République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Cycle de Formation d'Ingénieurs
dans la Discipline
Génie Informatique

ST-EN07/00

Projet de Fin d'Etudes
N 2021 / DIMA 33

MEMOIRE

Présenté à

L'Ecole Nationale d'Ingénieurs de Sfax
(Département de Génie Informatique et de Mathématiques Appliquées)

En vue de l'obtention

Du Diplôme National d'Ingénieur en Génie Informatique

Par

Emna AYEDI

Etude et Développement des nouvelles fonctionnalités pour l'ERP SAPTUN

Soutenu le 29 juin 2021, devant la commission d'examen :

Mr.	**Bechir ZALILA**	**Président**
Mr.	**Riadh BEN HALIMA**	**Examinateur**
Mme.	**Imene LAHIANI**	**Encadrante**
Mme.	**Mouna TOURJMEN**	**Encadrante**

Dédicace

Qu'il me soit permis à travers ce modeste travail d'exprimer ma plus grande reconnaissance

À

Mon cher Père **JALEL** Pour ses encouragements, ses considérables sacrifices, ses conseils qu'il m'a prodigués tout au long de ma vie pour que je parvienne à ce niveau. Qu'il trouve ici l'expression de ma gratitude, mon grand respect et mon amour infini.

Ma chère mère **RAOUDHA** Qu'aucun mot ne puisse exprimer ce que je lui dois pour son amour, sa bienveillance, son affection, sa patience, ses sacrifices… Trésor de bonté et de générosité et de tendresse, que Dieu la protège.

Ma chère grand-mère **Hamida** Pour ses prières et son amour inconditionnel, que Dieu lui donne bonne santé et longue vie parmi nous.

Ma sœur **MARIEM**, son mari **MOHAMED** et leur anges **MAYMAH** et **HMAYMAD**.

Ma sœur **KAWTHAR**, son mari **ADAM** et leur ange **ZAYDOUN**.

Mon frère **ALI**,

Mon mari **MOADH,**

Mes beaux-parents **RADHIA** et **SAID.**

Que Dieu nous garde toujours unis !

Finalement, je dédie mon travail à toute personne qui m'a aidé à faire ce projet.

Remerciements

J’étais honoré de faire mes études supérieures dans l’école nationale ingénieurs de Sfax. Je tiens pour cela, à remercier tous mes professeurs intervenants dans ma formation. Ensuite, j’exprime un remerciement spécial à mon encadrante académique **Mme IMENE LAHYANI** pour m’avoir encadré, orienté, aidé et conseillé. Elle a eu confiance en moi et m’a soutenu et encouragé. Je la remercie également pour sa patience et sa disponibilité durant toute la période de réalisation de ce projet.

C’est avec gratitude et respect que je tiens à exprimer mes profonds remerciements à **Mr**.**AHMED DRIRA** et **Mme. MARIEM SMAOUI** de m’avoir intégrée dans leur équipe et de m’avoir conseillé et aidé tout au long du stage.

Je tiens à remercier également toute la famille SOFTODO pour leur bienveillance et pour la bonne ambiance de travail

Je tiens à remercier, **Mr BECHIR ZALILA** pour l'honneur qu’il me fait de présider le jury de ce mémoire. Je suis particulièrement reconnaissante à **Mr RIADH BENHALIMA**, d'avoir bien voulu rapporter ce travail.

Finalement, mes remerciements s’adressent à tous mes collègues et amis de classe.

Table de Matières

Introduction générale **1**

Chapitre 1 : Etude préalable 3

1. Introduction 3
2. Présentation de l'organisme d'accueil 3
3. Contexte du projet 3
   - 3.1 Description générale du projet 4
   - 3.2 Etude de l'existant 4
   - 3.3 Critique de l'existant 5
   - 3.4 Solution proposée 6
4. Méthodologie adoptée pour la réalisation du projet 6
   - 4.1 Définition et avantages de la méthodologie SCRUM : 7
   - 4.2 Equipe Scrum 8
5. Planification du projet 8
6. Conclusion 9

Chapitre 2 : Spécification des besoins et analyse 10

1. Introduction 10
2. Spécification des besoins 10
   - 2.1 Identification des acteurs 10
   - 2.2 Les besoins fonctionnels 10
   - 2.3 Les besoins non fonctionnels 11
3. Choix technologique 12
   - 3.1 Environnement logiciel 12
   - 3.2 Environnement de développement 15
4. Description et Avantages de l'architecture générale du projet 20

5. Conclusion 21

Chapitre3 : Développement de nouvelles fonctionnalités pour l'ERP SAPTUN 22

1. Introduction 22

2. Développement des nouvelles fonctionnalités de l'ERP SAPTUN 22

2.1 Planification des Sprints 22

2.2 Conception et réalisation du Sprint1 23

2.3 Conception et réalisation du Sprint 2 39

3. Conclusion 53

Chapitre 4 : Déploiement et Surveillance de l'ERP SAPTUN 54

1. Introduction 54

2. La méthodologie DevOps 54

3. La chaîne DevOps de l'ERP SAPTUN 55

3.1 Intégration continue (CI) 55

3.2 Déploiement continue (CD) 56

3.3 Gitlab 56

4. Les stages du pipeline CI/CD : 58

4.1 Compilation du code 58

4.2 Test fonctionnel de l'application 58

4.3 Stockage de l'mage Docker dans le registre du GitLab 65

4.4 Déploiement de l'application sur le serveur de test 66

5. La supervision des performances 67

5.1 Prometheus 67

5.2 Eléments de performance 69

5.3 Grafana 70

5.4 La mise en place des notifications d'alerte 72

6. Conclusion 73

Conclusion et perspectives 74
Webgraphie 75

Liste des Tableaux

Tableau 1 : Planification du Projet 9
Tableau 2 : Backlog des fonctionnalités à ajouter 23
Tableau 3 : Description textuelle du CU « Effectuer un bon de transfert de produits entre dépôt » 24
Tableau 4 : Description textuelle du CU « Automatiser l'entrée numérique des produits » 26
Tableau 5 : Description textuelle du CU« Attribuer un compte bancaire au client » 40
Tableau 6 : Description textuelle du CU « Suspendre toutes les opérations de vente pour un client » 42

Table des Figures

Figure 1 logo SOFTODO 3
Figure 2 Architecture Modulaire de l'ERP SAPTUN 4
Figure 3 Le processus Scrum 7
Figure 4 Logo de Intellij 12
Figure 5 Logo de mongoDB 12
Figure 6 Logo de Gitlab 13
Figure 7 Logo de Jaspersoft Studio 13
Figure 8 Logo de Grafana 14
Figure 9 Logo de MobExterm 14
Figure 10 Logo de Docker 14
Figure 11 Logo de StartUML 15
Figure 12 Logo de Jira Software 15
Figure 13 Logo de Angular 16
Figure 14 Logo de NPM 16
Figure 15 Logo de Angular Material 16
Figure 16 Logo de HTML 17
Figure 17 Logo de CSS 17
Figure 18 Logo de Bootstrap 17
Figure 19 Logo de nodeJs 18
Figure 20 Logo de TypeScript 18
Figure 21 Logo de springBoot 18
Figure 22 Logo de mongoDB 19
Figure 23 Logo de Sélénium 19
Figure 24 Logo de JUnit 19
Figure 25 Architecture générale du projet 20
Figure 26 Diagramme de cas d'utilisation du Sprint 1 24
Figure 27 Diagramme de séquence du CU Effectuer un bon de transfert 28
Figure 28 Diagramme de séquence du CU «Automatiser l'entrée numérique des produits» 29
Figure 29 Diagramme de séquence du CU « Etablir l'état d'une caisse durant une période » 30

Figure 30 Diagramme de séquence de l'Interaction Créer un état de caisse *31*
Figure 31 diagramme de génération d'un rapport Jasper *32*
Figure 32 Diagramme de classes *33*
Figure 33 Ajouter un Bon de transfert *34*
Figure 34 Liste des bons de transfert *34*
Figure 35 Liste des bons de transfert qu'un dépôt a reçu *35*
Figure 36 Interface de traitement d'un bon de transfert *35*
Figure 37 Automatiser l'entrée numérique des produits *36*
Figure 38 Liste des produits refusés *36*
Figure 39 Liste des produits acceptés *37*
Figure 40 Interface de création d'un bon de retour *37*
Figure 41 Interface de la creation de l'etat de caisse *38*
Figure 42 La liste des réglements *38*
Figure 43 La liste des tickets *39*
Figure 44 Le rapport de l'état de caisse *39*
Figure 45 Diagramme de cas d'utilisation *40*
Figure 46 Diagramme de séquence de CU « Conversion des tickets en Bon de livraison » .. *43*
Figure 47 Diagramme de séquence de CU « Attribuer un compte bancaire au client » *44*
Figure 48 Diagramme de séquence de CU « Ajouter un nouveau compte bancaire au client » *44*
Figure 49 Diagramme de séquence de CU Suspendre les opérations de vente *45*
Figure 50 Diagramme de classes du Sprint 2 ..... *46*
Figure 51 L'interface des tickets effectues pour le client passager *47*
Figure 52 L'interface de conversion des tickets en bon de livraison *47*
Figure 53 L'interface de choix du client *48*
Figure 54 L'interface de fiche client *48*
Figure 55 Interface de suivi de la variation de l'état du client *49*
Figure 56 « Attribuer un compte bancaire au client » *50*
Figure 57 La possibilité d'avoir un ou plusieurs comptes bancaires *50*
Figure 58 Interface des comptes bancaires que possède un client *51*
Figure 59 Interface de la liste des clients *51*

Figure 60 Etablir l'état des factures par période *52*
Figure 61 Interaction pour établir l'état des factures par période *52*
Figure 62 La différence entre les méthodologies DevOps et Agile *54*
Figure 63 La chaine DevOps de l'ERP SAPTUN *55*
Figure 64 La chaine de l'intégration continue de l'ERP SAPTUN *55*
Figure 65 Les composants nécessaires du gitlab_ci.yml de l'ERP SAPTUN *57*
Figure 66 La description de l'étape de build dans le fichier gitlab_ci.yml *58*
Figure 67 La liste des "Feature File" du Test fonctionnel *60*
Figure 68 Un exemple de code du "Feature File" *60*
Figure 69 Exemple de code du fichier Java *61*
Figure 70 L'interface du test fonctionnel de l'ajout des comptes bancaires *62*
Figure 71 L'interface du test fonctionnel de l'ajout des comptes bancaires *63*
Figure 72 L'interface qui illustre le test fonctionnel de la création d'un bon de transfert *63*
Figure 73 La liste des produits refusés *64*
Figure 74 La liste des produits acceptés *64*
Figure 75 La description de l'étape de package dans le fichier gitlab_ci.yml *65*
Figure 76 Les différents images crées sur le serveur de test *67*
Figure 77 Le fichier prometheus.yml *68*
Figure 78 Les cibles de Prometheus *68*
Figure 79 La source de données pour Grafana *70*
Figure 80 Tableau de bord de supervision de l'application *72*
Figure 81 Tableau de bord de supervision du serveur *72*
Figure 82 Email d'alerte concernant l'utilisation de la mémoire *73*

Introduction générale

Le commerce est un domaine qui est devenu de plus en plus complexe et changeant vu l'évolution des exigences du marché en termes de qualité, de flexibilité, de délai et de prix.

Dans ce contexte déstabilisant, l'entreprise commerciale doit faire preuve d'adaptation afin de rester un générateur de profit, de maintenir la satisfaction clientèle et de garder une compétitivité durable.

Pour ce faire, être consciente de l'impact de la gestion des ressources internes sur sa performance pousse l'entreprise à investir des sommes considérables dans l'implantation de technologies logicielles.

Les progiciels ERP (Entreprise Ressources Planning) définis selon le célèbre éditeur, SAP, comme « un outil de gestion des ressources de l'entreprise qui permet d'assembler toutes les activités dans un modèle ou un processus qui automatise et concorde toutes les transactions qui lui sont associées » (SAP, 1996), viennent supporter de telles orientations organisationnelles et stratégiques.

Les enjeux de la mise en place des logiciels sont sensiblement reliés à la cohérence et l'homogénéité de l'information interne et surtout l'aboutissement à une meilleure agilité vis-à-vis du client par l'identification rapide de nouveaux prospects.

Dans ce cadre, la société informatique SOFTODO a développé un ERP de gestion commerciale SAPTUN qui permet de gérer le cycle de vie des produits et les différents services d'achat et de vente.

Notre stage de fin d'études est effectué chez SOFTODO qui consiste à développer de nouvelles fonctionnalités pour l'ERP SAPTUN, effectuer des tests fonctionnels pour ce qui a été développé, mettre en place un déploiement efficace de l'application en créant un conteneur de l'application avec Docker et en suivant la démarche DevOps et enfin mettre en place un système de supervision de l'application et du serveur de test.

Le présent rapport s'articule également autour de quatre chapitres dans lesquels nous essayerons de synthétiser les différentes tâches réalisées. Le premier chapitre est consacré à présenter l'entreprise d'accueil, le contexte du projet, la méthodologie adaptée pour atteindre les objectifs et enfin la planification du projet. Le deuxième chapitre concerne la spécification

des besoins fonctionnels et non fonctionnels, l'analyse du choix technologique ainsi que l'architecture technique de notre projet. Le troisième chapitre traite les nouvelles fonctionnalités ajoutées au ERP SAPTUN en présentant les diagrammes de conception et les interfaces de réalisation de ces fonctionnalités par sprint. Quant au quatrième chapitre, il est consacré à l'optimisation du déploiement de notre système en présentant la chaine DevOps adoptée, la description de chaque étape du cycle de vie de l'ERP ainsi que la mise en place d'un système de supervision de l'application et du serveur de test. Enfin, nous clôturons notre rapport par une conclusion accompagnée par quelques perspectives.

Chapitre 1 : Etude préalable

1. Introduction

Dans de ce chapitre nous présentons le cadre général du projet. En premier lieu, nous présentons l'organisme d'accueil, puis nous découvrons le contexte dans lequel le sujet a été proposé et enfin nous dévoilons la méthodologie adoptée et le plan général pour la réalisation de ce projet.

2. Présentation de l'organisme d'accueil

SOFTODO est une société Tuniso-Allemande qui a été fondu en 2012 et compte aujourd'hui une vingtaine de développeurs dirigées par le directeur de production « Mahmoud Siala » qui a plus de 10 ans d'expérience en Outsourcing de solutions logiciels.

Figure 1 logo SOFTODO

Un positionnement innovant sur le marché d'ingénierie et de développement de logiciel offshore est marqué par SOFTODO qui se traduit par une coopération avec les partenaires et par la haute qualité des solutions d'externalisation offshore proposées à eux.

SOFTODO propose une large gamme de solutions personnalisées telles que le développement web (**Java, php, Typo3, SpringBoot, Angular** ,**Laravel**) , le traitement de données et le test d'applications.

Développeurs et consultants collaborent également sur la conception d'une solution ERP.

Cette société vise à offrir les services informatiques les mieux adaptés aux besoins du client en s'appuyant sur les méthodologies les plus appropriées et en utilisant les nouvelles technologies.

3. Contexte du projet

Dans ce qui suit nous présentons une analyse de l'application existante. Ensuite, nous dégageons les différentes imperfections dans le système actuel et nous décrivons notre solution proposée.

3.1 Description générale du projet

L'ERP SAPTUN est un projet développé chez SOFTODO depuis décembre 2018 pour une gestion complète et personnalisable dédié à des sociétés commerciales. Toutes les briques nécessaires pour la mise en production de SAPTUN est assurée avec le gestionnaire de projet **Gitlab** qui offre une gestion optimale de versions et un système performant d'intégration continue. Cette performance est un facteur clé chez l'équipe SOFTODO pour garder la continuité relationnelle avec leurs partenaires.

L'établissement, le développement et le maintien de la qualité de service sont des principes de l'équipe qui donnent toujours naissance à des nouveaux besoins fonctionnels et non fonctionnels. L'origine de ces besoins peut être issue des envies du collaborateur actuel qui arriveront en cours de route ou bien issue des changements du pratiques et de la méthodologie de livraison de logiciels que SOFTODO vise à adopter dans le but d'accélérer la mise sur le marché et d'attirer des nouveaux clients.

Dans ce contexte, des nouvelles fonctionnalités qui concernent les différents modules de l'ERP tels que le module CRM (Client Ressources Management), vente, achat et stock sont demandées aujourd'hui de la part du client.

Vient également s'ajouter à ce fait la nécessité d'une amélioration du concept et des étapes de livraison du produit. Une nouvelle stratégie de déploiement du logiciel privilégiera un environnement de démarrage rapide et isolée chez le client.

3.2 Etude de l'existant

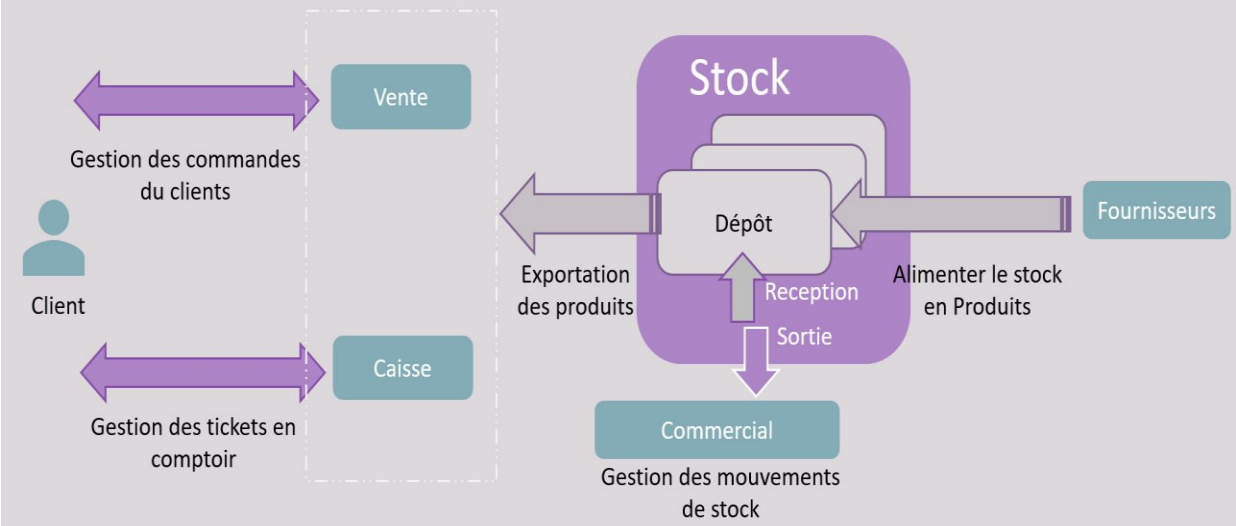


Figure 2 Architecture Modulaire de l'ERP SAPTUN

La figure 2 présente l'architecture modulaire de l'ERP SAPTUN qui dévoile la manière dont l'ERP SAPTUN optimise la gestion commerciale. En effet, recevoir les produits auprès des fournisseurs est l'opération principale de service d'achat qu'offre l'ERP pour alimenter le stock. Celui-ci est distribué sur plusieurs dépôts que chacun possède des caisses conçues pour accomplir une vente directe en comptoir avec le client. Par la suite, le caissier affecte au client un ticket de caisse afin de décrémenter le stock du dépôt et de calculer la somme à payer.

De plus, le client peut passer des commandes que l'administrateur traite en examinant la disponibilité des produits puis il poursuit l'opération de vente par la génération des bons de livraison et des factures de vente. De ce fait, des mouvements quotidiens de produits sont effectués par des commerciaux inclus dans l'ERP.

De ce fait, une cohérence et une homogénéité de l'information sont offertes, ce qui permet la communication rapide entre les parties prenantes de la gestion commerciale et donc un cycle de vente fluide et automatisé avec le client.

L'utilisation de ce progiciel permet de restreindre fortement le risque d'erreur lié à la saisie manuelle des factures, de garantir une traçabilité des encaissements et leurs règlements et donc d'éviter les éventuels litiges entre le vendeur et l'acheteur.

De plus, l'utilisateur de SAPTUN peut sécuriser tous les mouvements des produits par la création des documents contenant les informations nécessaires au voyage comme l'ordre de mission pour accorder la mission au **commercial**, la feuille de route pour légaliser la sortie et un acquittement pour prouver que la livraison est effectuée avec succès.

SAPTUN est un vrai gestionnaire qui a pu couvrir plusieurs tâches de l'administrateur, du caissier et du commercial et en offrir des fonctionnalités qui facilite énormément ces tâches.

3.3 Critique de l'existant

Grâce à une visibilité en temps réel sur ses opérations, le dirigeant de l'entreprise commerciale arrive à identifier rapidement les nouvelles exigences. Dans ce contexte, SAPTUN doit être en mesure de répondre aux besoins présents de partenaire, mais aussi à ceux qu'il connaît dans plusieurs mois, voire plusieurs années.

Au fur et à mesure de son évolution, l'organisation commerciale souffre aujourd'hui d'un accroissement de la charge de dépôts et une amplification du flux de règlements et de factures ce qui a entravé le processus de la vente et le management de la relation clientèle.

Malgré le progrès de l'ERP SAPTUN, son déploiement reste une phase très pénible. En effet, l'équipe passe beaucoup de temps à déboguer et à analyser les différences entre les environnements de développement et de production. Ce contraignant processus entrave la production de nouvelles fonctionnalités et peut entraîner des problèmes de satisfaction du client au fil du temps.

La gestion d'un flux considérable de données provoque une dégradation du temps de réponse aux requêtes utilisateur et parfois une indisponibilité totale d'un service ou encore un accès impossible au serveur. C'est un risque qui peut conduire à la baisse de l'activité commerciale.

3.4 Solution proposée

Face aux anomalies existantes, le premier objectif de ce projet est de transformer les nouvelles exigences du client en fonctionnalités intégrées dans l'ERP. En effet, les nouvelles fonctionnalités permettent un meilleur approvisionnement du stock grâce à l'interaction physique des dépôts du stock. Ainsi une relation avancée avec le client est offerte pour maintenir sa satisfaction. En outre une rapide analyse de service de comptabilité et de l'état des clients est fournie à travers la création d'une vue transversale et précise des flux d'informations de vente.

D'autre part, le deuxième objectif consiste à mettre en place un déploiement efficace de l'application, et de manière adaptable, sur n'importe quel serveur en créant un conteneur de l'application avec Docker et en suivant la démarche DevOps de façon à avoir un environnement isolé et rapide du développement au déploiement.

Enfin, dans le but de garder et améliorer la performance de l'ERP, le dernier objectif de notre projet est la mise en place d'un système de supervision de l'application et du serveur de test avec la possibilité de faire des alertes pour faciliter la mise en discussion de l'information avec les différents acteurs impliqués. Plus le système est performant plus l'expérience utilisateur est satisfaisante et la qualité est assumée.

4. Méthodologie adoptée pour la réalisation du projet

Le succès de l'ERP SAPTUN est la conséquence d'une vraie agilité dans le processus du travail. En effet, le client intervient à différentes étapes du projet pour visualiser et valider. Des ajustements dans la planification du projet s'introduisent en cours de route en fonction des

priorités du client. Souple, itérative et collaborative, la méthode agile peut conduire au changement avec un maximum d'efficacité

4.1 Définition et avantages de la méthodologie SCRUM :

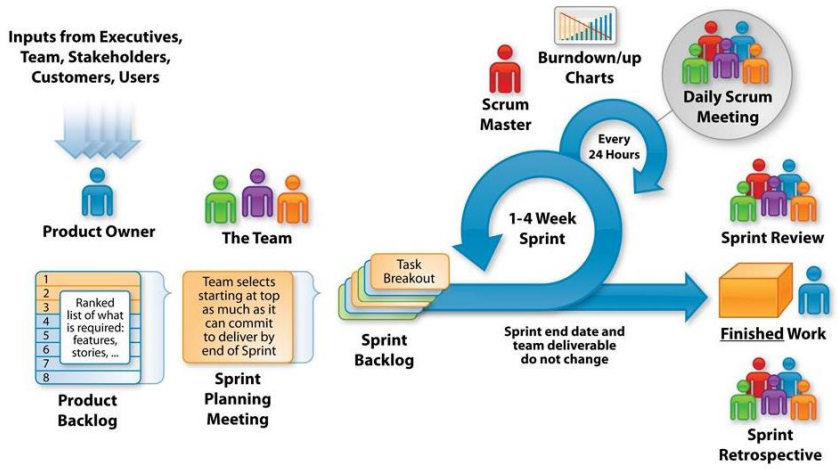


Figure 3 Le processus Scrum

La figure 3 montre que la méthodologie Scrum est un modèle planifié par des itérations à durée courte (de 1 à 4 semaines) appelés « sprint » à la fin desquels l'équipe Scrum, composée de 3 rôles clés (Scrum master, product owner et équipe de développement), fait livrer des incréments fonctionnels du logiciel. De ce fait, cette méthodologie offre la rapidité de test des nouveaux développements qui permet une immédiate correction des erreurs et donc une meilleure gestion des risques. L'amélioration de la productivité d'une équipe est un autre avantage que l'offre Scrum en exigeant une certaine responsabilité des développeurs. En effet, l'équipe est autoorganisée c'est à dire les membres peuvent décider entre eux "Qui fait Quoi et Comment ?" pour atteindre un objectif commun définit au début de chaque Sprint.

L'équipe de l'ERP SAPTUN est composée de trois stagiaires et deux encadrants donc le partage des connaissances permet de monter en compétences et d'avancer beaucoup plus vite pour réaliser un but commun qui est le développement et l'évolution de l'ERP. De plus effectuer des réunions quotidiennes avec les encadrants permet une adaptation rapide aux besoins changeants des encadrants et un control efficace de l'avancement des taches de chaque stagiaire, les difficultés rencontrées et les objectifs atteints.

Par conséquence, la méthodologie Scrum sera le bon choix pour la réalisation de notre projet. Elle se contente d'offrir un cadre de gestion pour l'ERP SAPTUN organisé autour de « sprints » de développement d'une durée de 10 à 15 jours.

4.2 Equipe Scrum

- AHMED DRIRA, un encadrant qui représente le Product Owner (PO) : responsable de la conception du produit à réaliser. Il est en interaction directe avec l'équipe de développeurs qui doit suivre ses instructions. C'est lui qui établit la priorité des fonctionnalités à développer ou à corriger au début de chaque sprint et la validation de celles terminées à la fin de celui-ci. Le PO est responsable de la gestion du product backlog.
- MARIEM SMAOUI, une encadrante qui représente le Scrum Master (SM) : un membre de l'équipe projet qui n'a pas vocation à la diriger, mais à la guider afin de maximiser la productivité et le savoir-faire de l'équipe. Il maîtrise parfaitement la méthode Scrum pour une mise en œuvre efficace de ses principes et ses valeurs. En effet, son rôle est de former les membres aux pratiques agiles et d'animer les différentes réunions de Scrum.
- Trois stagiaires qui représentent l'équipe de développement : chargée de transformer les besoins définis par le Product Owner en fonctionnalités réalisables et fournir un incrément « Fini » potentiellement publiable (Releasable) à la fin de chaque Sprint. Il n'y a pas de notion de hiérarchie, toutes les décisions sont prises ensemble. C'est une équipe autopilotée qui collabore en restant autonome.

5. Planification du projet

La réussite d'un projet fondé sur les pratiques agiles réside dans la planification prévisionnelle des sprints pour une estimation de la composition et de la date d'achèvement des sprints. Dans ce contexte, nous avons proposé une planification pour la réalisation de nos objectifs attendus de ce projet.

Objectif	Sprint	Details	Période
Se familiariser avec l'ERP : environnements et outils	0	- Formation SpringBoot et Angular. - Découverte des fonctionnalités existantes de l'ERP SAPTUN.	Du 15/02/2021 Au 26/02/2021
Développement des nouvelles exigences métier pour l'ERP	1	- Rédiger le backlog avec le client -Faire la découpe en User story	Du 01/03/2021 Au 31/03/2021
	2		
Test du fonctionnalités ajoutées à l'ERP	3	- Formation Sélénium	Du 01/04/2021 Au 16/04/2021

Se Familiariser avec les notions devops	4	- Comprendre les notions CI/CD	Du 19/04/2021 Au 30/04/2021
Mettre en place un pipeline CI/CD pour l'ERP avec Docker	5	- Formation docker	Du 03/05/2021 Au 14/05/2021
Mettre en place un système de supervision de l'ERP et du serveur	6	- Formation sur les métriques de performance - Formation sur l'utilisation de Prometheus et Grafana	Du 17/05/2021 Au 04/06/2021

Tableau 1 : Planification du Projet

6. Conclusion

Après la présentation du cadre, de la méthodologie choisie à adopter et de la planification du projet, nous allons entamer la phase d'analyse en citant les besoins fonctionnels et non fonctionnels de notre projet et les outils utilisées. Ceci fera l'objet du chapitre suivant.

Chapitre 2 : Spécification des besoins et analyse

1. Introduction

La phase de spécification et d'analyse des besoins est la première étape de notre processus de développement. En effet, nous commençons par la spécification des besoins fonctionnels et non fonctionnels décelés puis nous identifions l'environnement logiciel et celui de développement de notre système et enfin nous présentons l'architecture technique de notre système et ses avantages.

2. Spécification des besoins

Dans cette section, nous définissons en détails l'ensemble des fonctionnalités offertes par notre application. Nous commençons par identifier des acteurs suivis par la présentation des besoins fonctionnels relatifs à ces acteurs ainsi que les besoins non fonctionnels.

2.1 Identification des acteurs

Les différents acteurs susceptibles d'interagir avec notre solution sont l'administrateur de l'organisation commerciale et le magasinier.

Administrateur : Il a le privilège de profiter de toutes les fonctionnalités offertes tel que l'approvisionnement du stock, le contrôle des états du clients et la suivie des opérations de vente.

Magasinier : Réceptionne et stocke les produits reçus par le dépôt auquel il appartient.

2.2 Les besoins fonctionnels

L'ERP SAPTUN répond aux besoins du client par l'implémentation de ces fonctionnalités :

- Retourner des produits au stock à la suite d'un refus de la part du client.
- Affecter des comptes bancaires au client lors ou après son enregistrement.
- Suspendre l'ajout des commandes, des tickets et des bons de livraison pour un client.
- Suivre l'état du client via un graphe qui illustre son historique et l'évolution de celle-ci (Solde, Achats).
- Convertir des tickets de caisse d'un passager en bon de livraison avec une identification précise de ses coordonnées (adresse, téléphone, compte bancaire, etc.).
- Automatiser l'entrée numérique des produits.

- Etablir l'état d'une caisse de dépôt le long d'une période précise pour avoir une vue condensée sur le nombre des tickets et des règlements effectués durant cet intervalle de temps, ainsi que la somme totale des tickets et le montant total payé.
- Etablir la liste des factures effectuées le long d'une période.

La performance de l'ERP SAPTUN est optimisée par l'implémentation d'une chaine DevOps qui permet de :

- Tester le bon fonctionnement des nouvelles fonctionnalités ajoutées afin d'éviter les mauvaises surprises qui peuvent survenir.
- Mettre en place un déploiement facile et adaptable avec n'importe quel serveur par l'injection de l'application et tous ses dépendances dans un conteneur docker.
- Surveiller l'application et le serveur de test :
  - L'usage CPU, RAM
  - La latence des requêtes http
  - Le nombre de threads en cours d'utilisation, d'instances actives, de classes chargées, etc.

2.3 Les besoins non fonctionnels

Après avoir identifié les besoins fonctionnels de notre système, nous avons veillé également à la prise en considération de certains besoins non fonctionnels tout au long de notre processus de développement à savoir :

- Maintenabilité, extensibilité et scalabilité :
  - Un système modulaire : faire un développement par composition et se conformer à des standards pour permettre par la suite l'amélioration de l'application d'une manière facile, c'est-à dire qu'il pourra y avoir une possibilité d'ajouter ou de modifier de nouvelles fonctionnalités.
  - Un code source lisible, commenté et compréhensible afin d'assurer son état évolutif et extensible par rapport aux besoins de l'entreprise. Respecter les 5 principes SOLID au cours du développement.
- L'ergonomie et la simplicité :
  - Pour faciliter l'exploitation et la manipulation de l'application même par des non experts, des interfaces simples et conviviales doivent être fournies.
  - Le tableau de monitoring doit être compris de façon intuitive, c'est-à-dire les courbes de mesure de performance doivent être affichés de façon claire et sans ambiguïté.

3. Choix technologique

Dans cette partie, nous énumérons les langages et technologies utilisés pour la modélisation, l'implémentation et la gestion de notre projet.

3.1 Environnement logiciel

Nous présentons les différents logiciels selon le contexte de l'utilisation.

3.1.1 Logiciels de développement

- **Intellij IDEA**

[1] « C'est un environnement de développement intégré (IDE) Java développé par JetBrains .Intellij IDEA optimise la productivité des développeurs. En effet, il offre une saisie automatique et pertinente du code et une analyse à la volée de celui-ci. De plus, il permet une étroite intégration avec les outils de développement les plus répandus tels que des gestionnaires de version comme Git et CVS, des outils de construction de projets comme Ant et Maven et des outils de test comme JUnit et TestNg ».

Figure 4 Logo de Intellij

- **MongoDB Compass**

[2] « C'est une interface graphique pour MongoDB qui permet d'envisager de nouvelles perspectives de gestion pour les bases de données NoSQL orientées documents ».

Figure 5 Logo de mongoDB

- **Gitlab**

[3] « GitLab est une plateforme de développement open source qui couvre tous les briques nécessaires au développement et à la production. En effet, elle englobe l'intégralité des fonctionnalités propres à la méthodologie Devops : commençant par la planification du développement, passant par la compilation et l'intégration du code, ensuite le lacement des tests unitaires et la mesure de qualité du code et enfin la gestion des livrables pour obtenir les artefacts. Se basant sur les fonctionnalités du logiciel Git, elle permet de gérer des dépôts de code source et leurs différentes versions ».

Figure 6 Logo de Gitlab

- **JasperSoft Studio**

[4] « Jaspersoft Studio est un générateur de rapports de développement open source basé sur Java. Il permet rapidement de créer, structurer des rapports variés avec une mise en page raffinée et obtenir le fichier jrxml qui définit la présentation du rapport ».

Figure 7 Logo de Jaspersoft Studio

3.1.2 Logiciels de Supervision :

- **Grafana**

[5] « Grafana est une plateforme de supervision open source, créé en janvier 2014 par Torkel Ödegaard et orientée vers la surveillance des données systèmes en temps réel. Elle peut être déployé avec Docker et peut être exécuté en Gnu/Linux, Windows et MacOs.

[6] Cet outil est indispensable pour visualiser, consulter et aussi établir des alertes en présentant une grande quantité de données issues de sources différentes dans un ou plusieurs tableaux de bord et d'une manière simple, efficace et agréable ».

Figure 8 Logo de Grafana

3.1.3 Logiciels de Déploiement :

- **MobaXterm**

[7] « MobaXterm est un émulateur de terminal Linux pour Windows. Celui-ci vous offre la possibilité de vous connecter à distance au serveur du Département par la commande ssh. Un serveur X est intégré au logiciel, donc vous pouvez l'utiliser pour démarrer des logiciels employant l'interface graphique ».

Figure 9 Logo de MobExterm

- **Docker**

[8] « Docker est un logiciel libre qui permet de lancer des applications dans des conteneurs linux : un seul package qui regroupe l'application avec tous ses besoins en bibliothèques et autres dépendances. Par conséquent, Ces conteneurs ont l'avantage d'être portables sur les différentes distributions Linux ».

Figure 10 Logo de Docker

3.1.4 Logiciels de modélisation

- **StartUML**

[9] « StartUML est un logiciel de modélisation UML (Unified Modeling Language) open source qui permet de concevoir les applications grâce à sa génération de la plupart des diagrammes UML 2.0 ».

Figure 11 Logo de StartUML

3.1.5 Logiciels de gestion de projet

- **Jira Software**

[10] « Centré sur la gestion de projet de développement logiciel agiles, Jira Software est un puissant outil qui facilite la planification des tâches et leur suivi. Il prend en charge toute méthodologie Agile (tableau Scrum, tableau Kanban) et permet de créer le backlog et la feuille de route du projet pour une planification claire et visible par toute l'équipe. De plus, Jira offre une restitution des données liées à l'activité de l'équipe pour contrôler les contributions de chaque membre grâce à des tableaux de bord configurables et conviviales ».

Figure 12 Logo de Jira Software

3.2 Environnement de développement

Nous présentons les environnements de développement selon le contexte d'utilisation.

3.2.1 Partie Frontend

- **Angular**

[11] « Angular est un Framework coté client open source développé par les équipes de Google, basé sur TypeScript. Il permet la création d'applications web accessibles via une page web unique pour éviter les chargements de pages à chaque nouvelle action. ». Il offre une interface très réactive et dynamique qui permet de fluidifier l'expérience utilisateur.

Figure 13 Logo de Angular

- **NPM**

[12] « NPM est le gestionnaire de paquets officiel de Node.js. C'est le plus grand registre de logiciel au monde. Il permet de gérer les dépendances pour une application et d'installer des application Node.js disponibles sur son dépôt ».

Figure 14 Logo de NPM

- **Angular Material**

[13] « Angular Material est un module d'Angular. Il s'agit d'un module d'intégration qui permet d'obtenir facilement une interface responsive harmonieuse et unie dans le style 'flat' de google ».

Figure 15 Logo de Angular Material

- **HTML**

[14] « L'HyperText Markup Language, généralement abrégé HTML, est le langage de balisage conçu pour représenter les pages web. Il permet également de structurer sémantiquement et logiquement le contenu des pages et d'inclure des ressources multimédias dont des images, des formulaires de saisie, etc. ».

Figure 16 Logo de HTML

- **CSS**

[15] « Les feuilles de style en cascade1, appelées CSS « Cascading Style Sheets », forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000 ».

Figure 17 Logo de CSS

- **Bootstrap**

[16] « Bootstrap est une infrastructure de développement frontale, gratuite et open source pour la création de sites et d'applications Web. Le Framework Bootstrap repose sur HTML, CSS et JavaScript pour fournir aux développeurs un outils facile et efficace pour la création rapide des sites web responsives qui peuvent s’adapter sur n’importe quelle taille d’écran ».

Figure 18 Logo de Bootstrap

- **Node.js**

[17] « Node.js est une plateforme logicielle libre construite sur le moteur JavaScript V8 de Chrome qui permet de développer des applications en utilisant du JavaScript. Il se distingue

des autres plateformes grâce à une approche non bloquante permettant d'effectuer des entrées/sorties (I/O) de manière asynchrone » .

Figure 19 Logo de nodeJs

- **TypeScript**

[19] « TypeScript est un langage compilé et typé qui génère du JavaScript compréhensible par tous les navigateurs. TypeScript permet d'associer un type à un élément et empêche cet élément de changer de type. Cela offre une stabilité au niveau du code, dès lors il est possible d'anticiper le type d'un élément et ses différentes valeurs possibles ».

Figure 20 Logo de TypeScript

3.2.2 Partie Backend

- **SpringBoot**

[20] « Spring Boot est un framework qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée en jar, totalement autonome ». Il réduit la complexité de la configuration des projets, facilite la gestion des versions grâce à l'ajout automatique des indépendances compatibles et simplifie la prise en main.

Figure 21 Logo de springBoot

3.2.3 Base de données

- **MongoDB**

[21] « MongoDB (de l'anglais humongous qui peut être traduit par « énorme ») est un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Il est écrit en C++. Il fait partie de la mouvance NoSQL ». Il offre une souplesse d'utilisation et une évolutivité vu qu'une BDD orientée documents est plus scalable qu'une BDD relationnelle et la structure d'un document est proche de celle des classes et des objets du langage de programmation utilisé.

Figure 22 Logo de mongoDB

3.2.4 Partie Test

- **Sélénium**

[22] « Selenium est un framework de test automatisé open source développée en Java et utilisé pour valider le bon fonctionnement des applications Web sur différents navigateurs tel que Google Chrome grâce au chromedriver ou Mozilla Firefox avec Gecko ».

Figure 23 Logo de Sélénium

- **JUnit**

[23] **« JUnit** est un framework de test unitaire Java créé par Kent Beck et Erich Gamma ».

Figure 24 Logo de JUnit

4. Description et Avantages de l'architecture générale du projet

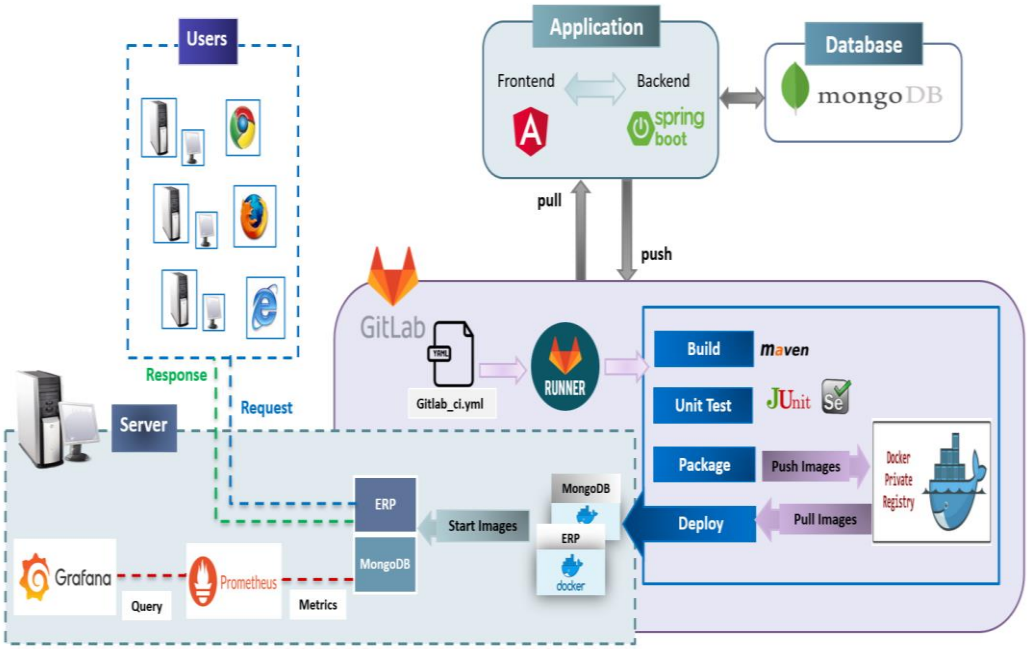


Figure 25 Architecture générale du projet

Spring Boot et **Angular** sont deux frameworks particulièrement pris pour le développement de l'ERP SAPTUN. Ils offrent un cadre structurant pour un développement productif, testable et facile à déployer. Spring Boot, orienté back-end, réduit la complexité de la configuration des projets. Angular, orienté front-end, offre une interface très réactive et dynamique. **MongoDB** est choisi comme base de données. Elle adopte un modèle de données de type document qui lui confère une grande souplesse d'utilisation et une vraie évolutivité. Un système de gestion de projet **GitLab** est utilisée. Il dispose d'un dépôt Git qui centralise le code source en gardant l'historique des versions, des « commits », etc. Le cycle de vie de l'ERP se repose sur les étapes qui constituent la chaine Devops tel que la compilation et le **build** du code source, l'exécution des **tests unitaires**, puis la création et le sauvegarde d'une **image Docker** dans un **registre docker** et enfin le déploiement dans un **serveur** distant pour l'obtention de cette application et ses dépendances dans un **conteneur docker**. Toute poussée en amont dans le référentiel Git sera détectée par un orchestrateur appelé « **GitlabCI** ». Un exécuteur appelé « **GitLab Runner** » intervient et permet d'automatiser ces briques grâce à un manifeste « **gitlab-ci.yml** » qui définit à son tour toute la configuration du cycle de vie de l'ERP. Une visualisation des données de performance de l'ERP sous forme graphique est offerte par l'utilisation de **Prometheus** qui interroge à intervalle régulier l'application pour

récupérer les données. **Grafana** propose une liste de dashboards pré-générés pour récupérer les informations en provenance de Prometheus et exposer les mesures de métriques de performance dans des graphes conviviales.

5. Conclusion

Pour conclure, la description exhaustive des fonctionnalités, l'identification du choix technologique et la description de l'organisation générale du système et sa décomposition en sous-systèmes nous permet d'avoir une vision claire et précise de ce qui sera réalisé. Le chapitre suivant est consacré pour la conception et la réalisation de la partie développement de nouvelles fonctionnalités métiers.

Chapitre3 : Développement de nouvelles fonctionnalités pour l'ERP SAPTUN

1. Introduction

Après avoir décortiqué les besoins et dévoilé l'architecture, nous rentrons dans le vif du sujet avec la conception et la réalisation du projet. Dans ce chapitre nous présentons le processus du développement des fonctionnalités ajoutées à l'ERP SAPTUN.

2. Développement des nouvelles fonctionnalités de l'ERP SAPTUN

Dans cette partie, nous présentons le processus de développement de toutes les fonctionnalités ajoutées au ERP SAPTUN de la planification jusqu'au réalisation.

2.1 Planification des Sprints

Lors de la génération du backlog, l'estimation de la complexité de chaque user story engendre une répartition des fonctionnalités à développer sur deux sprints SCRUM.

User Story	**Sous-taches**
Sprint 1	
En tant qu'utilisateur, je veux effectuer un bon de transfert de produits entre dépôts.	
En tant que magasinier, je veux traiter l'état d'un bon de transfert de produits destiné au dépôt auquel il appartient.	Accepter un bon de transfert Refuser un bon de transfert
En tant qu'administrateur, je veux effectuer un bon de retour de produits.	
En tant qu'administrateur, je veux établir l'état d'une caisse durant une période.	Extraire les données des tickets Extraire les données des règlements

	Générer un rapport pour les règlements de caisse
En tant qu'administrateur, je veux automatiser l'entrée numérique des produits.	
Sprint 2	
En tant qu'administrateur, je veux bloquer un client.	
En tant qu'administrateur, je veux affecter des comptes bancaires à un client.	Créer des comptes bancaires Modifier des comptes bancaires Supprimer des comptes bancaires
En tant qu'administrateur, je veux établir l'état des factures par période.	Extraire les données des factures Générer un rapport pour les factures
En tant qu'administrateur, je veux suivre la variation de l'état du client par période.	Suivre les achats du client Suivre le solde du client
En tant qu'administrateur, je veux changer le profil du client passager à la conversion des tickets en bon de livraison.	

Tableau 2 : Backlog des nouvelles fonctionnalités à ajouter

2.2 Conception et réalisation du Sprint1

Après la planification des sprints, nous présentons la conception de l'ensemble de « User Story » du Sprint 1 puis nous exposons ce que nous avons réalisé sur le plan pratique.

2.2.1 Conception du Sprint 1

Dans cette section, nous allons aborder la conception des fonctionnalités à travers la présentation du diagramme de cas d'utilisation, l'exploration des diagrammes de séquences relatifs aux cas d'utilisations et enfin la présentation du diagramme de classe des entités persistantes.

2.2.1.1 Diagramme de cas d'utilisation

Pour décrire les scénarios les plus importants dans cette phase, nous nous sommes basés sur le diagramme des cas d'utilisation et la description textuelle de ces derniers.

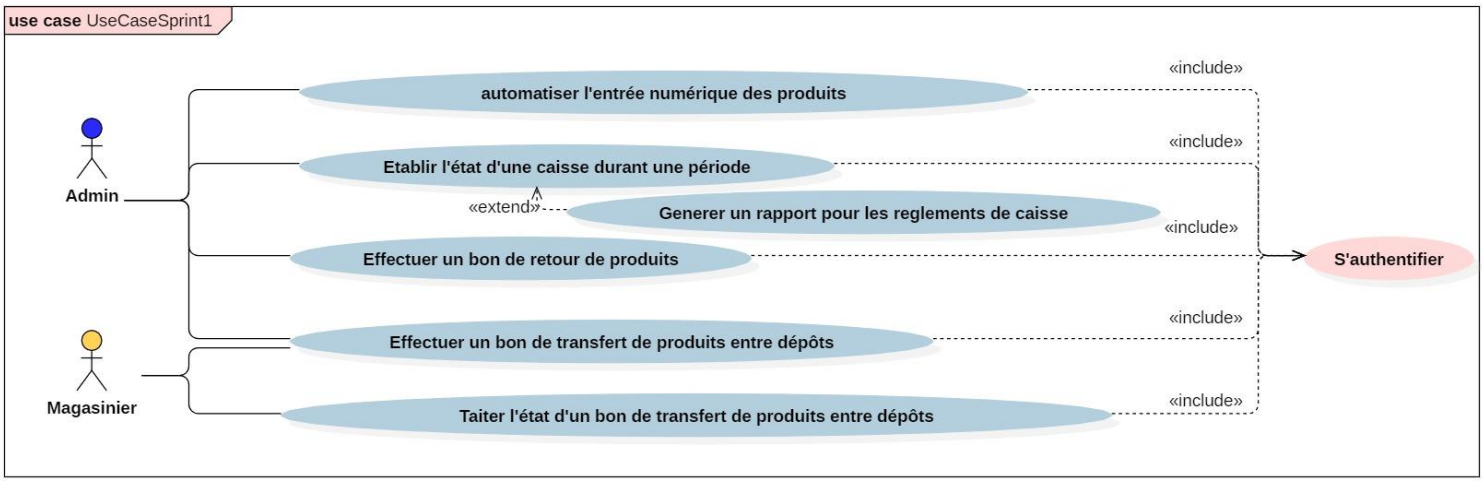


Figure 26 Diagramme de cas d'utilisation du Sprint 1

Tableau 3 : Description textuelle du CU « Effectuer un bon de transfert de produits entre dépôt »

Sommaire	
Acteur	Utilisateur (Administrateur ou magasinier)
Résumé	L'utilisateur effectue un bon de transfert de produits entre deux dépôts.
Description de l'enchaînement	
Pré-condition	L'utilisateur doit être authentifié et doit accéder à l'interface de la liste des bons de transfert
Post-condition	Un bon de transfert est créé
Scénario nominal	1- L'utilisateur clique sur le bouton « Ajouter un bon de transfert ». 2- Le système redirige l'utilisateur vers l'interface de création du bon de transfert

	3- L'utilisateur choisit le dépôt de source 4- Le système télécharge la liste des produits du dépôt source 5- L'utilisateur choisit le dépôt de destination 6- Le système vérifie la non-coïncidence entre le dépôt de source et le dépôt de destination 7- L'utilisateur entre le code ou la description dans les champs de ligne produit 8- Le système complète la ligne produit par le champ de la quantité en entrant 1 par défaut 9- L'utilisateur entre la quantité désirée à transférer 10- Le système vérifie la validité de la valeur entrée 11- Les étapes 7, 8 ,9 et 10 peuvent se répéter 12- L'utilisateur clique sur le bouton « Sauvegarder » 13- Le système vérifie les données entrées 14- Le système enregistre le bon de transfert
Scénario alternatif	**A1- Au point 6 : La coïncidence du dépôt de destination avec celui de source** 1- Le système réinitialise le champ du dépôt de destination à vide 2- Le scenario nominal reprend au point 5 **A2- Au point 10 : La quantité entrée dépasse la quantité du produit stockée dans le dépôt de source** 1- Le système affiche un message d'attention 2- Le scenario nominal reprend au point 9 **A3- Au point 13 : Les données sont incomplètes** 1- Le système affiche un message d'erreur

	2- Le scenario nominal reprend au point 3, 5 ou 7 selon l'issue de l'erreur

Tableau 4 : Description textuelle du CU « Automatiser l'entrée numérique des produits »

Sommaire	
Acteur	Administrateur
Résumé	L'administrateur automatise l'entrée numérique des produits à partir d'un fichier Excel
Description de l'enchaînement	
Pré-condition	L'administrateur doit être authentifié et doit accéder à l'interface de la liste des produits
Post-condition	Des nouveaux produits sont ajoutés automatiquement
Scénario nominal	1- L'administrateur clique sur le bouton « Import Excel File ». 2- Le système ouvre un navigateur en affichant seulement les fichiers Excel existants 3- L'administrateur sélectionne un fichier Excel et clique sur le bouton « open » 4- Le système vérifie séparément la validité des données de chaque ligne du fichier 5- Le système enregistre les données des lignes sous forme d'entités produit 6- Le système affiche à l'utilisateur la liste des produits enregistrés automatiquement et celle des produits refusés, s'ils existent

Scénario d'erreur	**E1- Au point 4 : Les données de tous les lignes du fichier Excel sont invalides :** 1- Le système affiche les données du fichier Excel en mentionnant la cause de l'erreur de chaque ligne.

2.2.1.2 Diagramme de séquence du CU « Effectuer un bon de transfert de produits entre dépôts »

Ci-dessous nous présentons le diagramme de séquence cas d'utilisation « Effectuer un bon de transfert de produits entre dépôts »

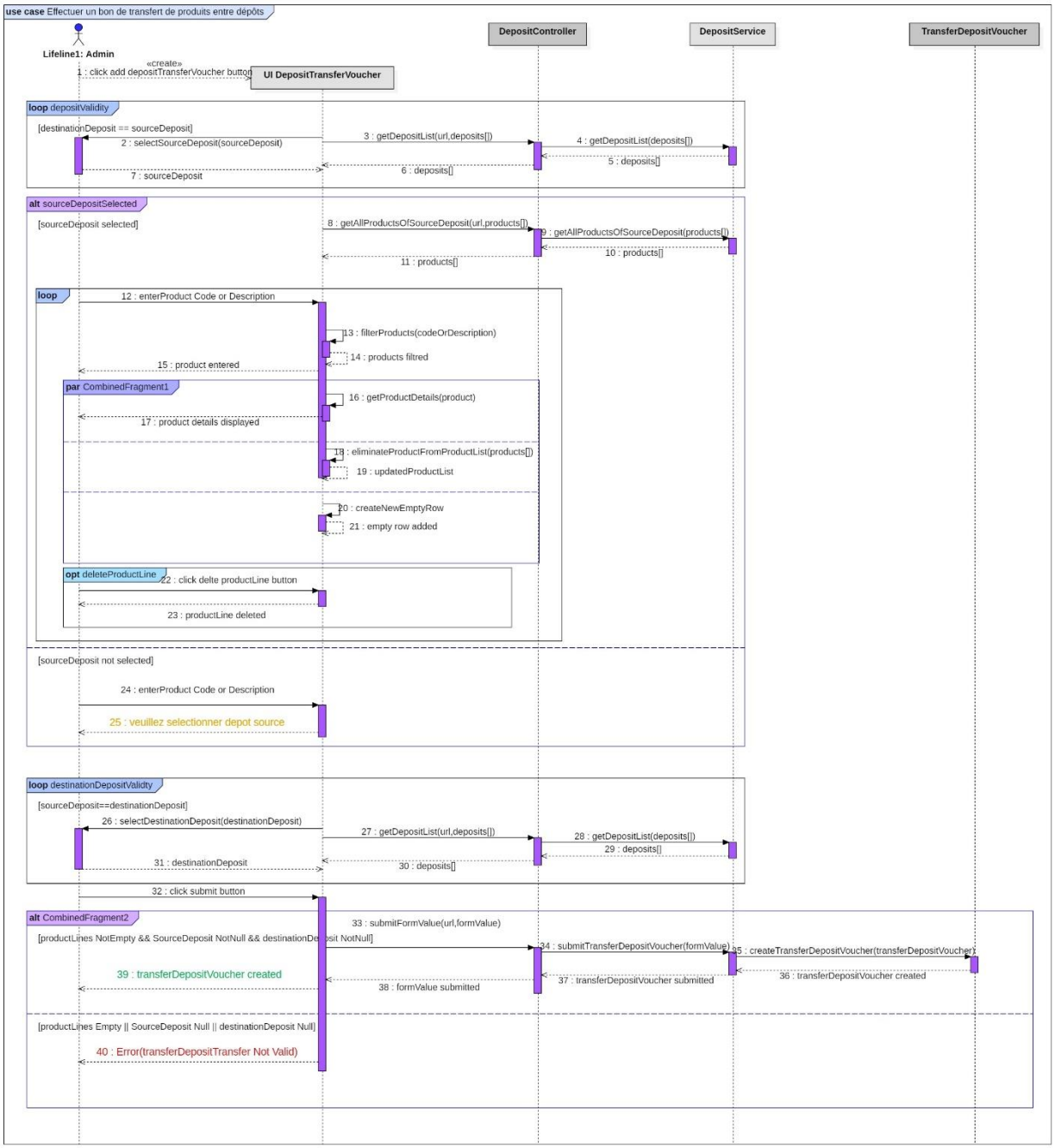


Figure 27 Diagramme de séquence du CU Effectuer un bon de transfert

2.2.1.3 Diagramme de séquence du cas d'utilisation « Automatiser l'entrée numérique des produits »

Ci-dessous nous présentons le diagramme de séquence cas d'utilisation *« Automatiser l'entrée numérique des produits »*

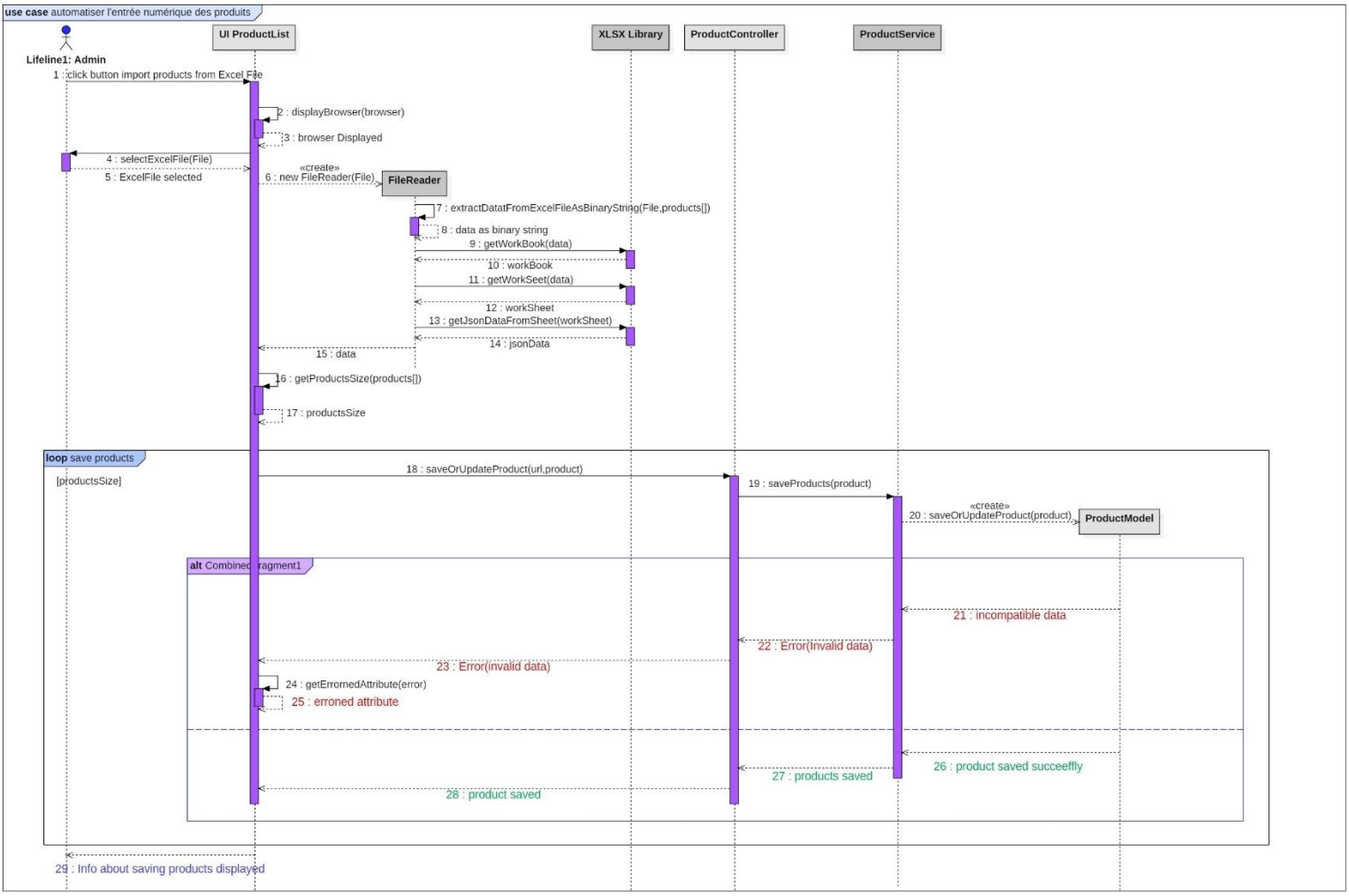


Figure 28 Diagramme de séquence du CU «Automatiser l'entrée numérique des produits»

2.2.1.4 Diagramme de séquence du cas d'utilisation « Etablir l'état d'une caisse durant une période »

Le cas d'utilisation « *Automatiser l'entrée numérique des produits* » fait l'objet du diagramme de séquence ci-dessous.

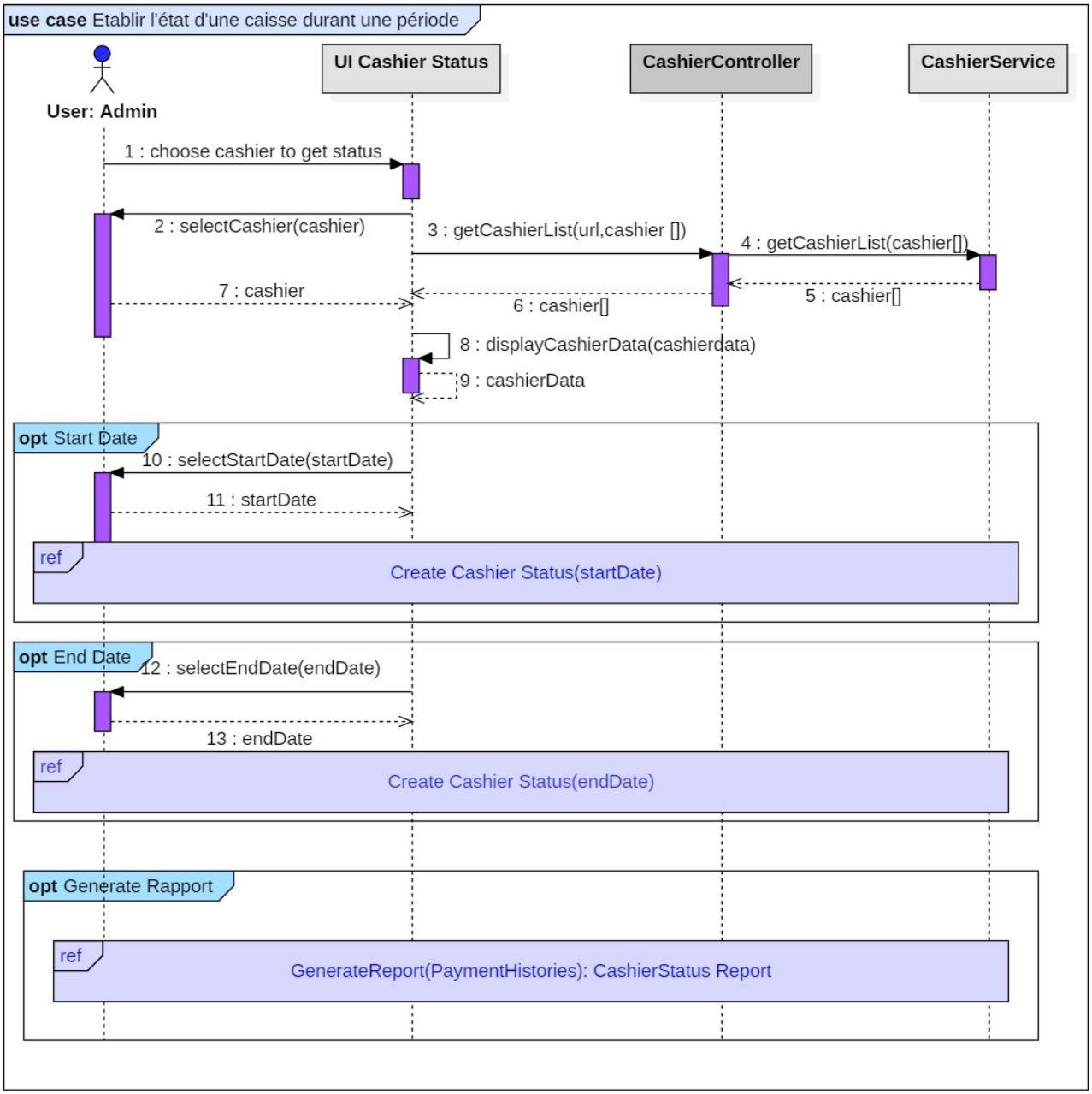


Figure 29 Diagramme de séquence du CU « Etablir l'état d'une caisse durant une période »

L'interaction Ref « Create Cashier » fait référence au diagramme de séquence ci-dessous en passant un paramètre selon le besoin.

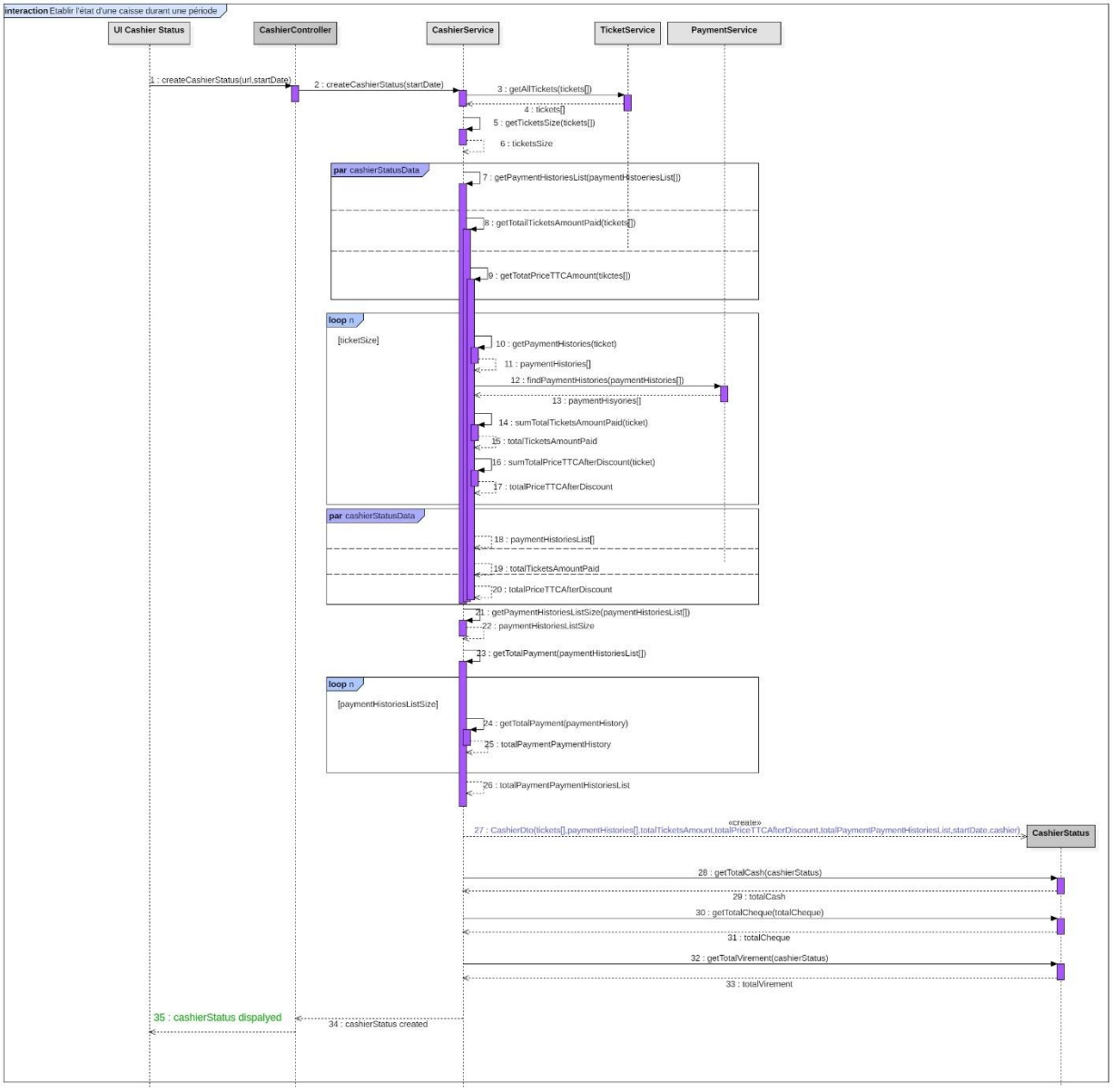


Figure 30 Diagramme de séquence de l'Interaction Créer un état de caisse

L'interaction Ref « Generate Report » fait référence au diagramme de séquence ci-dessous pour la génération d'un rapport Jasper.

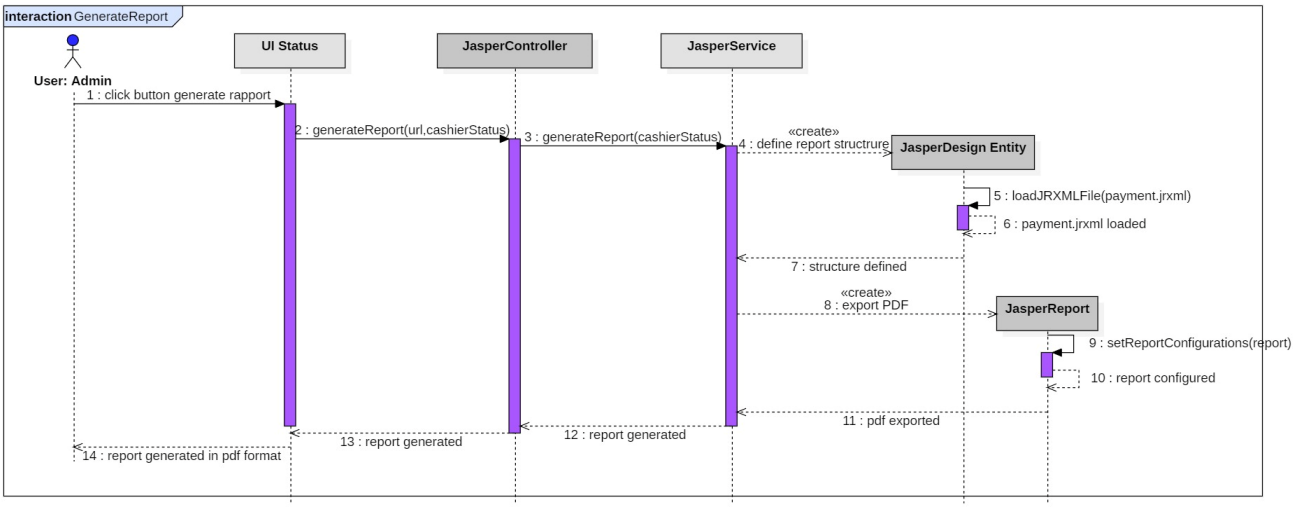


Figure 31 diagramme de génération d'un rapport Jasper

2.2.1.5 Diagramme de classes

Le diagramme de classes est un digramme modélisant les entités persistantes du système ainsi que les relations établies entre elles. La Figure 32 représente le diagramme de classes utilisé lors du Sprint 1. **L'administrateur** commercial peut importer un **fichier Excel,** effectuer un **bon de transfert** entre dépôts et passer un **bon de livraison** pour un client. Ces trois entités sont composées d'un ensemble de lignes que chacune représente un seul **produit.** En outre, l'administrateur peut établir l'état d'une caisse qui est accordée à un seul caissier durant une période précise avec la possibilité de génération d'un rapport Jasper. Celui-ci représente la liste des **règlements** permettant le payement des tickets de cet état. Un payement est accordé à un seul ticket. Un ticket peut être payé par un ou plusieurs règlements. Le bon de transfert peut être créé par un **magasiner** du dépôt source et obligatoirement traité par un autre magasinier du dépôt de destination.

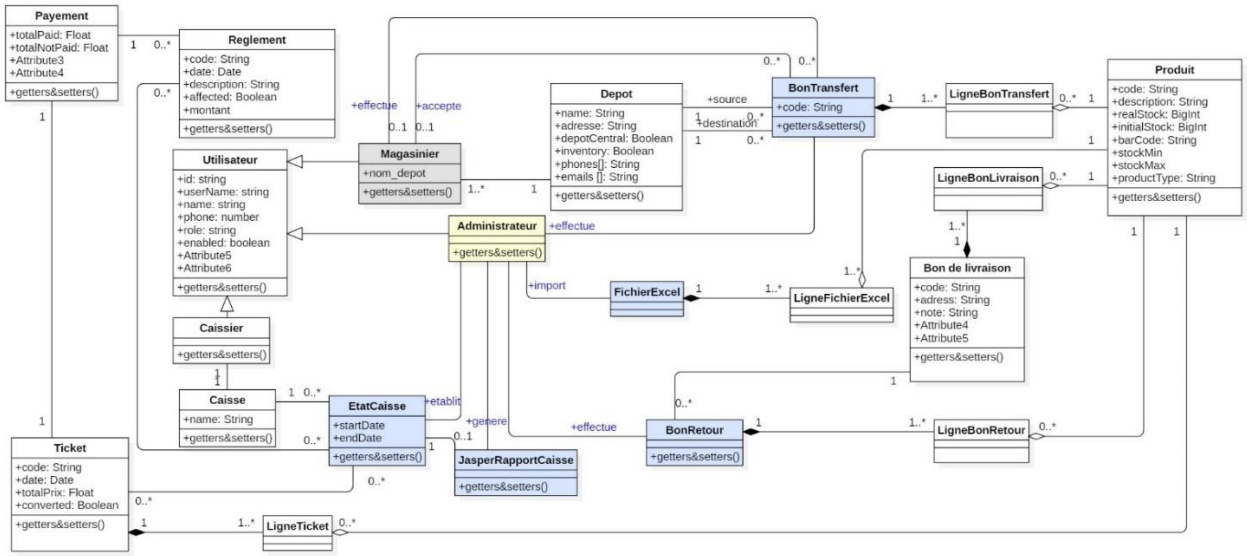


Figure 32 Diagramme de classes

2.2.2 Réalisation du Sprint1

Dans cette section, nous allons présenter les différentes interfaces graphiques de notre application pour montrer ce que nous avons réalisé sur le plan pratique

2.2.2.1 Effectuer un bon de transfert de produits entre dépôts

Au fur et à mesure de l'évolution de l'organisation commerciale une répartition du stock sur plusieurs dépôts s'impose. Impérativement, des transferts de produits entre ces dépôts existent toujours en passant par deux étapes : l'enregistrement du bon de transfert puis la réception des produits transférés par un magasinier. La figure ci-dessous illustre la création du bon de transfert qui est caractérisé par un dépôt de source, un dépôt de destination, une liste de produits à transférer, la date de création et enfin l'état du bon. Ceci peut être fait par un administrateur ou par un magasinier qui appartient au dépôt source. En accédant à l'interface de création comme étant un magasinier, le dépôt source est par défaut celui auquel appartient le magasinier sinon c'est au choix de l'administrateur.

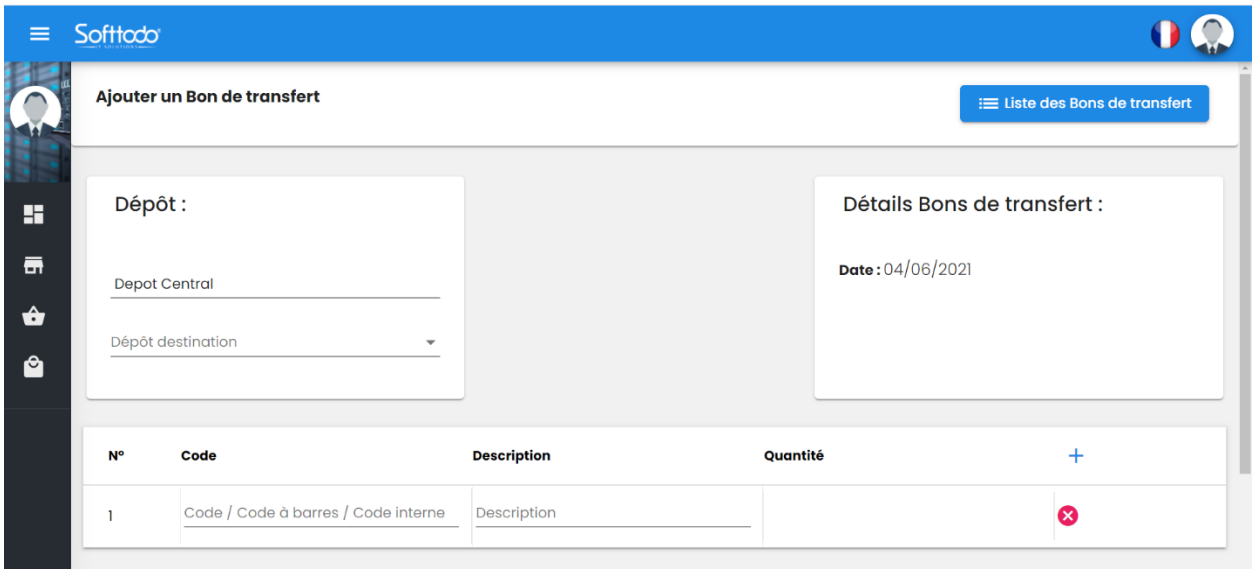


Figure 33 Ajouter un Bon de transfert

La Figure 34 représente la liste des bons de transfert ajoutés depuis un dépôt de source par des magasiniers appartenant à ce dépôt ou par l'administrateur de l'ERP. En outre, ces acteurs peuvent modifier un bon de transfert en cliquant sur l'icône « stylo » ou le supprimer en cliquant sur l'icône « poubelle ».

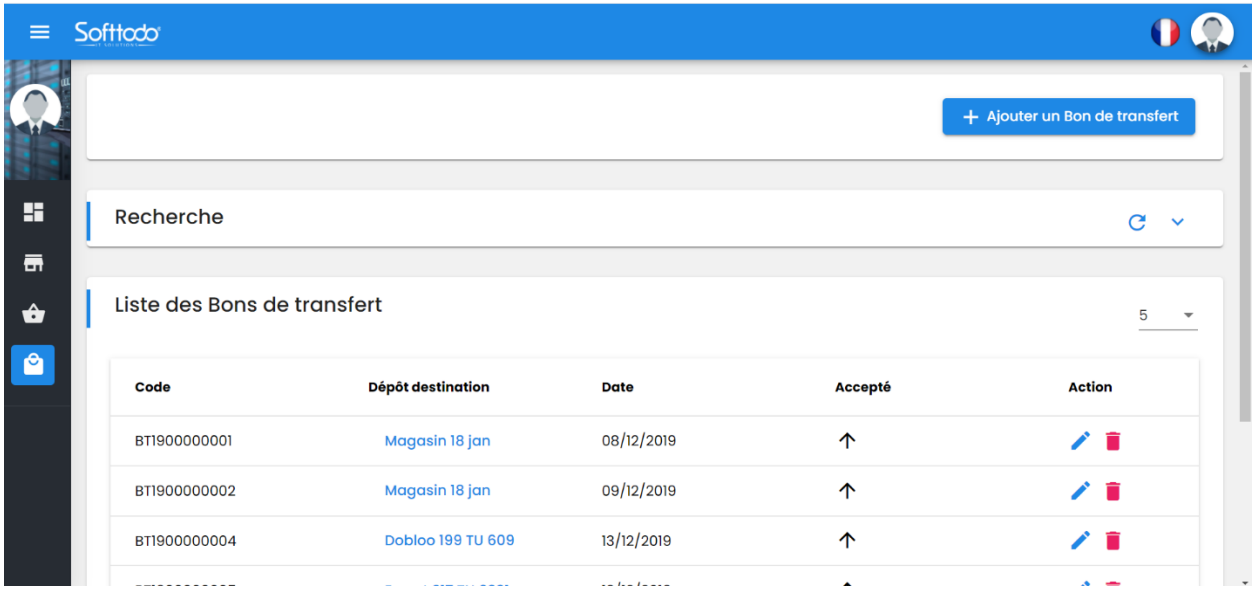


Figure 34 Liste des bons de transfert

2.2.2.2 Traiter un bon de transfert de produits par un magasinier

L'état du bon de transfert veut dire qu'à l'arrivée des produits au dépôt de destination, un magasinier vérifie la quantité réelle reçue par rapport à celle déclarée dans le bon de transfert

puis retourne au ERP pour accepter ce bon si tout va bien ou le refuser si c'est le cas contraire. La Figure ci-dessous présente la liste des bons de transfert qu'un dépôt a reçu.

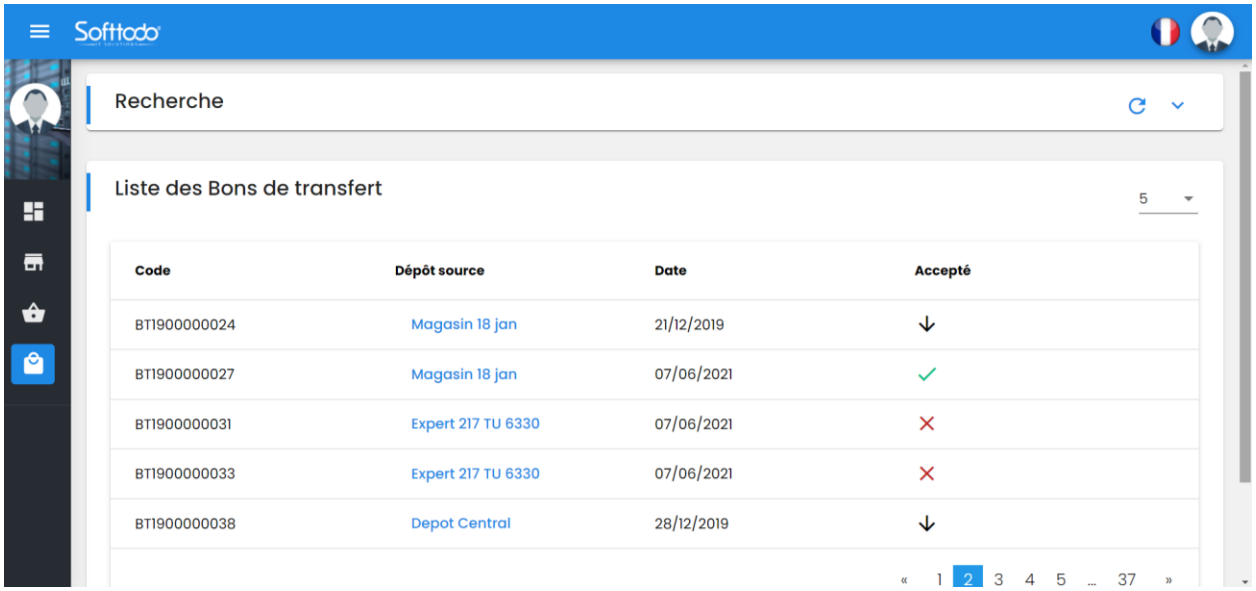


Figure 35 Liste des bons de transfert qu'un dépôt a reçu

Comme le montre la Figure ci-dessous, en cliquant sur l'un de ces bons de transfert une interface est affichée présentant ses détails avec un bouton d'acceptation et un autre de refus.

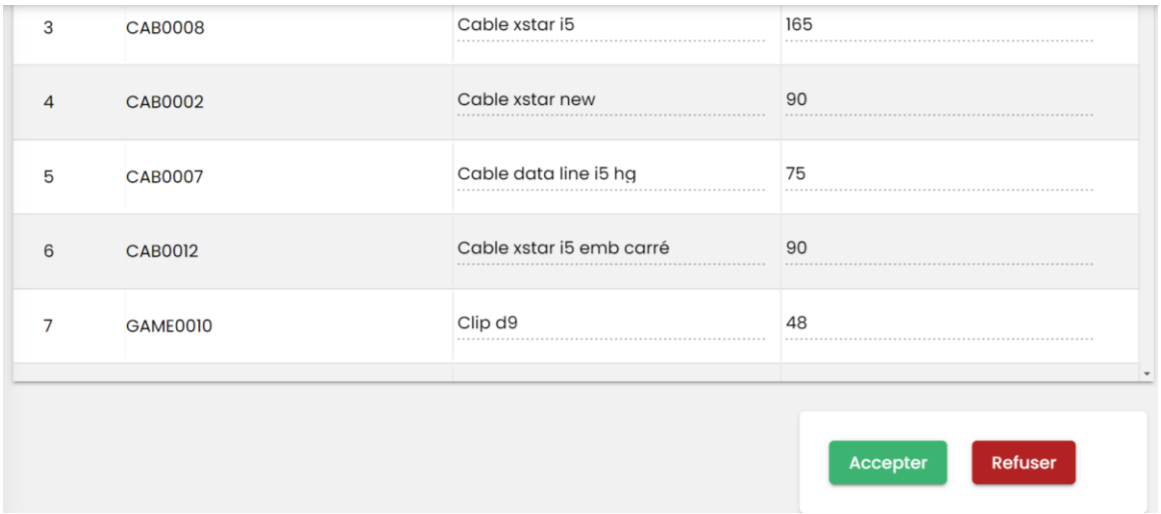


Figure 36 Interface de traitement d'un bon de transfert

Après la vérification du stock, le magasinier clique sur l'un des deux boutons pour accepter ou refuser le bon de transfert et enfin l'état de ce bon est définit avec succès.

2.2.2.3 Automatiser l'entrée numérique des produits

Parmi les fonctionnalités de gestion de stock qui assurent une migration facile de la gestion manuelle vers l'utilisation de l'ERP, nous citons l'automatisation de l'entrée numérique d'une liste de produits. Ceci est réalisé à travers la sélection d'un fichier Excel que chaque ligne forme un produit. Comme le montre la Figure 37, en cliquant sur le bouton « + Import From Excel » un navigateur s'affiche donnant la main à l'administrateur pour sélectionner le fichier « .csv ».

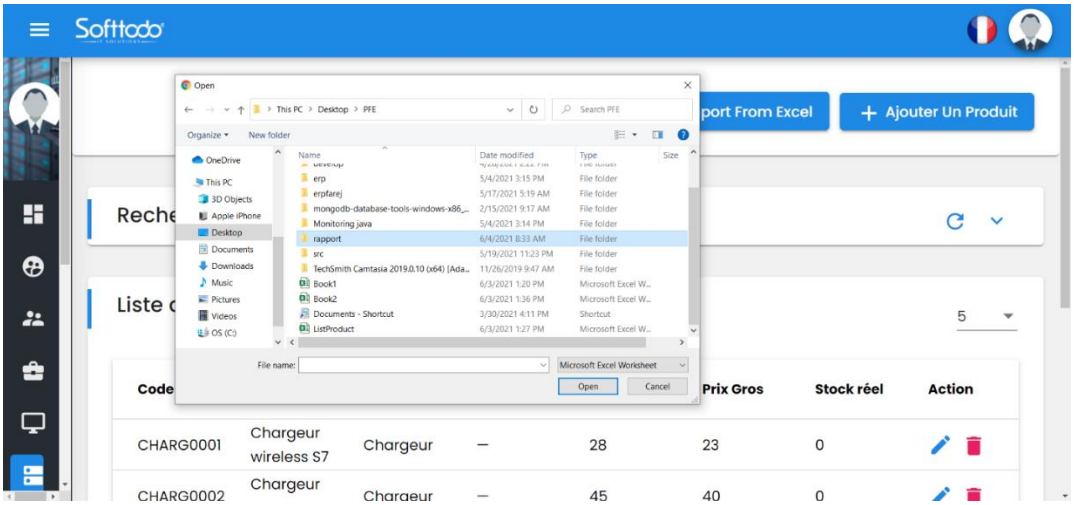


Figure 37 Automatiser l'entrée numérique des produits

En cliquant sur le bouton « Open », le système commence à importer les produits de tout le fichier Excel. Après le téléchargement, le système affiche à l'administrateur un compte rendu de l'opération d'importation contenant la liste des produits refusés et celle des produits acceptés et enregistrés avec succès comme le montre les Figures 38 et 39.

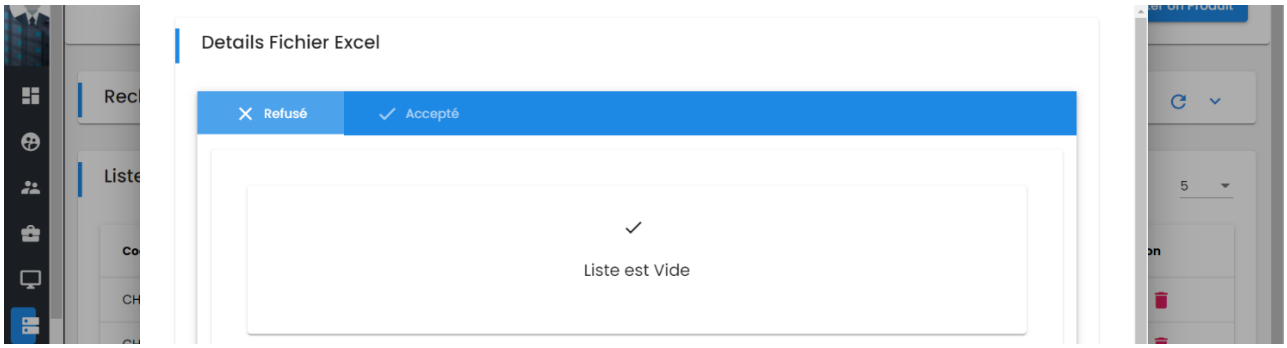


Figure 38 Liste des produits refusés

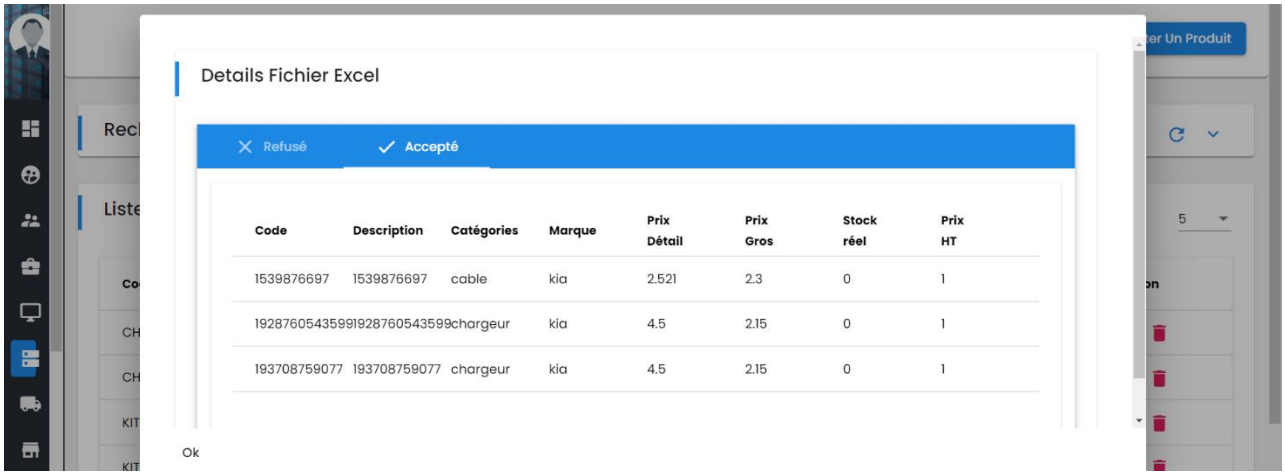


Figure 39 Liste des produits acceptés

2.2.2.4 Retourner un produit

La possibilité de retour des marchandises fait partie des moyens de l'ERP SAPTUN pour fidéliser ses clients. En effet, une interface est dédiée à la création d'un bon de retour à partir d'un bon de livraison que l'administrateur choisit. A la sélection d'un bon de livraison, tous les détails du bon sont affichés. Alors le système donne la main à l'administrateur d'entrer un ou plusieurs produits de l'ensemble des produits livrés.

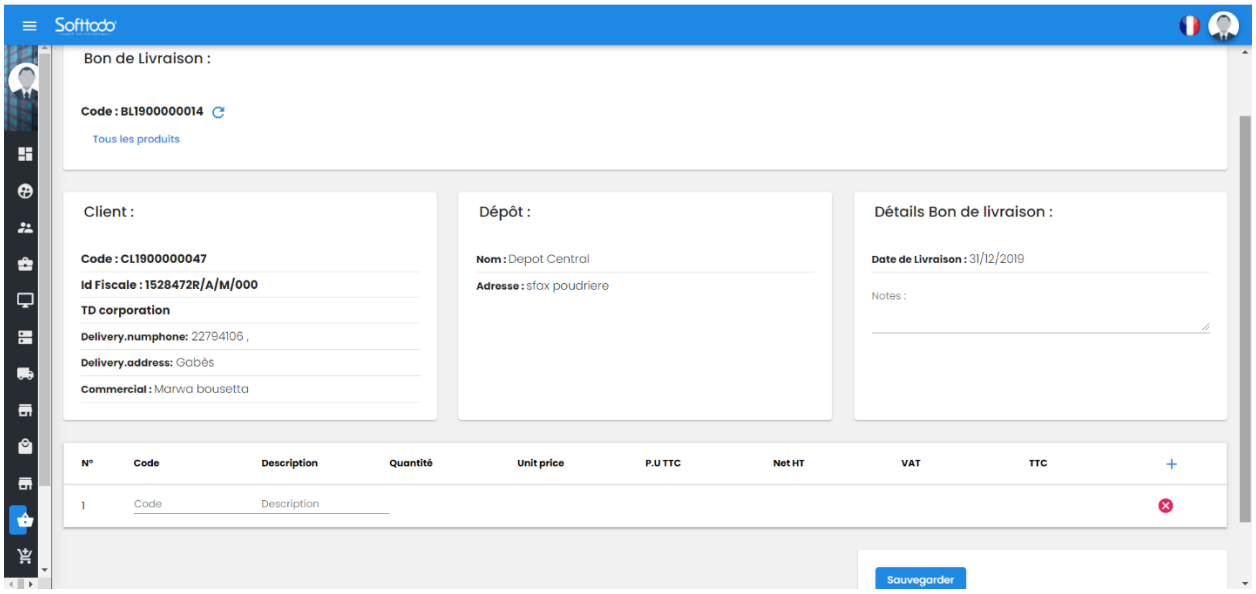


Figure 40 Interface de création d'un bon de retour

Le système offre une optimisation pour la création d'un bon de retour. En effet, il fournit un système d'auto-complétion à la saisie du code de produit et il affiche la quantité livrée en identifiant celle à retourner. En cliquant sur le bouton « sauvegarder » le système enregistre le

bon de retour et redirige l'administrateur à la page de la liste des bons de retour en lui affichant un message de succès.

2.2.2.5 Etablir l'état d'une caisse durant une période

Une caisse doit être tenue et contrôlée avec rigueur. De ce fait, pour vérifier les opérations passées, l'administrateur effectue quotidiennement un inventaire de caisse d'une manière efficace grâce à un collecte facile des tickets et des règlements. A travers l'interface présenté ci-dessous l'administrateur commence par choisir la caisse cible et le système affiche les détails de celle-ci tel que le dépôt auquel est associée et son unique caissier.

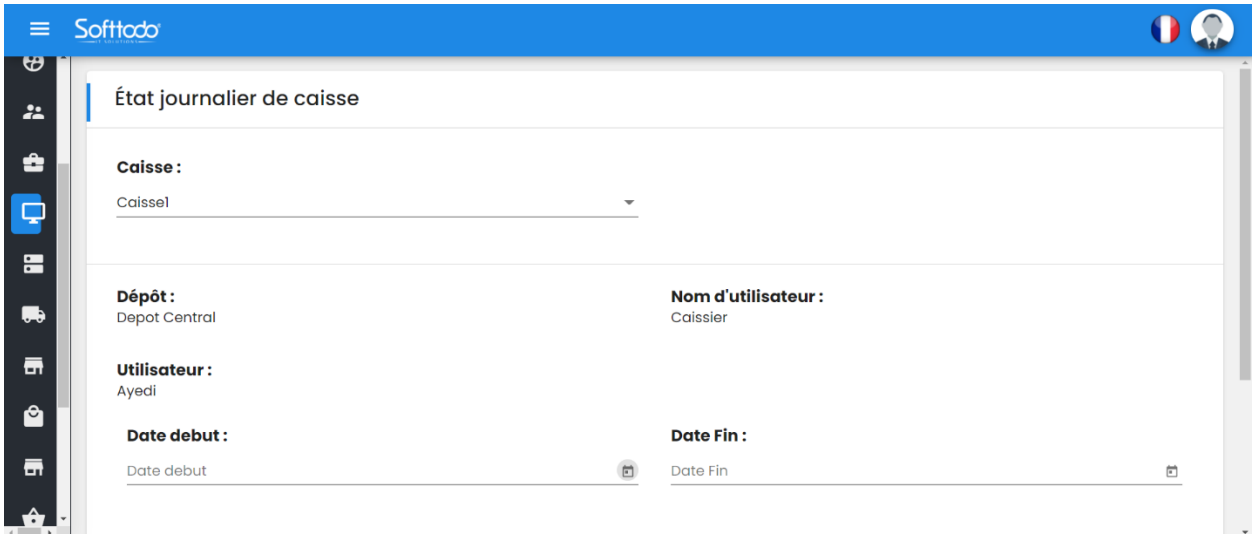


Figure 41 Interface de la creation de l'etat de caisse

Ensuite, l'administrateur fait entrer la période de l'inventaire. Alors, le système affiche la liste des règlements effectués en précisant la somme totale payée et la contribution de chaque type de payement dans cette somme : en espèce, avec chèque ou avec virement bancaire comme le montre la figure ci-dessous.

Figure 42 La liste des réglements

De plus, une deuxième liste qui présente l'ensemble des tickets passés à la caisse s'affiche. En outre, le montant total des tickets et celui payé par les clients qui peuvent payer par facilité comme le montre la Figure 43.

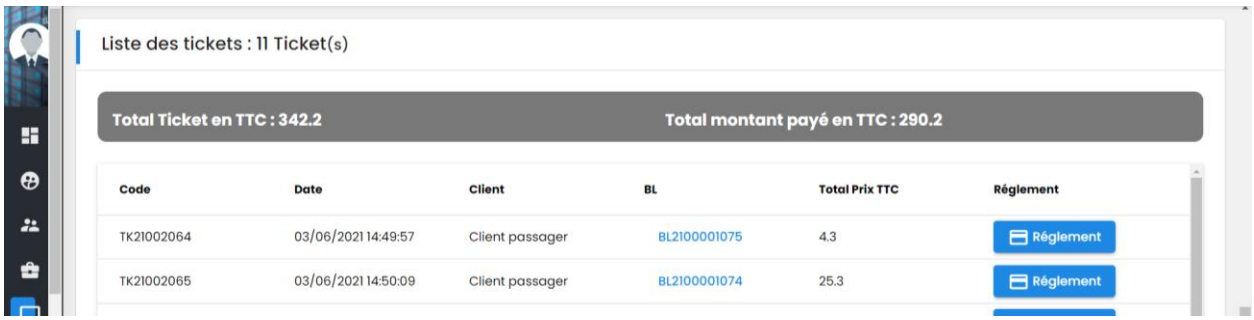


Figure 43 La liste des tickets

2.2.2.6 Générer un rapport pour les règlements de caisse

La Figure 44 signale la possibilité de générer un rapport selon une structure bien définie. En cliquant sur le bouton « générer rapport » une fenêtre s'affiche présentant le rapport et donnant la possibilité de le télécharger en PDF ou de l'imprimer pour avoir une version sur papier.

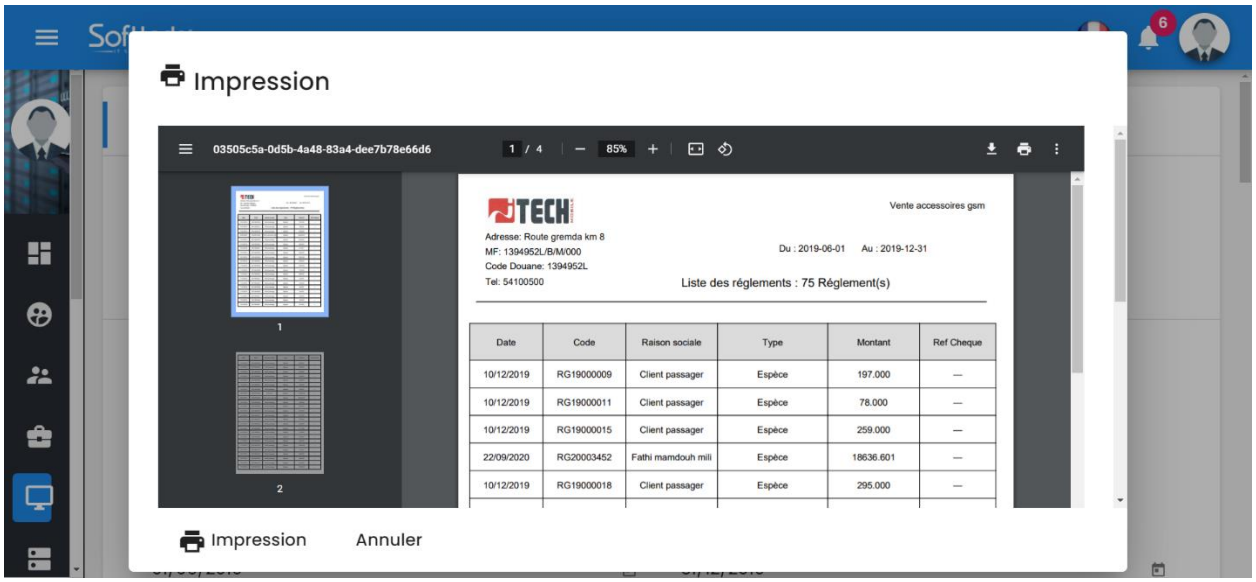


Figure 44 Le rapport de l'état de caisse

2.3 Conception et réalisation du Sprint 2

Dans le Sprint 2, nous avons suivi la même démarche de travail faite dans le Sprint1.

2.3.1 Conception du Sprint 2

Dans cette partie, nous allons aborder la conception des fonctionnalités du Sprint 2 à travers la présentation du diagramme de cas d'utilisation, des diagrammes de séquences relatifs à ces cas d'utilisations et du diagramme de classe des entités persistantes.

2.3.1.1 Diagramme de cas d'utilisation

Les principales tâches effectuées par l'administrateur durant le Sprint 2 sont les cas d'utilisation de notre système que nous présentons dans ce diagramme. Puis nous en faisons la description textuelle.

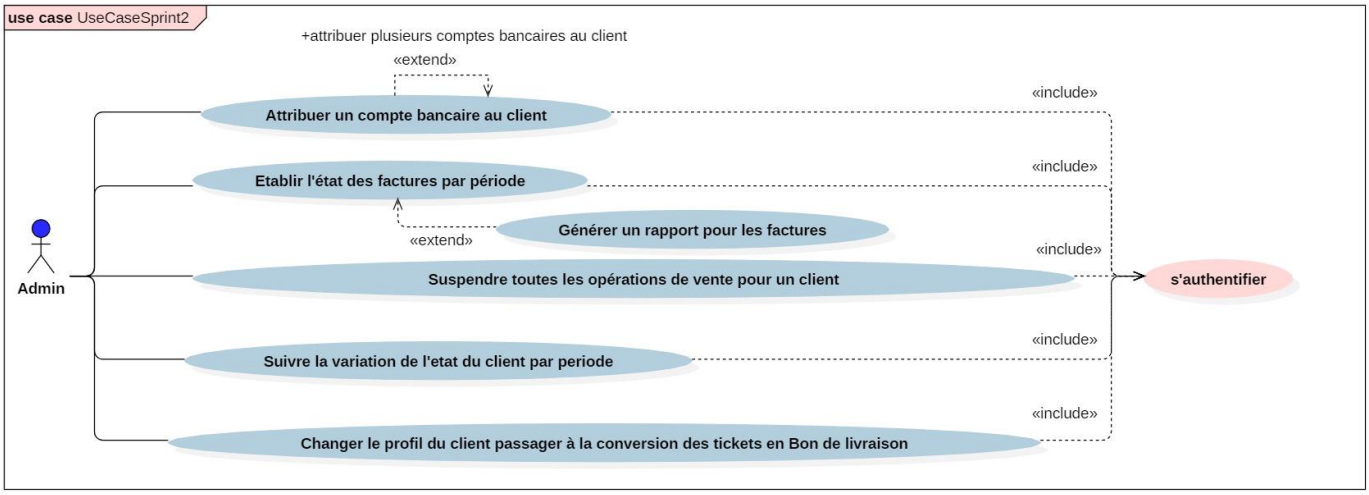


Figure 45 Diagramme de cas d'utilisation

Tableau 5 : Description textuelle du CU « Attribuer un compte bancaire au client »

Sommaire	
Acteur	Administrateur
Résumé	L'administrateur se permet d'attribuer un ou plusieurs comptes bancaires au client
Description de l'enchaînement	

Pré-condition	L'administrateur doit être authentifié et doit accéder à l'interface de la liste des clients
Post-condition	Un client admet un ou plusieurs comptes bancaires
Scénario nominal	1- L'administrateur clique sur l'icône « Stylo » pour modifier un client. 2- Le système redirige l'administrateur vers l'interface de modification de ce client 3- L'administrateur choisit une banque 4- L'administrateur fait entrer le RIB 5- Le système vérifie les données entrées 6- L'administrateur clique sur le bouton « + Compte bancaire » 7- Le système ajoute une autre ligne composée d'un champ de banque et un autre de RIB 8- Les étapes 3,4,5 et 6 se répètent 9- Les étapes 7 et 8 peuvent se répéter selon le nombre de comptes bancaires que l'administrateur souhaite les attribuer à ce client 10- L'administrateur clique sur le bouton « modifier » 11- Le système vérifie les données modifies 12- Le système enregistre les modifications apportées au client
Scénario alternatif	A1- Au point 5 : Duplication du compte bancaire 1- Le système affiche un message d'attention 2- Le système réinitialise la deuxième ligne à vide 3- Le scenario nominal reprend au point 3 A2- Au point 11 : Les données sont incomplètes 1- Le système affiche un message d'erreur

	2- Le scenario nominal reprend au point 9

Tableau 6 : Description textuelle du CU « Suspendre toutes les opérations de vente pour un client »

Sommaire	
Acteur	Administrateur
Résumé	L'administrateur peut suspendre toutes les opérations de vente pour un client tel que la création d'une commande, d'un ticket ou d'un bon de livraison.
Description de l'enchaînement	
Pré-condition	L'administrateur doit être authentifié et doit accéder à l'interface des paramètres généraux
Post-condition	Un client est suspendu de toute opération de vente
Scénario nominal	1- L'administrateur clique sur le bouton « Modifier les paramètres généraux » 2- Le système redirige l'administrateur vers l'interface de modification des paramètres généraux 3- L'administrateur clique sur le bouton de l'option « bloquer client » 4- L'administrateur clique sur le bouton de « sauvegarde » 5- Le système enregistre les modifications apportées sur les paramètres généraux

	6- L'administrateur se dirige vers l'interface de modification d'un client 7- L'administrateur active le bouton radio « bloquer » 8- L'administrateur clique sur le bouton « modifier » 12- Le système enregistre les modifications apportées au client
Scénario d'erreur	A1- Au point 6 : Le client choisit est le client passager 1- L'administrateur ne trouve pas le bouton radio d'activation du blocage

2.3.1.2 Diagramme de séquence du cas d'utilisation « Changer le profil du client passager à la conversion des tickets en Bon de livraison »

Ci-dessous nous présentons le diagramme de séquence cas d'utilisation « Changer le profil du client passager à la conversion des tickets en Bon de livraison »

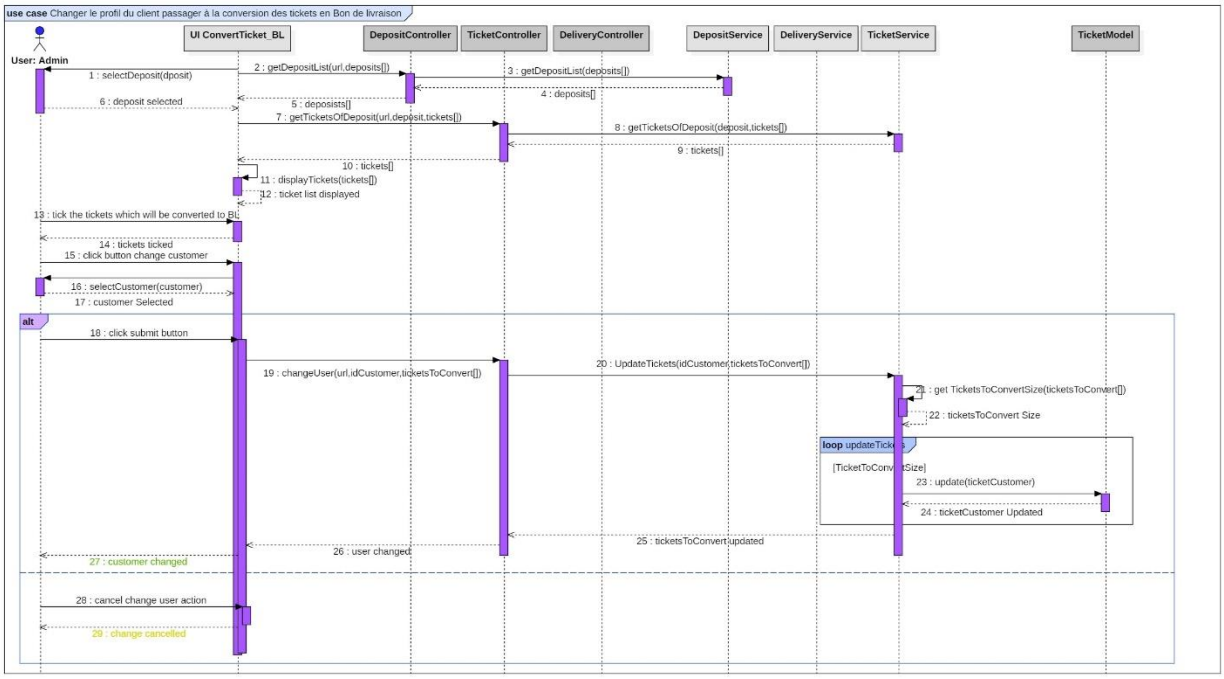


Figure 46 Diagramme de séquence de CU « Conversion des tickets en Bon de livraison »

3.3.1.3 Diagramme de séquence du cas d'utilisation « Attribuer un compte bancaire au client »

Ci-dessous nous présentons le diagramme de séquence cas d'utilisation « Attribuer un compte bancaire au client »

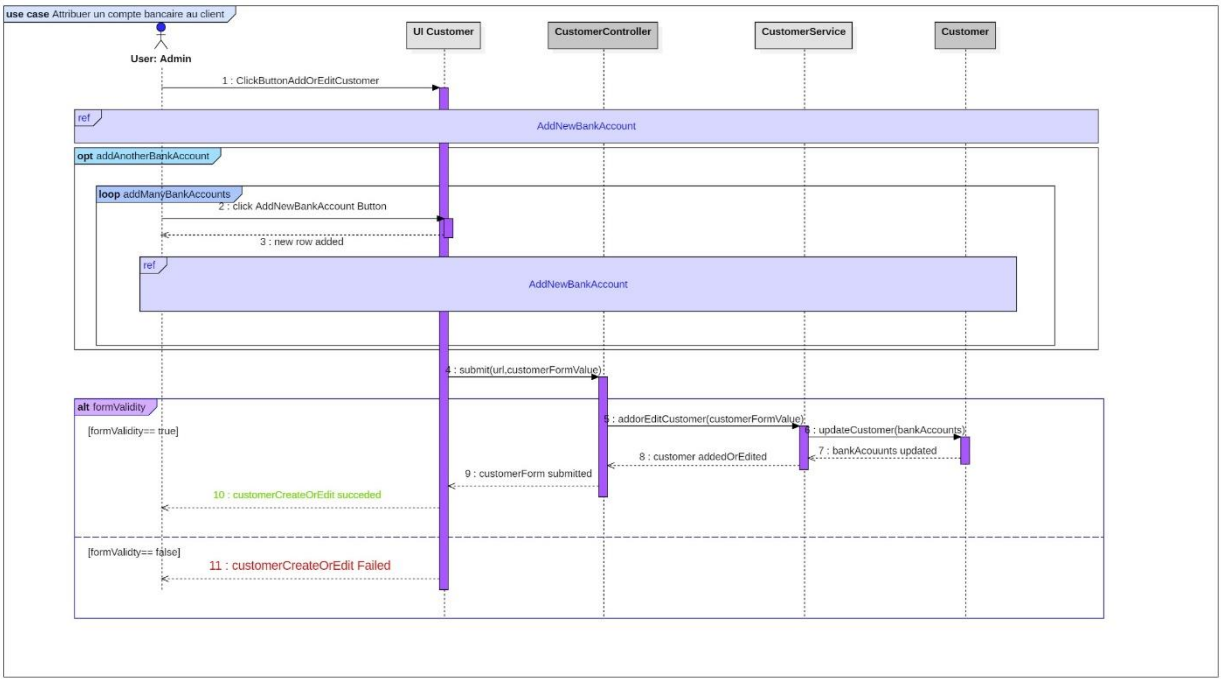


Figure 47 Diagramme de séquence de CU « Attribuer un compte bancaire au client »

Ci diagramme ci-dessous représente l'interaction Ref « Add New Bank Account »

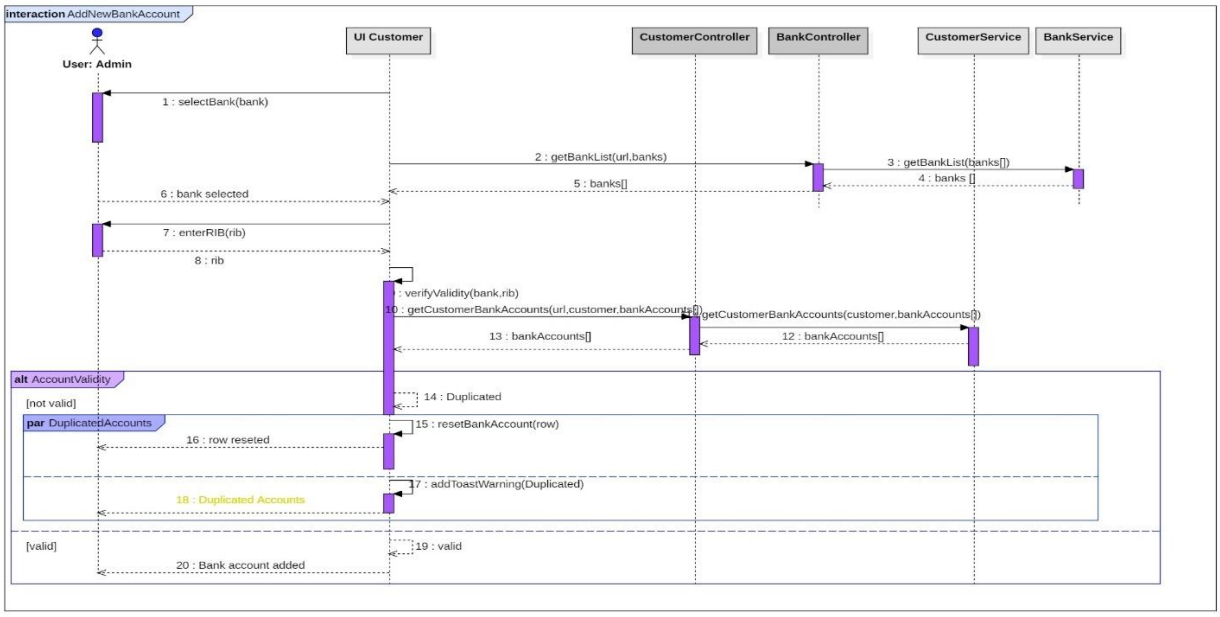


Figure 48 Diagramme de séquence de CU « Ajouter un nouveau compte bancaire au client »

3.3.1.4 Diagramme de séquence du cas d'utilisation « Suspendre toutes les opérations de vente pour un client »

Ci-dessous nous présentons le diagramme de séquence cas d'utilisation « Suspendre toutes les opérations de vente pour un client »

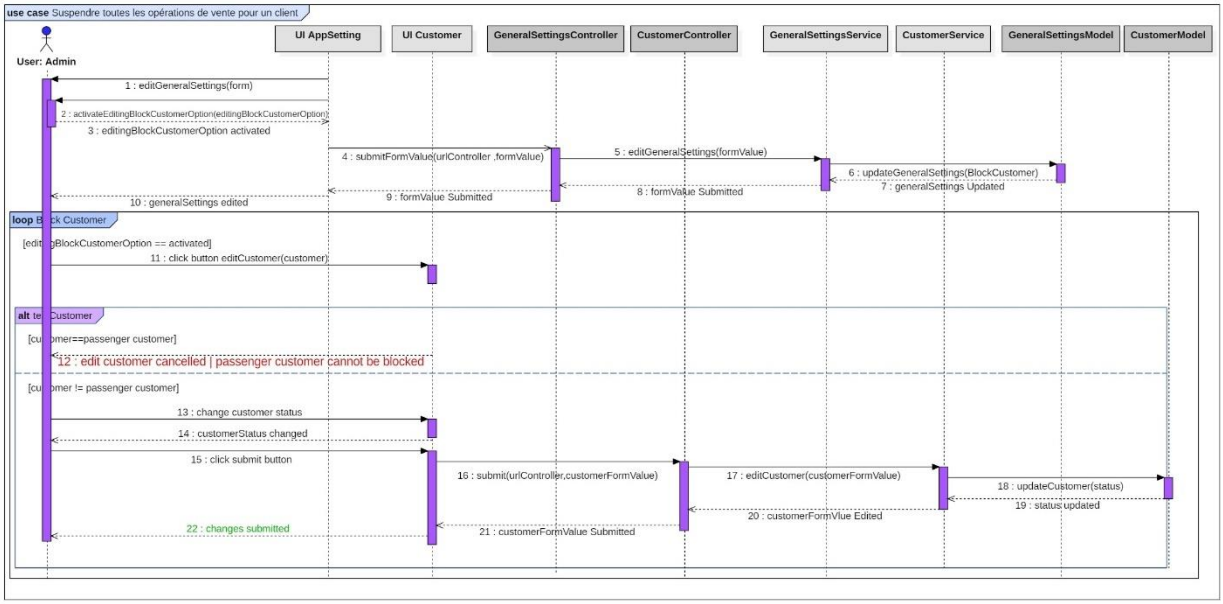


Figure 49 Diagramme de séquence de CU Suspendre les opérations de vente

3.3.1.5 Diagramme de classes

La Figure 50 représente le diagramme de classes utilisées lors du Sprint 2. **L'administrateur** commercial se permet de suivre **l'état** d'un **client**. De plus, il a la possibilité d'établir **l'état des factures** le long d'une période précise et en admettant l'offre de génération d'un **rapport Jasper** sur cet état. En outre, l'administrateur peut attribuer un ou plusieurs **comptes bancaires** à un client. Chaque compte est associé à une seule **banque** et caractérisé par un RIB. Ce diagramme nous montre la transformation d'un ou plusieurs **tickets** en un **bon de livraison** avec la nécessité de changer l'identité du client au cas où ces tickets sont associés à un client passager.

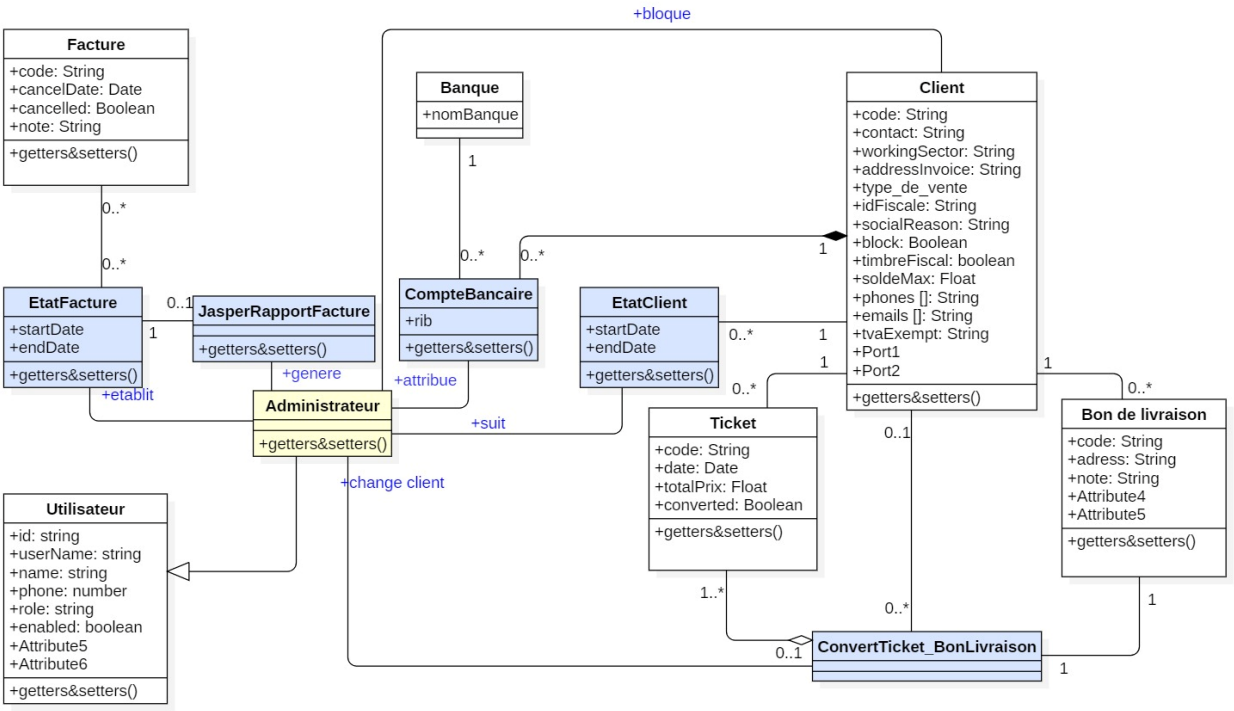


Figure 50 Diagramme de classes du Sprint 2

2.3.2 Réalisation du Sprint 2

Dans cette partie, nous présentons les interfaces de réalisation des fonctionnalités élaborées lors du sprint 2.

2.3.2.1 Changer le profil du client passager à la conversion des tickets en Bon de livraison

Un passager est alloué d'acheter des produits à la caisse. Ceci est assuré par le système en passant un ticket au profit du client passager représentant tous les passagers. Cette figure nous montre la liste des tickets associés au client passager.

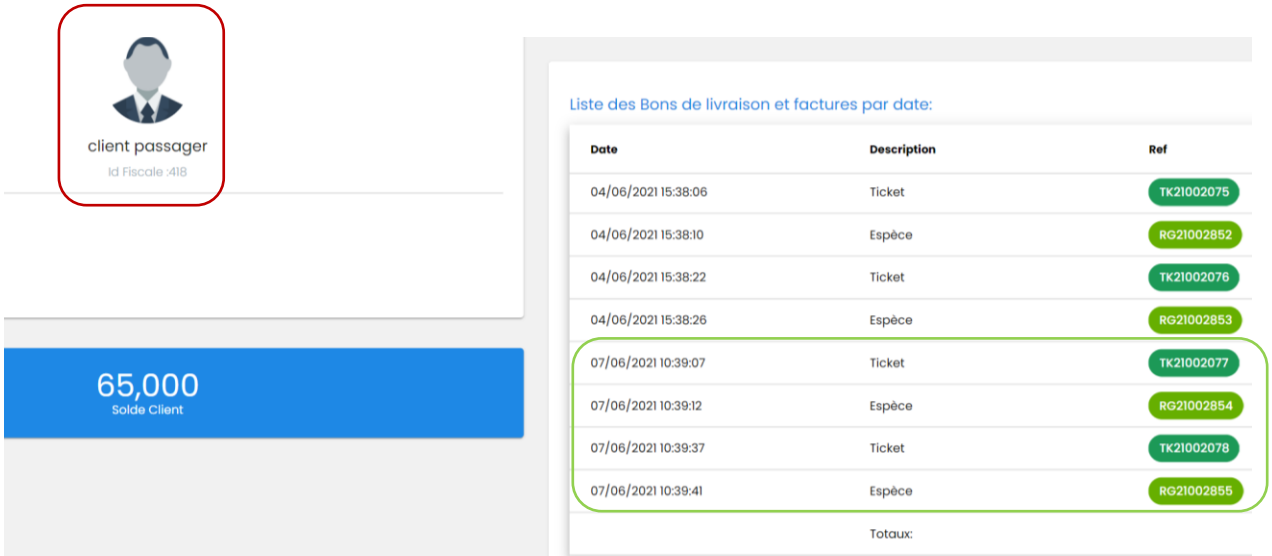


Figure 51 L'interface des tickets effectues pour le client passager

Mais la conversion de ces tickets en bon de livraison ne peut pas se faire pour un client passager faute des coordonnées de la personne cible (téléphone, adresse, etc.). Comme le montre la figure ci-dessous, l'administrateur peut également éviter cette mauvaise traçabilité des données en changeant le client passager par un autre client bien défini.

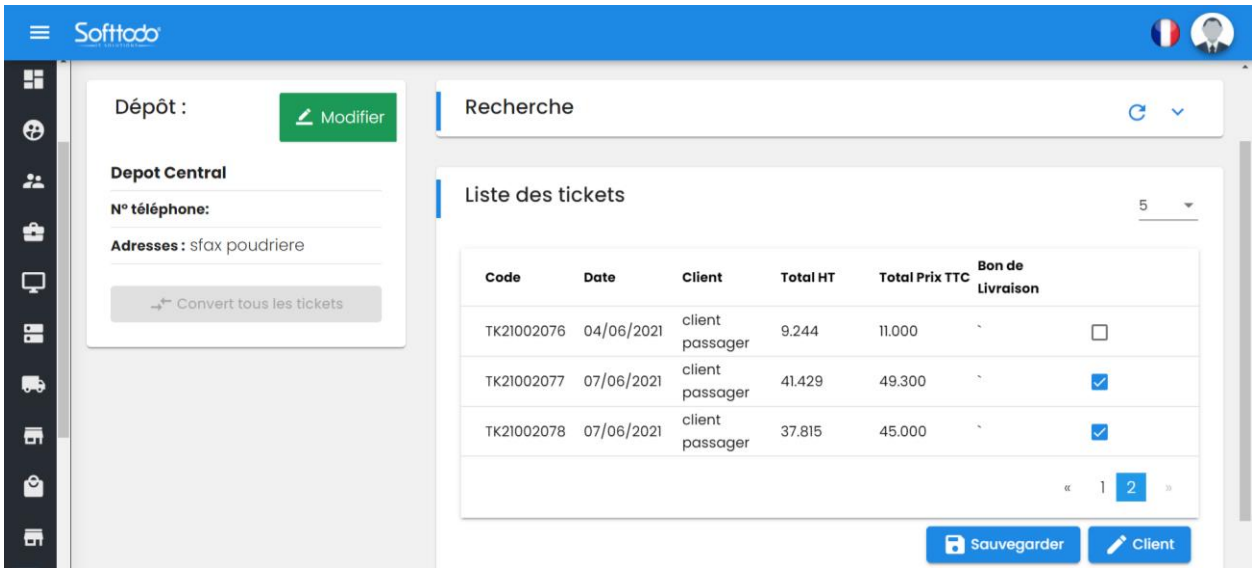


Figure 52 L'interface de conversion des tickets en bon de livraison

Après la sélection des tickets, l'administrateur clique sur le bouton « Modifier client » et une fenêtre s'affiche lui donnant la main pour choisir un client autre que le client passager.

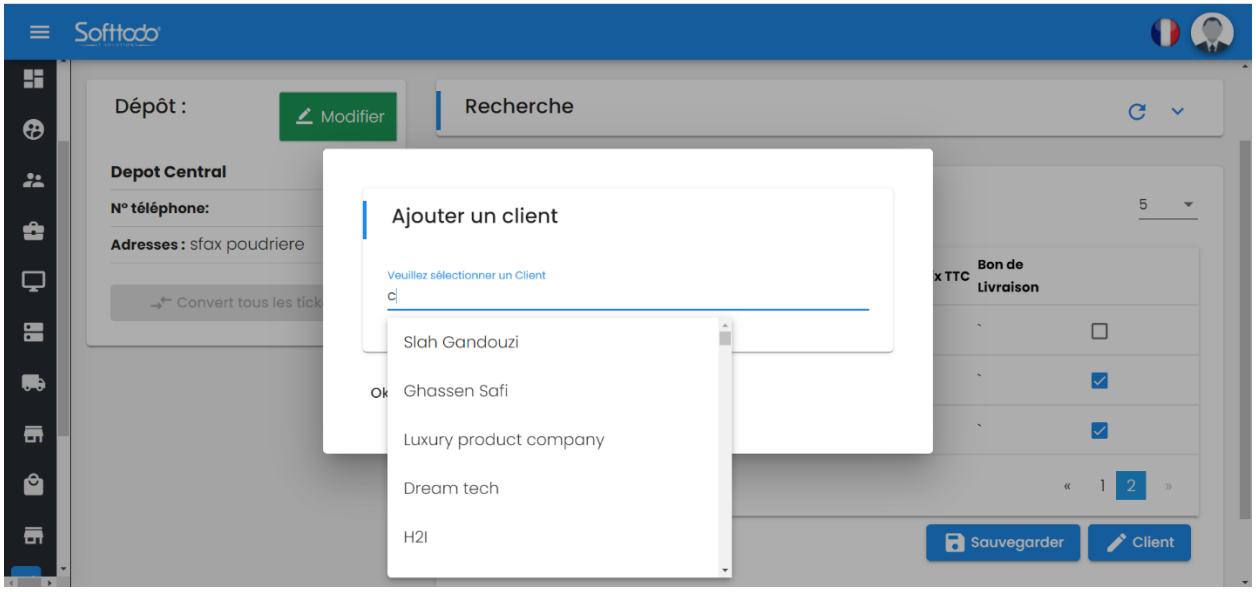


Figure 53 L'interface de choix du client

La conversion est effectuée en cliquant sur le bouton « sauvegarder ». D'où un message de succès est affiché. En accédant à l'interface de la fiche du client, l'administrateur peut remarquer la suppression des tickets. Par ailleurs, il remarque un transfert de leurs règlements vers le compte du client choisi et un nouveau bon de livraison est ainsi créé et associé à ce client dont le téléphone et l'adresse sont bien définis.

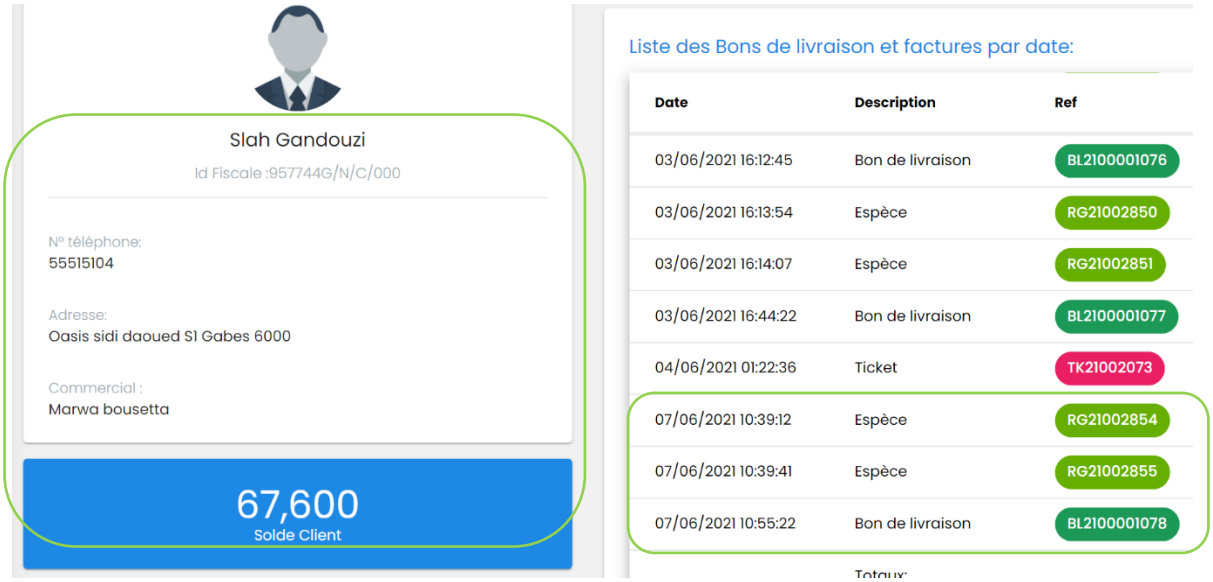


Figure 54 L'interface de fiche client

2.3.2.2 Suivre la variation de l'état du client par période

L'ERP SAPTUN expose à l'administrateur un graphe clair pour consulter la variation de l'état du client durant une période précise. Nous pouvons définir la variation de l'état par l'ensemble

des opérations d'encaissements et de décaissements réalisées le long de cette période ce qui permet d'avoir une visibilité accrue sur la santé financière du client et de découvrir la nature de son compte (créditeur ou débiteur).

La figure 55 montre ou l'administrateur choisit un intervalle de temps ainsi le système affiche la courbe de variation du solde et celle d'effectuation des achats avec la possibilité de restreindre la visualisation sur l'une des courbes en cliquant sur « purchase » ou sur « solde ».

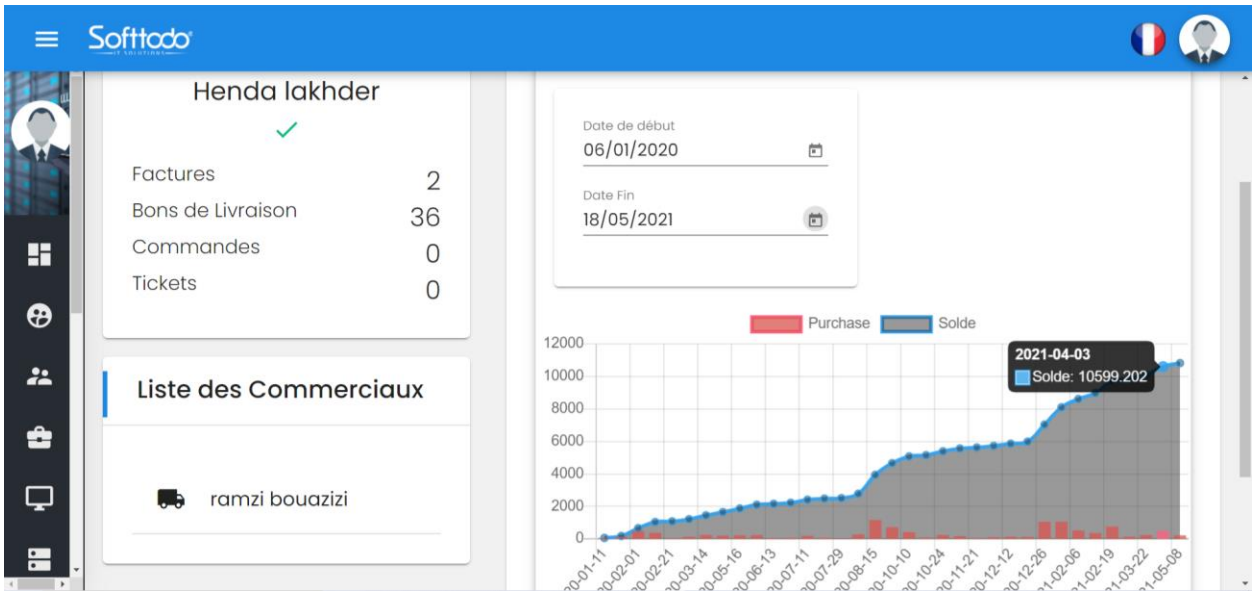


Figure 55 Interface de suivi de la variation de l'état du client

2.3.2.3 Attribuer des comptes bancaires au client

Une utilisation fréquente des payements par chèque ou par virement a donné la nécessité d'affecter des comptes bancaires au client dès l'enregistrement ou même après. En accédant aux interfaces de création ou de modification d'un client, un champ de sélection de la banque et celui d'entrée du RIB existent. Ces deux champs permettent d'identifier un compte bancaire comme le montre la figure ci-dessous.

Figure 56 « Attribuer un compte bancaire au client »

La Figure 57 montre que le client a la possibilité d'avoir un ou plusieurs comptes bancaires. Un bouton « + compte bancaire » est dédié pour ce faire. De plus, l'administrateur peut cliquer sur l'icône croix pour supprimer un compte.

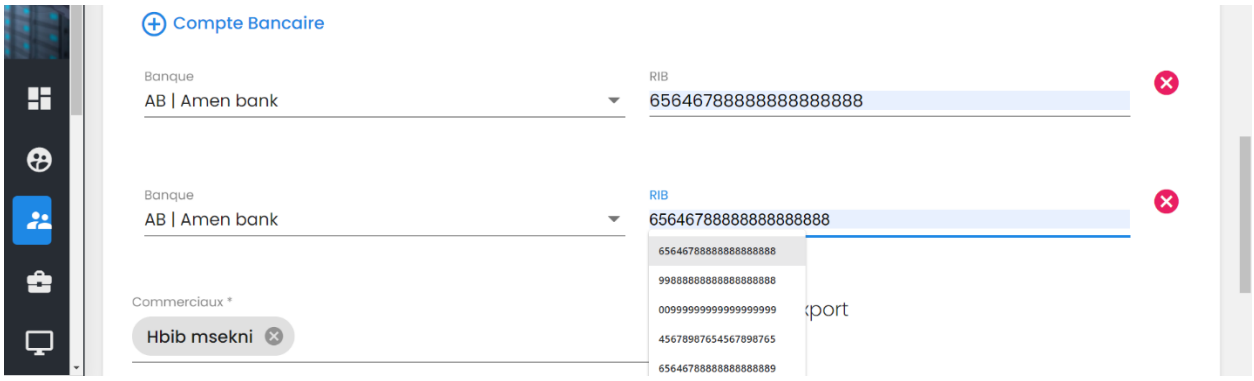


Figure 57 La possibilité d'avoir un ou plusieurs comptes bancaires

L'administrateur de l'ERP peut consulter les comptes bancaires d'un client en accédant à l'interface de ses détails.

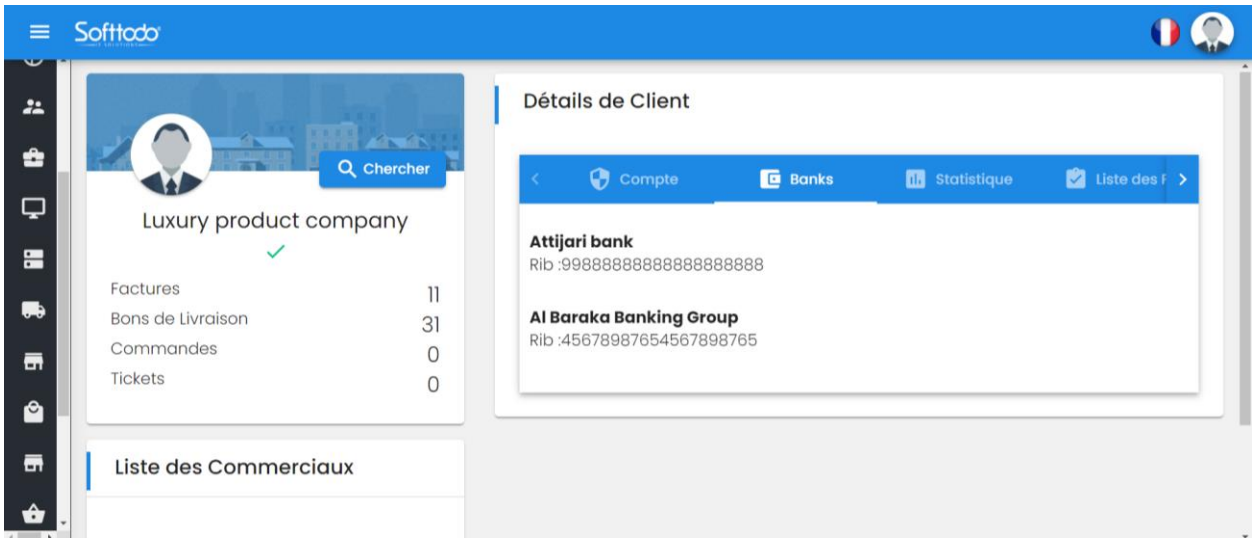


Figure 58 Interface des comptes bancaires que possède un client

2.3.2.4 Suspendre toutes les opérations de vente pour un client

Un client mauvais payeur ou en mauvaise santé financière doit être bloqué pour suspendre toutes les opérations d'ajout de commande, de ticket ou de bon de livraison pour lui jusqu'à résoudre son problème et le débloquer. Le client passager ne peut pas être bloqué car c'est un compte qui représente tous les passagers. Pour bloquer un client, l'administrateur doit accéder aux paramètres généraux de l'application pour activer cette option du blocage des clients ce qui résulte l'apparition d'un bouton radio dans l'interface de modification d'un client pour le bloquer ou le débloquer. À la sauvegarde des changements, le système redirige l'administrateur vers la liste des clients et donc un changement d'état du client est effectué avec succès ce qui est illustré par la figure ci-dessous.

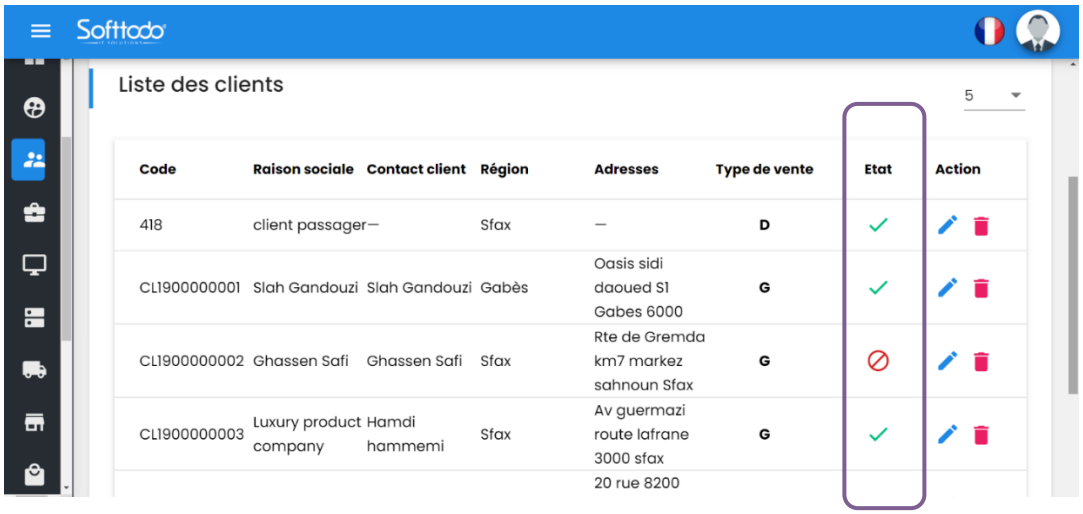


Figure 59 Interface de la liste des clients

2.3.2.5 Etablir l'état des factures par période

Faire un rapport sur l'état des factures de vente donne une bonne tenue de la comptabilité. Mais avoir une liste énorme de factures peut gêner le bon fonctionnement de cette tâche. A travers la Figure ci-dessous l'administrateur peut obtenir la liste des factures effectuées le long d'une période.

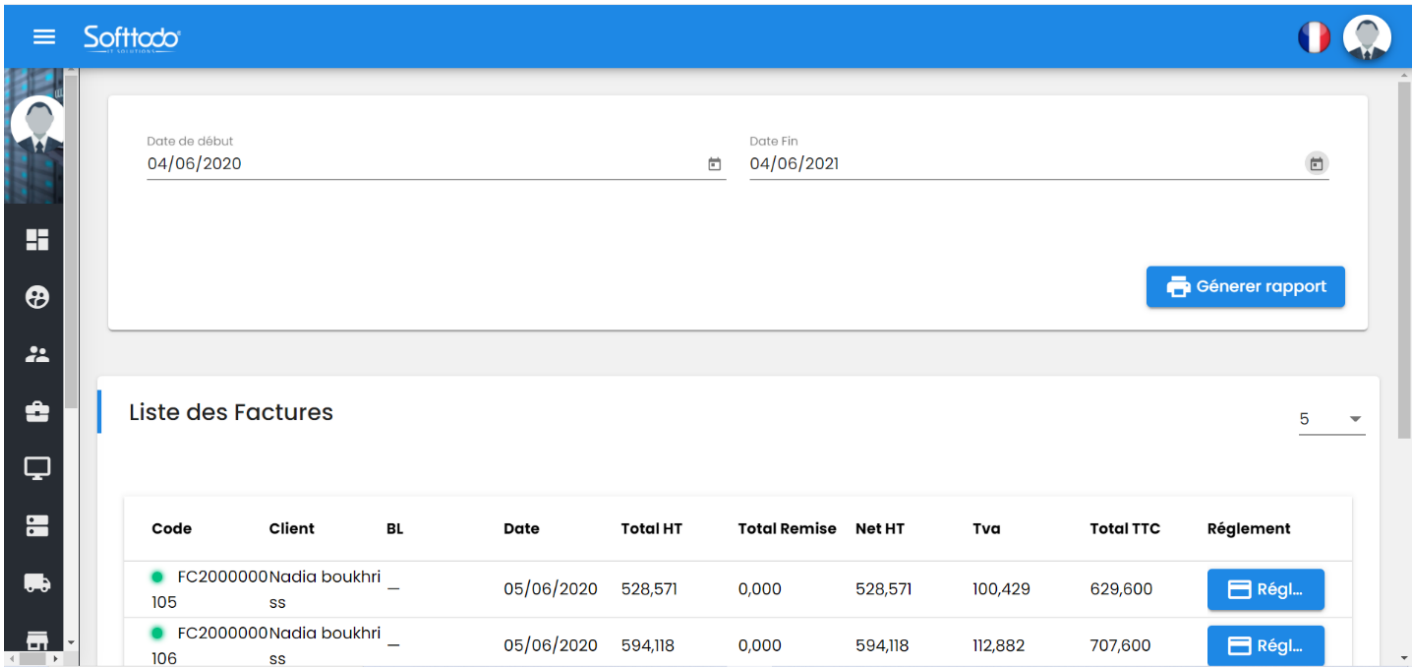


Figure 60 Etablir l'état des factures par période

Comme la génération du rapport de caisse, l'administrateur peut également générer un rapport pour la liste des factures soit pour le téléchargement en PDF ou bien pour l'impression. La Figure 61 représente la génération de ce rapport.

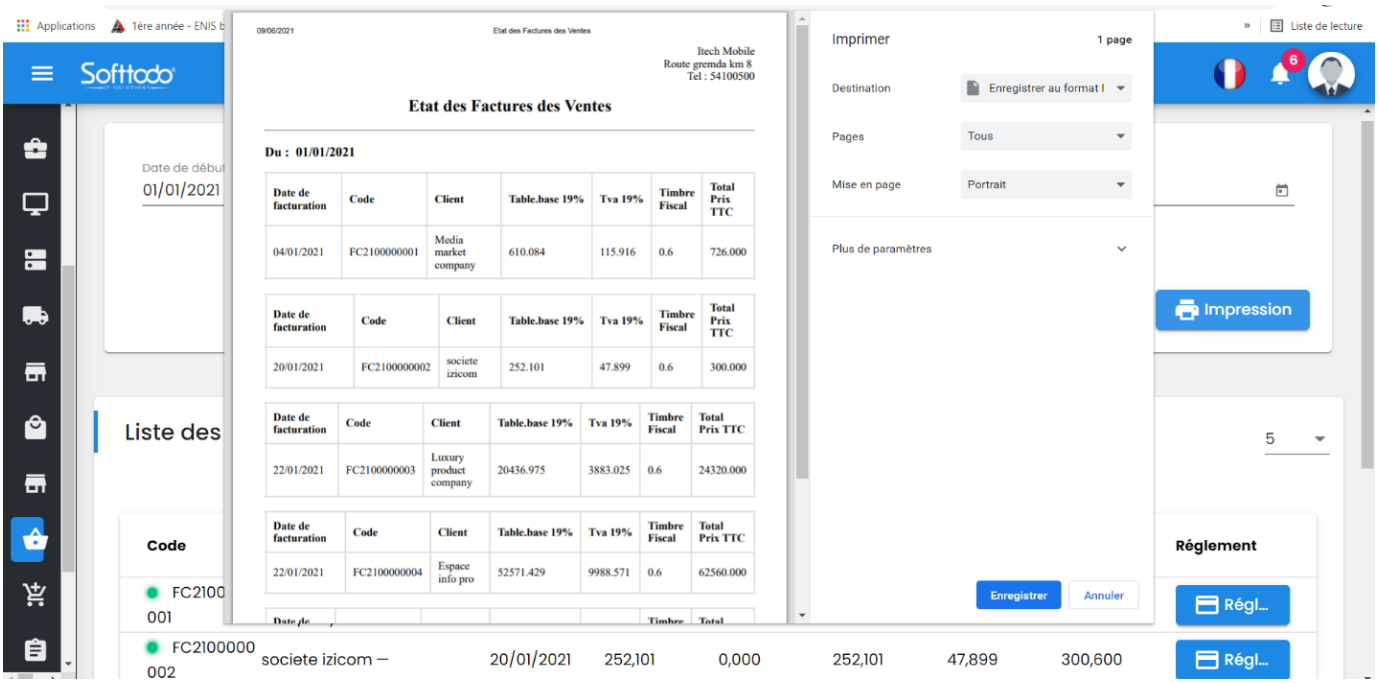


Figure 61 Interaction pour établir l'état des factures par période

3. Conclusion

Pour conclure, il importe de souligner que tout au long de ce chapitre nous avons présenté les phases de planification, de conception et de réalisation des fonctionnalités ajoutées au ERP SAPTUN en adoptant la méthodologie Scrum. Maintenant que nous sommes bons au niveau de l'application, il est temps d'optimiser son déploiement et de contrôler son fonctionnement pour garantir son efficacité. Ceci fera l'objet du chapitre suivant.

Chapitre 4 : Déploiement et Surveillance de l'ERP SAPTUN

1. Introduction

Après avoir développé les nouvelles fonctionnalités de l'ERP SAPTUN nous pouvons achever les phases indispensables à sa production. Dans ce dernier chapitre, nous décrivons la notion DevOps, puis nous détaillons le processus d'exécution de chaque étape du cycle DevOps et nous finissons par la mise en place d'un tableau de monitoring du système.

2. La méthodologie DevOps

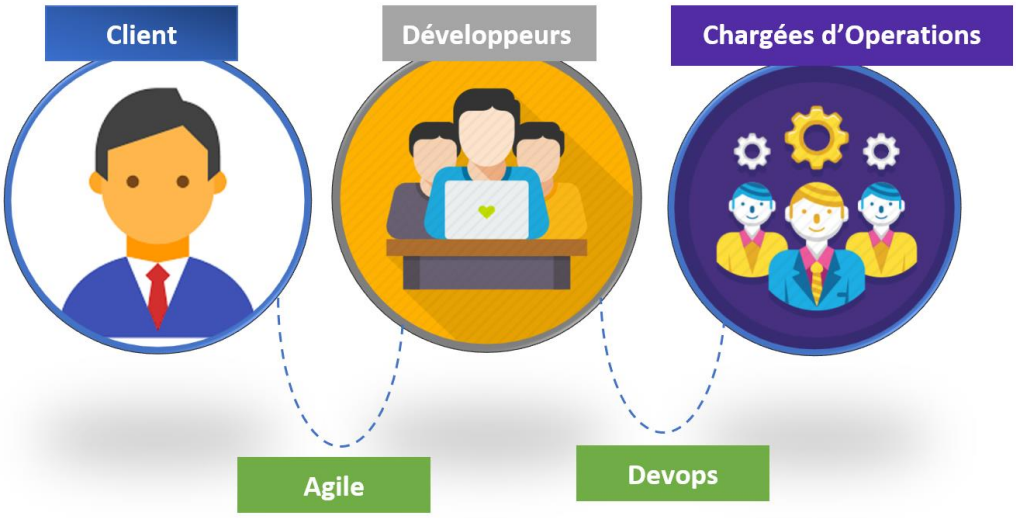


Figure 62 La différence entre les méthodologies DevOps et Agile

L'équipe développeur de l'ERP SAPTUN a adoptée d'une part l'approche Agile pour répondre aux besoins du client et d'autre part elle suit la démarche DevOps qui lui permet de combler l'écart avec les personnels chargés des opérations informatiques. En effet, ces deux profils poursuivent historiquement des objectifs opposés. Le développeur fait évoluer l'application en ajoutant de nouvelles fonctionnalités ou en corrigeant des bugs le plus rapidement, à moindres coûts et parfois au détriment de sa performance et de sa stabilité. Quant aux chargés d'opérations, ils se chargent de la mise en production du système en se portant garant de sa disponibilité et de son temps de réponse et en sacrifiant souvent le coût et le temps nécessaires au déploiement d'une nouvelle version.

DevOps se concentre sur une livraison constante tandis que le processus Agile se concentre lui sur des changements constants donc l'un ne remplace pas l'autre. Il s'agit de deux modes de

collaboration qui complètent le cycle de vie du projet pour s'assurer de la livraison en continu d'un logiciel fonctionnel et opérationnel.

3. La chaîne DevOps de l'ERP SAPTUN

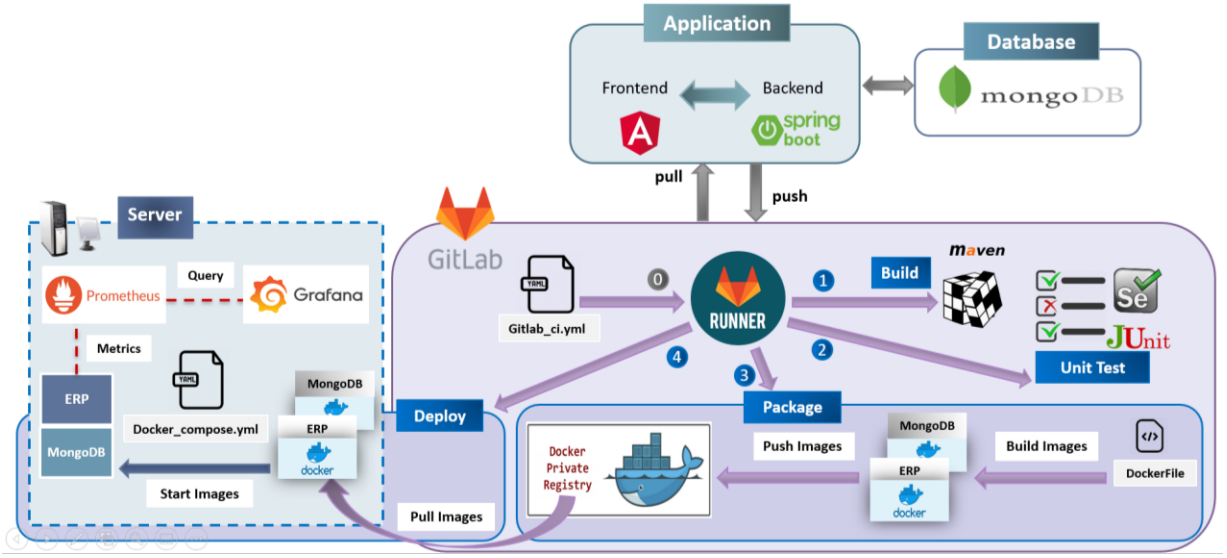


Figure 63 La chaine DevOps de l'ERP SAPTUN

Les différentes étapes du cycle DevOps forment le pipeline CI/CD (Continuous Integration and Continuous Delivery) en français Intégration continue et Déploiement continue. Ces deux notions néanmoins doivent être bien maîtrisées pour que la configuration du pipeline soit bien établie et fonctionnelle.

3.1 Intégration continue (CI)

[24] « L'intégration continue est un ensemble de pratiques utilisées en génie logiciel, consistant à vérifier, à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée ».

Le pipeline de l'intégration continue de notre projet commence dès la poussée en amont du code dans le serveur de gestion de version Git et prend fin lorsque les artefacts du code sont créés.

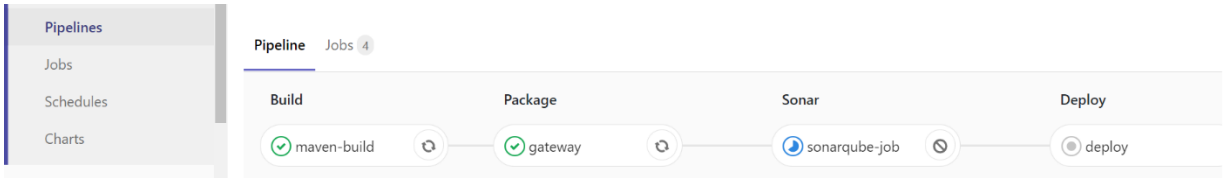


Figure 64 La chaine de l'intégration continue de l'ERP SAPTUN

3.2 Déploiement continue (CD)

[25] « La livraison continue est une discipline où l'application est construite de manière à pouvoir être mise en production à n'importe quel moment ».

Le pipeline du déploiement est la suite logique de celui de l'intégration continue. Il aura lieu grâce à un code stocké avec le code de l'application ce qui permet la description de l'infrastructure du système à déployer :

1. La codification de l'infrastructure avec l'**Infrastructure-as-Code**.
2. Le **déploiement** de l'application.

3.3 Gitlab

GitLab est une application dédiée pour automatiser le cycle DevOps en fournissant des solutions qui permettent l'exécution du pipeline CI/CD sans rentrer dans la mise en place complexe des outils, ainsi que les connexions associées.

3.3.1 Gitlab CI

Toutes les étapes du pipeline sont automatisées par un orchestrateur Gitlab CI, qui permet de gérer les dépendances entre elles. De plus, l'utilisation d'un orchestrateur permet de donner accès à tous, et à tout moment, à un tableau de bord qui donnera l'état de santé des étapes CI/CD. Ainsi, les développeurs ont au plus tôt la boucle de feedback nécessaire, afin de garantir que l'application soit prête au déploiement.

3.3.2 Gitlab Runner

Le GitLab Runner va gérer les jobs du pipeline CI/CD et les lancer automatiquement quand une branche sera envoyée sur le dépôt ou lorsqu'elle sera mergée.

3.3.3 Le manifeste gitlab_ci.yml

gitlab_ci.yml est la pierre angulaire de l'implémentation d'un pipeline dans GitLab. En fait, c'est un manifeste que nous trouvons à la racine du référentiel git et écrit en YAML. YAML l'acronyme de « Yet Another Markup Language» fournit de puissants paramètres de configuration très simple à lire, facile à comprendre et facile à éditer grâce auxquels nous définissons les scripts à exécuter.

```
1   image: docker:latest
2
3   services:
4     - docker:dind
5
6   cache:
7       paths:
8       - .m2/repository
9
10  variables:
11      MAVEN_OPTS: "-Dmaven.repo.local=${CI_PROJECT_DIR}/.m2/repository"
12      DOCKER_DRIVER: overlay2
13
14  stages:
15    - build
16    - test
17    - package
18    - deploy
```

Figure 65 Les composants nécessaires du gitlab_ci.yml de l'ERP SAPTUN

Le GitLab Runner peut utiliser des images Docker pour prendre en charge nos pipelines. Le mot clé « **image** » définit le nom de l'image Docker que nous allons utiliser. Les images valides sont celles hébergées dans le Docker Engine local ou sur Docker Hub. Le mot clé **services** définit des images Docker supplémentaires qui sont liées au conteneur principal. Le bloc avec le mot clé « **cache** » est utilisé pour accélérer toutes nos étapes. Effectivement, dans le cas d'une compilation Java avec Maven, cette compilation récupère beaucoup de dépendances et de librairies externes. Ces librairies sont stockées dans le répertoire « .m2 ». Les **variables** CI/CD constituent un type de variable d'environnement. Nous pouvons les utiliser pour contrôler le comportement des tâches, stocker les valeurs que nous souhaitons réutiliser ou encore éviter la répétition d'un code lourd très utilisé comme le chemin d'un répertoire.

- Le DOCKER_DRIVER signale au Docker Engine quel pilote de stockage à utiliser. Nous utilisons overlay2.
- $CI_PROJECT_DIR, est un répertoire commun à toutes les étapes du pipeline.

Dans le bloc « **stages** », nous recitons dans l'ordre les étapes qui constituent le pipeline vu qu'elles s'exécutent en séquentiel. Ici, la première étape va être le build du projet, puis l'exécution des tests fonctionnels, ensuite la création d'une image Docker et enfin le déploiement de l'application. Chaque étape est représentée par un bloc indépendant et unique appelé **« Job »** qui est l'élément le plus fondamental dans un fichier gitlab-ci.yml et qui détermine ce qui doit être exécuté et fixe les contraintes et les conditions de l'action à mettre en place. Dans les jobs, nous voyons que nous avons trois différentes lignes :

- **Stage** : c'est le nom de l'étape qui va apparaître dans notre pipeline d'intégration continue.

- **Script** : ce sont les lignes de script à lancer
- **Image** : c'est l'image Docker qui va être lancée par GitLab afin d'exécuter les lignes de script que nous avons définies.

4. Les stages du pipeline CI/CD :

4.1 Compilation du code

Les développeurs appliquent régulièrement leurs modifications sur le référentiel partagé de GitLab. Cela permet de ne plus se soucier car si la compilation ne se fait pas, le code est alors rejeté, et le développeur doit corriger ses erreurs.

```
20  maven-build:
21    image: maven:3-jdk-8
22    stage: build
23    script:
24      - "mvn clean install"
25    only:
26      - PFE2021
27    artifacts:
28      paths:
29        - $CI_PROJECT_DIR/target/*.jar
30      expire_in: 24 hrs
```

Figure 66 La description de l'étape de build dans le fichier gitlab_ci.yml

Ici, l'image maven:3-jdk-8, qui contient déjà Java 8, va être lancée afin de pouvoir exécuter le script qui à son tour va télécharger Maven, l'outil de compilation, toutes les dépendances de l'application et lancer la compilation du projet . Les artefacts sont les résultats d'une étape que nous souhaitons reporter à d'autres étapes. Dans ce cas, nous demandons que tous les livrables .jar générés lors du build du projet soient sauvegardés pour les prochaines étapes. Après cette première étape, le code devient plus sûr et le dépôt de code source garantit qu'à chaque instant, un développeur récupère un code qui compile. En revanche, les tests ne sont pas encore exécutés et le code peut donc être de mauvaise qualité. A la suite de la compilation, l'orchestrateur se charge de lancer les tests fonctionnels codés par les développeurs.

4.2 Test fonctionnel de l'application

Dans le cahier de charge, le client annonce ses besoins à l'équipe de production en exigeant des règles métier c'est-à-dire des critères qui décrivent le résultat attendu de chaque situation pouvant exister. Une fois le besoin est développé par l'équipe de production logicielle, avant de livrer au client, il faut s'assurer de la qualité du produit final.

4.2.1 Définition de Test Fonctionnel

Les tests fonctionnels sont destinés à s'assurer que l'application réponde au cahier des charges et aux spécifications du point de vue de l'utilisateur. Ils ont pour objectif de dérouler un scénario composé d'une liste d'actions et pour chaque action une liste de vérifications validant la conformité de ce qui se passe en réel avec l'attendu. Ce sont des tests en boite noire qui ne dépendent pas de la structure interne du programme ou du code. De ce fait, écrire les scénarios très tôt est fortement souhaitable car cela facilite la tâche du développeur en lui offrant la bonne analyse fonctionnelle.

4.2.2 Les bases d'un test fonctionnel automatisé

4.2.2.1 Développement pilotée par le comportement

La phase de test fonctionnel est considérée comme une partie intégrante du développement étant donné qu'elle exploite le développement piloté par le comportement (en anglais behaviour-driven development ou BDD).

[26] « C'est une méthode de programmation agile qui encourage la collaboration entre les développeurs, les ingénieurs qualité et les intervenants non techniques ou commerciaux participant à un projet logiciel ».

Cette méthode permet à un utilisateur de définir le comportement d'une application en anglais simple, ce qui permet aux non-programmeurs de mieux comprendre les critères d'acceptation et aux programmeurs de modifier le code de l'application et prendre en compte toutes les entrées, erreurs et sorties possibles jusqu'à ce que le test réussisse.

4.2.2.2 Framework d'automatisation cucumber avec du sélénium.

L'utilisation de frameworks dédiés à l'automatisation des tests fonctionnels ne nécessite aucune modification dans le code à tester. Elle assure une séparation totale et claire entre le code et leurs tests.

- **Décrire le comportement en langage naturel**

Cucumber est un framework de test open source qui prend en charge le développement axé sur le comportement BDD pour les tests d'automatisation des applications Web. En effet, les spécifications définies par l'utilisateur pour valider l'application sont regroupés en fonctionnalités systèmes dont chacune est décrite dans un fichier distinct appelé **« Feature**

File ». La Figure 80 représente la liste des différents « fetaures » élaborés le long de notre projet.

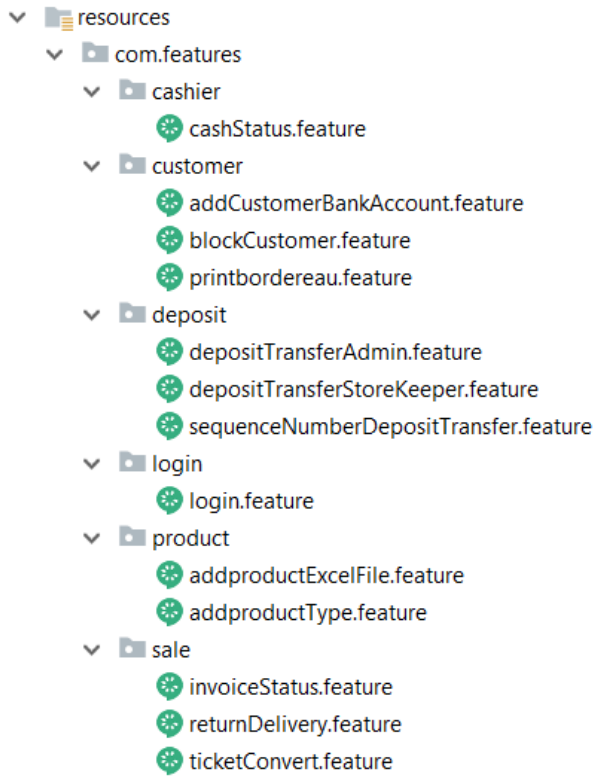


Figure 67 La liste des "Feature File" du Test fonctionnel

Le « Feature File » est écrit à l'aide de Gherkin Language qui est à son tour basé sur un ensemble prédéfini de mots-clés contextuels simples pour écrire les scénarios de comportement de l'application. Chaque scénario est décrit par des étapes facultatives, ordonnées, bien connus des praticiens BDD écrites par l'auteur du test :

- **Given :** La condition préalable du test.
- **When :** L'interaction du testeur pour exercer le comportement à tester
- **Then :** Le résultat attendu du test lorsque les conditions initiales sont remplies et l'action est effectuée.
- **And :** Signifie une autre étape équivalente à la précédente.

```
Feature: Test login Functionality

  Scenario: Check login is successfully with valid credentials
    Given user is on login page
    When  user enters username and password
    Then  user click on login and login should be successful
```

Figure 68 Un exemple de code du "Feature File"

❖ Implémenter le code de test des steps

Sélénium est un outil d'automatisation pour les tests fonctionnels de l'application Web. Le framework Cucumber peut être utilisé pour tester les applications avec Sélénium. Le cœur de Sélénium est le WebDriver qui pilote un navigateur de manière native, comme le ferait un utilisateur. Chaque étape du « feature File » est liée à une méthode dans un fichier Java grâce à des annotations spécifiques : @Given, @When, @Then et à l'aide d'une correspondance de chaîne à quelques caractères spéciaux près. Cette méthode devient exécutable dès qu'on lui associe le code qui est une traduction algorithmique du comportement attendu.

```
public class LoginAdmin {
    Constant cte = new Constant();
    WebDriver driver = Driver.getInstance();
    String baseUrl;

    @Given("^user is on login page$")
    public void user_is_on_login_page() throws Throwable {

        baseUrl = "https://www.google.com/";
        driver.manage().timeouts().implicitlyWait( time: 30, TimeUnit.SECONDS);
        driver.get(cte.url);



    }

    @When("^user enters username and password$")
    public void user_enters_username_and_password() throws Throwable {
        driver.findElement(By.id("email")).click();
        driver.findElement(By.id("email")).clear();
        driver.findElement(By.id("email")).sendKeys(cte.loginAdmin);
        driver.findElement(By.id("suivant")).click();

        driver.findElement(By.id("password")).clear();
        driver.findElement(By.id("password")).sendKeys(cte.passwordAdmin);

    }

    @Then("^user click on login and login should be successful$")
    public void user_click_on_login_and_login_should_be_successful() throws Throwable {

        driver.findElement(By.id("connecter")).click();
```

Figure 69 Exemple de code du fichier Java

Cette portion de code Java montre que Selenium utilise des localisateurs pour interagir avec les éléments de la page Web. Les localisateurs sont les propriétés HTML d'un élément Web qui indiquent à Selenium l'élément dont il a besoin pour effectuer l'action. Pour identifier cet élément web avec précision, nous avons différents types de localisateurs, à savoir :

- Identifiant (driver.findElement (By.id(<<>>)
- Nom (driver.findElement (By.name(<<>>)
- Sélecteur CSS (driver.findElement (By.cssSelector<<>>)
- XPath (driver.findElement (By.xpath <<>>)

4.2.2.3 Interfaces valorisantes le test des fonctionnalités développées

Il existe deux types de test fonctionnel :

- **Le test fonctionnel de progression :** réalisé sur une exigence qui vient d'être développée. C'est donc la première fois que ce test est réalisé, avant que le produit ne soit livré en production.
- **Le test fonctionnel de non-régression :** réalisé sur une exigence existant déjà avant les développements en cours. Il permet de s'assurer que les évolutions et les corrections effectuées dans une nouvelle version n'ont pas entraîné de régressions sur les fonctionnalités existantes.

Il est nécessaire de tester des valeurs correctes ainsi que des valeurs intentionnellement fausses afin que les fonctionnalités soient résistantes aux usages imprévus. Dans la partie suivante nous présentons les différentes interfaces qui montrent les règles métiers respectés par notre programme en se focalisant sur le test fonctionnel de **progression**.

❖ Attribuer des comptes bancaires au client

D'après la Figure83 nous constatons que l'ajout est contrôlé par des contraintes pour éviter le risque de redondance des données. Si l'administrateur entre le même compte alors le système réinitialise les champs à vide et affiche un message d'attention.

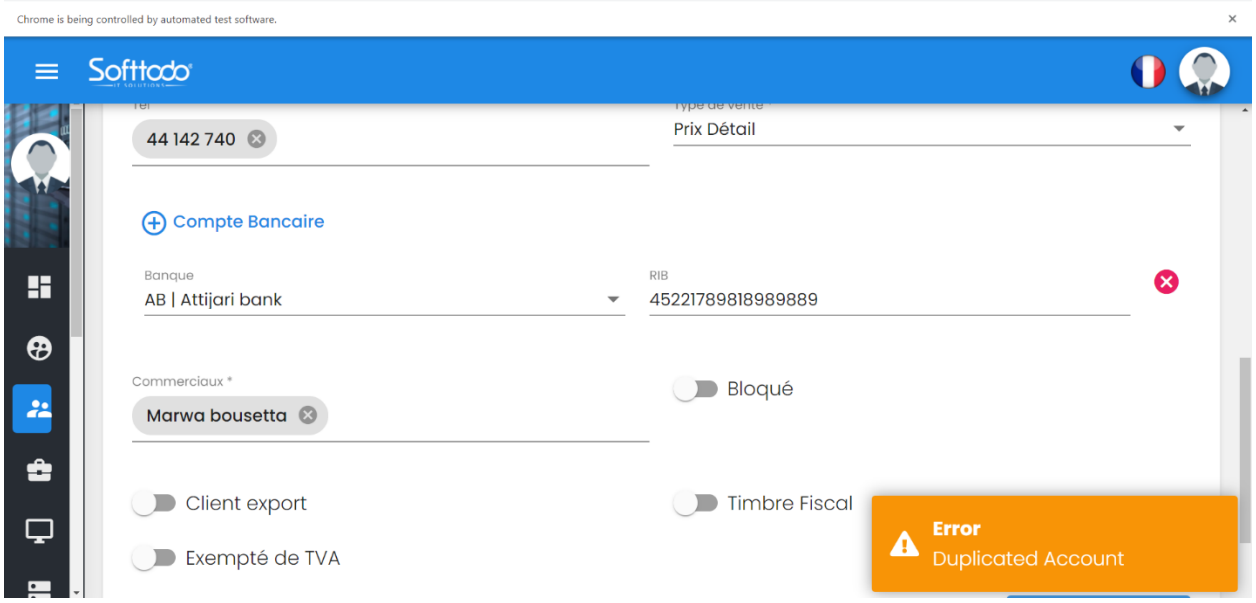


Figure 70 L'interface du test fonctionnel de l'ajout des comptes bancaires

Une création non complète des comptes bancaires empêche l'action de création ou de modification du client et le système affiche un message d'erreur.

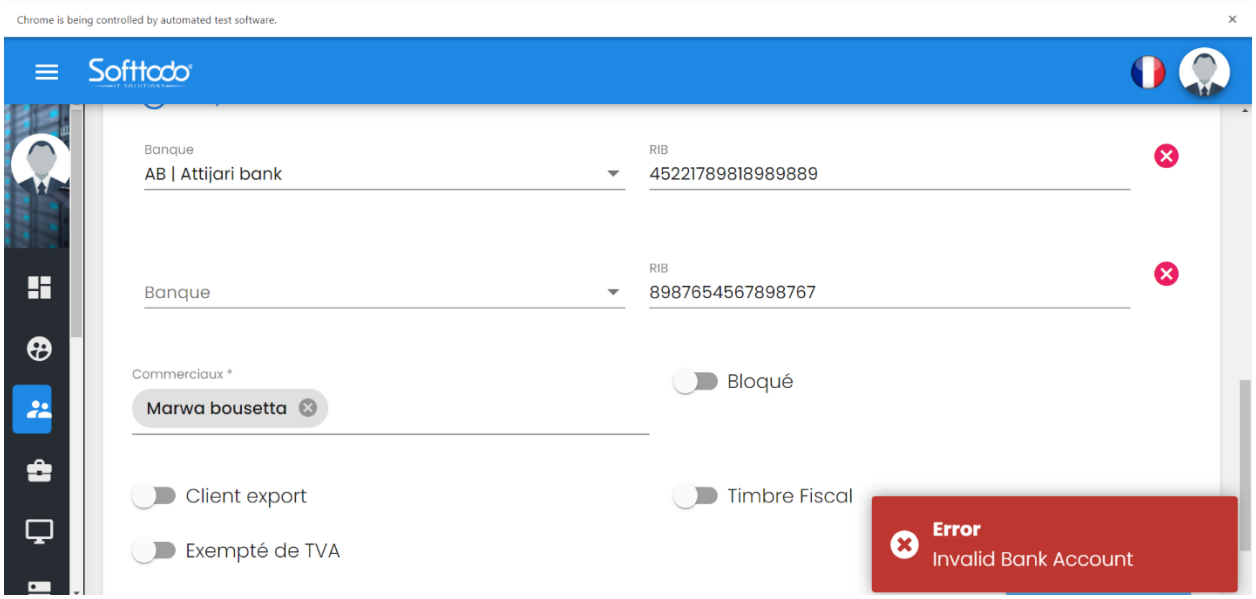


Figure 71 L'interface du test fonctionnel de l'ajout des comptes bancaires

❖ Effectuer un bon de transfert de produits entre dépôts

Dans le but de fiabiliser les données et de rester aligné avec les objectifs métiers de l'entreprise si l'utilisateur de l'ERP entre les produits à transférer et leurs différentes quantités et essaye de passer le bon de transfert alors le système affiche un message d'errer mentionnant que le dépôt de source et/ou celui de destination ne sont pas encore sélectionné(s). Ainsi l'enregistrement d'un bon de transfert dont la liste de produits est vide est interdit. Ceci engendre une erreur et un message significatif s'affiche à l'utilisateur ce qui montre la figure ci-dessous.

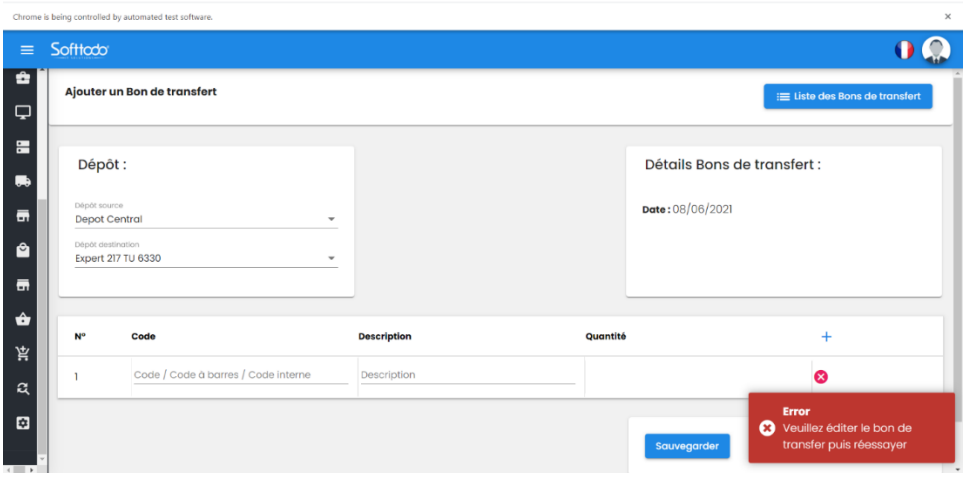
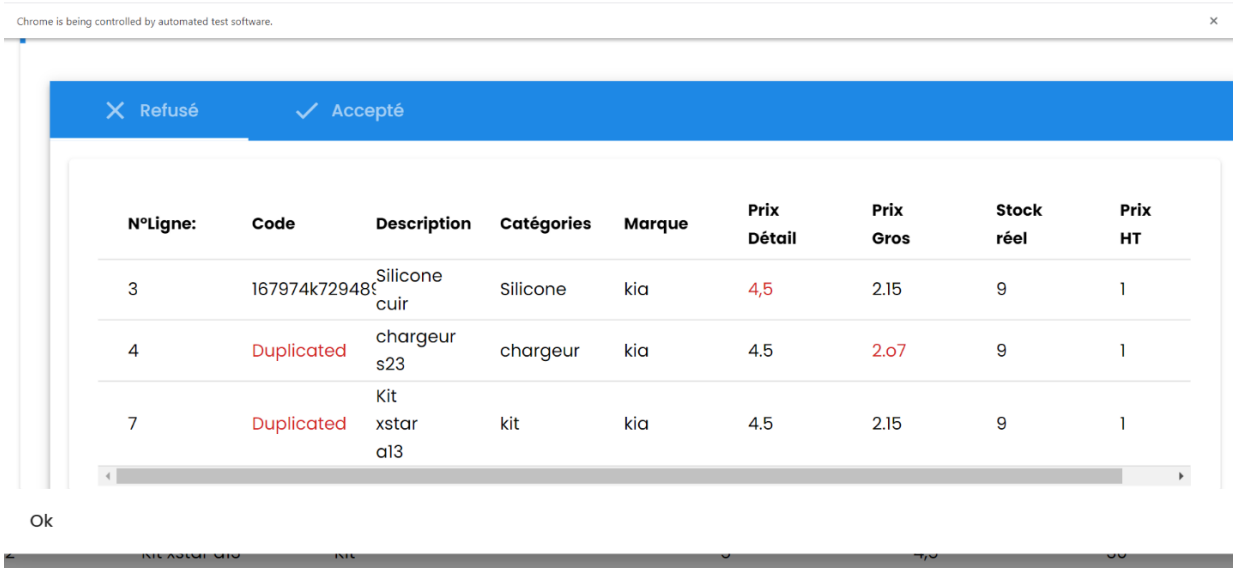


Figure 72 L'interface qui illustre le test fonctionnel de la création d'un bon de transfert

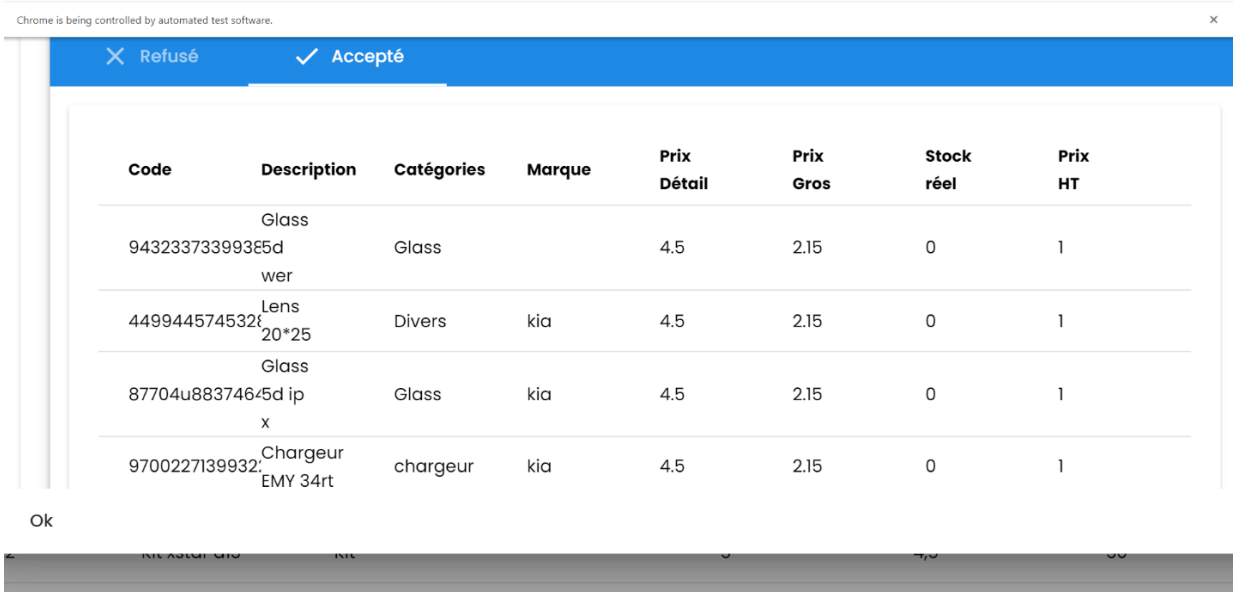
❖ Importer une liste de produits à travers un fichier Excel

L'enregistrement des données écrites dans le fichier Excel peut engendrer des erreurs terribles dans la base de données si cette opération n'est pas bien étudiée. La Figure 73 montre que le système traite le fichier ligne par ligne. L'entrée des coordonnées erronées d'un produit engendre le refus de celui-ci et l'affiche dans la liste des produits refusés en mentionnant à l'utilisateur l'issue de l'erreur et le numéro de ligne dans le fichier Excel sinon tout va bien et le produit est ajouté avec succès et affiché dans la liste de ceux acceptés.



N°Ligne:	Code	Description	Catégories	Marque	Prix Détail	Prix Gros	Stock réel	Prix HT
3	167974k72948	Silicone cuir	Silicone	kia	4,5	2.15	9	1
4	Duplicated	chargeur s23	chargeur	kia	4.5	2.o7	9	1
7	Duplicated	Kit xstar a13	kit	kia	4.5	2.15	9	1

Figure 73 La liste des produits refusés



Code	Description	Catégories	Marque	Prix Détail	Prix Gros	Stock réel	Prix HT
9432337339935	Glass 5d wer	Glass		4.5	2.15	0	1
449944574532	Lens 20*25	Divers	kia	4.5	2.15	0	1
87704u883746	Glass 5d ip x	Glass	kia	4.5	2.15	0	1
9700227139932	Chargeur EMY 34rt	chargeur	kia	4.5	2.15	0	1

Figure 74 La liste des produits acceptés

4.3 Stockage de l'mage Docker dans le registre du GitLab

```
43    gateway:
44      stage: package
45      before_script:
46        - cp $CI_PROJECT_DIR/target/erp*.jar $CI_PROJECT_DIR/target/erp.jar
47        - docker login -u $CI_REGISTRY_USER -p $CI_BUILD_TOKEN $CI_REGISTRY
48      script:
49        - docker build --tag $CI_REGISTRY_IMAGE/pfe2021:latest  .
50        - docker push $CI_REGISTRY_IMAGE/pfe2021:latest
51      only:
52        - PFE2021
53      tags:
54        - pfe2021
55      dependencies:
56        - maven-build
```

Figure 75 La description de l'étape de package dans le fichier gitlab_ci.yml

Le code, une fois compilé et testé, doit être déployé dans un dépôt de livrables. Nous choisissons la technologie de conteneurisation Docker qui permet de déployer l'application dans un conteneur docker.

4.3.1 Docker

Docker est une plateforme virtuelle pour Linux créée en 2013 permettant de construire, déployer et lancer des applications cloisonnées.

4.3.1.1 Conteneur Docker

Un conteneur Docker contient ce dont une application a besoin (fichiers binaires, librairies et systèmede fichier). Les conteneurs en cours d'exécution sur une seule machine partagent tous le même noyau du système d'exploitation pour commencer instantanément et faire une utilisation plus efficace de la RAM. Vu ses points forts, sa légèreté, son portabilité et l'isolation qu'il offre, nous choisissons docker comme solution de virtualisation pour mettre en place notre plateforme d'intégration continue.

4.3.1.2 Docker File

Les « Dockerfiles » décrivent un système de packages, de fichiers de configuration, de ports utilisés, etc qui permettent de construire le conteneur applicatif, assurant ainsi un environnement compatible. Ces dockerfiles sont présents dans le répertoire Docker du projet et partent d'une image qui contient déjà la version 8 de Java et copie tous les .jar des différents projets, afin de construire les différentes images associées. La dernière étape de ce dockerfile est de lancer la commande Java avec le jar associé.

4.3.2 Gitlab avec Docker

GitLab vient avec un registre Docker inclus, ce qui nous permet de packager l'application et ses dépendances et de créer facilement des artefacts au sein de GitLab. La création et la publication d'images sont assurées avec l'exécution de ces commandes docker :

1 - cp $CI_PROJECT_DIR/target/erp*.jar $CI_PROJECT_DIR/target/erp.jar

- ⇨ Copier les artefacts de build construits dans le répertoire du projet.

2 - docker login -u $CI_REGISTRY_USER -p $CI_BUILD_TOKEN $CI_REGISTRY

- ⇨ Se connecter au registre de gitlab en utilisant $CI_REGISTRY_USER et $CI_BUILD_TOKEN $CI_REGISTRY des variables sécurisées et enregistrées au niveau de gitlab CI.

3 - Une fois connecté au registre, nous pouvons construire l'image docker en utilisant la commande **docker build** et l'option **--tag** pour versionner l'image.

4 - Enfin, nous partageons l'image docker construite dans le registre du Gitlab avec la commande **docker push**.

Ce processus nous permettra de figer le code compilé dans un package immuable. De ce fait, nous pouvons facilement redéployer le même code compilé sur n'importe quel autre environnement. Notre code est compilé en continu, testé, packagé et prêt à être déployé sur de nouveaux environnements.

4.4 Déploiement de l'application sur le serveur de test

L'étape la plus importante de la livraison continue est le **déploiement du package** que nous avons précédemment créé lors de l'intégration continue. Les avantages d'utiliser un outil pour **automatiser le déploiement** de l'application sont nombreux :

- Permet à l'équipe de se concentrer sur le développement
- Les déploiements deviennent beaucoup moins sujets aux erreurs et beaucoup plus reproductibles
- Le déploiement sur un nouvel environnement est facile et peut être très fréquent.

❖ **Docker Compose.yml**

Un fichier unique qui spécifie l'environnement, la configuration et l'accès pour notre application est archivé avec le code source. La figure ci-dessous présente les différentes images en exécution sur notre serveur de test.

```
erp@erp:~$ docker ps
CONTAINER ID        IMAGE                                                   COMMAND                  CREATED             STATUS              PORTS
                    NAMES
55e2240b91d5        grafana/grafana                                         "/run.sh"                3 hours ago         Up 3 hours          0.0.0.0:
3000->3000/tcp      grafana
7eb2ccf618cd        prom/prometheus                                         "/bin/prometheus --c…"   24 hours ago        Up 21 hours         0.0.0.0:
9090->9090/tcp      interesting_leakey
f9d4afb4b985        quay.io/prometheus/node-exporter:latest                 "/bin/node_exporter …"   46 hours ago        Up 46 hours
                    adoring_austin
642757c96117        mongo:latest                                            "docker-entrypoint.s…"   10 days ago         Up 21 hours         0.0.0.0:
27017->27017/tcp    mongodb
c3772f9be3a2        gitlab-registry.soft2do.de/softtodo-erp/erp/pfe2021:latest   "java -Xmx2000m -jar…"   10 days ago         Up 21 hours (healthy)   0.0.0.0:
4444->4000/tcp      erp
```

Figure 76 Les différents images crées sur le serveur de test

5. La supervision des performances

Afin de savoir si le déploiement d'un système s'est bien déroulé, il est nécessaire de superviser son fonctionnement, ainsi d'activer des notifications en cas de problème pour avoir un feedback rapide. Le monitoring, ou supervision, intervient une fois que notre système est déployé. Le principe est de récupérer certaines métriques significatives pour ceux qui interviennent sur l'application. Cela peut être par exemple le nombre de connexions HTTP, le temps de réponse de certaines pages,etc.

5.1 Prometheus

Prometheus, un excellent outil pour collecter les métriques à partir de notre application, est basé sur la création d'un fichier de configuration écrit en yaml à partir duquel les jobs sont configurés. L'image Docker Prometheus est disponible dans la section des versions du référentiel GitHub.

La commande ci-dessous permet l'execution de l'image Prometheus :

docker run -d -p 9090:9090 -v $PWD/prometheus.yml prom/Prometheus

Prometheus interroge un ensemble de cibles configurées pour récupérer par intermittence leurs valeurs métriques. Dans la terminologie de Prometheus, cette interrogation est appelée grattage (qui se produit par défaut toutes les 15 secondes). Notre application expose ses métriques pour Prometheus sur le point de terminaison « /prometheus ».

Prometheus Alerts Graph Status Help Classic UI

```
global:
  scrape_interval: 15s
  scrape_timeout: 10s
  evaluation_interval: 1m
scrape_configs:
- job_name: prometheus
  honor_timestamps: true
  scrape_interval: 5s
  scrape_timeout: 5s
  metrics_path: /metrics
  scheme: http
  follow_redirects: true
  static_configs:
  - targets:
    - 192.168.1.207:9090
- job_name: node_exporter
  honor_timestamps: true
  scrape_interval: 5s
  scrape_timeout: 5s
  metrics_path: /metrics
  scheme: http
  follow_redirects: true
  static_configs:
  - targets:
    - 192.168.1.207:9100
- job_name: spring_micrometer
  honor_timestamps: true
  scrape_interval: 5s
  scrape_timeout: 5s
  metrics_path: prometheus
  scheme: http
  follow_redirects: true
  static_configs:
  - targets:
    - 192.168.1.207:4444
```

Figure 77 Le fichier prometheus.yml

La Figure 78 présente les différentes cibles que Prometheus se permet d'extraire leurs métriques.

Prometheus Alerts Graph Status Help Classic UI

Targets

All Unhealthy Collapse All

node_exporter (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.1.207:9100/metrics	UP	instance="192.168.1.207:9100" job="node_exporter"	1.809s ago	33.446ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.1.207:9090/metrics	UP	instance="192.168.1.207:9090" job="prometheus"	2.451s ago	14.221ms	

spring_micrometer (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.1.207:4444/prometheus	UP	instance="192.168.1.207:4444" job="spring_micrometer"	1.520s ago	3.549ms	

Figure 78 Les cibles de Prometheus

5.2 Eléments de performance

5.2.1 Threads

D’une part, les threads démons sont des threads à faible priorité conçus pour effectuer des tâches spécifiques ou complexes et qui s'exécutent toujours en arrière-plan. D’autre part, les threads utilisateur (Peak) sont des threads à haute priorité utilisés pour effectuer des tâches de support et qui s'exécutent toujours au premier plan. JVM n'attend pas que le thread démon se termine mais attend le thread utilisateur (Peak). Plus il y a de threads, plus le système va devoir switcher. De plus, un thread requiert des ressources pour s'exécuter notamment un espace mémoire nommé pile. Il est donc nécessaire de contrôler le nombre de threads qui sont lancés dans une même JVM car l’élévation de ce nombre peut ralentir l’application, et même le serveur sur lequel elle s'exécute.

5.2.2 Mémoire

Il existe deux types importants de mémoire JVM à surveiller : la mémoire Heap et non-Heap. Chacune ayant son propre objectif. La mémoire Heap est l'endroit où la JVM stocke les données d'exécution représentées par les instances allouées et peut être libérée lorsque le Garbage Collector s'exécute. Lorsque l'espace de Heap est épuisé, la JVM renvoie une erreur OutOfMemoryError. Par conséquent, il est très important de surveiller l'évolution de la mémoire Heap libre et utilisée pour empêcher la JVM de ralentir et éventuellement de planter. La mémoire non-Heap est l'endroit où la JVM stocke les informations au niveau de la classe telles que les champs et les méthodes d'une classe, le code de méthode, le pool de constantes d'exécution et les chaînes internalisées. Un manque de mémoire non-Heap peut indiquer qu'il y a un grand nombre de chaînes en cours d'internalisation ou une fuite du chargeur de classe.

5.2.3 Requêtes HTPP

Surveiller le trafic HTTP entrant pour savoir la latence des demandes est offert par la métrique http_server_requests_seconds_count qui définit le nombre de requêtes http. **rate (http_server_requests_seconds_count [1m])** renverrait le taux de requêtes par seconde moyenné au cours de la dernière minute. Nous avons effectué une requête simple qui nous donne la durée moyenne des requêtes entrantes

rate (http_server_requests_seconds_sum [1m]) / rate (http_server_requests_seconds_count [1m])

5.2.4 CPU

Le CPU (abréviation en anglais de « Central Processing Unit », en français « processeur ») est responsable du traitement des données et de la transmission des instructions. De ce fait, la vitesse à laquelle l'ensemble des programmes et des processus fonctionnent dépend des processus trop gourmands qui fonctionnent tous en même temps et que le CPU atteigne par conséquent sa limite d'utilisation. Il est donc important de suivre de près la charge CPU. **process_cpu_seconds_total** est la métrique qui nous permet de visualiser la variation de l'utilisation du CPU.

5.3 Grafana

Nous pouvons installer et exécuter Grafana à l'aide de l'image Docker officielle.

docker run -ti --user root --volume "grafana" --entrypoint bash grafana/grafana:latest

La figure ci-dessous montre la configuration de la source des données pour la supervision de notre système à l'aide de prometheus.

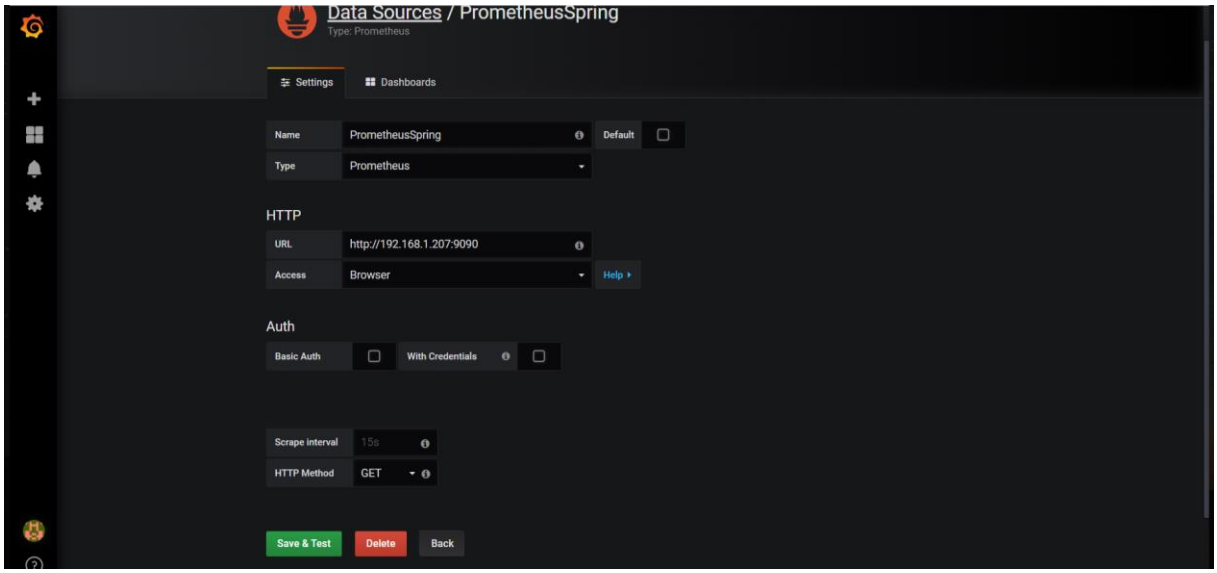


Figure 79 La source de données pour Grafana

5.3.1 Supervision de l'application

La supervision de l’application est offerte par un tableau de bord fourni par Grafana ,qui permet de personnaliser les métriques et les graphiques comme nous souhaitons, et conçu spécialement pour contrôler le fonctionnement des applications Spring Boot.

La figure ci-dessous nous montre plusieurs panneaux chacun, représente la variation d’une métrique extraite de l’application grâce à l’exploitation d’une source de données Prometheus configurée par l’équipe de l’ERP SAPTUN. Nous commençons par la métrique « jvm_threads_current », représentant les Thread en cours d’exécution, qui donne comme pourcentage 38%, une valeur acceptable en termes de charge que le processeur doit gérer.

Pour chaque zone mémoire, nous pouvons voir combien de mémoire a été utilisé à partir du panneau « Memory Used » représentant la métrique « jvm_memory_used_bytes » qui varie en continu de 50 Mb jusqu’à 80 Mb. Alors cette zone de stabilité est considérée comme la zone verte de variation de l’utilisation de la mémoire par notre application. Tout dépassement est nécessairement une défaillance que l’équipe doit traiter.

Les métriques Spring MVC peuvent également calculer des quantiles et des centiles, ce qui peut être utile lorsque nous souhaitons évaluer la lenteur de la durée de la demande du chargement d’une API. Comme le montre notre tableau de bord, « http_request_duration_seconds_count » nous donne la durée moyenne des requêtes entrantes qui débute d’une valeur de 125 ms et augmente au fil de temps.

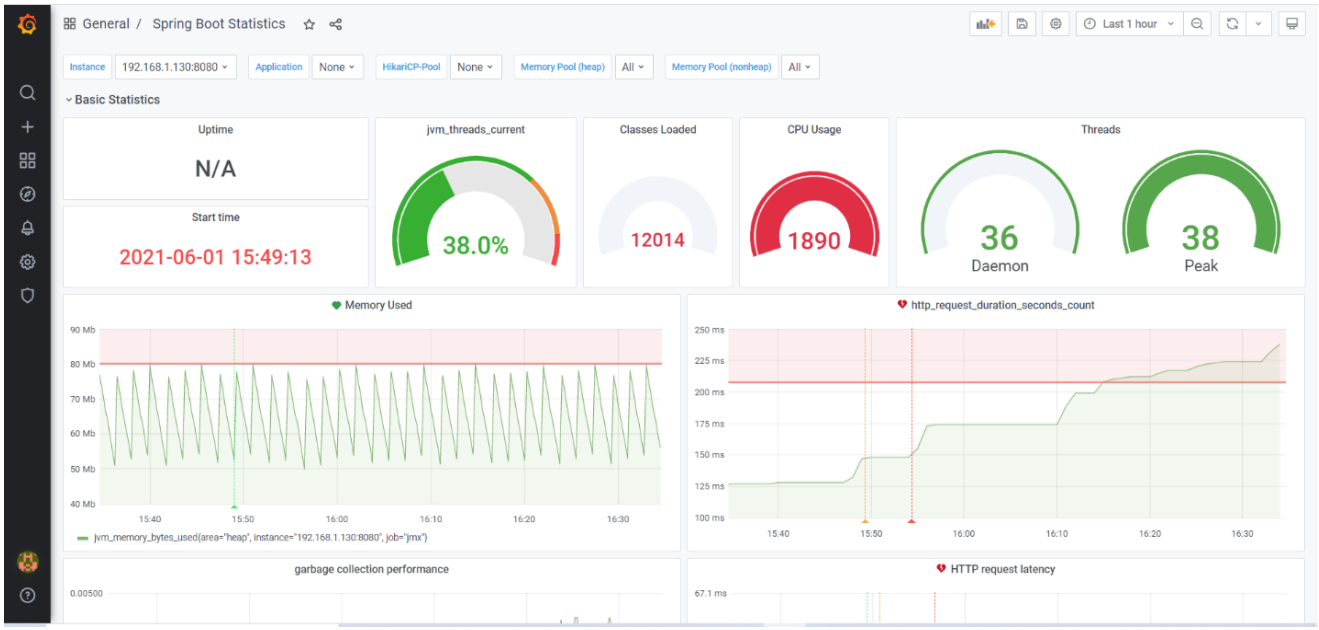


Figure 80 Tableau de bord de supervision de l'application

5.3.2 Supervision du serveur

Le serveur est ainsi supervisé. La Figure ci-dessous représente les différentes métriques exposées grâce au Prometheus tel que l'usage du CPU, une valeur de 24% très acceptable par rapport à la valeur critique adoptée par plusieurs organisations :70 % pendant 15 minutes. De plus, la mémoire utilisée est de 74 % une valeur inadmissible qui montre une grande inefficacité du processus de libération de la mémoire.

Figure 81 Tableau de bord de supervision du serveur

5.4 La mise en place des notifications d'alerte

5.4.1 Configuration de la chaine de notification

La dernière étape est l'étape de feedback rapide. Cette étape est celle qui va nous permettre de faire le lien entre la production (ops), et les développeurs (dev) en favorisant une communication très active. En effet, c'est une étape qui donne de la visibilité aux développeurs sur des problèmes qu'il peut y avoir en production. Plus rapide est la détection des problèmes, plus rapide est leur correction. L'intégration la plus simple est l'intégration E-mail.

5.4.2 Alertes reçues

La figure ci-dessous présente l'email reçu de la part de Grafana qui indique le dépassement de la valeur critique de l'utilisation de la mémoire.

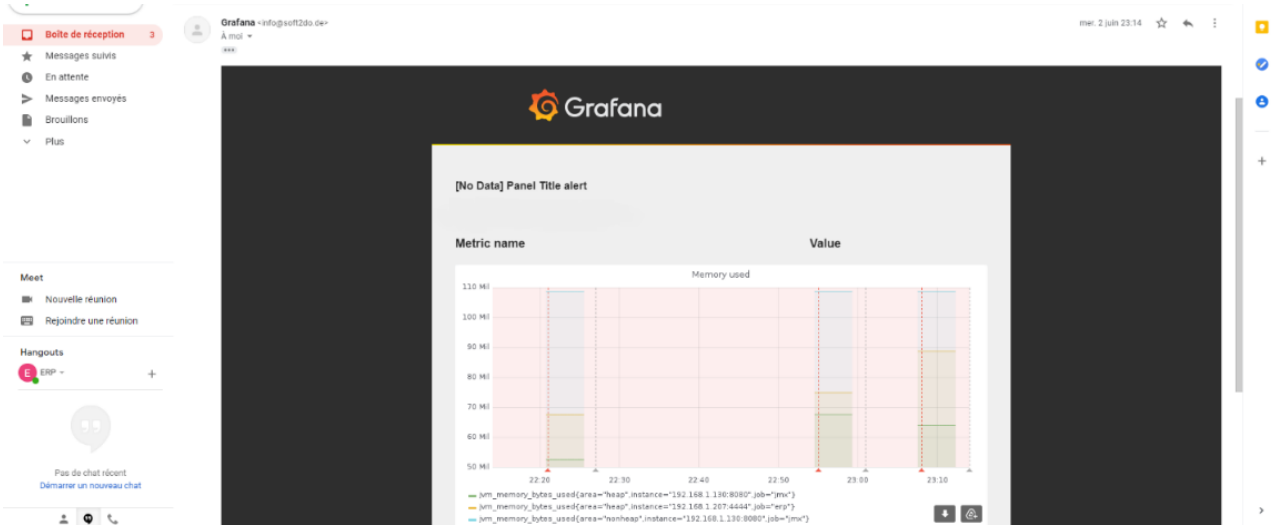


Figure 82 Email d'alerte concernant l'utilisation de la mémoire

6. Conclusion

Au cours de ce chapitre, nous avons définis l'intégration continue et le déploiement continue en passant par la notion de virtualisation avec la technologie Docker et les avantages qu'offre l'utilisation de Gitlab. Puis nous avons décrit la supervision de plusieurs métriques de l'application et du serveur de test grâce à Prometheus et Grafana. Enfin, nous avons présenté la configuration effectuée pour que des notifications d'alerte soient automatiquement envoyées en cas de problème.

Conclusion et perspectives

Nous touchons ainsi à la fin de notre rapport élaboré dans le cadre du projet de fin d'étude et à partir duquel nous avons essayé de présenter nos objectifs du projet, d'expliquer les principales notions auxquelles nous avons fait référence et de décrire en détail toutes les étapes pour atteindre ces objectifs.

Ce stage a été très bénéfique sur le plan professionnel et personnel. C'était d'une part l'occasion d'appliquer les atouts acquis durant toutes les années de notre formation à l'École Nationale d'Ingénieurs de Sfax ENIS, et d'autre part de se familiariser avec la vie professionnelle et le travail de groupe puisqu'il constitue une véritable porte d'entrée motivante et ambitieuse au monde professionnel.

En termes de contact quotidien avec l'équipe de l'ERP SAPTUN, ce stage nous a permis, non seulement d'approfondir les connaissances et le savoir-faire, mais aussi d'avoir une vision beaucoup plus générale sur l'architecture d'un ERP et son cycle de vie surtout en appliquant la méthode Scrum qui est une vraie opportunité pour apprendre à être responsable et autoorganisé.

Néanmoins, comme tout projet informatique, nous étions exposés à quelques contraintes qu'il fallait surmonter tout au long de la réalisation de ce travail. Elles ont constitué un défi pour acquérir de nouvelles connaissances techniques et de développer les capacités opérationnelles.

Malgré les difficultés techniques rencontrées et les contraintes de temps, nous pouvons considérer que nous avons réussi à atteindre l'objectif du projet et développer toutes les fonctionnalités spécifiées dans le cahier de charge.

En guise de perspective, notre solution a l'avantage d'être ouverte et extensible. Nous pouvons ainsi envisager à exploiter les informations intéressantes en des analyses prédictives. La prédiction se fait par des algorithmes permettant de transformer les données en connaissances et d'évaluer les probabilités futures. Cette notion s'appelle le data mining qui permet de rechercher des liens de causalité des données afin d'aider les décideurs à prendre des décisions adéquates.

Webgraphie

[1] https://www.jetbrains.com/fr-fr/idea/

[2] https://www.mongodb.com/products/compass

[3] https://about.gitlab.com/

[4] https://community.jaspersoft.com/project/jaspersoft-studio

[5] https://grafana.com/

[6] https://mobaxterm.mobatek.net/

[7] https://www.docker.com/

[8] https://www.docker.com/

[9] https://staruml.io/

[10] https://www.atlassian.com/software/jira

[11] https://www.angularjswiki.com/angular/angular-material-icons-list-mat-icon-list/

[12] https://www.npmjs.com/

[13] https://developer.mozilla.org/en-US/docs/Web/HTML

[14] https://www.w3schools.com/css/

[15] https://getbootstrap.com/

[16] https://nodejs.org/en/

[17] https://npm/view

[18] https://typescript/1

[19] https://fr.wikipedia.org/wiki/Angulars.

[20] https://spring.io/projects/spring-boot

[21] https://fr.wikipedia.org/wiki/Selenium_(informatique)

[22] https://fr.wikipedia.org/wiki/Node.js

[23] https://fr.wikipedia.org/wiki/JUnit.js

[24] https://en.wikipedia.org/wiki/Continuous_integration

[25] https://www.padok.fr/blog/devops-deploiement-continu

[26] https://www.tecmint.com/understand-linux-load-averages-and-monitor-performance

Etude et Développement des nouvelles fonctionnalités pour l'ERP SAPTUN

Emna AYEDI

الخلاصة : يأتي هذا العمل في إطار مشروع نهاية الدراسات للحصول على دبلوم مهندس وطني في هندسة الحاسب المنفذ داخل شركة "سوفتودو ."الهدف من هذا المشروع هو دراسة وتطوير وظائف جديدة لـ في الواقع ، خلال دورة حياته ، يمر هذا البرنامج SAPTUN ERP. عن طريق إنشاء حاويات مع SAPTUN نشر بعدة مراحل من التطوير والتطور بفضل الأساليب الرشيقة التي اعتمدها فريقه .نتيجة لذلك ، يتم إضافة وظائف العمل واختبارها مع تطور احتياجات الشريك .وبالتالي ، تم تحسين البرنامج وجميع تبعياته .أخيرًا ، يتم تقديم مراقبة التطبيق عامل إرساء يشمل والخادم لمراقبة تقدم النشر مع تنشيط الإخطارات من أجل الحصول على ملاحظات سريعة في حالة حدوث مشكلة

Résumé Ce travail s'inscrit dans le cadre de projet de fin d'études pour l'obtention du diplôme National d'Ingénieur en génie informatique réalisé au sein de la société « SOFTODO ». L'objectif de ce projet consiste à étudier et développer des nouvelles fonctionnalités pour l'ERP SAPTUN. En effet, durant son cycle de vie, ce logiciel passe par plusieurs étapes de développement et évolution grâce aux méthodes agiles adoptées par son équipe. De ce fait, des fonctionnalités métiers sont ajoutées et testées au fur et à mesure de l'évolution des besoins du partenaire. Ainsi, le déploiement de l'ERP SAPTUN est optimisé en créant des conteneurs avec docker englobant le programme et tous ses dépendances. Enfin, une supervision de l'application et du serveur est introduite pour suivre le bon déroulement du déploiement avec l'activation des notifications afin d'avoir un feedback rapide en cas de problème.

Abstract: This work is part of the end of studies project for obtaining the National Engineer diploma in computer engineering carried out within the company "SOFTODO". The objective of this project is to study and develop new functionalities for the SAPTUN ERP. Indeed, during its life cycle, this software goes through several stages of development and evolution thanks to the agile methods adopted by its team. As a result, business functionalities are added and tested as the needs of the partner evolve. Thus, the deployment of the SAPTUN ERP is optimized by creating containers with docker encompassing the program and all its dependencies. Finally, monitoring of the application and the server is introduced to monitor the progress of the deployment with the activation of notifications in order to have rapid feedback in the event of a problem.

المفاتيح :

التكامل المستمر الانتشار المستمر DevOps تخطيط موارد المؤسسات ؛ رشيق؛ تطوير؛ اختبار الوحدة تعيين ؛ عامل ميناء؛ إشراف؛

Mots clés : ERP ; Agile; Développement ; Test Unitaire ; Déploiement ; Docker ; Supervision ; DevOps ; Intégration continue ; Déploiement continue;

Key-words : ERP ; Agile; Development ; Unit Test ; Deploiement ; Docker ; Supervision ; DevOps ; Continuous Integration; Continuous Deployement;