RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira – Béjaïa

Faculté des Sciences Exactes

Département d'Informatique

Option – Génie Logiciel

PROJET DE FIN DE CYCLE

En vue de l'obtention du diplôme de Master en Génie Logiciel

Thème

Conception et Développement d'un Système de Gestion des Centres de Livraison Régionaux, incluant la Planification des Commandes Clients et la Prédiction Quantitative

Réalisé par :
Mlle:MEZNAD Imene
Mlle:MASSIOUN Celia

Encadré par :
M.TOUAZI Djoudi
M.SALHI Nadir
M.BENNAI Brahim

Devant le jury composé de :

Mme.	**HAMZA Lamia**	- Président	(Univ. A/Mira Bejaia)
M.	**AKILAL Karim**	- Examinateur	(Univ. A/Mira Bejaia)
Mme.	**YESSAD Nawel**	- Examinateur	(Univ. A/Mira Bejaia)
Mme.	**TASOULT Nadia**	- Examinateur	(Univ. A/Mira Bejaia)

Année Universitaire 2024 – 2025

Dédicace

En amour, et en reconnaissance, je dédie ce travail :

À la mémoire de mes très chers parents, **mon père et ma mère**,
pour leur amour éternel, leurs sacrifices et leurs prières.

Que Dieu les accueille dans Son vaste paradis.

À mes sœurs : **Myriam, Samira, Zoulikha, Souhila, Fouzia et Radia**,
pour leur soutien, leur tendresse et leur présence constante.

À mon frère jumeau **Mourad**,
mon double, mon repère, pour sa confiance inébranlable. À mes amis les plus sincères, dont la présence et le soutien ont embelli ce parcours. À tous ceux qui me sont chers, de près ou de loin, et à tous ceux qui ont contribué à la réalisation de ce travail,
je vous exprime ma profonde gratitude.

Massioun Celia

En signe de reconnaissance et d'amour, je dédie ce travail à :

À celle qui a illuminé ma vie par son sourire, qui m'a donné la force de continuer dans les moments difficiles et qui reste ma source d'inspiration , ma très chère **maman**
À l'homme de ma vie, mon pilier et celui qui m'a enseigné la persévérance mon cher **papa**
À mon âme sœur, celle qui partage mes joies et mes peines, ma meilleure amie, ma précieuse sœur **Lydia**
À mon petit frère **Zakaria**, pour sa présence bienveillante et à mon adorable chat **Soucicel**, compagnon fidèle de mes journées studieuses
À ma grande famille, mes grands-parents, mes oncles, mes tantes et mes cousins, pour leur chaleur et leur bienveillance
À mes amis sincères, qui ont rendu ce parcours plus beau

Puisse Allah vous protéger et vous combler de Ses bienfaits.

Meznad Imene

Remerciements

Nous exprimons, avant tout, notre profonde gratitude envers le Tout-Puissant, pour nous avoir guidées, soutenues et inspirées tout au long de la réalisation de ce travail.

Nos remerciements les plus sincères s'adressent à :

- M. TOUZI Djoudi, notre encadrant universitaire, pour son accompagnement rigoureux, ses conseils éclairés, sa bienveillance et sa disponibilité continue tout au long de ce projet.

- M. SALHI Nadir notre co-encadrant pour sa présence, son soutien et ses contributions précieuses.

– M. BENNAI Brahim, qui nous a encadrés durant notre stage, pour son accueil chaleureux, sa disponibilité ainsi que le partage de son savoir-faire, qui ont grandement enrichi notre réflexion et notre démarche.

– M. KHERBACHI Samy, pour ses conseils avisés, son aide précieuse et son soutien tout au long de cette expérience.

– M. KESSIRI Allili, pour ses conseils pertinents et ses explications claires des problématiques liées à notre thème, qui nous ont permis de mieux comprendre les enjeux du sujet.

– Ainsi que l'ensemble de l'équipe Supply Chain et du service SI de CEVITAL, pour leur accueil et leur collaboration.

Nous adressons également nos remerciements respectueux aux membres du jury — Mme Lamia HAMZA, M. Karim AKILAL, Mme Nawel YESSAD et Mme Nadia TASOULT — pour l'honneur qu'ils nous font en acceptant d'évaluer ce travail, et pour leurs observations pertinentes qui contribueront sans nul doute à notre progression.

Nos pensées les plus reconnaissantes vont à nos familles, pour leur soutien moral indéfectible, leurs encouragements constants et leur patience tout au long de cette période exigeante.

Enfin, nous remercions toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire.

Table des matières

Introduction Générale **1**

I Présentation de l'organisme & Étude de l'existant **3**
I.1 Introduction . . . 4
I.2 Présentation de l'organisme d'accueil . . . 4
I.2.1 Présentation du Groupe Cevital . . . 4
I.2.2 Presentation de Cevital Agro-Industrie . . . 4
I.2.3 Presentation de cadre d'etude . . . 5
I.3 Etude de l'existant . . . 6
I.3.1 Présentation du domaine de travail . . . 6
I.3.2 Problématique . . . 7
I.3.3 Impact global sur l'entreprise . . . 7
I.4 Proposition de solution améliorée . . . 7
I.4.1 Planification et suivi des livraisons . . . 8
I.5 Conclusion . . . 9

II Approche Méthodologique & Choix Technologiques **10**
II.1 Introduction . . . 11
II.2 Le processus de développement logiciel : . . . 11
II.2.1 Le Framework Scrum . . . 11
II.2.2 Le Processus Unifié (UP) . . . 11
II.2.3 L'association de Scrum et du Processus Unifié (UP) . . . 12
II.3 Langage de modélisation . . . 13
II.4 Outils utilisés dans le projet . . . 13
II.4.1 Éditeur de code . . . 13
II.4.2 Langages & bibliothèques . . . 13
II.4.3 Frontend . . . 14
II.4.4 Backend . . . 14
II.4.5 Base de données & ORM . . . 15
II.4.6 Sécurité & authentification . . . 15
II.4.7 Modélisation et style . . . 15
II.4.8 Outils de test et API . . . 16
II.4.9 Outils de prédiction . . . 16
II.4.10 Le modèle MVC (Model-View-Controller) . . . 16
II.5 Conclusion . . . 16

III Sprint zéro **17**
III.1 Introduction . . . 18
III.2 Spécification des besoins . . . 18
III.2.1 Identification des acteurs . . . 18
III.2.2 Réalisation du diagramme de contexte dynamique . . . 18
III.2.3 Collecte des besoins fonctionnels . . . 20
III.2.4 Diagramme de cas d'utilisation . . . 21

III.2.5 Collecte des besoins non fonctionnels . . . 22
III.3 Pilotage du projet avec Scrum . . . 23
III.3.1 Équipe Scrum . . . 23
III.3.2 Product Backlog . . . 23
III.3.3 Structure et découpage du projet . . . 25
III.4 Charte graphique . . . 26
III.4.1 Présentation du logo de l'application web . . . 26
III.4.2 Palette des couleurs . . . 27
III.5 Conclusion . . . 27

IV Sprint UN **28**
IV.1 Introduction . . . 29
IV.2 Backlog du Sprint 1 . . . 29
IV.3 Spécification . . . 30
IV.3.1 Diagramme de cas d'utilisation - Sprint 1 . . . 30
IV.3.2 Description textuelle des cas d'utilisation - Sprint 1 . . . 31
IV.4 Analyse . . . 33
IV.4.1 diagramme d'activité - Sprint 1 . . . 33
IV.4.2 Diagramme de séquence système - Sprint 1 . . . 35
IV.5 Conception . . . 36
IV.5.1 diagramme d'activité Authentification - Sprint 1 . . . 37
IV.5.2 Le diagramme de séquence détaillé - Sprint 1 . . . 37
IV.5.3 Diagramme de classes - Sprint 1 . . . 39
IV.6 Implementation . . . 40
IV.6.1 Dictionnaire de données - Sprint 1 . . . 40
IV.6.2 Modèle relationnel de données - Sprint 1 . . . 41
IV.7 Test & presentation d'interfaces . . . 42
IV.8 Bilan du Sprint 1 . . . 43
IV.9 Conclusion . . . 43

V Sprint deux **44**
V.1 Introduction . . . 45
V.2 Backlog du Sprint 2 . . . 45
V.3 Spécification . . . 47
V.3.1 Diagramme de cas d'utilisation - Sprint 2 . . . 47
V.3.2 Description textuelle des cas d'utilisation - Sprint 2 . . . 48
V.4 Analyse . . . 52
V.4.1 Analyse du stock d'alerte et du stock maximum des articles dans chaque dépôt. . . . 52
V.4.2 diagramme d'activité - Sprint 2 . . . 52
V.4.3 Les diagrammes de séquence système - Sprint 2 . . . 54
V.5 Conception . . . 55
V.5.1 Les diagrammes de séquence détaillés - Sprint 2 . . . 55
V.5.2 Diagramme de classes - Sprint 2 . . . 57
V.6 Implementation . . . 57
V.6.1 Dictionnaire de données - Sprint 2 . . . 57
V.6.2 Modèle relationnel de données - Sprint 2 . . . 60

V.7 Test & presentation d'interfaces . . . 61
V.8 Bilan du Sprint 2 . . . 61
V.9 Conclusion . . . 61

VI Sprint trois **62**
VI.1 Backlog du Sprint 3 . . . 63
VI.2 Spécification . . . 64
VI.2.1 Diagramme de cas d'utilisation - Sprint 3 . . . 64
VI.2.2 Description textuelle des cas d'utilisation - Sprint 3 . . . 65
VI.3 Analyse . . . 69
VI.3.1 diagramme d'activité - Sprint 3 . . . 69
VI.3.2 Les daiagrammes de séquence système - Sprint 3 . . . 70
VI.4 Conception . . . 71
VI.4.1 Les diagrammes de séquence détaillés - Sprint 3 . . . 71
VI.4.2 Diagramme de classes - Sprint 3 . . . 75
VI.5 Implementation . . . 76
VI.5.1 Dictionnaire de données - Sprint 3 . . . 76
VI.5.2 Modèle relationnel de données - Sprint 3 . . . 78
VI.6 Test & presentation d'interfaces . . . 78
VI.7 Bilan du Sprint 3 . . . 79
VI.8 Conclusion . . . 80

VII Sprint quatre **81**
VII.1 Introduction . . . 82
VII.2 Backlog du Sprint 4 . . . 82
VII.3 Spécification . . . 83
VII.3.1 Diagramme de cas d'utilisation - Sprint 4 . . . 83
VII.3.2 Description textuelle des cas d'utilisation - Sprint 4 . . . 83
VII.4 Analyse . . . 84
VII.5 Conception . . . 85
VII.5.1 Les diagrammes de séquence détaillés - Sprint 4 . . . 85
VII.5.2 Diagramme de classes - Sprint 4 . . . 86
VII.6 Implementation . . . 86
VII.6.1 Modèle relationnel de données - Sprint 4 . . . 86
VII.7 Test & presentation d'interfaces . . . 87
VII.8 Bilan du Sprint 4 . . . 88
VII.9 Conclusion . . . 88

Conclusion Générale **89**

Annexes **90**
Protection des routes selon les rôles . . . 90
Fichier de commandes importé . . . 90
Bon de livraison généré . . . 91

Table des figures

I.1 Organigramme géneral de Cevital Agro-Industrie 5
II.1 Processus Scrum 11
II.2 Processus Unifié(UP) 12
II.3 Logo UML 13
II.4 Logo VS Code 13
II.5 Logo JavaScript 13
II.6 Logo JSX 14
II.7 Logo React 14
II.8 Logo Tailwind 14
II.9 Logo Node.js 14
II.10 Logo Express 14
II.11 Logo Sequelize 15
II.12 Logo PostgreSQL 15
II.13 Logo JWT 15
II.14 Logo Draw.io 15
II.15 Logo Canva 15
II.16 Logo Postman 16
II.17 Logo python 16
II.18 Logo SARIMA 16
III.1 Diagramme de contexte dynamique 19
III.2 Diagramme de cas d'utilisation global 22
III.3 L'équipe Scrum de notre projet 23
III.4 Logo de StockFlow 26
III.5 Palette des couleurs 27
IV.1 Diagramme cas d'utilisation du SPRINT 1 31
IV.2 diagramme d'activité - Sprint 1 34
IV.3 Diagramme de séquence système "Authentification" 35
IV.4 Diagramme de séquence système "Ajouter un utilisateur" 36
IV.5 diagramme d'activité - Authentification 37
IV.6 Diagramme de séquence détaillé "Authentification" 38
IV.7 Diagramme de séquence détaillé "Ajouter utilisateur" 39
IV.8 Diagramme de classes - Sprint 1 40
IV.9 Login page 42
IV.10page Afficher les utilisateurs 43
V.1 Diagramme de cas d'utilisation – Sprint 2 48
V.2 diagramme d'activité - Sprint 2 53
V.3 Diagramme de séquence système "Affecter un utilisateur à un dépôt" 54
V.4 Diagramme de séquence détaillé " Gérer les entrées" 55
V.5 Diagramme de séquence détaillé "Affecter utilisateurs aux dépôts" 56
V.6 Diagramme de classes - Sprint 2 57
V.7 Interface d'affectation des utilisateurs aux dépôts 61
VI.1 Diagramme de cas d'utilisation – Sprint 3 65
VI.2 diagramme d'activité - Sprint 3 70
VI.3 diagramme séquence système - Gérer les livraisons 71

VI.4 Diagramme de séquence détaillé "Planifier commande" 73
VI.5 Diagramme de séquence détaillé "Importer commandes" 74
VI.6 diagramme de classe SPRINT 3 et Final 75
VI.7 Interface de comparaison des quantités 79
VI.8 Interface de planification des livraisons 79
VII.1Diagramme de cas d'utilisation - Sprint 4 83
VII.2Diagrammes de séquence détaillé "Consulter les prédictions" 86
VII.3Interface de prédiction 88
4 Protection des routes selon les rôles dans le backend 90
5 Exemple de fichier de commandes clients fourni par CEVITAL 91
6 Exemple de bon de livraison fourni par CEVITAL 91

Liste des tableaux

III.1 Messages émis et reçus par le système 19
III.2 Tableau des besoins fonctionnels 20
III.3 Planification des sprints 24
III.4 Répartition chronologique des sprints 26
IV.1 Backlog Sprint 1 - Description détaillée des tâches 29
IV.2 Règles de gestion globale 31
IV.3 Description textuelle du cas : Authentification 31
IV.4 Description textuelle du cas : Gérer les utilisateurs 32
IV.5 Table : Utilisateur 40
V.1 Backlog Sprint 2 - Description détaillée des tâches 45
V.2 Règles de gestion du Sprint 2 49
V.3 Description textuelle du cas : Affectation 50
V.4 Description textuelle du cas : Enregistrer une entrée multiple 51
V.5 Dictionnaire de données de Sprint 2 57
VI.1 Backlog Sprint 3 - Description détaillée des tâches 63
VI.2 Règles de gestion globale – Sprint 3 65
VI.3 Description textuelle du cas : Gérer les commandes. 66
VI.4 Description textuelle du cas : Gérer le réapprovisionnement. 67
VI.5 Description textuelle du cas : Gérer les livraisons. 68
VI.6 Dictionnaire de données des tables – Commandes et Livraisons 76
VII.1Backlog Sprint 4 - Description synthétique des tâches 82
VII.2Règles de gestion essentielles – Sprint 4 84
VII.3Description textuelle du cas : Prédire les besoins. 84

Liste des acronymes

ADB	Administrateur de Base de Données
API	Application Programming Interface
BDD	Base de Données
CEVITAL	Groupe industriel algérien multisectoriel
CLR	Centre de Livraison Régional
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DSI	Direction des Systèmes d'Information
ID	Identifiant
IDE	Integrated Development Environment
JS	JavaScript
JSX	JavaScript XML
JWT	JSON Web Token
MAX	Maximum
MIN	Minimum
ORM	Object-Relational Mapping
SI	Système d'Information
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language
UP	Unified Process
U.S	User Story
UX	User Experience
VS Code	Visual Studio Code

Introduction Générale

Le secteur de la gestion des dépôts occupe une place centrale dans la performance logistique des grandes entreprises industrielles. Un système efficace de gestion des stocks, des livraisons, des réapprovisionnements et des flux de marchandises permet non seulement d'optimiser les coûts, mais aussi de garantir une meilleure satisfaction client.

Dans ce contexte, le groupe **CEVITAL**, l'un des plus grands groupes industriels privés en Algérie et un acteur majeur dans plusieurs secteurs stratégiques (agroalimentaire, distribution, logistique, etc.), accorde une grande importance à la modernisation de ses processus logistiques par la digitalisation. La gestion des dépôts représente ainsi un enjeu critique pour assurer la fluidité et la traçabilité des produits entre les différentes entités du groupe.

Cependant, la complexité des opérations au sein des Centres de Livraison Régionaux (CLR) de CEVITAL et la multiplicité des acteurs impliqués soulèvent plusieurs problématiques. La gestion actuelle des stocks repose en grande partie sur des méthodes manuelles et non automatisées, ce qui engendre un manque de réactivité, une visibilité réduite sur les niveaux de stock en temps réel, ainsi qu'une difficulté à anticiper précisément les besoins en approvisionnement.

Ces faiblesses opérationnelles augmentent les risques de rupture ou de surstock, nuisent à l'efficacité logistique et peuvent impacter négativement la satisfaction des clients, ainsi que la performance globale de la chaîne d'approvisionnement.

Pour répondre à ces défis, ce mémoire propose la conception et la mise en œuvre d'un **système intelligent de gestion des dépôts**, basé sur une démarche combinant le **Processus Unifié (UP)** et une approche **agile (Scrum)**. Cette combinaison permet d'assurer à la fois une modélisation structurée et progressive du système et une livraison itérative et incrémentale des fonctionnalités. Le projet est structuré en plusieurs **sprints**, chacun représentant une étape du développement orientée vers un objectif précis.

L'objectif est de développer une solution modulaire et évolutive couvrant la gestion des utilisateurs, des articles, des clients, des dépôts, des véhicules liés aux différents dépôts pour les livraisons, ainsi que des commandes et du réapprovisionnement. , tout en intégrant à terme une composante d'intelligence artificielle pour anticiper les besoins.

Ce mémoire est structuré en sept chapitres :

— **Chapitre 1 : Présentation de l'organisme d'accueil et de la problématique**
Ce chapitre introduit le groupe CEVITAL et le service concerné. Il présente également la problématique identifiée au niveau de la gestion des dépôts et expose la solution proposée dans le cadre de ce projet.

— **Chapitre 2 : Outils, langages et processus utilisés**
Ce chapitre détaille les environnements de développement, les langages de programmation, les frameworks, ainsi que les méthodes de gestion de projet adoptées : le Processus Unifié et Scrum.

— **Chapitre 3 : Sprint 0 – Analyse des besoins fonctionnels**
Ce chapitre est consacré à l'identification des besoins fonctionnels du système

— **Chapitre 4 : Sprint 1 – Authentification et gestion des utilisateurs**
Ce chapitre décrit le développement de la partie authentification, ainsi que la gestion des utilisateurs selon leurs rôles (admin fonctionnel, admin dépôt, planificateur.).

— **Chapitre 5 : Sprint 2 – Gestion des entités principales**

Ce chapitre présente la mise en œuvre des modules de gestion des clients, des dépôts, des articles, des familles et sous-familles, des véhicules, ainsi que l'enregistrement des entrées.

— **Chapitre 6 : Sprint 3 – Gestion des commandes, livraisons et réapprovisionnement**
Ce chapitre couvre la gestion des commandes clients, leur planification, les bons de livraison et de retour, ainsi que les commandes de réapprovisionnement des dépôts.

— **Chapitre 7 : Sprint 4 – Prédiction des besoins et amélioration du réapprovisionnement**
Ce dernier chapitre introduit une approche basée sur l'intelligence artificielle pour prédire les besoins futurs en fonction de l'historique des commandes et améliorer la stratégie de réapprovisionnement.

Ce projet vise ainsi à améliorer significativement la performance logistique de CEVITAL en apportant une solution numérique intelligente, adaptée aux défis actuels de la gestion des dépôts.

Chapitre I

Présentation de l'organisme & Étude de l'existant

I.1 Introduction

Dans un contexte mondial en mutation, le développement économique repose sur des entreprises performantes et innovantes. En Algérie, le groupe CEVITAL incarne cette dynamique grâce à une stratégie d'expansion et une diversification réussie.

Notre stage de fin d'études s'est déroulé au sein de CEVITAL Agro-Industrie, filiale spécialisée dans la transformation agroalimentaire, encadré par la Direction des Systèmes d'Information (DSI), équipe Supply Chain. Le projet a porté sur la gestion des stocks et des approvisionnements dans un environnement industriel exigeant. Nous avons opté pour une étude de l'existant, visant à identifier les problèmes rencontrés, puis à proposer des solutions numériques adaptées aux besoins de l'entreprise. Avant d'aborder ces aspects, il est essentiel de présenter le cadre de travail, ses missions et ses enjeux stratégiques.

I.2 Présentation de l'organisme d'accueil

I.2.1 Présentation du Groupe Cevital

Fondé en 1998 par Issad Rebrab, ce premier groupe privé en Algérie est un acteur majeur de l'industrie. Cevital affirme qu'il est un acteur de premier plan dans différents domaines tels que l'agroalimentaire, les équipements électroménagers, la sidérurgie, l'industrie du verre, la distribution et la logistique. Grâce à une approche stratégique et des investissements ambitieux, il a connu une expansion constante qui s'est étendue sur plusieurs continents. Dotée de plus de 18 000 employés, CEVITAL reflète l'énergie économique algérienne et participe de manière active à la création d'emplois et à la génération de richesse, renforçant ainsi sa présence sur les scènes nationale et internationale [1].

I.2.2 Presentation de Cevital Agro-Industrie

Implantée au port de Béjaïa, Cevital Agro-Industrie est la branche spécialisée du groupe dans la transformation agroalimentaire. Dotée d'installations industrielles modernes, elle produit du sucre, des huiles, des boissons et autres produits de grande consommation. Elle répond aux besoins du marché national tout en développant une présence à l'international, notamment en Europe, au Maghreb et en Afrique de l'Ouest.

l'organisation de Cevital Agro-Industrie

L'organisation de Cevital Agro-Industrie repose sur la mobilisation des ressources humaines, matérielles et financière pour atteindre ses objectifs. La direction générale est constituée d'un secrétariat et de 19 directions principales :

L'organigramme ci-dessous présente les principales directions de Cevital Agro-Industrie, chacune contribuant de manière essentielle à la performance et à la compétitivité de l'entreprise :

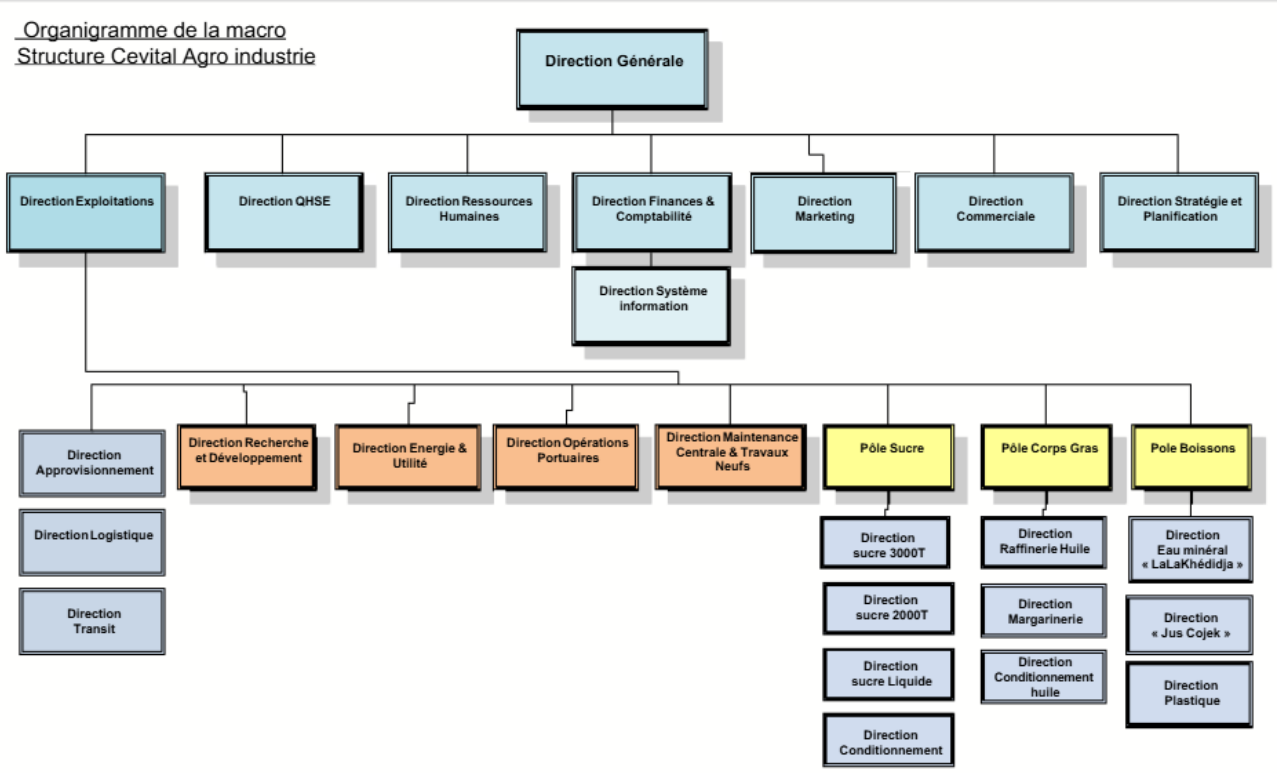

FIGURE I.1 – Organigramme géneral de Cevital Agro-Industrie

I.2.3 Presentation de cadre d'etude

Dans le cadre de ce stage, l'étude est réalisée au sein du département SI Supply Chain, rattaché à la Direction des Systèmes d'Information (DSI) de CEVITAL. Ce département occupe un rôle stratégique en assurant la mise en œuvre, le développement et la gestion des solutions informatiques dédiées à l'optimisation des processus logistiques et d'approvisionnement de l'entreprise.

Les départements de la DSI et leur contribution

La DSI de CEVITAL est organisée en plusieurs départements, chacun ayant un rôle spécifique dans la gestion et l'optimisation des systèmes d'information de l'entreprise. Parmi eux, on distingue notamment le Département Applications Métiers, le Département Technique, le Département Transformation Digitale et Business Intelligence, ainsi que le Département Sécurité SI. Cependant, notre étude s'intéresse particulièrement au Département SI Supply Chain.

Département SI Supply Chain

Ce département est chargé du développement et de la gestion des solutions informatiques permettant d'optimiser plusieurs processus clés, notamment :

- la gestion des dépôts,
- la gestion du réapprovisionnement,

- la gestion des livraisons,

Grâce à ces solutions, le Département SI Supply Chain contribue à l'optimisation des opérations logistiques et de distribution, en améliorant la gestion des stocks, la fluidité des approvisionnements et la réactivité de la chaîne d'approvisionnement.

I.3 Etude de l'existant

I.3.1 Présentation du domaine de travail

Le domaine de travail concerné par cette étude est la gestion de stock Les Centres de Livraison Régionaux (CLR) sont des infrastructures stratégiques qui assurent la distribution des produits de CEVITAL vers les différents points de vente. Cette activité est essentielle pour assurer la disponibilité des produits finis, tout en évitant les ruptures ou les surstocks qui peuvent impacter la performance globale de la chaîne logistique.

La gestion de stock comprend plusieurs volets clés tels que :

- Le suivi des niveaux de stock en temps réel .
- Le contrôle des entrées et sorties de marchandises .
- Les suggestions de réapprovisionnement automatisées, basées sur les seuils définis et l'historique des consommations.
- L'analyse des mouvements de stock pour la prise de décision.

La bonne gestion de stock contribue directement à la réduction des coûts logistiques, à l'amélioration du service client et à la fluidité des opérations d'approvisionnement et de distribution.

Systèmes actuels, outils et méthodes

Actuellement, la communication entre les planificateurs des dépôts repose essentiellement sur des échanges oraux ou téléphoniques pour coordonner la prise, la livraison des produits et le réapprovisionnement. Ce mode de fonctionnement manuel expose le processus à des risques fréquents d'erreurs, de malentendus et de pertes d'informations.

Chaque matin, les planificateurs et administrateurs des dépôts procèdent à un échange de listes de niveaux de stock sous format papier ou fichier excel, une tâche répétitive et contraignante, peu adaptée aux exigences d'efficacité et de réactivité d'une grande entreprise comme CEVITAL.

De plus, chaque soir, le service transport procède à l'affectation des véhicules en fonction de leur disponibilité, et s'efforce d'identifier à l'avance les créneaux d'indisponibilité, afin de mieux organiser les livraisons à planifier.

Par ailleurs, les prévisions de stock sont réalisées de manière empirique, principalement en fonction des saisons et de l'expérience des planificateurs. Cette approche manuelle augmente

le risque d'erreurs d'estimation, pouvant entraîner soit des ruptures de stock, soit des surplus coûteux à gérer.

I.3.2 Problématique

- La gestion actuelle des stocks au sein des Centres de Livraison Régionaux (CLR) de CEVITAL repose majoritairement sur des méthodes manuelles et non automatisées.

- Cette approche entraîne :
  — un manque de réactivité,
  — une visibilité réduite sur les niveaux de stock en temps réel,
  — ainsi qu'une difficulté à anticiper avec précision les besoins en approvisionnement,
  — ainsi qu'une complexité accrue à planifier efficacement les livraisons.

- Ces faiblesses augmentent :
  — les risques de rupture ou de surstock,
  — nuisent à l'efficacité logistique,
  — et peuvent impacter la satisfaction des clients et la performance globale de la chaîne d'approvisionnement.

I.3.3 Impact global sur l'entreprise

L'absence d'un système d'information performant dans la gestion de la supply chain a plusieurs répercussions négatives sur l'ensemble de l'organisation :

- **Financier** : Augmentation des coûts opérationnels due aux erreurs humaines, aux retards de traitement, et à une perte globale de rentabilité.

- **Opérationnel** : Les délais de traitement peuvent être multipliés par 2 à 3 par rapport à un système automatisé, ce qui nuit à l'efficacité globale.

- **Client** : Diminution du taux de satisfaction client, en raison des retards de livraison et de l'indisponibilité fréquente des produits.

- **Concurrence** : Risque accru de perte de parts de marché au profit de concurrents mieux digitalisés et technologiquement avancés.

I.4 Proposition de solution améliorée

Pour optimiser la gestion des stocks et améliorer la coordination entre les différents dépôts, la solution suivante est proposée :

Gestion centralisée des dépôts, articles, clients, utilisateurs et vehivules

- Une **gestion efficace** des dépôts et des articles qui y sont associés ;

- L'**affectation structurée des clients et vehicules** à chaque dépôt, facilitant la répartition des commandes et leur suivi ;
- L'**administration des utilisateurs responsables** (Planificateurs dépôts, Admin dépôts, Admins fonctionnels.), avec la possibilité de suivre leurs tâches et interventions ;
- La **conservation d'un historique complet** des opérations (mouvements de stock, livraisons, actions des utilisateurs, etc.), essentielle pour assurer la traçabilité et faciliter la prise de décision ;
- Une **recherche simplifiée** dans les données historiques, garantissant un accès rapide et efficace aux informations clés ;
- Une **interface intuitive** et conviviale, permettant une gestion fluide et performante au quotidien.

Système de gestion automatisé

Un système informatisé assurant la **mise à jour automatique** des niveaux de stock à chaque entrée ou sortie de marchandise, réduisant les erreurs humaines et garantissant une traçabilité en temps réel.

Suggestions automatiques de réapprovisionnement

Le système génère automatiquement des recommandations de commandes lorsque les niveaux de stock atteignent un seuil critique, aidant ainsi les gestionnaires à décider quand et quoi réapprovisionner afin de garantir une disponibilité continue des produits.

Interface de suivi en temps réel

Mise en place d'un **tableau de bord interactif et centralisé**, accessible aux gestionnaires et administrateurs, offrant :

- Une **visualisation claire** des niveaux de stock par dépôt ;
- Des **alertes instantanées** en cas de rupture ou de risque de pénurie.

I.4.1 Planification et suivi des livraisons

Le système facilitera également la **planification des livraisons** et le **suivi en temps réel** de l'acheminement des commandes, contribuant à une **organisation logistique optimisée** et à une **satisfaction client accrue**.

I.5 Conclusion

La présentation de CEVITAL et de sa filiale Agro-Industrie met en lumière son rôle clé dans l'économie nationale, notamment en matière de souveraineté alimentaire et d'innovation.

Ce cadre professionnel structuré nous a permis de mieux comprendre les enjeux de la gestion des stocks et des approvisionnements, point de départ de notre projet, qui vise à analyser les pratiques actuelles et proposer une solution numérique adaptée.

Chapitre II

Approche Méthodologique & Choix Technologiques

II.1 Introduction

Les méthodes agiles, axées sur la collaboration, la communication et l'adaptabilité, favorisent un développement itératif et une livraison rapide. Elles permettent de mieux répondre besoins changeants des clients et d'améliorer la qualité du produit final. Dans le cadre de notre projet, l'adoption d'une méthode agile — en particulier Scrum — constitue un choix stratégique pour en assurer l'efficacité. Ce chapitre présente cette approche ainsi que les outils de modélisation et de développement utilisés.

II.2 Le processus de développement logiciel :

Il s'agit d'un ensemble d'étapes structurées de l'analyse des besoins jusqu'au déploiement , qui nous permettent de produire un logiciel fonctionnel et adapté aux attentes de l'utilisateur.

II.2.1 Le Framework Scrum

Scrum est une méthode agile qui aide les équipes à gérer efficacement les projets complexes. Elle s'appuie sur une structure claire : des rôles bien définis, des réunions régulières et des outils de suivi appelés artefacts. Cette approche favorise une organisation souple et une amélioration continue [2].
L'illustration ci-dessous synthétise de manière structurée les rôles et les cérémonies clés du cadre Scrum :

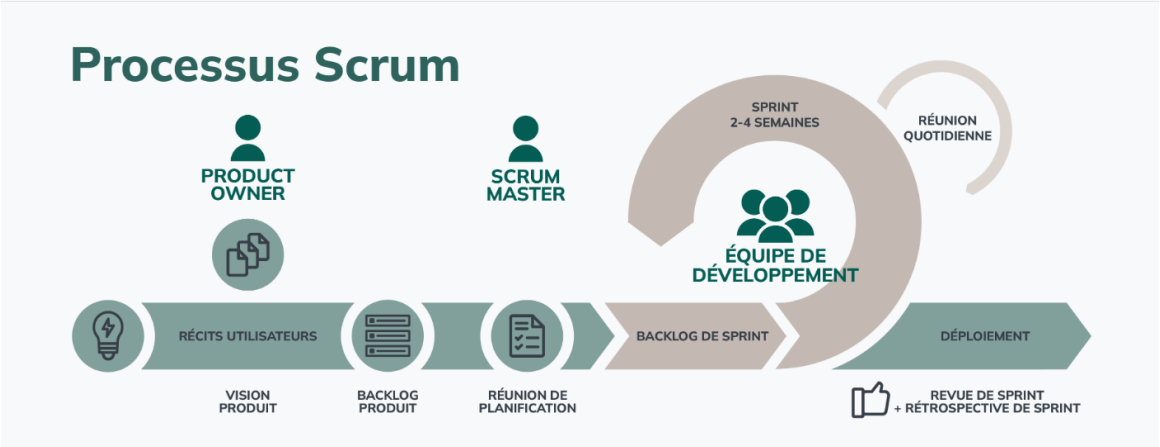

Figure II.1 – Processus Scrum

II.2.2 Le Processus Unifié (UP)

Le Processus Unifié est une approche méthodique qui s'adapte aux projets de toute taille. Il permet de structurer l'organisation du développement logiciel en définissant clairement les tâches et les responsabilités au sein de l'équipe. [3].

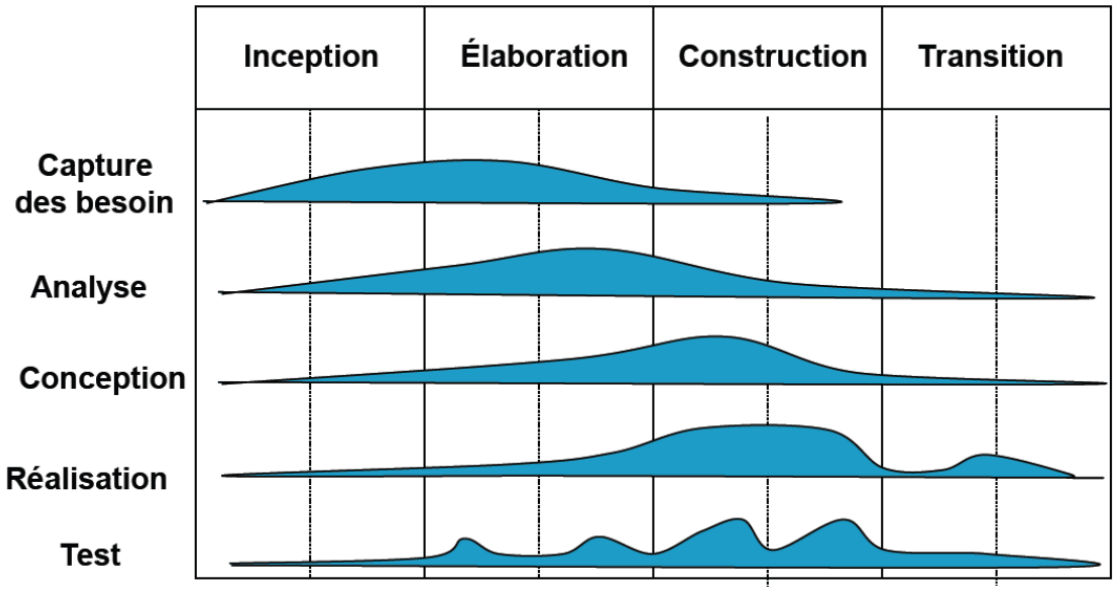

FIGURE II.2 – Processus Unifié(UP)

Activités du Processus Unifié

- **Spécification :** Cette étape consiste à élaborer le diagramme de cas d'utilisation pour le sprint en cours, accompagné d'une description textuelle. Elle permet d'identifier les interactions entre les acteurs et le système, posant ainsi les fondations fonctionnelles du sprint.

- **Analyse :** Elle inclut la réalisation d'un diagramme d'activité global ainsi que des diagrammes de séquence pour les cas d'utilisation clés. Cette phase aide à modéliser le comportement dynamique du système.

- **Conception :** À ce stade, nous modélisons les processus internes via des diagrammes d'activités et de séquence plus détaillés. Un diagramme de classes est également produit pour représenter les entités manipulées durant le sprint.

- **Implémentation :** Elle repose sur la création du dictionnaire de données et du modèle relationnel correspondant. Cette phase assure la cohérence entre les modèles conceptuels et la structure de la base de données.

- **Test :** Enfin, cette phase permet de valider les éléments réalisés par des relectures, des vérifications croisées et des ajustements. Les livrables sont confirmés avant le passage au développement technique.

II.2.3 L'association de Scrum et du Processus Unifié (UP)

Dans notre projet, nous avons choisi de combiner les avantages du Processus Unifié (UP) et de Scrum. Le cadre structuré du UP nous a permis de planifier clairement les différentes phases du développement (spécification, analyse, conception, etc.). En parallèle, Scrum nous a offert

une approche itérative et adaptative, idéale pour intégrer les retours du client à chaque sprint. Cette complémentarité a renforcé notre capacité à suivre l'avancement du projet, à répondre rapidement aux changements et à mieux gérer les exigences fonctionnelles et techniques[4].

II.3 Langage de modélisation

FIGURE II.3 – Logo UML

Le langage UML est un standard largement adopté pour modéliser les systèmes logiciels orientés objet. Dans le cadre de notre projet, qui repose sur des technologies comme JavaScript, React et ExpressJS, l'utilisation d'UML s'est révélée pertinente. Grâce à ses différents types de diagrammes (cas d'utilisation, séquence, activités, classes), UML nous a permis de représenter de manière claire et structurée les fonctionnalités du système, telles que la gestion des stocks, les transferts. Il a également facilité la conception des entités de la base de données et amélioré la communication technique au sein de l'équipe de développement[5].

II.4 Outils utilisés dans le projet

II.4.1 Éditeur de code

VS Code : a été utilisé comme éditeur principal pour le développement du projet. Il a permis de gérer à la fois le frontend en ReactJS (création de composants, gestion du routage, stylisation) et le backend en Node.js avec Express et Sequelize (modèles, routes, contrôleurs). Son terminal intégré s'est révélé particulièrement pratique pour le lancement des serveurs et l'exécution des commandes.

FIGURE II.4 – Logo VS Code

II.4.2 Langages & bibliothèques

FIGURE II.5 – Logo JavaScript

JavaScript :a été le langage principal utilisé pour l'ensemble du projet, aussi bien côté client (ReactJS) que côté serveur (Node.js/Express).

JSX : une extension syntaxique , a permis de concevoir les interfaces utilisateur de façon déclarative, en intégrant le HTML directement dans le code JavaScript.

FIGURE II.6 – Logo JSX

II.4.3 Frontend

FIGURE II.7 – Logo React

React :a été utilisé pour construire l'interface utilisateur sous forme de composants dynamiques et réutilisables.

Tailwind CSS : un framework utilitaire de stylisation, a permis d'appliquer rapidement des styles directement dans le JSX, facilitant la cohérence visuelle de l'interface.

FIGURE II.8 – Logo Tailwind

II.4.4 Backend

FIGURE II.9 – Logo Node.js

Node.js :a été utilisé pour exécuter le code JavaScript côté serveur et assurer le traitement des requêtes backend.

Express :un framework minimaliste pour Node.js, a facilité la structuration des routes et la communication entre le frontend et notre base de données.

FIGURE II.10 – Logo Express

II.4.5 Base de données & ORM

FIGURE II.11 – Logo Sequelize

Sequelize : est un ORM utilisé pour gérer les interactions entre les modèles du projet et la base de données relationnelle.

PostgreSQL : PostgreSQL est un système de gestion de base de données relationnelle avancé et open-source.

FIGURE II.12 – Logo PostgreSQL

II.4.6 Sécurité & authentification

FIGURE II.13 – Logo JWT

JWT :a été utilisé pour sécuriser l'authentification des utilisateurs à travers un système de jetons d'accès signés.

II.4.7 Modélisation et style

Draw.io : a été l'outil principal pour la création de schémas UML (cas d'utilisation, séquence, activité, etc.).

FIGURE II.14 – Logo Draw.io

FIGURE II.15 – Logo Canva

Canva : est une plateforme en ligne utilisé pour créer les maquettes et le logo du projet.

II.4.8 Outils de test et API

Postman :a servi à tester les routes de l'API et vérifier les échanges de données de l'application.

FIGURE II.16 – Logo Postman

II.4.9 Outils de prédiction

FIGURE II.17 – Logo python

python : a été utilisé comme langage de programmation pour le développement et l'entraînement du modèle de prédiction SARIMA.

SARIMA : a été utilisé comme algorithme de machine learning pour la prédiction des besoins en réapprovisionnement, à partir des données historiques de consommation.

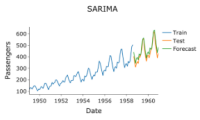

FIGURE II.18 – Logo SARIMA

II.4.10 Le modèle MVC (Model-View-Controller)

Dans notre application, nous suivons l'architecture MVC qui qui est composée de trois couches : Le Modèle gère les données, la Vue affiche l'interface, et le Contrôleur fait le lien entre eux.

II.5 Conclusion

Pour conclure, ce chapitre a permis de présenté l'approche Agile, en mettant l'accent sur le framework Scrum. Nous avons également détaillé les outils de modélisation (UML) et les technologies utilisées dans notre projet, tant pour le frontend que le backend. Ces choix méthodologiques et technologiques assurent une organisation efficace du travail et une réalisation adaptée aux exigences de notre projet.

Chapitre III

Sprint zéro

III.1 Introduction

La phase initiale, souvent appelée Sprint 0, vise à définir les besoins fonctionnels et non fonctionnels du projet. Elle permet de comprendre les attentes des utilisateurs, d'identifier les contraintes techniques et d'assurer une vision partagée au sein de l'équipe.

Ce moment de préparation est crucial pour organiser les priorités, structurer le Product Backlog et planifier efficacement les sprints à venir. En alignant les acteurs dès le départ, on limite les risques de malentendus et les changements de direction en cours de projet.

III.2 Spécification des besoins

La spécification des besoins représente une étape essentielle visant à identifier, analyser et structurer les attentes des utilisateurs ainsi que celles des parties prenantes. Elle constitue la base sur laquelle repose la définition claire des fonctionnalités du futur système.

III.2.1 Identification des acteurs

Les acteurs désignent les entités externes qui interagissent avec le système dans le but de réaliser des actions spécifiques. Ces entités peuvent être humaines, organisationnelles, logicielles ou matérielles [6].

Dans le cadre du projet **StockFlow**, plusieurs acteurs principaux ont été identifiés :

- **Admin Fonctionnel** : responsable de la configuration globale et de la gestion des utilisateurs du système.
- **Planificateur** : chargé d'organiser les commandes et de planifier les livraisons en fonction des capacités disponibles.
- **Admin de Dépôt** : superviseur de l'activité quotidienne au sein des CLR, y compris la gestion des entrées de stock et l'organisation des livraisons.
- **ADB (Administrateur de Base de Données)** : acteur ayant un accès complet à l'ensemble de l'application, y compris les données critiques, les paramètres système et les opérations de maintenance de la base de données.
- **Assistant Client (Keep Contact)** : acteur externe intervenant principalement dans le suivi ou la communication, sans interaction directe avec les fonctionnalités principales du système.

III.2.2 Réalisation du diagramme de contexte dynamique

Le diagramme de contexte dynamique permet de représenter le système logiciel comme une seule entité, en mettant en évidence les interactions avec les acteurs externes, sans détailler les traitements internes [7].

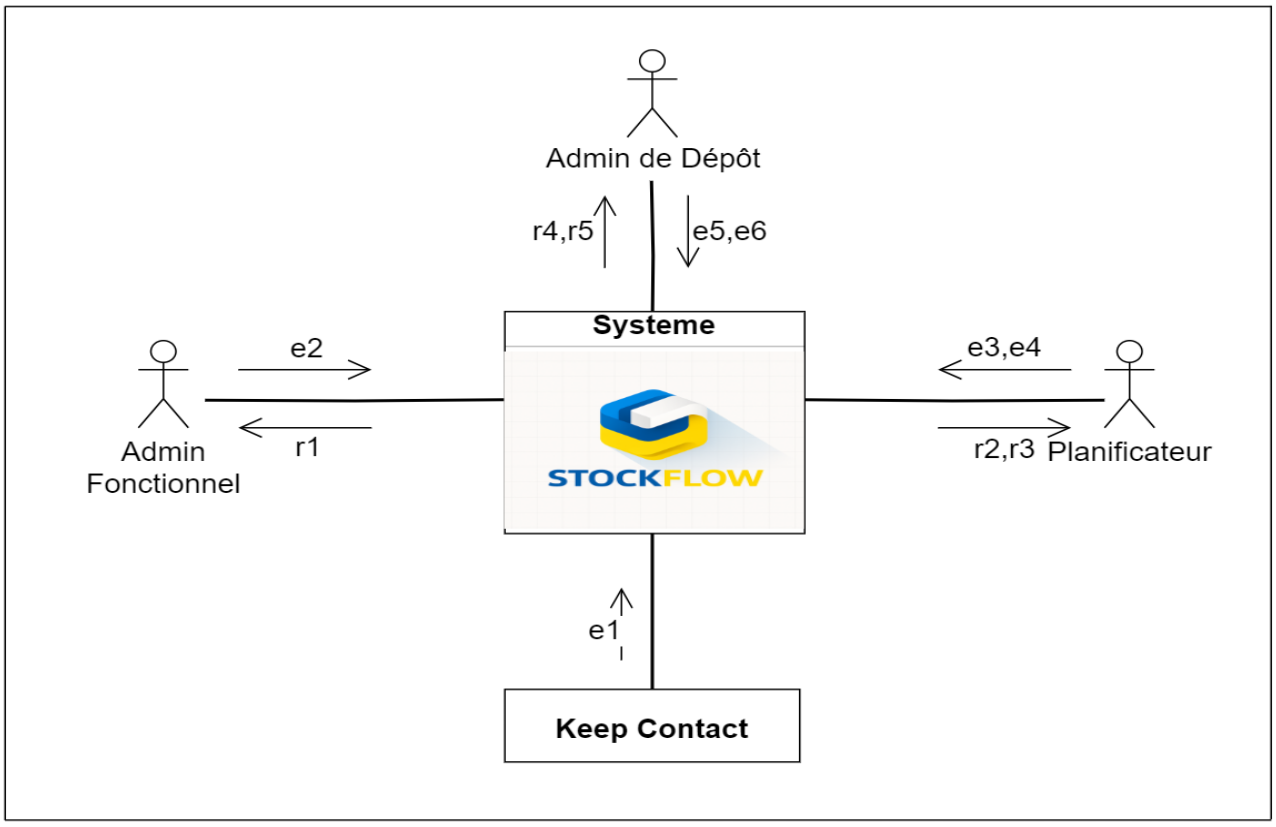

FIGURE III.1 – Diagramme de contexte dynamique

Description des messages échangés.

Ce diagramme (figure III.1) met en évidence les interactions entre les différents acteurs identifiés et le système **StockFlow**. Les flèches **entrantes(e)** représentent les messages ou actions envoyés par les acteurs vers le système, tandis que les flèches **sortantes(r)** indiquent les réponses ou résultats fournis par le système.

TABLE III.1 – Messages émis et reçus par le système

ID	Description
Messages envoyés au système et Messages reçus par le système	
e1	Envoyer les commandes clients au système pour traitement.
e2	Demande de création, modification ou suppression, ainsi que consultation des statistiques globales.
r1	Confirmation des différentes opérations.
e3	Importer les commandes et demander leur validation.

Table III.1 - Messages émis et reçus par le système(Suite)

ID	Description
r2	Enregistrement et affichage des articles disponibles et de leurs quantités.
e4	Planifier la livraison des commandes et valider les demandes de réapprovisionnement.
r3	Enregistrer la planification et déclencher la demande de réapprovisionnement.
e5	Visualisation des demandes planifiées et déclenchement de la livraison.
r4	Génération du bon de livraison et mise à jour automatique du stock après enregistrement. .
e6	Requête de consultation des prédictions.
r5	Affichage des prédictions.

III.2.3 Collecte des besoins fonctionnels

Les besoins fonctionnels correspondent aux services que le système doit offrir aux utilisateurs pour répondre aux objectifs définis. Le tableau suivant présente l'ensemble des fonctionnalités attendues du système StockFlow :

Besoins	Fonctionnalités
Espace d'authentification	Permet la gestion de la connexion et de la déconnexion des utilisateurs, avec vérification des rôles et des permissions.
Espace de gestion globale	Permet la gestion des utilisateurs, des articles, les familles et sous-familles,des dépôts, des véhicules et des clients.
Espace de gestion des commandes, leur validation et Prédiction des besoins	Permet l'importation des commandes, la vérification des stocks, la planification et l'affectation des véhicules, ainsi que la Prédiction des besoins.
Espace de gestion des livraisons	Permet le suivi des livraisons, la gestion des documents de livraison et l'enregistrement des retours.
Espace de gestion du réapprovisionnement	Permet le suivi des stocks critiques, la demande de réapprovisionnement et la mise à jour après réapprovisionnement.

Table III.2 – Tableau des besoins fonctionnels

III.2.4 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation constitue un outil fondamental de la modélisation fonctionnelle dans l'approche UML. Il permet de représenter visuellement les principales fonctionnalités du système, de définir ses limites et de mettre en évidence les interactions entre les acteurs et le système [5].

Liste des cas d'utilisation par acteur

1. **Admin Fonctionnel :**
   - Gérer les dépôts.
   - Gérer les utilisateurs.
   - Gérer les clients.
   - Gérer les articles.
   - Gérer les véhicules.
   - Gérer les familles et sous-familles.
   - Consulter les prédictions.
2. **Planificateur :**
   - Gérer le réapprovisionnement.
   - Gérer les commandes validées.
   - Consulter les prédictions.
3. **Admin de Dépôt :**
   - Gérer les entrées de stock.
   - Gérer les livraisons.
   - Consulter les prédictions.
4. **ADB (Administrateur de la base de données) :**
   - Accéder à l'ensemble des données du système.
   - Superviser et maintenir l'intégrité des bases.
   - Surveiller les performances et résoudre les incidents techniques.

Remarque : Le cas d'utilisation *« Prédire les besoins saisonniers »* est automatisé et géré par le système lui-même à travers un module d'intelligence prédictive, sans intervention directe d'un acteur.

Diagramme de cas d'utilisation global

Le diagramme suivant représente l'ensemble des interactions entre les acteurs du système et les fonctionnalités identifiées.

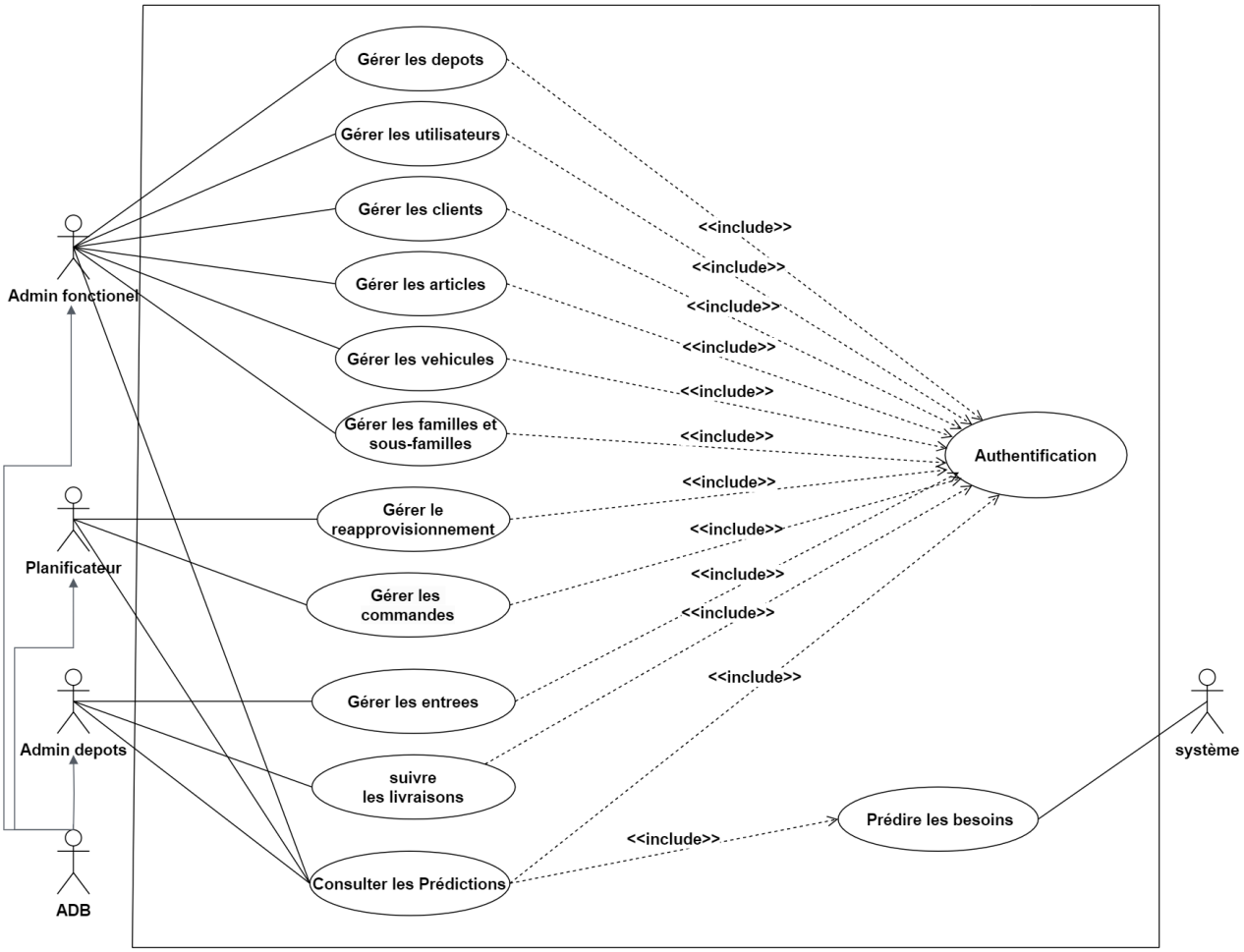

FIGURE III.2 – Diagramme de cas d'utilisation global

III.2.5 Collecte des besoins non fonctionnels

Le système StockFlow doit répondre aux critères suivants :

- **Performance et réactivité :** Les opérations liées à la gestion et à la consultation des stocks doivent être exécutées en moins de deux secondes,

- **Accessibilité et ergonomie :** L'interface utilisateur doit être claire, intuitive et adaptée aux conditions réelles de travail des différents acteurs du système.

- **Sécurité des données :** La protection des données sera assurée par le hachage des mots de passe, l'authentification via JWT (JSON Web Token) et une gestion stricte des accès fondée sur les rôles attribués.

- **Fiabilité des prévisions :** Le système devra produire des prévisions d'approvisionnement cohérentes et précises, alignées sur les tendances saisonnières observées dans les historiques de commandes.

III.3 Pilotage du projet avec Scrum

Cette section décrit concrètement la mise en œuvre de la méthode Scrum, adoptée pour organiser et suivre efficacement l'avancement des travaux tout au long du projet.

III.3.1 Équipe Scrum

Dans le cadre de notre projet de fin d'études au sein de **CEVITAL**, nous avons adopté la méthodologie **Scrum**, qui repose sur une répartition claire des responsabilités pour une gestion agile et efficace.

Le rôle de *Product Owner* a été assuré par notre **client en entreprise**, chargé de prioriser les besoins métier et de valider les livrables.

La fonction de *Scrum Master*, garante du bon déroulement du processus, a été partagée entre **M. Bennai Brahim** (tuteur entreprise) et **M. Touazi Djoudi** (encadrant universitaire), assurant la coordination entre les parties prenantes.

L'**équipe de développement**, composée de **Meznad Imene** et **Massioun Celia**, a pris en charge la conception, le développement et les tests du système, en participant aux différents événements *Scrum* : planifications, revues et rétrospectives.

Cette organisation itérative a favorisé un suivi régulier de l'avancement, une bonne réactivité aux retours, et le respect des délais.

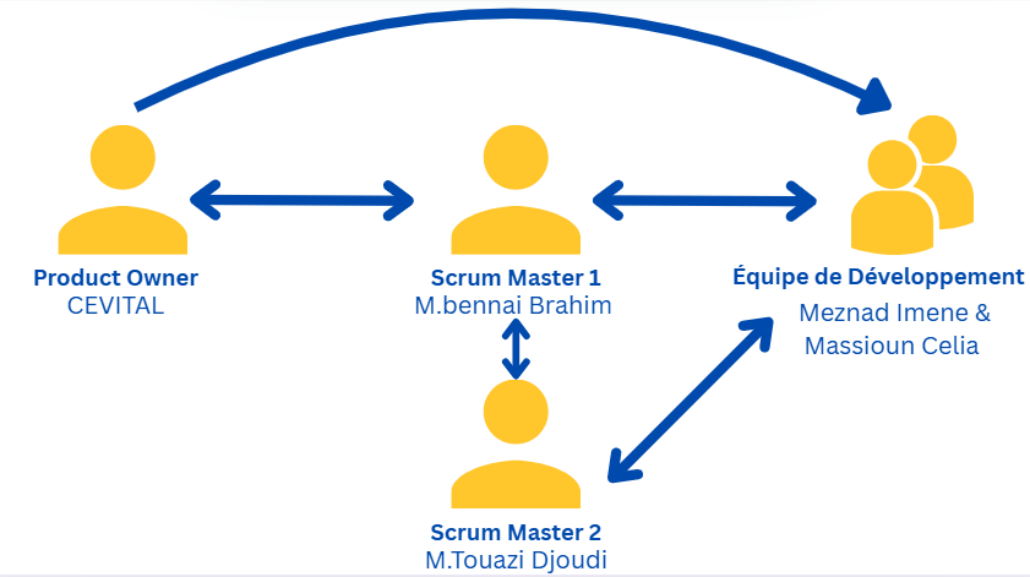

FIGURE III.3 – L'équipe Scrum de notre projet

III.3.2 Product Backlog

Le *Product Backlog* est défini par le *Product Owner*. Il regroupe l'ensemble des fonctionnalités à développer, classées par priorité, et constitue la base de planification des sprints.

Le tableau ci-dessous présente le backlog initialement établi au démarrage du projet :

Table III.3 – Planification des sprints

ID	Thèmes (features)	Id U.S	User story	Importance
2	Authentification et gestion des utilisateurs	1.1	Admin Fonctionnel : Ouvrir l'application, s'authentifier avec ses identifiants, puis accéder à son espace de travail personnalisé selon son rôle.	Élevée
		1.2	Admin Fonctionnel : Gestion des utilisateurs (ajout, modification, suppression).	Moyenne
		1.3	planificateur/Admin Dépôt : Accéder à l'espace après authentification.	Élevée
		1.4	Admin Fonctionnel : Gestion des rôles et des accès.	Moyenne
2	Gestion des ressources	2.1	Admin Fonctionnel : Création, modification et consultation des dépôts.	Élevée
		2.2	Admin Fonctionnel : Gestion des articles, familles, sous-familles et véhicules.	Moyenne
		2.3	Admin Fonctionnel : Gestion des clients (ajout, modification, suppression).	Moyenne
		2.4	Admin Dépôt : Gestion des entrées des dépôts.	Élevée
		2.5	Admin Fonctionnel : Affecter chaque planificateur/Admin Dépôt et les articles à leur dépôt.	Moyenne
3	Gestion des commandes, réapprovisionnement et livraisons	3.1	planificateur : Importer, valider et suivre les commandes.	Élevée
		3.2	planificateur : Affecter les véhicules disponibles aux commandes.	Moyenne
		3.3	planificateur : Envoyer la planification à l'Admin Dépôt.	Facile

Planification des sprints (suite)

ID	Thèmes (features)	Id U.S	User story	Importance
		3.4	planificateur : Vérifier l'état des livraisons.	Moyenne
		3.5	planificateur : Gérer les réapprovisionnements.	Moyenne
		3.6	Admin Dépôt : Suivre l'état des livraisons, et générer les bons de livraison.	Élevée
		3.7	Admin Dépôt : Gérer les retours et mettre à jour le statut du véhicule.	Moyenne
4	Prédiction et visualisation	4.1	Admin Fonctionnel/planificateur/Admin Dépôt : Consulter les prévisions de besoins basées sur l'historique.	Moyenne
		4.2	Admin Fonctionnel/planificateur/Admin Dépôt : Visualisation centralisée des données logistiques avec graphiques et prévisions.	Élevée

III.3.3 Structure et découpage du projet

Dans le tableau (III.4) ci-dessous, nous illustrons la répartition chronologique des sprints pour notre projet de gestion des dépôts CEVITAL. Au total, nous avons planifié quatre (4) sprints, chacun ayant une durée de 4 semaines environ.

- **Le premier Sprint** est consacré à la mise en place de l'authentification sécurisée et à la gestion des utilisateurs.
- **Le deuxième Sprint** porte sur la gestion des ressources, notamment les dépôts, les articles, les véhicules, ainsi que les clients. Il inclut également l'affectation et la gestion des entrées de stock.
- **Le troisième Sprint** est centré sur la gestion des commandes, des réapprovisionnements et des livraisons.
- **Le quatrième Sprint** est dédié à la prédiction des besoins, à la visualisation des données et à la finalisation de l'application.

Numéro de Sprint	Tâche	Durée
1	**Authentification et gestion des utilisateurs**	15 février - 19 mars
2	**Gestion des ressources et affectation**	20 mars - 16 avril
3	**Gestion des commandes, réapprovisionnement et livraisons**	17 avril - 14 mai
4	**Prédiction, visualisation et finalisation**	15 mai - 10 juin

TABLE III.4 – Répartition chronologique des sprints

III.4 Charte graphique

La charte graphique constitue un référentiel visuel qui définit les règles d'utilisation des éléments graphiques d'une application ou d'un site web en spécifiant notamment les couleurs, typographies, logos, et styles graphiques utilisés [8].

III.4.1 Présentation du logo de l'application web

Le nom de notre application, **StockFlow**, a été soigneusement choisi pour illustrer l'objectif principal du système. Le terme « Stock » fait référence à la gestion des inventaires, tandis que « Flow » symbolise la fluidité et la régularité des mouvements de marchandises.

FIGURE III.4 – Logo de StockFlow

III.4.2 Palette des couleurs

Pour notre application web, voici les couleurs essentielles :

- **Bleu foncé** (#002855) : pour l'en-tête, les boutons principaux et les éléments interactifs.
- **Jaune** (#FFC72C) : pour les accents, boutons secondaires et icônes.
- **Blanc** (#FFFFFF) : pour les fonds et les zones de contenu.

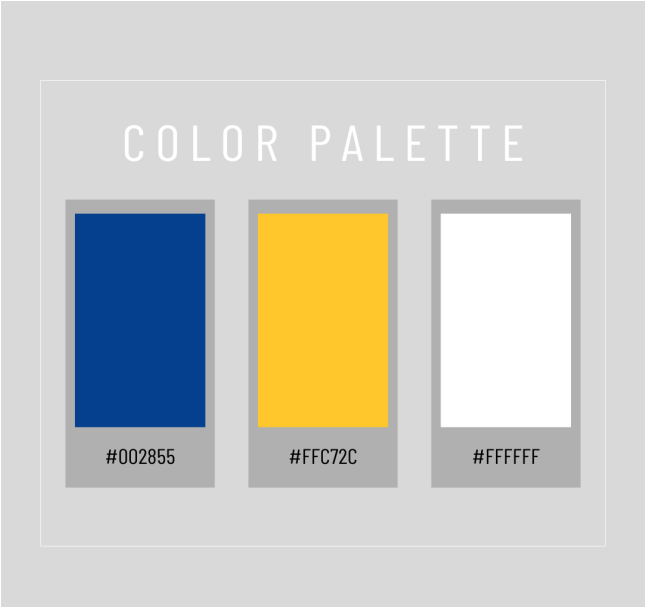

FIGURE III.5 – Palette des couleurs

III.5 Conclusion

Nous avons procédé à la spécification complète des besoins du système, en définissant les acteurs impliqués, les différents cas d'utilisation, ainsi que le cas d'utilisation global. L'équipe Scrum est désormais constituée, avec des rôles bien attribués, et un *Product Backlog* structuré a été élaboré pour encadrer le développement. Par ailleurs, une charte graphique a été mise en place afin d'assurer une identité visuelle cohérente à l'application. Ces éléments nous permettent d'aborder le **Sprint 1** dans de bonnes conditions, en amorçant la réalisation des premières fonctionnalités.

Chapitre IV

Sprint UN

IV.1 Introduction

Dans ce chapitre, nous allons entrer au cœur de notre démarche en détaillant les fonctionnalités définies dans le backlog, telles qu'elles ont été planifiées pour le premier sprint. Cette étape s'inscrit pleinement dans le cadre du Processus Unifié (UP), qui nous guide dans l'organisation rigoureuse et progressive du projet. L'objectif est de mettre en place les fondations fonctionnelles essentielles, en garantissant une livraison rapide et de qualité, tout en restant à l'écoute des besoins du client.

Nous vous proposons ainsi de découvrir comment nous avons sélectionné et réparti les tâches, comment chaque fonctionnalité a été pensée pour répondre aux exigences, et comment cette première phase constitue un jalon clé dans la réalisation globale du système.

IV.2 Backlog du Sprint 1

Pour le **Sprint 1**, nous avons choisi de nous concentrer sur les fonctionnalités clés liées à la gestion des utilisateurs. Cela inclut tout ce qui touche à l'authentification : se connecter et garder la session active . En parallèle, nous avons aussi travaillé sur la gestion des rôles notamment pour que les administrateurs fonctionnels puissent gérer les utilisateurs. Ces éléments, tirés du *Product Backlog*, étaient nos priorités à terminer pendant ce sprint.

Table IV.1 – Backlog Sprint 1 - Description détaillée des tâches

Fonctionnalité	**ID**	**Tâche**	**Par qui**	**Durée (j)**
Authentification et gestion des rôles	A1	Rédiger les cas d'utilisation pour la fonctionnalité d'authentification, incluant les scénarios d'accès.	Celia Massioun	0.5
	A2	Concevoir les diagrammes de séquence UML pour le processus d'authentification et créer les classes nécessaires pour la gestion des utilisateurs et des rôles.	Celia Massioun	1
	A3	Développer l'interface utilisateur d'authentification, incluant les formulaires de connexion et les interactions frontend, ainsi que le backend .	Imene Meznad	3
	A4	Implémenter la gestion des rôles avec JWT pour sécuriser l'accès, réaliser les tests pour validation complete.	Imene Meznad	3

Suite à la page suivante

Table IV.1 Backlog Sprint 1 - Description détaillée des tâches

Fonctionnalité	ID	Tâche	Par qui	Durée (j)
Gestion des utilisateurs	B1	Rédiger les cas d'utilisation couvrant la création, modification, suppression et consultation des comptes utilisateurs.	Imene Meznad	0.5
	B2	Concevoir les diagrammes de séquence UML décrivant les interactions associées aux opérations CRUD sur les utilisateurs.	Imene Meznad	0.5
	B3	Développer les interfaces (création, modification, suppression, affichage) avec les fonctionnalités de validation, recherche, tri, et intégration frontend/backend.	Celia Massioun	2
	B4	Réaliser les tests fonctionnels pour valider le bon fonctionnement de l'ensemble des opérations sur les utilisateurs.	Celia Massioun	1

IV.3 Spécification

La phase de spécification est essentielle pour poser les bases du système de gestion des dépôts CEVITAL. Elle permet de clarifier ce que l'application doit faire et comment les utilisateurs vont interagir avec elle.

Dans cette phase, on va se concentrer sur les fonctionnalités principales et les interactions des différents utilisateurs avec le système dans le sprint 1. Pour rendre tout ça plus visuel et compréhensible, on va utiliser des diagrammes de cas d'utilisation, accompagnés de descriptions simples et détaillées.

IV.3.1 Diagramme de cas d'utilisation - Sprint 1

Diagramme de cas d'utilisation sprint 1

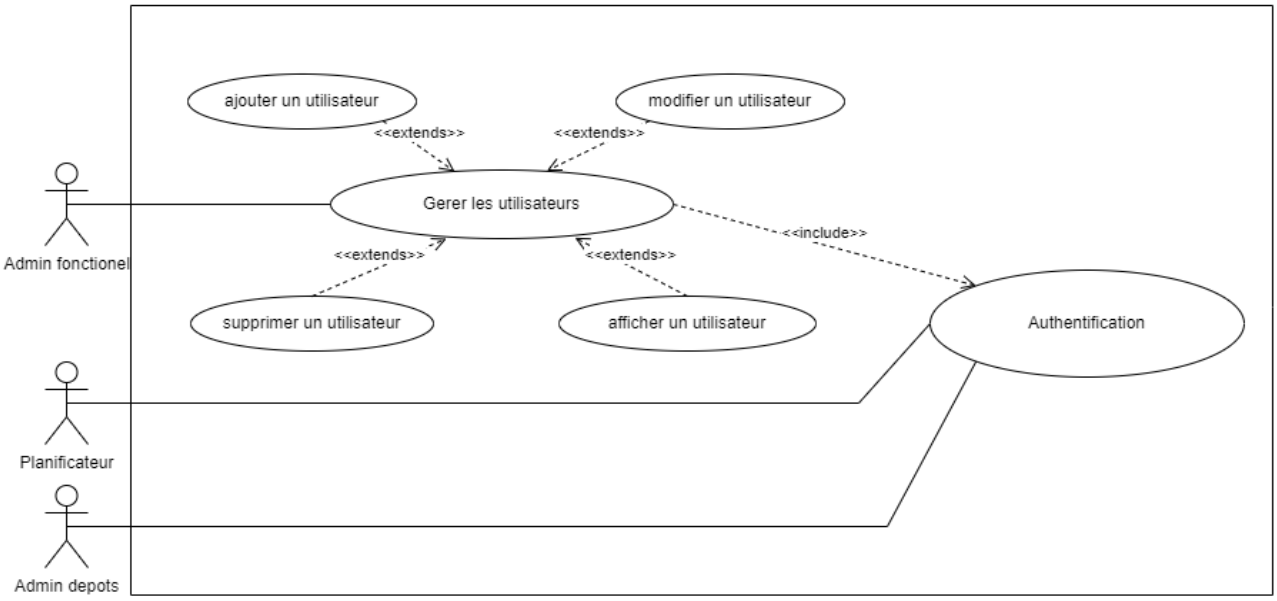

FIGURE IV.1 – Diagramme cas d'utilisation du SPRINT 1

IV.3.2 Description textuelle des cas d'utilisation - Sprint 1

Règles de gestion globale

TABLE IV.2 – Règles de gestion globale

Règle	Description
RG1	Un utilisateur doit avoir un rôle (Admin Fonctionnel, Planificateur, Admin Dépôts).
RG2	Seul l'Admin Fonctionnel peut ajouter un nouvel utilisateur et lui attribuer un rôle.
RG3	Le système vérifie l'unicité de l'email avant l'ajout.

Description textuelle des cas d'utilisation - Sprint 1

CU : Authentification

TABLE IV.3 – Description textuelle du cas : Authentification

CU	Authentification

Acteur	Tout utilisateur
Précondition	L'utilisateur est enregistré dans le système et n'est pas désactivé.
Objectif	Permettre à l'utilisateur d'accéder à l'application de manière sécurisée.
Postcondition	L'utilisateur est authentifié et redirigé vers son interface, en fonction de son rôle.
Enchaînement principal	— L'utilisateur accède à l'écran de connexion. — Il saisit son identifiant (email) et son mot de passe. — Il clique sur le bouton de connexion. — Le système vérifie l'authenticité des informations. — Si les informations sont correctes, l'utilisateur est redirigé vers son espace personnel (selon son rôle).
Enchaînements alternatifs	— Si les informations sont incorrectes (email, mot de passe ou les deux), le système affiche un message d'erreur et invite à une nouvelle saisie. — Si l'utilisateur a oublié son mot de passe, il doit contacter le service administratif pour réinitialisation.
Exception	Après trois tentatives échouées, l'utilisateur est temporairement bloqué. Un message l'informe du délai de blocage.

TABLE IV.4 – Description textuelle du cas : Gérer les utilisateurs

CU	CU02 : Gérer les utilisateurs
Acteur	Admin Fonctionnel
Précondition	L'utilisateur est authentifié en tant qu'Admin Fonctionnel.
Objectif	Ajouter, modifier, supprimer , consulter ou desactiver les informations d'un utilisateur.
Postcondition	La liste des utilisateurs est mise à jour.

Enchaînement principal	— L'Admin accède à l'interface de gestion des utilisateurs. — Il choisit une action : ajouter, modifier, désactiver, supprimer ou consulter. — **Ajouter un utilisateur** : • Clique sur "Ajouter", remplit les champs requis (nom, prénom, email, rôle). • Le système valide les données, enregistre l'utilisateur et affiche un message de succès. — **Modifier un utilisateur** : • Sélectionne un utilisateur, modifie les informations, puis valide. • Le système met à jour les données et affiche un message de confirmation. — **Désactiver ou supprimer un utilisateur** : • Sélectionne l'utilisateur et choisit une date de fin ou clique sur supprimer . • Le système désactive l'utilisateur à la date indiquée ou le supprime et confirme l'action. — **Consulter un utilisateur** : • Sélectionne l'utilisateur et affiche ses informations détaillées.
Enchaînements alternatifs	— Si l'email est déjà utilisé, un message d'erreur est affiché. — Si l'utilisateur est déjà affecté à un dépôt, la suppression est interdite. — En cas de données invalides ou incomplètes, le système demande une correction.
Exception	Si la connexion au serveur est perdue, l'action est annulée et un message d'erreur s'affiche.

IV.4 Analyse

La phase d'analyse vise à identifier, comprendre et documenter les besoins des parties prenantes. Elle permet de transformer les attentes exprimées en exigences formelles, qui serviront de référence tout au long du processus de développement logiciel.[9].

IV.4.1 diagramme d'activité - Sprint 1

Le diagramme d'activités est un type de diagramme comportemental utilisé en UML. Il permet de modéliser graphiquement les différentes étapes d'un processus ou le déroulement logique d'un cas d'utilisation[5].

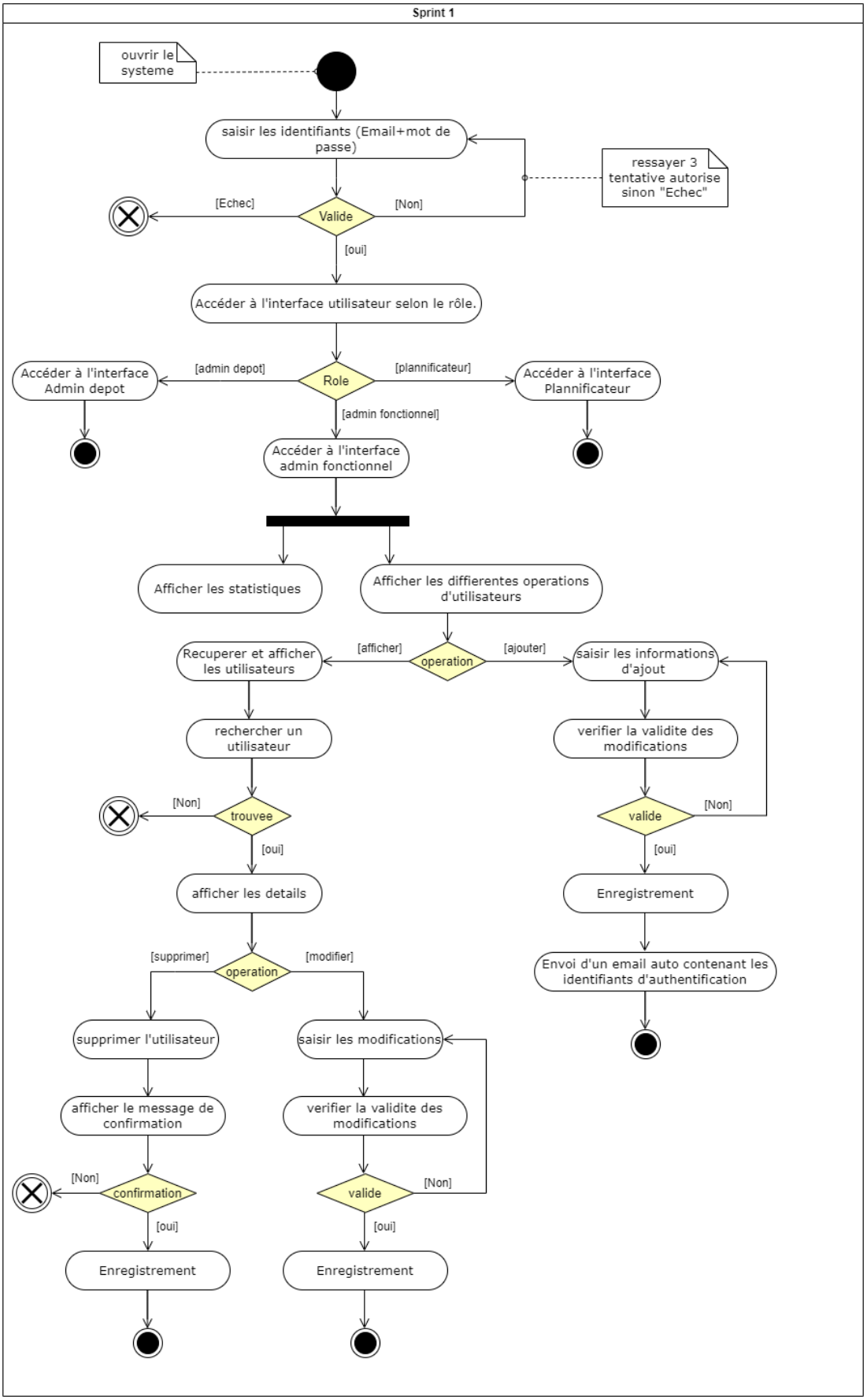

FIGURE IV.2 – diagramme d'activité - Sprint 1

IV.4.2 Diagramme de séquence système - Sprint 1

un outil de modélisation dynamique qui permet de représenter, de manière chronologique, les échanges entre un acteur externe et le système étudié[10].

Afin de ne pas alourdir le contenu du mémoire, nous avons choisi de ne présenter que deux diagrammes de séquence système jugés essentiels :

- **Diagramme de séquence système "Authentification"** :
  Ce scénario décrit le processus d'authentification, point d'entrée essentiel du système. L'utilisateur saisit son email et son mot de passe. Le système vérifie les informations saisies. Si elles sont correctes, l'utilisateur est connecté avec succès. Sinon, un message d'erreur est affiché et il peut réessayer. Cette boucle se répète jusqu'à trois fois. Après trois échecs consécutifs, le système bloque temporairement le compte de l'utilisateur pendant quinze minutes. Durant cette période, les champs email et mot de passe deviennent gris et désactivés, empêchant toute nouvelle saisie, même si les identifiants sont corrects.

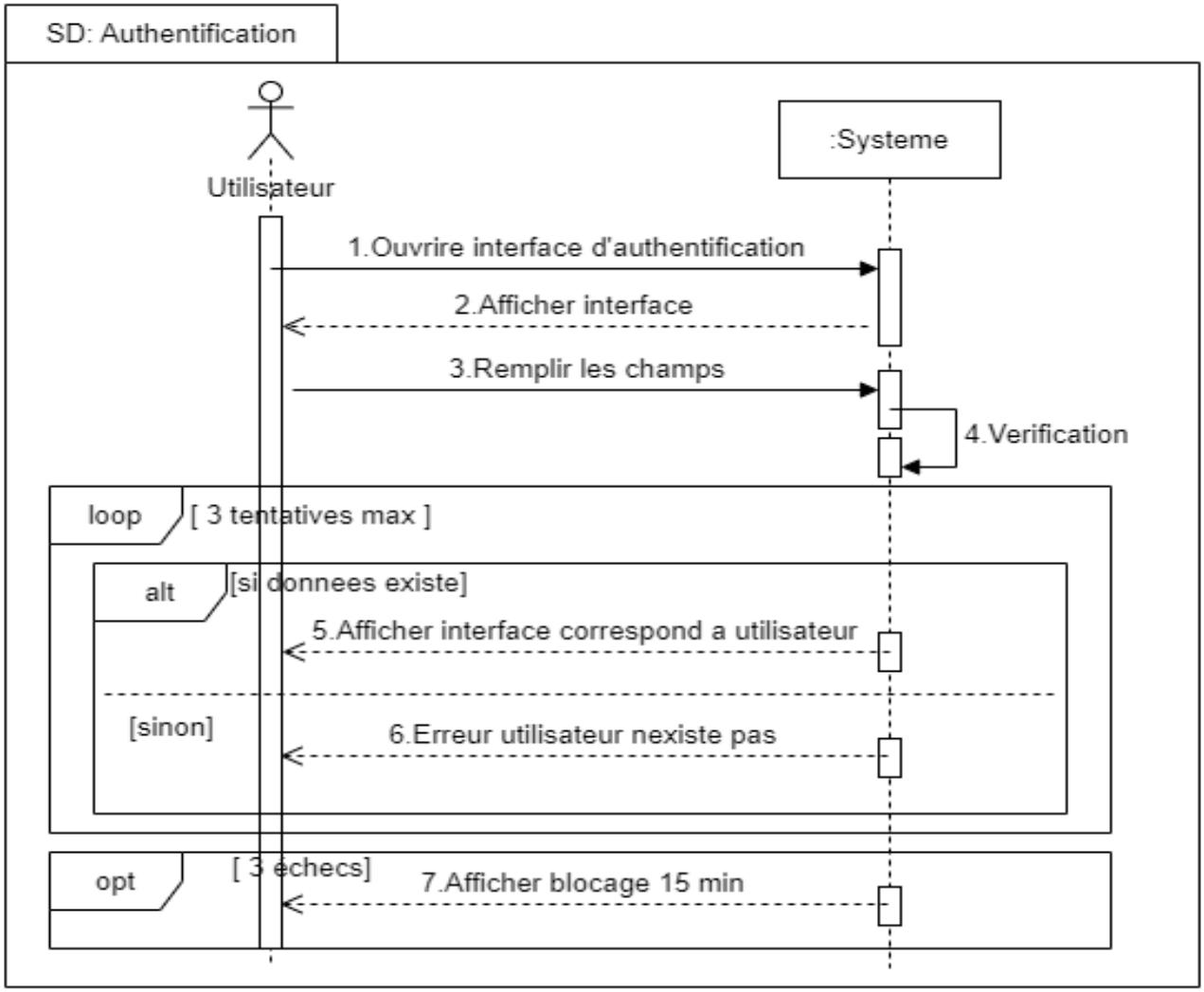

FIGURE IV.3 – Diagramme de séquence système "Authentification"

- **Diagramme de séquence – Ajouter un utilisateur** :
  Ce scénario illustre le processus d'ajout d'un nouvel utilisateur. Après authentification, l'administrateur fonctionnel accède à l'interface d'ajout, saisit le nom, le prénom, l'email et les autres informations nécessaires. Le système vérifie que les champs sont valides, notamment l'absence de chiffres dans le nom et le prénom, ainsi que l'unicité de l'email. Un mot de passe est généré automatiquement. Le compte est ensuite enregistré et un email est envoyé à l'utilisateur contenant ses identifiants. Un message de confirmation s'affiche en cas de succès.

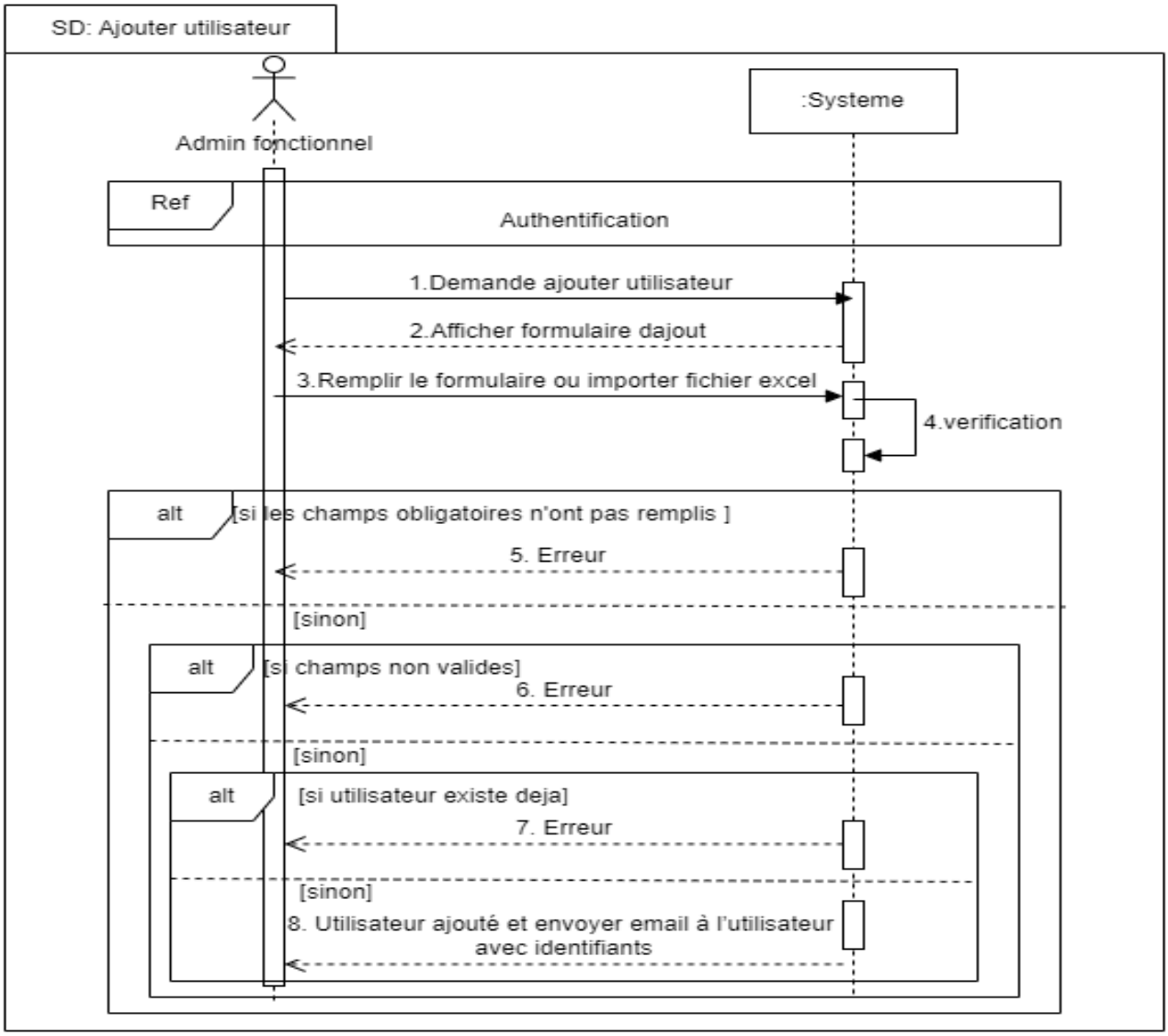

FIGURE IV.4 – Diagramme de séquence système "Ajouter un utilisateur"

IV.5 Conception

La phase de conception constitue une étape essentielle dans le cadre du Processus Unifié. Elle vise à définir l'architecture interne du système, ainsi que les solutions techniques permettant de satisfaire les exigences fonctionnelles et non fonctionnelles identifiées lors de l'analyse [11].

IV.5.1 diagramme d'activité Authentification - Sprint 1

Ce diagramme représente le déroulement des actions, des choix et des interactions effectués par un utilisateur lors du processus d'authentification au sein du système[12].

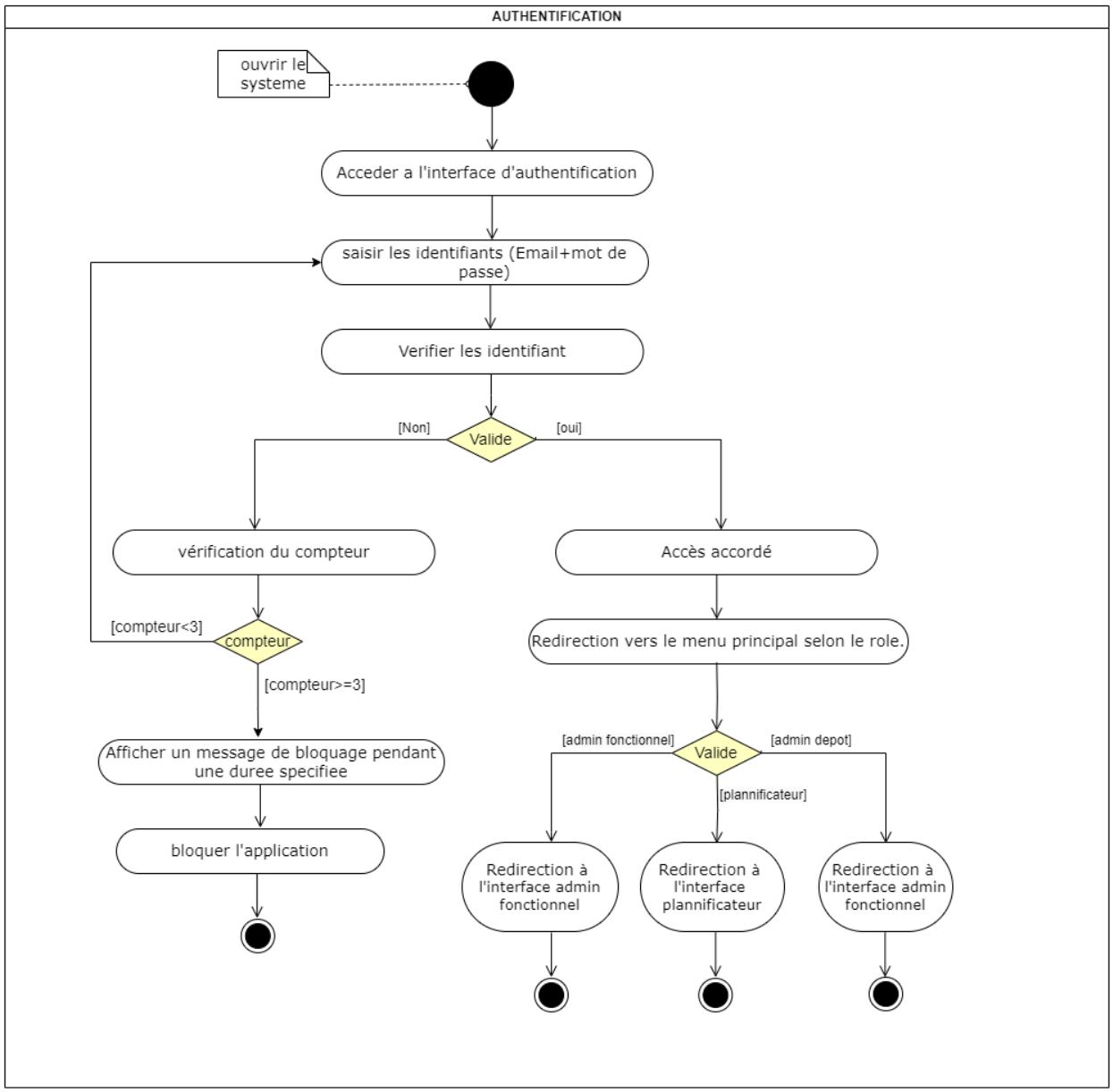

FIGURE IV.5 – diagramme d'activité - Authentification

IV.5.2 Le diagramme de séquence détaillé - Sprint 1

Un diagramme de séquence détaillé est un diagramme d'interaction UML qui expose le déroulement précis des échanges entre l'interface utilisateur, les contrôleurs, les services et la base de données, en représentant l'ordre temporel des messages, l'utilisation de fragments et les traitements internes [13]. Dans cette étude, nous réaliserons deux diagrammes détaillés : un

pour l'authentification renforcée (avec UI, AuthController, AuthMidleware, compteur d'échecs, blocage, redirection selon rôle) et un autre pour l'ajout d'utilisateur (avec AdminUI, UtilisateurController, vérifications, mailService, génération de mot de passe).

- **Diagramme de séquence détaillé "Authentification"** :

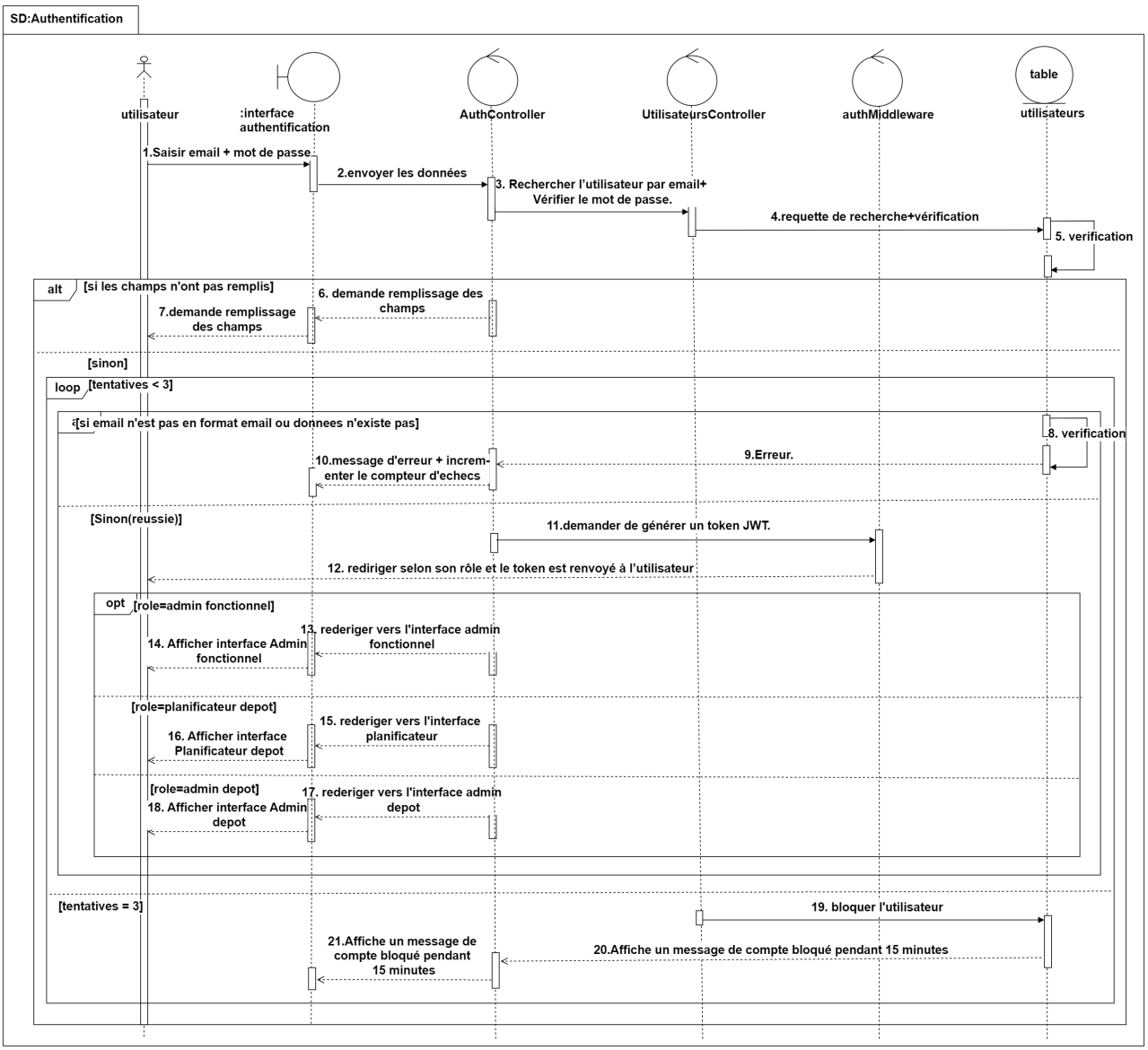

FIGURE IV.6 – Diagramme de séquence détaillé "Authentification"

- **Diagramme de séquence détaillé "Ajouter utilisateur"** :

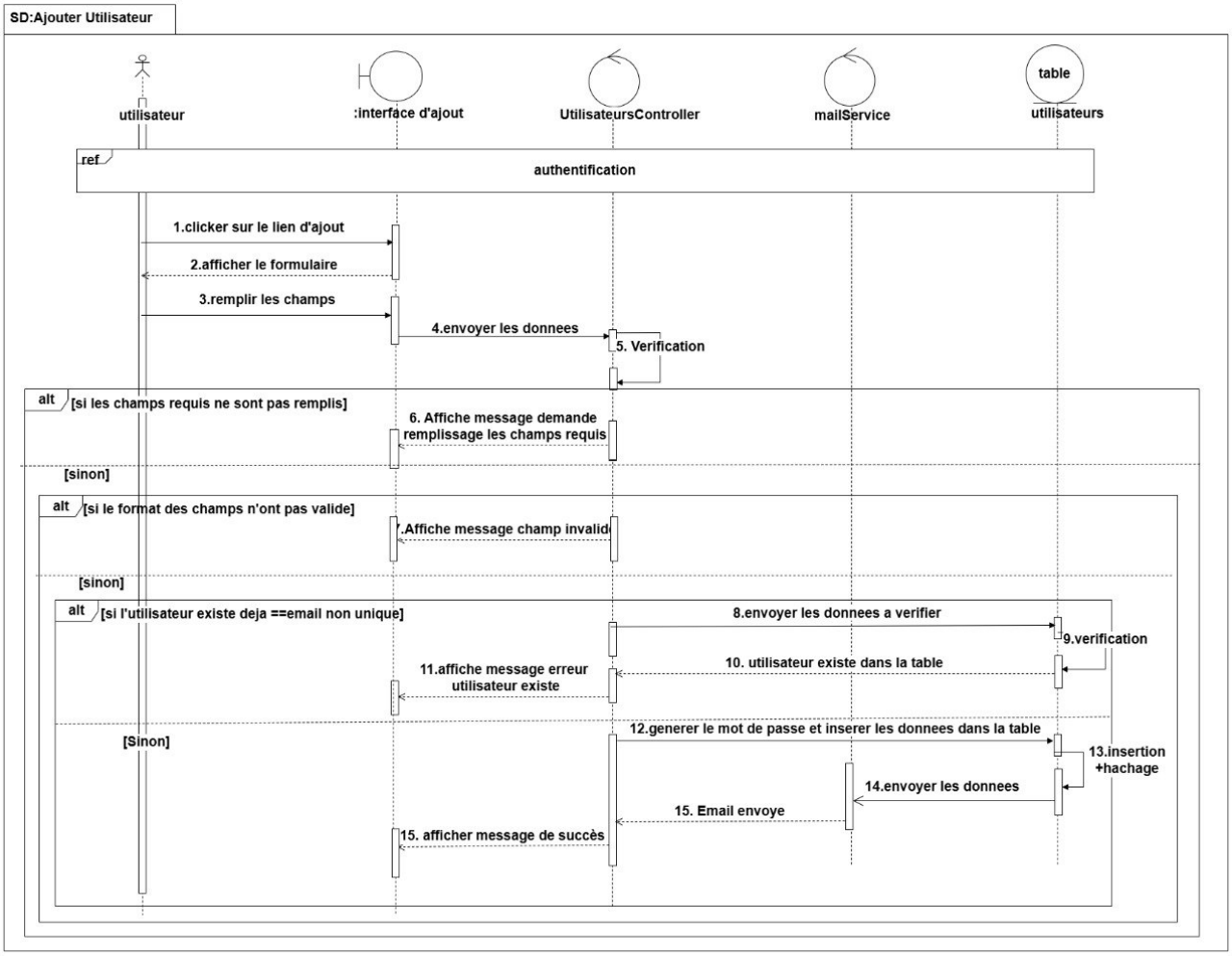

FIGURE IV.7 – Diagramme de séquence détaillé "Ajouter utilisateur"

IV.5.3 Diagramme de classes - Sprint 1

Ce diagramme représente la structure statique du système en illustrant les principales classes, leurs attributs, leurs méthodes, ainsi que les relations (héritage, association) qui les lient. Il établit les bases techniques de l'implémentation en définissant les objets essentiels et leurs connexions [14].

Le diagramme de classes présenté ci-dessus illustre la modélisation orientée objet des utilisateurs du système dans le cadre du Sprint 1 du projet de gestion des dépôts CEVITAL. Il montre les entités principales, leurs attributs, leurs méthodes, ainsi que les relations d'héritage entre elles.

La classe mère Utilisateur regroupe les informations de base communes à tous les types d'utilisateurs, tandis que les classes filles Admin fonctionnel, Planificateur et Admin dépôt définissent des rôles spécifiques. Une énumération Statut permet de gérer l'état des utilisateurs (actif ou inactif).

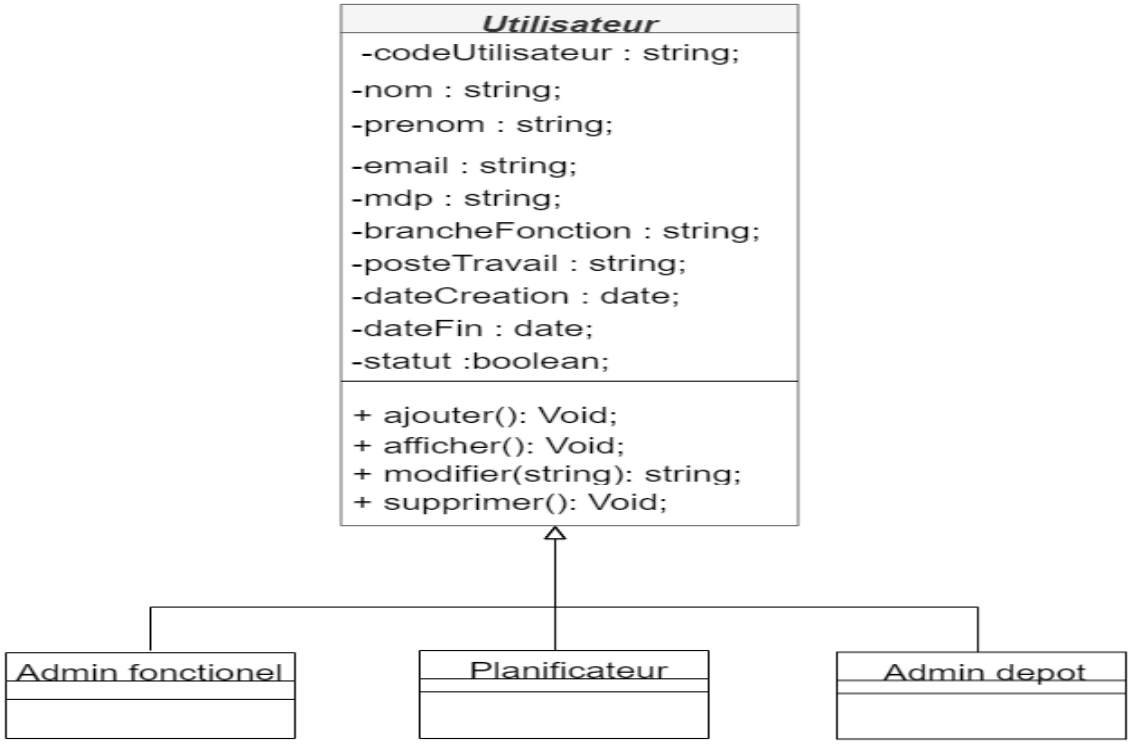

FIGURE IV.8 – Diagramme de classes - Sprint 1

IV.6 Implementation

IV.6.1 Dictionnaire de données - Sprint 1

Il s'agit d'un tableau structuré qui décrit précisément les éléments stockés dans la base de données. Pour chaque champ, il présente son nom, son type, ses contraintes (telles que clé primaire, unicité ou nullabilité), désignation, ainsi que sa taille le cas échéant. Ce tableau constitue un support essentiel pour la conception et la compréhension du modèle de données.[15].

TABLE IV.5 – Table : Utilisateur

Nom du champ	**Type**	**Contraintes**	**Taille**	**Désignation**
codeUtilisateur	VARCHAR	Clé primaire, unique	10	Identifiant unique de l'utilisateur
nom	VARCHAR	Non null	50	Nom de l'utilisateur

Nom du champ	Type	Contraintes	Taille	Désignation
prenom	VARCHAR	Non null	50	Prénom de l'utilisateur
email	VARCHAR	Unique, format valide	100	Adresse e-mail de l'utilisateur
mdp	VARCHAR	Non null	255	Mot de passe (hashé)
dateCreation	DATE	Défault = CURRENT_DATE	-	Date de création du compte
dateFin	DATE	Nullable	-	Date de désactivation du compte
brancheFonction	VARCHAR	Nullable	50	Département ou branche fonctionnelle
posteTravail	VARCHAR	Nullable	50	Intitulé du poste occupé
statut	BOOLEAN	Valeur par défaut "True"	-	Statut actif ou inactif du compte
role	ENUM('admin fonctionnel', 'planificateur', 'admin depot')	Obligatoire	-	Rôle attribué à l'utilisateur dans le système

IV.6.2 Modèle relationnel de données - Sprint 1

Le modèle relationnel organise les données sous forme de relations (tables) composées d'enregistrements (lignes) et d'attributs (colonnes). Les clés primaires assurent l'unicité de chaque ligne, tandis que les clés étrangères définissent les liens entre les tables. Ce modèle constitue la base logique à partir de laquelle est généré le schéma physique de la base de données [16].

Utilisateur(codeUtilisateur, nom, prenom, email, mdp, dateCreation, dateFin,brancheFonction, posteTravail, statut, role)

Explication de l'approche "Push Up" : L'héritage est remplacé par une colonne role pour indiquer le type d'utilisateur.

Les fonctionnalités spécifiques aux Admin fonctionnel (ex : modifier(), supprimer()) sont gérées via des contrôles applicatifs basés sur le role.

IV.7 Test & presentation d'interfaces

L'interface d'authentification constitue le point d'entrée sécurisé de l'application StockFlow, permettant aux utilisateurs de s'authentifier selon leurs identifiants. Elle offre une interface responsive et moderne intégrant un fond animé, le support du mode sombre, ainsi qu'une redirection automatique selon le rôle de l'utilisateur. Pour garantir la sécurité, les mots de passe sont hachés côté serveur, et un jeton JWT signé est généré lors de la connexion pour maintenir une session sécurisée. Des mesures de protection contre les attaques par force brute sont également appliquées côté backend pour prévenir les tentatives d'accès non autorisées.

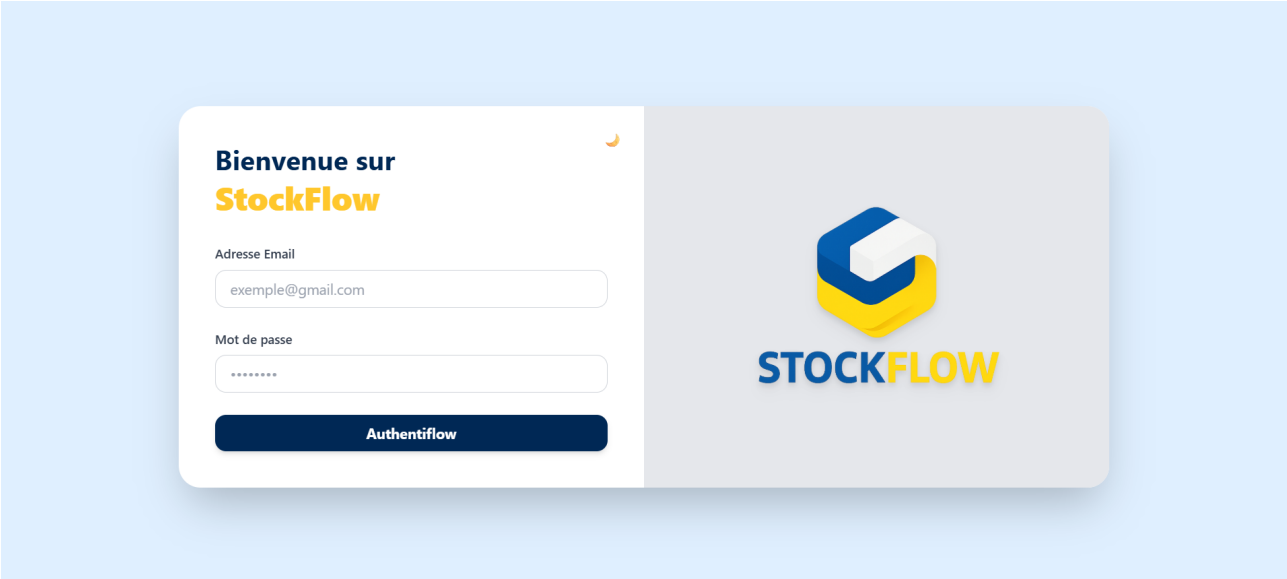

FIGURE IV.9 – Login page

L'interface Afficher les utilisateurs est responsive et intuitive. Elle affiche les utilisateurs par rôle (Admin Fonctionnel, Admin Dépôt, Planificateur) avec leurs informations clés : nom, prénom, email, rôle, état. Elle intègre un tableau interactif avec recherche, tri, filtres et actions (modifier, désactiver), facilitant la gestion rapide et claire des profils.

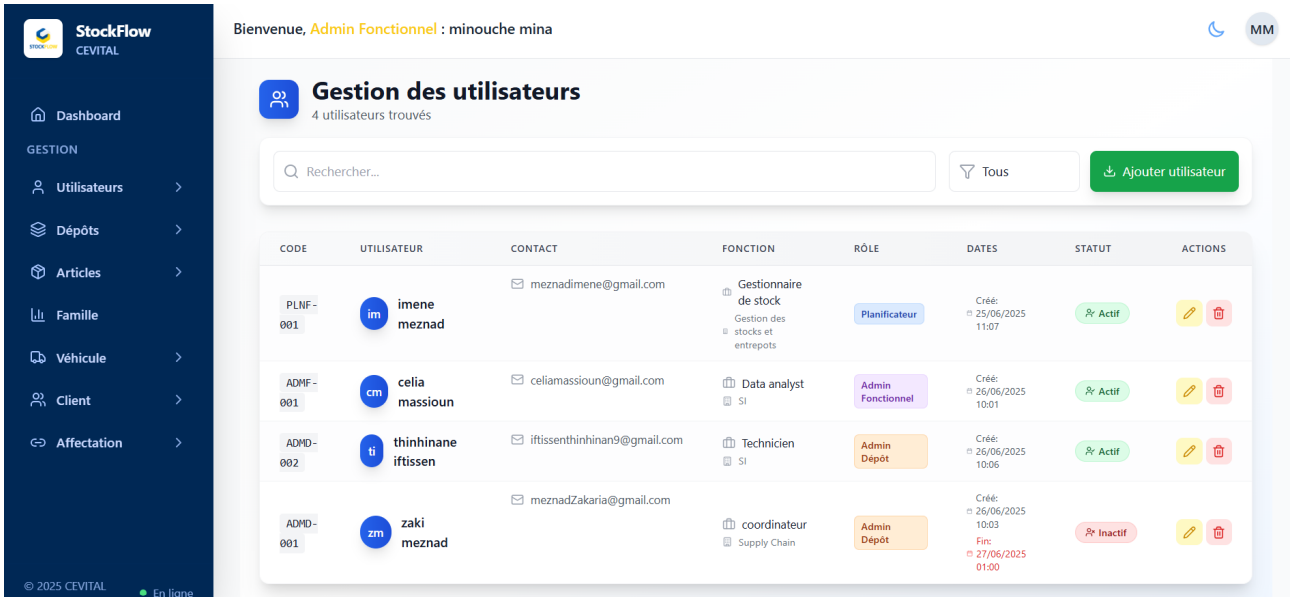

FIGURE IV.10 – page Afficher les utilisateurs

IV.8 Bilan du Sprint 1

Revue de Sprint : Toutes les fonctionnalités planifiées (authentification JWT et gestion des utilisateurs) ont été livrées avec succès. Les tests couvrent 90% des cas d'usage et l'interface (Fig. IV.9) a été validée par le client qui a également formulé quelques suggestions d'amélioration pour optimiser l'UX.

Rétrospective : Points forts : collaboration efficace et documentation complète. Axes d'amélioration : synchronisation des merges et prototypage plus précoce des interfaces.

IV.9 Conclusion

Le premier sprint nous a permis de modéliser les fonctionnalités de base du système, en particulier l'authentification des utilisateurs et leur gestion. Cette étape a posé les fondations nécessaires à la suite du développement. Nous entamerons à présent le Sprint 2, qui portera sur la gestion des ressources du système, telles que les dépôts, les familles, les sous-familles et les articles.

Chapitre V

Sprint deux

V.1 Introduction

Ce deuxième sprint s'inscrit dans la continuité du travail effectué lors du Sprint 1.Conformément à la démarche incrémentale adoptée, chaque sprint apporte de nouvelles fonctionnalités tout en s'appuyant sur les éléments déjà réalisés.

Afin d'alléger la lecture et d'éviter toute répétition, les définitions des différentes phases du processus de développement, ainsi que les concepts méthodologiques déjà expliqués dans le Sprint 1 (les definitions des diagrammes, etc.), ne seront pas reprises ici.

Ce deuxième sprint poursuit la progression du développement entamé lors du Sprint 1, en se concentrant cette fois sur des fonctionnalités essentielles à la gestion opérationnelle des ressources.

V.2 Backlog du Sprint 2

Le backlog du sprint 2 couvre précisément l' **affectation,gestion des dépôts, des articles, familles et sous-familles,des clients, des véhicules ainsi que la gestion des entrées** dans le système. Ces fonctionnalités sont cruciales pour assurer un suivi efficace des ressources matérielles et logistiques.

Table V.1 – Backlog Sprint 2 - Description détaillée des tâches

Fonctionnalité	ID	Tâche	Par qui	Durée (j)
Affectation des rôles et articles	C1	Rédiger les cas d'utilisation pour l'affectation des planificateurs, des administrateurs de dépôt ainsi que des articles aux dépôts correspondants.	Celia Massioun	1.5
	C2	Concevoir les diagrammes de séquence UML pour les processus d'affectation des utilisateurs.	Celia Massioun	2
	C3	Développer l'interface utilisateur pour la gestion des affectations, incluant les validations et notifications.	Imene Meznad	3
	C4	Implémenter la logique backend pour gérer les affectations avec tests fonctionnels.	Imene Meznad	2.5
Gestion des dépôts	D1	Rédiger les cas d'utilisation pour la création, modification, suppression et consultation des dépôts.	Imene Meznad	1.5
	D2	Développer l'interface frontend pour la gestion des dépôts avec formulaires et listes filtrables.	Celia Massioun	3

Suite à la page suivante

Table V.1 Backlog Sprint 2 - Description détaillée des tâches

Fonctionnalité	**ID**	**Tâche**	**Par qui**	**Durée (j)**
	D3	Implémenter le backend pour la gestion des dépôts, incluant la validation et les tests unitaires.	Celia Massioun	2.5
Gestion des articles, familles et sous-familles	E1	Rédiger les cas d'utilisation pour la gestion des articles, familles et sous-familles.	Celia Massioun	2
	E3	Développer l'interface frontend pour gérer articles, familles et sous-familles avec recherche et filtres.	Imene Meznad	3
	E4	Implémenter la logique backend pour ces entités avec validation et tests.	Imene Meznad	3
Gestion des clients	F1	Rédiger les cas d'utilisation pour la gestion des clients (création, modification, suppression, consultation).	Imene Meznad	1.5
	F2	Modéliser les processus via diagrammes de séquence UML.	Imene Meznad	1.5
	F3	Développer les interfaces frontend pour la gestion des clients.	Celia Massioun	2.5
	F4	Implémenter le backend associé et réaliser les tests.	Celia Massioun	2
Gestion des véhicules	G1	Rédiger les cas d'utilisation liés à la gestion des véhicules.	Celia Massioun	0.5
	G3	Développer l'interface frontend pour la gestion des véhicules.	Imene Meznad	1
	G4	Implémenter la logique backend pour les véhicules avec tests associés.	Imene Meznad	2
Gestion des entrées	H1	Rédiger les cas d'utilisation relatifs à la gestion des entrées en dépôt(liste des articles) .	Imene Meznad	1.5
	H2	Modéliser les séquences et interactions pour les entrées.	Imene Meznad	1.5
	H3	Développer l'interface utilisateur pour la gestion des entrées, incluant la validation et les notifications.	Celia Massioun	3
	H4	Implémenter le backend pour la gestion des entrées et réaliser les tests nécessaires.	Celia Massioun	2.5

V.3 Spécification

Dans cette phase, on va se concentrer sur les fonctionnalités principales et les interactions des différents utilisateurs avec la gestion des ressources dans le sprint 2.

V.3.1 Diagramme de cas d'utilisation - Sprint 2

Le diagramme de cas d'utilisation ci-dessous présente les fonctionnalités du Sprint 2, en mettant l'accent sur la gestion des ressources.

Les fonctionnalités du Sprint 1 sont intégrées de manière globale pour offrir une vue d'ensemble, mais ne sont pas détaillées, afin de maintenir la lisibilité du schéma.

Diagramme de cas d'utilisation sprint 2

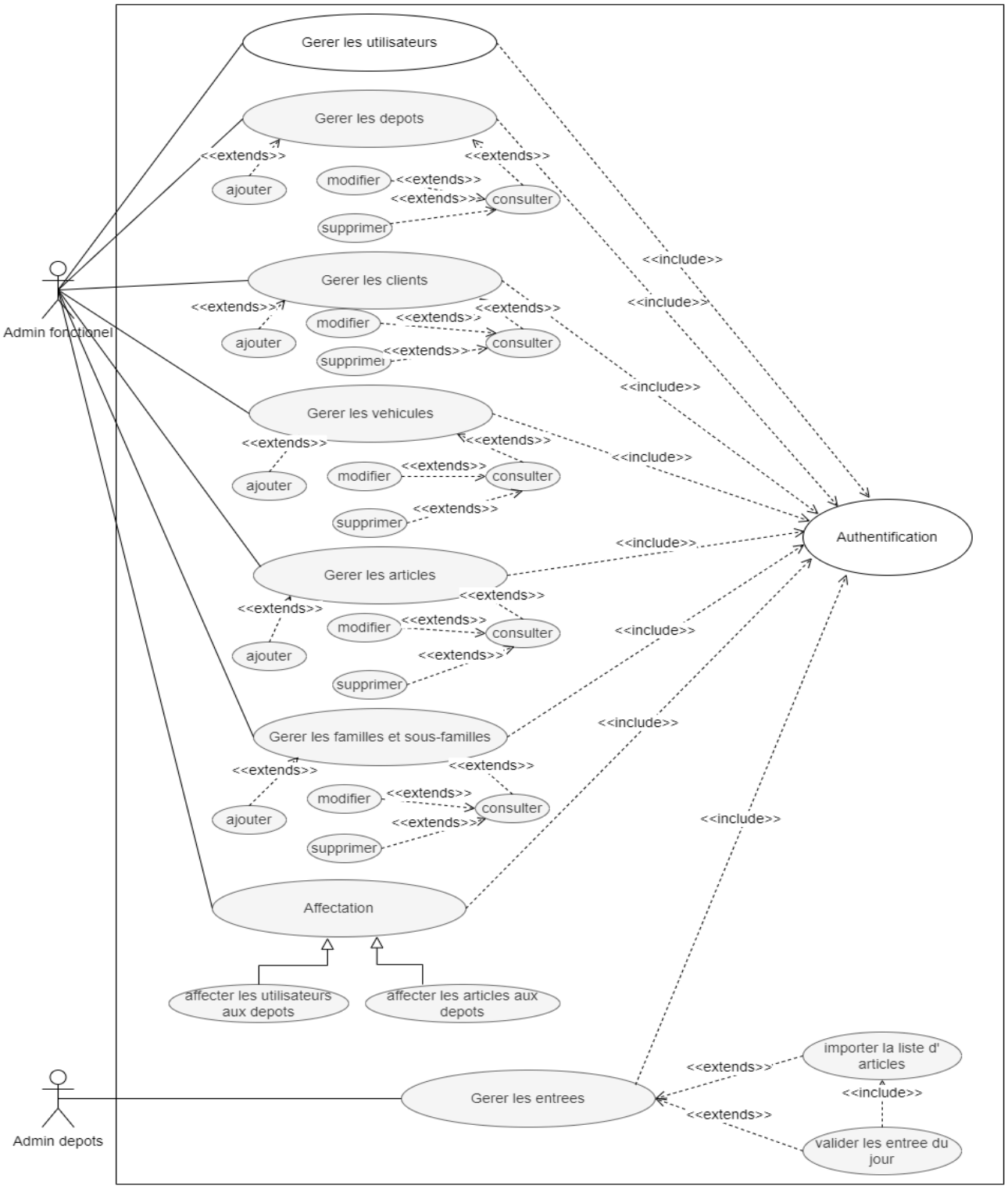

FIGURE V.1 – Diagramme de cas d'utilisation – Sprint 2

V.3.2 Description textuelle des cas d'utilisation - Sprint 2

Règles de gestion globale

Table V.2 – Règles de gestion du Sprint 2

Code	Règle de gestion
RG1	Seul l'Admin Fonctionnel peut gérer les utilisateurs, clients, familles, sous-familles, articles, véhicules et affectations.
RG2	Chaque dépôt doit avoir un code unique et une adresse obligatoire (région et wilaya).
RG3	Un client doit avoir un code unique, un nom et des coordonnées complètes.
RG4	Tout véhicule est identifié par une plaque d'immatriculation unique.
RG5	Un article doit appartenir à une famille/sous-famille, avoir un stock positif ou nul dans un dépôt, ou ne pas y exister.
RG6	Les familles/sous-familles ont des noms uniques. Une sous-famille ne peut appartenir qu'à une seule famille.
RG7	Toute entrée en stock doit être associée à un article valide, une quantité positive et une date.
RG8	Les articles ne peuvent être créés que si les familles et sous-familles sont déjà présentes dans le système.
RG9	La suppression d'une famille et sous-famille sont interdite si elle contient des sous-familles et des articles associés respectivement.
RG10	L'affectation d'un planificateur ou d'un administrateur dépôt est indispensable pour lui permettre d'effectuer les opérations liées uniquement à son dépôt.

Description textuelle des cas d'utilisation - Sprint 2

CU : Affectation

Table V.3 – Description textuelle du cas : Affectation

CU	Affectation (utilisateurs et articles aux dépôts)
Acteur	Admin Fonctionnel
Précondition	Les utilisateurs/articles et les dépôts existent dans le système.
Objectif	Lier les utilisateurs ou les articles à leurs dépôts respectifs.
Postcondition	— L'utilisateur avec le rôle Admin Dépôt est affecté à un seul dépôt. — L'utilisateur avec le rôle Planificateur est affecté à un ou plusieurs dépôts. — Un ou plusieurs articles sont affectés à un ou plusieurs dépôts.
Enchaînement principal	— L'Admin Fonctionnel accède à l'interface d'affectation. — Le système affiche la liste des utilisateurs. — L'Admin sélectionne un utilisateur. — Le système affiche la liste des dépôts disponibles. — Selon le rôle de l'utilisateur sélectionné : **utilisateur**Si l'utilisateur a le rôle **Admin Dépôt**, il ne peut être affecté qu'à un seul dépôt. Si l'utilisateur a le rôle **Planificateur**, il peut être affecté à un ou plusieurs dépôts. — Les dépôts déjà affectés à un autre utilisateur du même rôle sont affichés en mode désactivé. — Le système enregistre les affectations et affiche un message de confirmation. **article** — L'Admin sélectionne l'option "Affecter un article". — Le système affiche la liste des articles existants. — L'Admin sélectionne un ou plusieurs articles. — Le système affiche la liste des dépôts disponibles. — L'Admin sélectionne un ou plusieurs dépôts. — Le système enregistre l'affectation et affiche un message de confirmation.
Enchaînements alternatifs	— Si un utilisateur est déjà affecté à un dépôt, une nouvelle affectation est bloquée tant qu'il n'est pas dissocié. — Si aucun dépôt n'est sélectionné, l'action d'affectation est désactivée.

Exception	En cas de défaillance du système ou d'indisponibilité de la base de données, l'opération est annulée et un message d'erreur est affiché.

CU : Gérer les entrées

TABLE V.4 – Description textuelle du cas : Enregistrer une entrée multiple

CU	Gérer les entrées
Acteur	Admin Dépôts
Précondition	Les articles et les dépôts doivent être valides, et les articles doivent être déjà affectés à un ou plusieurs dépôts.
Objectif	Importer une liste d'articles accompagnés de leurs quantités reçues, afin de mettre à jour automatiquement le stock du dépôt concerné. Cette opération permet de gérer les entrées de stock de manière automatisée à partir d'une liste prédéfinie.
Postcondition	Les articles sont enregistrés avec leur date d'entrée, et les quantités sont mises à jour dans le stock du dépôt.
Enchaînement principal	1. L'Admin Dépôts importe une liste d'articles à partir d'un fichier Excel ou saisit manuellement les entrées reçues. 2. Le système lit chaque ligne et vérifie l'existence de l'article. 3. Chaque entrée est enregistrée avec la date d'entrée. 4. Le stock du dépôt est mis à jour automatiquement pour chaque article.
Enchaînement alternatif	- Si un article n'existe pas dans le système : rejet de la ligne et affichage d'un message d'erreur. - Si la quantité est inférieure à zéro : rejet de la ligne et message d'erreur. - Si le fichier est invalide : rejet total avec message d'erreur.

Exception	En cas de défaillance lors de la mise à jour de stock : annulation de l'opération et log d'erreur.

V.4 Analyse

V.4.1 Analyse du stock d'alerte et du stock maximum des articles dans chaque dépôt.

Dans le cadre de notre système de gestion des stocks, les seuils critiques tels que le **stock d'alerte** (`stockAlert`) et le **stock maximum** (`stockMax`) sont automatiquement calculés afin d'assurer une gestion optimale des articles au sein de chaque dépôt.

Le **stock d'alerte** représente la quantité minimale en stock à partir de laquelle une commande de réapprovisionnement doit être lancée afin d'**éviter toute rupture de stock**. Ce seuil est calculé selon la formule suivante :

$$Stock d'alerte = Consommation moyenne quotidienne \times Délai moyen d'approvisionnement$$

La consommation moyenne est déduite à partir des commandes clients sur une période donnée, tandis que le délai d'approvisionnement correspond au temps moyen entre deux commandes de réapprovisionnement.

De son côté, le **stock maximum** définit la quantité maximale qu'un dépôt peut stocker pour **éviter les surcharges** et les coûts associés au surstockage. Il est calculé en appliquant un **coefficient supérieur à 1** sur le stock d'alerte, par exemple :

$$Stock maximum = Stock d'alerte \times Facteur de sécurité (ex. 1{,}5 ou 2)$$

V.4.2 diagramme d'activité - Sprint 2

Ce diagramme offre une vue globale du flux d'exécution des activités telles que la gestion des entités (utilisateurs, clients, dépôts, véhicules, articles, familles et sous-familles), l'affectation des utilisateurs et la gestion des entrées. Il facilite ainsi la compréhension des séquences d'actions attendues et aide à identifier les transitions logiques entre les différentes opérations du système.

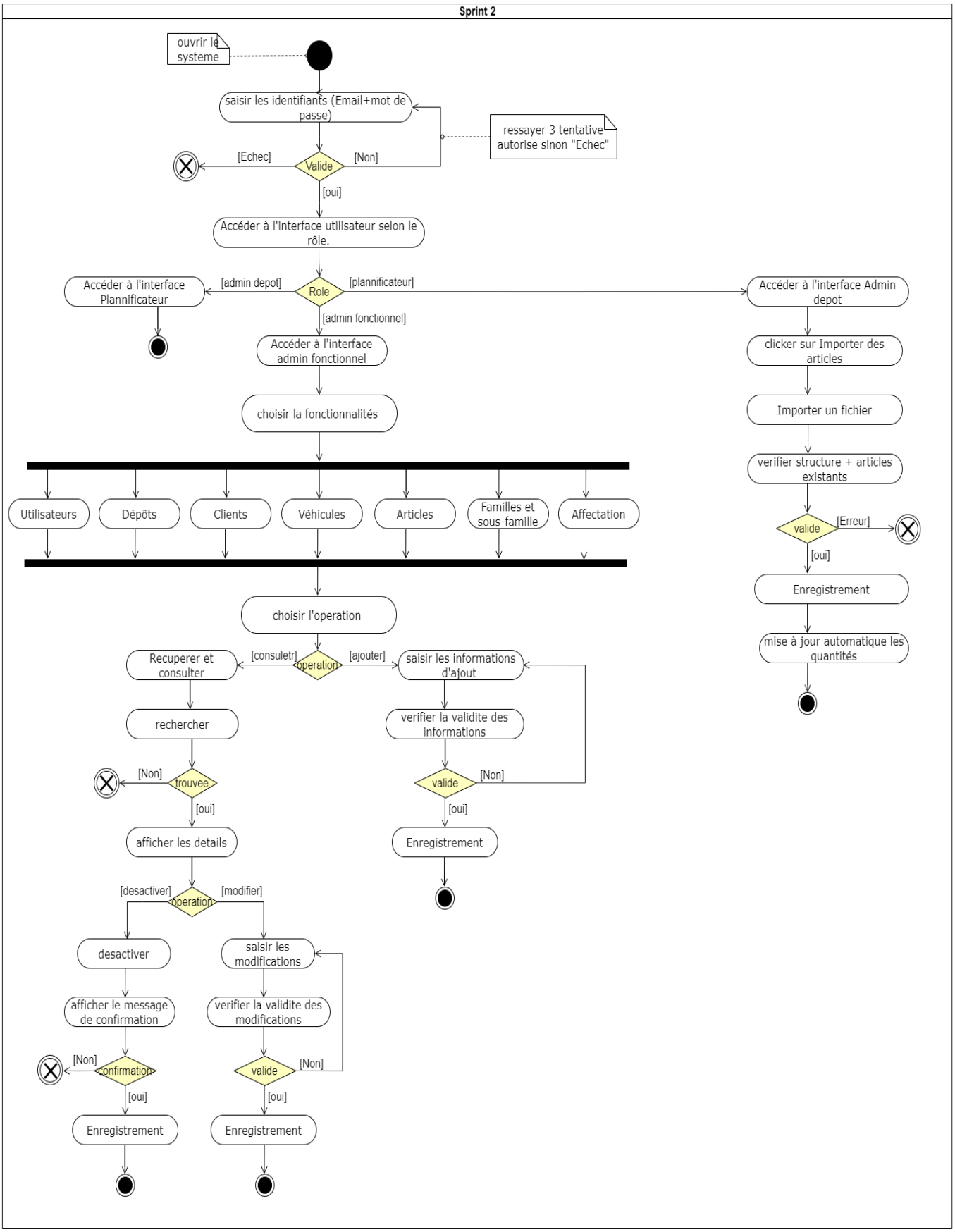

FIGURE V.2 – diagramme d'activité - Sprint 2

V.4.3 Les diagrammes de séquence système - Sprint 2

- **Diagramme de séquence système "Affecter un utilisateur à un dépôt"** :
  Ce diagramme Permet à l'Admin Fonctionnel d'affecter un ou plusieurs utilisateurs (ayant le rôle Admin Dépôt ou Planificateur) à un dépôt précis, afin de restreindre leur accès et visibilité uniquement à ce dépôt.

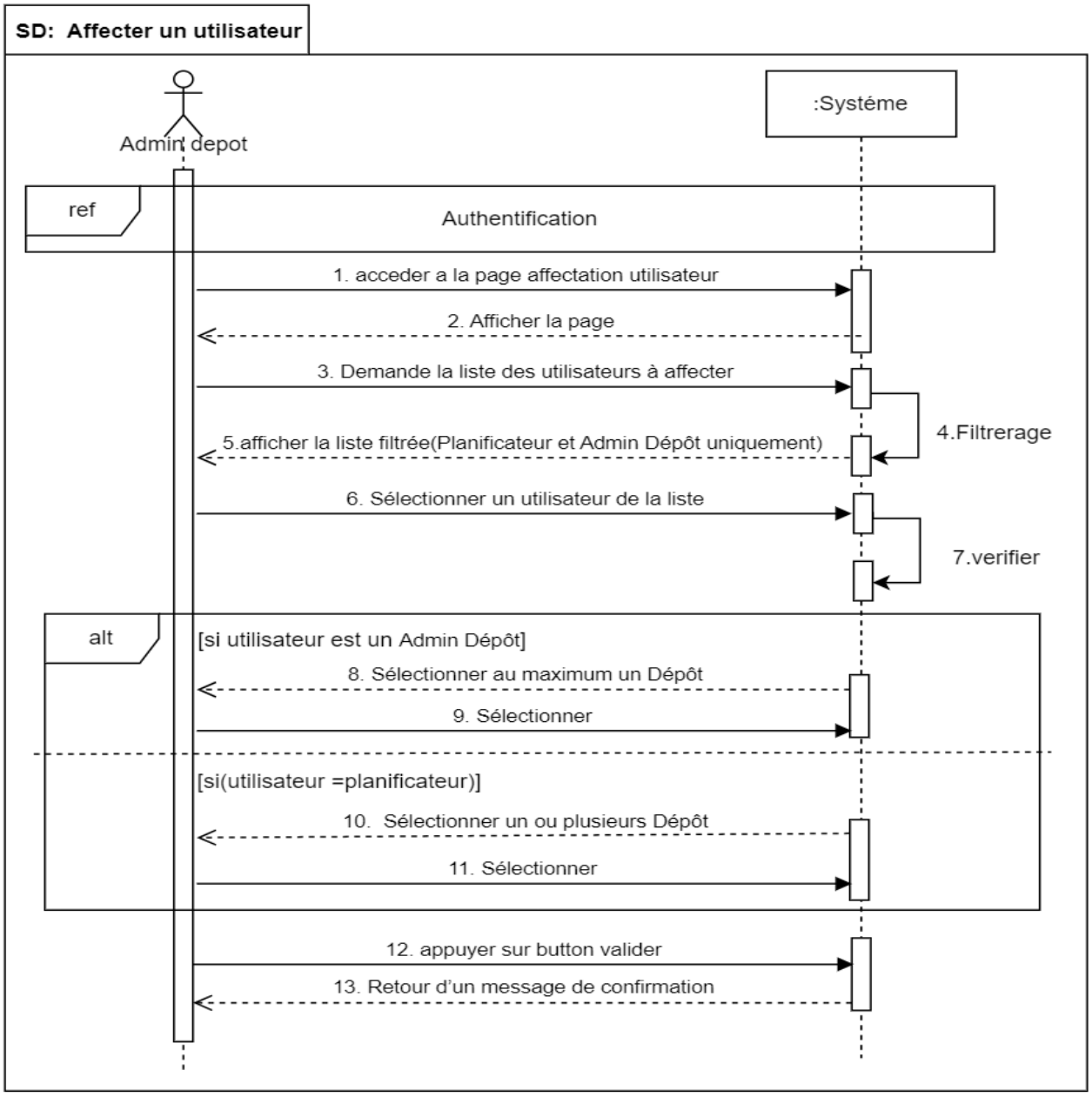

FIGURE V.3 – Diagramme de séquence système "Affecter un utilisateur à un dépôt"

V.5 Conception

V.5.1 Les diagrammes de séquence détaillés - Sprint 2

- **Diagramme de séquence détaillé " Gérer les entrées"** :
  Ce diagramme décrit le processus interne de gestion des entrées dans un dépôt. Un utilisateur autorisé accède à l'interface de saisie des entrées, sélectionne un article concerné et saisit les quantités reçues ou importe directement une liste d'entrées journalières via un fichier externe . L'interface transmet ces informations au depotController, qui effectue les vérifications nécessaires sur l'article (existence, validité des quantités) en interrogeant la table Articles. Si les données sont valides, l'entrée est enregistrée dans la table ArticleDepot, avec la date et le dépôt concerné. Un message de confirmation est retourné à l'utilisateur.

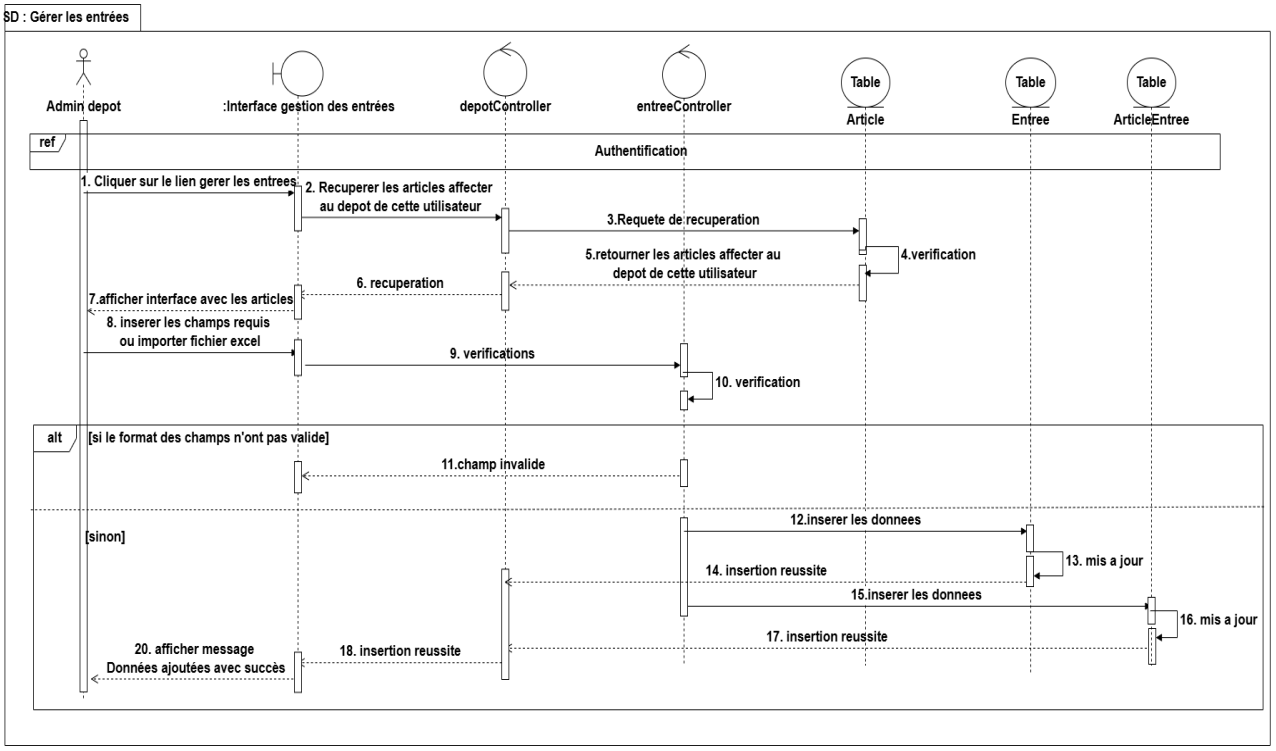

FIGURE V.4 – Diagramme de séquence détaillé " Gérer les entrées"

- **Diagramme de séquence détaillé "Affecter utilisateurs aux dépôts"** :

  Ce diagramme représente le processus détaillé d'affectation d'un utilisateur à un dépôt, effectué par l'Admin Fonctionnel. Après avoir accédé à l'interface dédiée, l'administrateur sélectionne un utilisateur depuis une liste générée par le"UtilisateurController", qui interroge la table Utilisateurs. Le système vérifie alors le rôle de l'utilisateur : si c'est un Admin Dépôt, un seul dépôt peut être affecté ; si c'est un Planificateur, plusieurs dépôts sont autorisés.

Le "DepotController" est ensuite sollicité pour fournir la liste des dépôts disponibles. Les dépôts déjà affectés à d'autres utilisateurs du même rôle sont désactivés dans l'interface. Lorsque l'Admin sélectionne un ou plusieurs dépôts valides, le système vérifie l'intégrité de la demande, puis enregistre l'affectation dans la table Affectationdepots. Enfin, un message de confirmation est renvoyé à l'interface pour notifier le succès de l'opération.

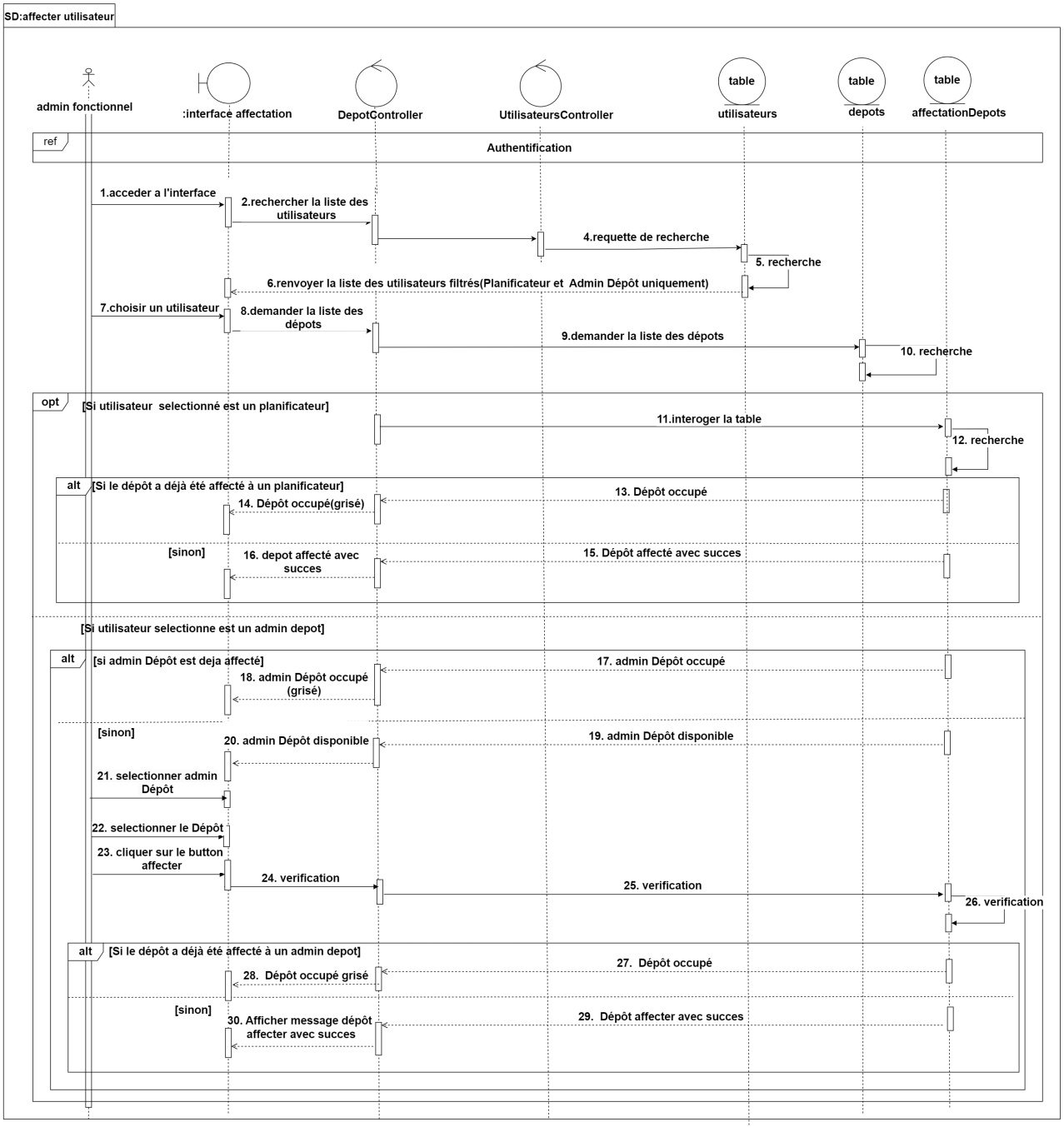

FIGURE V.5 – Diagramme de séquence détaillé "Affecter utilisateurs aux dépôts"

V.5.2 Diagramme de classes - Sprint 2

Le diagramme de classes modélise les entités principales du système de gestion des dépôts de CEVITAL. La classe *Admin Fonctionnel*, héritée de la classe *Utilisateur*, est responsable de la gestion des dépôts, des articles, des familles, des sous-familles et des clients. Les articles sont structurés de manière hiérarchique à travers les entités *Famille*, *SousFamille* et *Article*. Le stockage des articles dans les différents dépôts est assuré par la classe *ArticleDepot*, qui enregistre les quantités disponibles par dépôt ainsi que les informations relatives aux entrées, telles que la date et la quantité ajoutée. Ce modèle assure une organisation cohérente des données et une répartition claire des responsabilités.

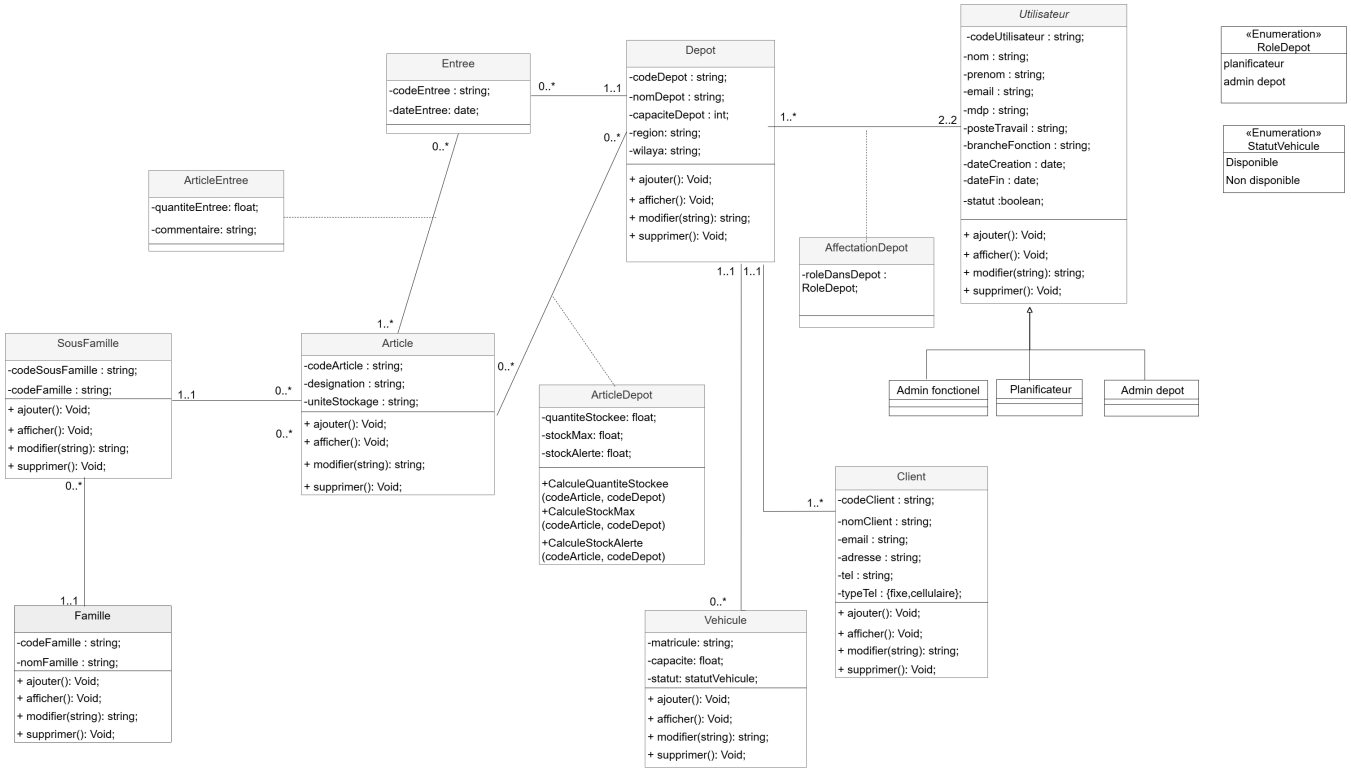

FIGURE V.6 – Diagramme de classes - Sprint 2

V.6 Implementation

V.6.1 Dictionnaire de données - Sprint 2

Le dictionnaire de données de sprint 2 :

TABLE V.5 – Dictionnaire de données de Sprint 2

Nom du champ	Type	Contraintes	Taille	Désignation
Table : AffectationDepot				

Nom du champ	Type	Contraintes	Taille	Désignation
codeUtilisateur	VARCHAR	Clé primaire partielle, clé étrangère	10	Identifiant de l'utilisateur affecté au dépôt .
codeDepot	VARCHAR	Clé primaire partielle, clé étrangère	10	Identifiant du dépôt concerné.
roleDansDepot	ENUM('Admin Depot', 'Planificateur Depot')	Valeurs prédéfinies	-	Rôle de l'utilisateur dans le dépôt
Table : Famille				
codeFamille	VARCHAR	Clé primaire, unique	10	Identifiant de la famille d'articles
nomFamille	VARCHAR	Non null	50	Nom de la famille d'articles
Table : SousFamille				
codeSousFamille	VARCHAR	Clé primaire, unique	10	Identifiant de la sous-famille
nomSousFamille	VARCHAR	Non null	50	Nom de la sous-famille
Table : Article				
codeArticle	VARCHAR	Clé primaire, unique	10	Identifiant de l'article
designation	VARCHAR	Non null	100	Désignation (nom) de l'article
uniteStockage	VARCHAR	Non null, par défaut = Palette	10	Unité de stockage utilisée (palette)
Table : Depot				
codeDepot	VARCHAR	Clé primaire, unique	10	Identifiant unique du dépôt
nomDepot	VARCHAR	Non null	100	Nom du dépôt
capaciteDepot	INT	Non null	-	Capacité maximale du dépôt
region	VARCHAR	Non null	50	Région géographique du dépôt

Nom du champ	Type	Contraintes	Taille	Désignation
wilaya	VARCHAR	Non null	50	Wilaya (division administrative) du dépôt
Table : ArticleDepot				
codeArticle	VARCHAR	Clé primaire partielle, clé étrangère	10	Identifiant de l'article stocké
codeDepot	VARCHAR	Clé primaire partielle, clé étrangère	10	Dépôt où l'article est stocké
quantiteStockee	FLOAT	Nullable	-	Quantité actuelle en stock
stockMax	FLOAT	Non null	-	Seuil maximum autorisé pour éviter les surcharges
stockAlerte	FLOAT	Non null	-	Seuil minimal pour anticiper les ruptures
Table : Vehicule				
matricule	VARCHAR	Clé primaire, unique	15	Numéro d'immatriculation du véhicule
capacite	INT	Non null	-	Capacité de chargement du véhicule
statut	ENUM('Disponible', 'Non-disponible')	Valeur par défaut = Disponible	-	Statut d'utilisation du véhicule
Table : Entree				
codeEntree	VARCHAR	Clé primaire, unique	15	Identifiant unique de l'entrée en stock
dateEntree	DATE	Non null	-	Date d'entrée des articles dans le stock
Table : ArticleEntree				
codeEntree	VARCHAR	Clé primaire partielle, clé étrangère	15	Référence de l'entrée liée
codeArticle	VARCHAR	Clé primaire partielle, clé étrangère	10	Article concerné par l'entrée

Nom du champ	Type	Contraintes	Taille	Désignation
quantiteEntree	FLOAT	Non null	-	Quantité ajoutée lors de l'entrée
commentaire	TEXT	Nullable	-	Remarque ou information complémentaire
Table : Client				
codeClient	VARCHAR	Clé primaire, unique	10	Identifiant du client
nomClient	VARCHAR	Non null	100	Nom du client
email	VARCHAR	Non null	100	Adresse email du client
adresse	VARCHAR	Non null	200	Adresse complète du client
typeTel	ENUM('Cellulaire', 'Fixe')	Valeurs prédéfinies	20	Type de numéro de téléphone
tel	VARCHAR	Non null	15	Numéro de téléphone du client

V.6.2 Modèle relationnel de données - Sprint 2

Le modèle relationnel correspondant au dictionnaire de données présenté ci-dessus est le suivant :

AffectationDepot(#codeUtilisateur, #codeDepot, roleDansDepot)

Famille(codeFamille, nomFamille)

SousFamille(codeSousFamille, nomSousFamille,#codeFamille)

Article(codeArticle, designation, statut, uniteStockage,#codeSousFamille)

Depot(codeDepot, nomDepot, capaciteDepot, region, wilaya)

ArticleDepot(#codeArticle, #codeDepot, quantiteStockee, stockMax, stockAlerte)

Vehicule(matricule, capacite, statut,#codeDepot)

Entree(codeEntree, dateEntree, #codeDepot)

ArticleEntree(#codeEntree, #codeArticle, quantiteEntree, commentaire)

Client(codeClient, nomClient, email, adresse, typeTel, tel,#codeDepot)

V.7 Test & presentation d'interfaces

Interface d'affectation

Interface permettant d'affecter un utilisateur (planificateur ou administrateur dépôt) à un ou plusieurs dépôts, afin de restreindre et organiser ses droits d'accès pour l'exécution des opérations liées à ses dépôts.

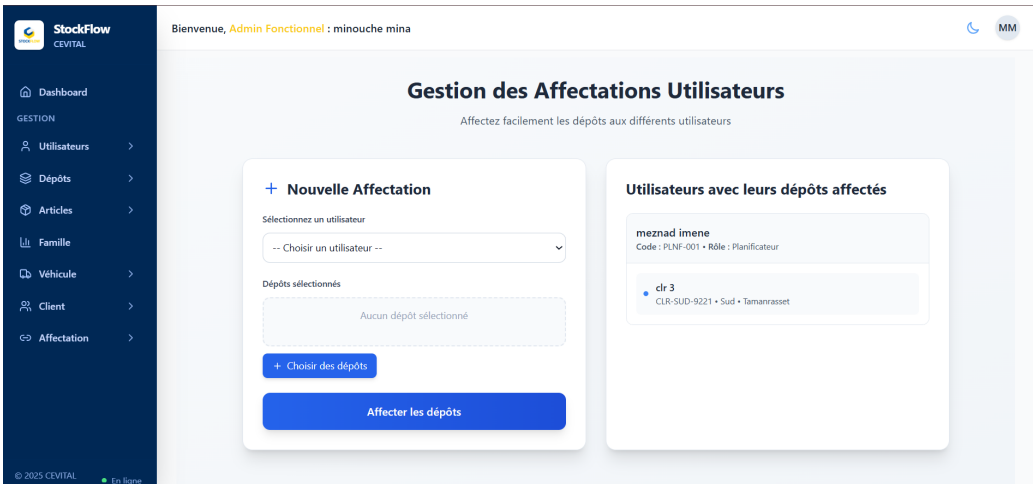

FIGURE V.7 – Interface d'affectation des utilisateurs aux dépôts

.

V.8 Bilan du Sprint 2

Revue de Sprint : Toutes les fonctionnalités clés (gestion des dépôts, articles, familles/sous-familles, clients, véhicules et entrées) ont été livrées. L'interface d'affectation (Fig. V.5) et le module d'entrées de stock sont opérationnels, avec une couverture de tests à 85%.

Rétrospective : Points forts : modélisation UML précise (Fig. V.6) et intégration réussie des contraintes métier (RG1-RG10). Axes d'amélioration : optimiser les temps de réponse des requêtes complexes et renforcer les tests de charge.

V.9 Conclusion

Ce deuxième sprint a permis de structurer efficacement la gestion des articles, des familles, des dépôts ainsi que le processus d'entrée de stock. Grâce aux diagrammes de classes, au modèle relationnel et au dictionnaire de données, une base solide a été mise en place pour assurer la cohérence et l'intégrité des données. Cette modélisation prépare le terrain pour les fonctionnalités plus avancées à venir, notamment la gestion des sorties et des transferts de stock, qui seront abordées dans le sprint suivant.

Chapitre VI

Sprint trois

Introduction

Le **Sprint 3** est dédié à la mise en place de la **gestion logistique avancée** du système. Il intègre les modules essentiels liés à la **gestion des commandes**, au **réapprovisionnement**, à la **livraison**, ainsi qu'à la **gestion des véhicules**.

Ce sprint a pour objectif d'assurer la **fluidité du processus**, depuis la commande du client jusqu'à la livraison finale, tout en garantissant la **disponibilité des stocks** et des **moyens de transport** nécessaires.

Les **principaux utilisateurs** concernés par ce sprint sont :

- **Le Planificateur**, qui se charge de la **gestion des réapprovisionnements** et de la **planification des sorties des commandes clients**.
- **L'Admin Dépôt**, qui s'occupe du **suivi des livraisons**, notamment de la **mise à jour des statuts de livraison**.

VI.1 Backlog du Sprint 3

Le backlog du Sprint 3 couvre les fonctionnalités logistiques clés telles que la gestion des commandes, du réapprovisionnement et des livraisons.

Table VI.1 – Backlog Sprint 3 - Description détaillée des tâches

Fonctionnalité	ID	Tâche	Par qui	Durée (j)
Gestion des commandes	G1	Rédiger le cas d'utilisation du Planificateur Dépôt, en incluant l'importation des commandes clients ainsi que leur planification.	Imene Meznad	0.5
	G2	Élaborer les diagrammes de séquence UML des cas d'utilisation.	Imene Meznad	2
	G3	Développer des interfaces utilisateur pour l'importation et la planification des commandes.	Celia Massioun	3
	G4	Implémenter la logique back-end pour l'importation et la planification des commandes, incluant la vérification des stocks, le partage équitable selon les stocks disponibles et les quantités commandées, ainsi que la gestion de la disponibilité et de l'indisponibilité des véhicules en fonction des horaires.	Celia Massioun	2.5
Gestion du réapprovisionnement	H1	Définir le cas d'utilisation du réapprovisionnement manuel pour un ou plusieurs articles.	Celia Massioun	0.5

Suite à la page suivante

Table VI.1 – suite

Fonctionnalité	ID	Tâche	Par qui	Durée (j)
	H2	Créer les diagrammes de séquence associés aux scénarios de réapprovisionnement.	Celia Massioun	1.5
	H3	Développer l'interface et le suivi du réapprovisionnement.	Imene Meznad	2.5
	H4	Implémenter les règles backend pour déclencher le réapprovisionnement.	Imene Meznad	2
Gerer les livraisons	I1	Rédiger les cas d'utilisation pour le gestion des livraisons Rédiger les cas d'utilisation pour la gestion des livraisons, incluant les états de livraison et la génération des documents associés.	Imene Meznad	0.5
	I2	Modéliser les interactions via diagramme de séquence.	Imene Meznad	0.5
	I3	Implémenter l'interface utilisateur pour le traitement des livraisons, incluant la gestion des états (« retournée »,« annulée », « livrée ») ainsi que la génération des documents associés.	Celia Massioun	1.5
	I4	Implémenter les traitements backend liés à l'état de livraison.	Celia Massioun	1

VI.2 Spécification

Dans cette phase, nous nous concentrons sur les fonctionnalités principales du Sprint 3, qui couvrent la gestion des commandes et des livraisons.

VI.2.1 Diagramme de cas d'utilisation - Sprint 3

Le diagramme de cas d'utilisation suivant illustre les principales interactions des acteurs avec le système dans le cadre du Sprint 3. Il met en évidence les processus liés à la planification des commandes clients, aux opérations de réapprovisionnement des dépôts, et à la gestion des livraisons.

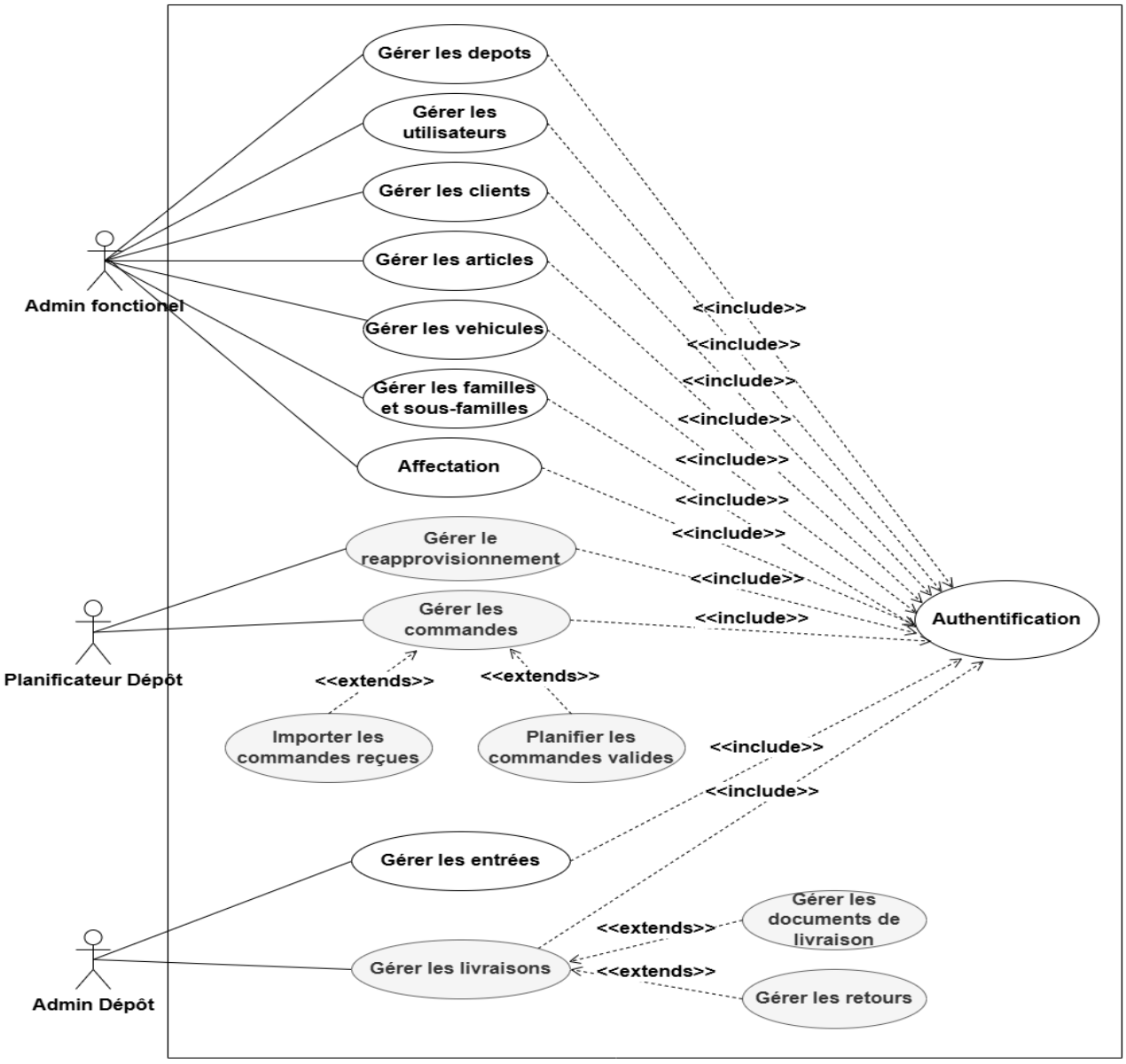

FIGURE VI.1 – Diagramme de cas d'utilisation – Sprint 3

VI.2.2 Description textuelle des cas d'utilisation - Sprint 3

Règles de gestion globale

TABLE VI.2 – Règles de gestion globale – Sprint 3

Code	Règle de gestion
RG1	Une commande doit être liée à un client existant et comporter un ou plusieurs articles existants, une quantité strictement positive et une date de commande.

Code	Règle de gestion
RG2	Le réapprovisionnement peut être déclenché manuellement lorsque le stock disponible est inférieur au stock d'alerte défini pour chaque article.
RG3	La planification d'une commande ne peut être effectuée que si la commande est validée, que les articles sont en stock suffisant, et qu'un véhicule est disponible.
RG4	Une livraison nécessite une commande planifiée, un véhicule affecté, et un bon de livraison généré.
RG5	La quantité livrée pour un article ne peut jamais dépasser la quantité commandée.
RG6	La somme des quantités transportées par les différents véhicules pour une commande client doit être égale à la quantité à livrer.
RG7	La quantité transportée par un véhicule ne doit pas dépasser sa capacité maximale.
RG8	Le stock est mis à jour automatiquement après chaque réapprovisionnement ou chaque livraison confirmée.
RG9	Seuls les utilisateurs authentifiés avec le rôle **planificateur dépôt** peuvent effectuer les opérations de planification des commandes et de réapprovisionnement.
RG10	Seuls les utilisateurs authentifiés avec le rôle **administrateur dépôt** peuvent gérer, suivre ou clôturer les livraisons.

Description textuelle des cas d'utilisation - Sprint 3

CU : Gérer les commandes

TABLE VI.3 – Description textuelle du cas : Gérer les commandes.

CU	Gérer les commandes
Acteur	Planificateur
Précondition	Le planificateur est authentifié et a accès aux données de ses dépôt.
Objectif	Permettre d'importer les commandes clients, de les analyser par rapport au stock disponible, d'attribuer les quantités et de planifier les livraisons.
Postcondition	Les commandes sont validées, réparties selon les disponibilités, et les livraisons sont planifiées.

Enchaînement principal	**1. Importer les commandes** 1.1 Le planificateur accède à l'interface d'import des commandes. 1.2 Il importe un fichier Excel. 1.3 Le système filtre les commandes du jour selon le dépôt concerné et les importe. **2. Comparer avec le stock** 2.1 Le système vérifie la disponibilité des articles demandés pour chaque commande. 2.2 Si le stock est suffisant, les quantités sont directement affectées. 2.3 Si le stock est insuffisant, un partage équitable est généré automatiquement. **3. Ajuster la répartition et valider** 3.1 Le planificateur visualise les répartitions proposées. 3.2 Il peut modifier manuellement la répartition. **4. Planifier la livraison** 4.1 Le planificateur programme la date prévue de livraison et les véhicules à affecter.. **4.2 Il valide le planning, qui sera ensuite transmis à l'administrateur de dépôt concerné.**
Enchaînement alternatif	- Si le fichier Excel contient des données invalides ou un format incorrect, un message d'erreur s'affiche et les données sont ignorées. - Si une commande contient un article ou client inexistant, elle est ignorée. -Les commandes qui ne concernent pas les dépôts du planificateur authentifié sont ignorées.
Exception	En cas de rupture de stock ou de problème système, le processus est interrompu et l'utilisateur est notifié.

CU : Gérer le réapprovisionnement

Table VI.4 – Description textuelle du cas : Gérer le réapprovisionnement.

CU	Gérer le réapprovisionnement
Acteur	Planificateur
Précondition	Le planificateur est authentifié et a accès aux données de son dépôt.

Objectif	Le planificateur déclenche un réapprovisionnement lorsque le stock est inférieur au seuil d'alerte.
Postcondition	Le stock est mis à jour à la réception des articles, qui sont considérés comme des entrées.
Enchaînement principal	**1. Créer une demande** 1.1 Le planificateur accède à l'interface de réapprovisionnement. 1.2 Il sélectionne l'article, la quantité, et soumet la demande. 1.3 Le système enregistre la demande. **2. Réception des articles** 2.1 Le système enregistre une entrée de stock pour l'article concerné.
Enchaînement alternatif	- Si l'article n'est pas reconnu, la demande est refusée. - Si la quantité demandée est invalide (négative, nulle ou non numérique), un message d'erreur s'affiche et la ligne concernée est ignorée.
Exception	Si une erreur système empêche le traitement de la demande, celle-ci est annulée et le planificateur en est informé.

CU : Gérer les livraisons

TABLE VI.5 – Description textuelle du cas : Gérer les livraisons.

CU	Gérer les livraisons
Acteur	Administrateur de dépôt
Précondition	L'administrateur de dépôt est authentifié et a reçu le programme de livraison, établi par le planificateur, concernant son dépôt.
Objectif	Préparer les articles à livrer, mettre à jour le stock, charger les véhicules, générer les bons de livraison et gérer les éventuels retours ou annulations via des bons de retour.
Postcondition	Les commandes sont livrées ou retournées selon les cas, le stock est mis à jour, les bons correspondants sont générés et les véhicules sont libérés après traitement.

Enchaînement principal	**1. Préparation des commandes** 1.1 L'administrateur de dépôt consulte la planification des livraisons du jour. 1.2 Il prépare les articles à livrer conformément à cette planification. 1.3 Les véhicules sont chargés avec les quantités affectées. **2. Livraison et confirmation** 2.1 Après livraison, l'administrateur confirme la réception du bon de livraison signé et mettre à jour l'état de la commande à « Livrée ». 2.2 Le véhicule utilisé est marqué comme « Disponible ». **3. Gestion des retours ou annulations** 3.1 En cas de retour d'articles (partiel ou total), l'administrateur enregistre un bon de retour. 3.2 Les articles retournés sont ajoutés au stock comme nouvelles entrées. 3.3 En cas d'annulation de commande, les quantités non livrées sont remises en stock et un bon de retour est généré. **4. Génération des documents** 4.1 Le système génère un bon de livraison pour chaque commande prévue à la livraison. 4.2 Un bon de retour est généré automatiquement en cas de retour ou d'annulation.
Enchaînement alternatif	- Si un article préparé est incorrect ou manquant, l'administrateur peut ajuster la commande avant validation. - Si le véhicule prévu n'est pas disponible, un autre véhicule peut être affecté. - Si un bon de livraison est refusé par le client, un retour est déclenché et un bon de retour est généré.
Exception	En cas de panne du système ou de rupture de stock imprévue, la livraison peut être annulée ou reportée. L'administrateur est notifié.

VI.3 Analyse

VI.3.1 diagramme d'activité - Sprint 3

Ce diagramme offre une vue globale du flux d'exécution des activités telles que la gestion des commandes clients, le contrôle des stocks, la planification des livraisons, la gestion du réapprovisionnement.

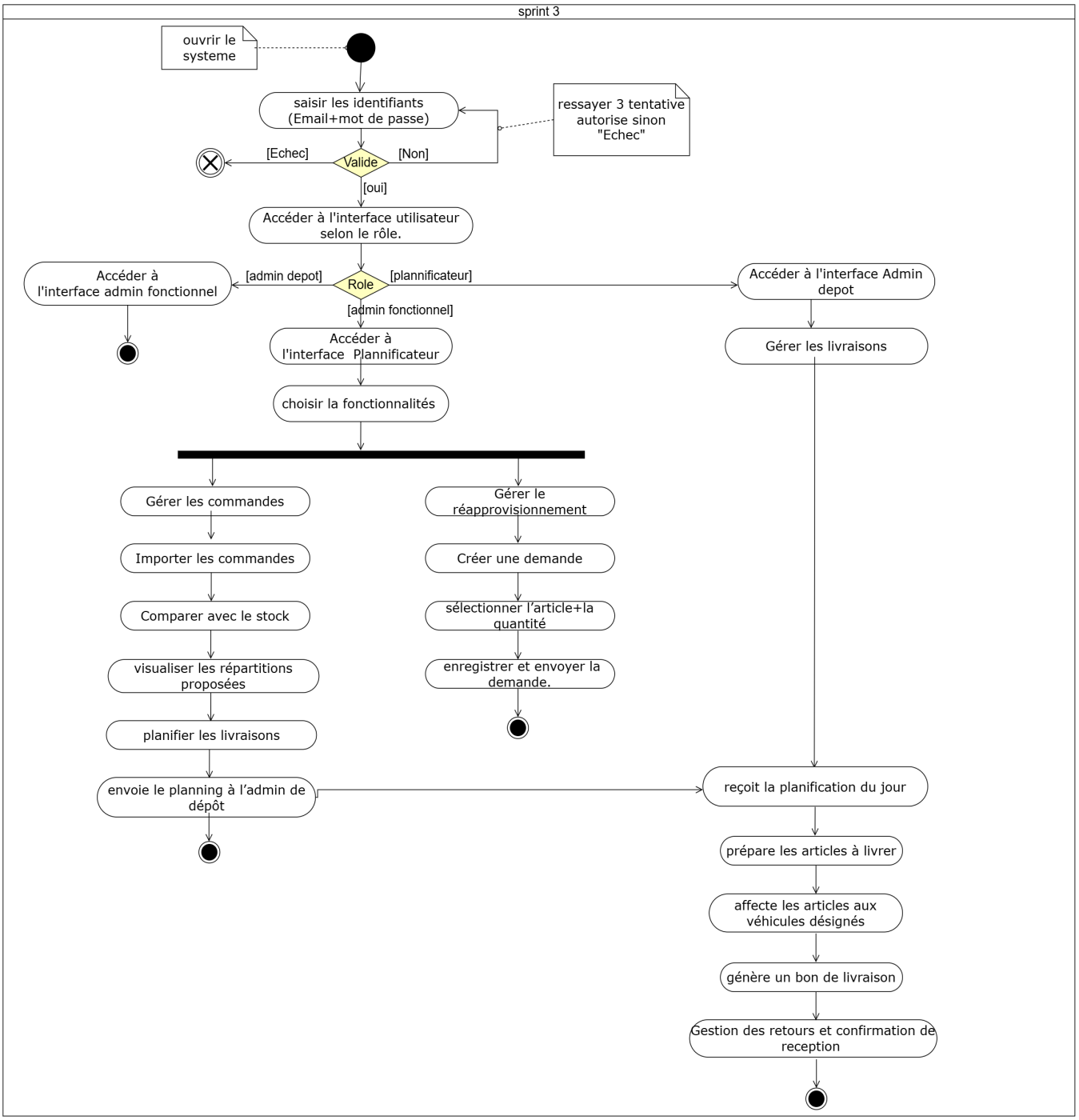

FIGURE VI.2 – diagramme d'activité - Sprint 3

VI.3.2 Les daiagrammes de séquence système - Sprint 3

Nous avons choisi de modéliser le cas de la gestion des livraisons. Ce diagramme montre l'échange de messages entre l'utilisateur et le système pour accomplir cette tâche.

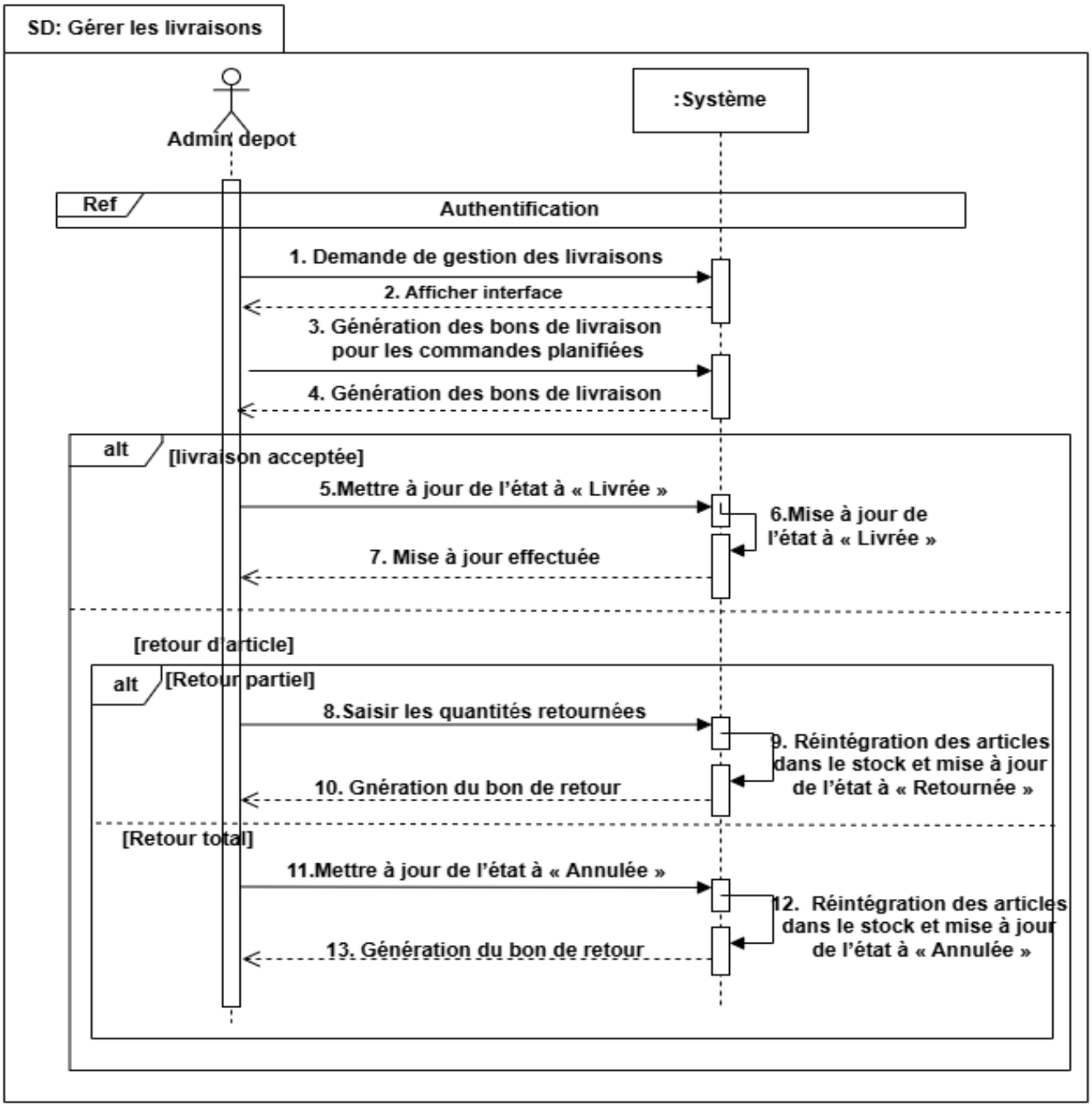

FIGURE VI.3 – diagramme séquence système - Gérer les livraisons

VI.4 Conception

VI.4.1 Les diagrammes de séquence détaillés - Sprint 3

Ces diagrammes décrivent, pour les fonctionnalité clé (Planifier commande, Importer commandes), la manière dont l'utilisateur interagit avec le système au fil du temps d'une maniere détaillés . Ils permettent d'illustrer les messages échangés, les données transmises, ainsi que les différentes interfaces impliquées.

- **Diagramme de séquence détaillé "Planifier commande"** :
  Ce diagramme illustre le processus de planification d'une commande client par un planificateur. Après authentification, celui-ci accède aux commandes liées à ses dépôts ainsi qu'aux suggestions de quantités à livrer, réparties équitablement pour chaque article en fonction du stock disponible. Le planificateur affecte ensuite les véhicules et précise les quantités à transporter, en tenant compte de la plage horaire disponible, du stock actuel et de la capacité de chaque véhicule. Le système vérifie la validité des données (dates, stock, capacités) avant d'enregistrer la planification et d'afficher un message de confirmation.

- **Diagramme de séquence détaillé "Imorter commandes"** :
  Ce diagramme illustre le processus d'importation des commandes client par un planificateur de dépôt. Après authentification, il importe un fichier Excel envoyé par le keep contact. Le système effectue alors plusieurs vérifications : présence des champs requis, validité des quantités, existence des articles et des clients, et absence de doublons. Les lignes incorrectes sont rejetées avec un message d'erreur. Si toutes les données sont valides, elles sont enregistrées dans la base de données, et un message confirme le succès de l'importation.

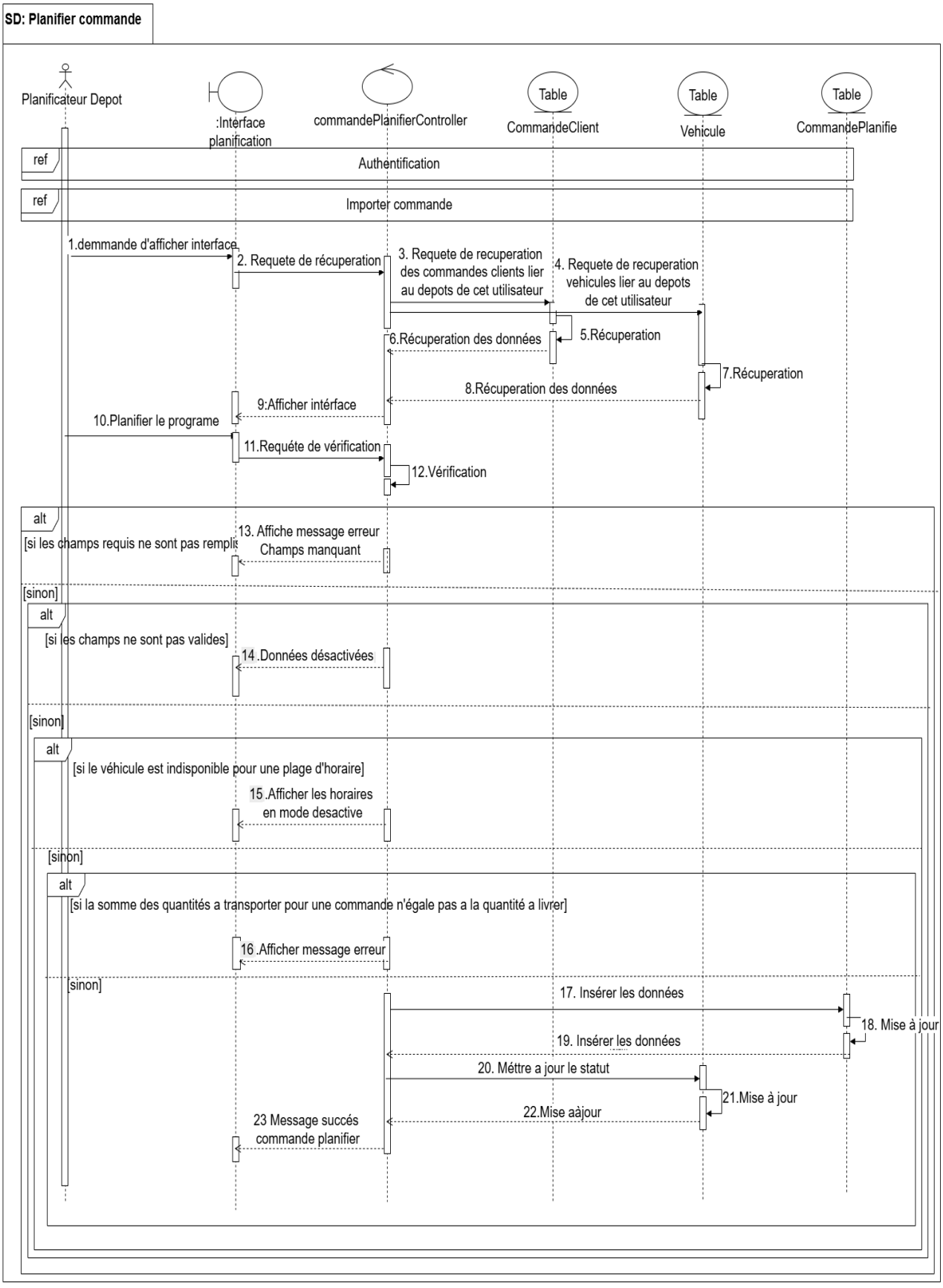

Figure VI.4 – Diagramme de séquence détaillé "Planifier commande"

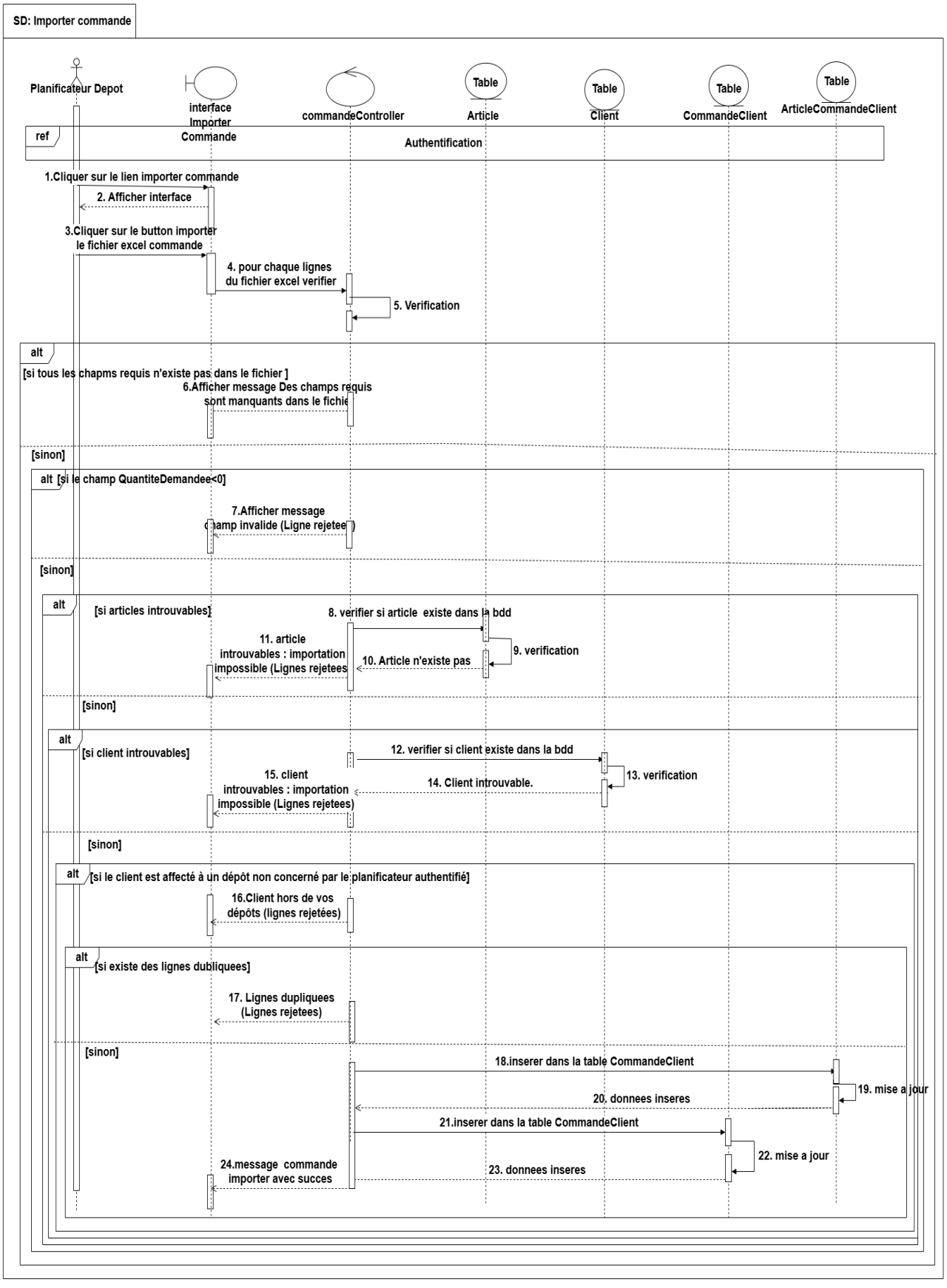

FIGURE VI.5 – Diagramme de séquence détaillé "Importer commandes"

VI.4.2 Diagramme de classes - Sprint 3

Le diagramme de classes du Sprint 3 modélise la gestion des commandes, livraisons, retours et réapprovisionnements. Les entrées d'articles sont enregistrées via **Entree** et **ArticleEntree**. Les clients passent des commandes (**CommandeClient**) contenant des articles (**ArticleCommandeClient**), qui sont planifiées pour livraison via **CommandePlanifie**. Les livraisons et retours sont suivis à l'aide des classes **BonLivraison** et **BonRetour**. Enfin, les dépôts peuvent effectuer des commandes de réapprovisionnement à travers **CommandeReapprovisionnement** et **ArticleCommandeReappro**.

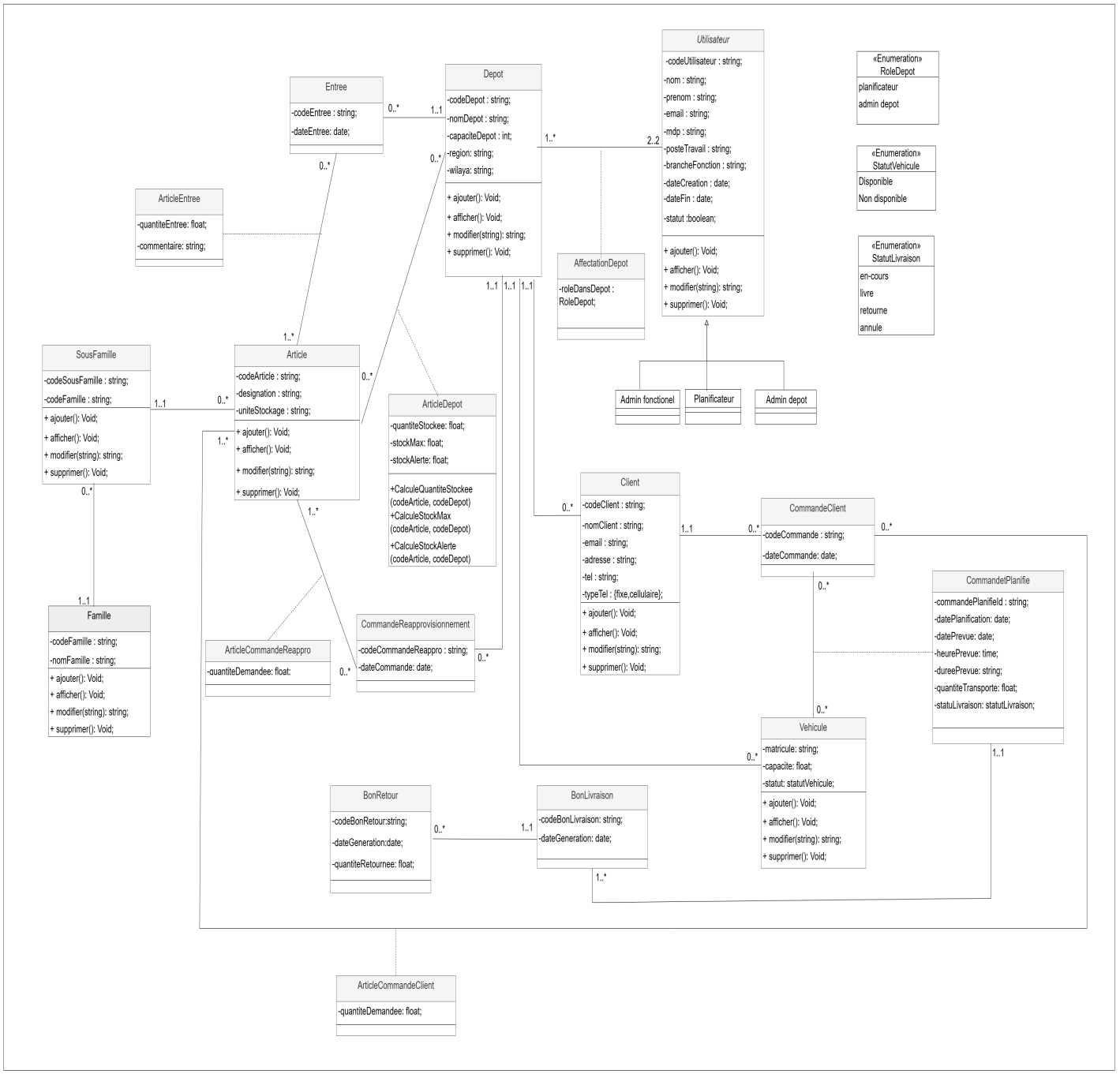

FIGURE VI.6 – diagramme de classe SPRINT 3 et Final

VI.5 Implementation

VI.5.1 Dictionnaire de données - Sprint 3

Le dictionnaire de données de sprint 3 :

Table VI.6 – Dictionnaire de données des tables – Commandes et Livraisons

Nom du champ	Type	Contraintes	Taille	Désignation
Table : CommandeClient				
codeCommande	VARCHAR	Clé primaire, unique	10	Identifiant unique de la commande client
dateCommande	DATE	Non null	-	Date de création de la commande client
Table : ArticleCommandeClient				
codeCommande	VARCHAR	Clé primaire partielle, clé étrangère	10	Référence de la commande client associée
codeArticle	VARCHAR	Clé primaire partielle, clé étrangère	10	Référence de l'article commandé
quantiteDemandee	FLOAT	Non null	-	Quantité d'article demandée dans la commande
Table : CommandePlanifie				
commandePlanifieId	VARCHAR	Clé primaire, unique	10	Identifiant de la commande planifiée
datePlanification	DATE	Non null, Défaut = CURRENT_DATE	-	Date de planification de la livraison
datePrevue	DATE	Non null	-	Date prévue de livraison
heurePrevue	TIME	Non null	-	Heure prévue de livraison
dureePrevue	STRING	Non null	10	Durée estimée de la livraison

Nom du champ	Type	Contraintes	Taille	Désignation
quantiteTransporte	FLOAT	Non null	-	Quantité transportée dans cette planification par vehicule
statutLivraison	ENUM	Non null	10	Statut de la livraison (En cours, Livre, Retourne)
Table : BonLivraison				
codeBonLivraison	VARCHAR	Clé primaire, unique	10	Identifiant du bon de livraison
dateGeneration	DATE	Non null, Défaut = CURRENT_DATE	-	Date de génération du bon de livraison
Table : BonRetour				
codeBonRetour	VARCHAR	Clé primaire, unique	10	Identifiant du bon de retour
dateGeneration	DATE	Non null, Défaut = CURRENT_DATE	-	Date de génération du bon de retour
Table : CommandeReapprovisionnement				
codeCommandeReappro	VARCHAR	Clé primaire, unique	10	Identifiant de la commande de réapprovisionnement
dateCommande	DATE	Non null	-	Date de la commande de réapprovisionnement
Table : ArticleCommandeReappro				
codeArticle	VARCHAR	Clé primaire partielle, clé étrangère	10	Identifiant de l'article réapprovisionné
codeCommandeReappro	VARCHAR	Clé primaire partielle, clé étrangère	10	Référence de la commande de réapprovisionnement associée

Nom du champ	Type	Contraintes	Taille	Désignation
quantiteDemandee	FLOAT	Non null	-	Quantité d'article commandée pour réapprovisionnement

VI.5.2 Modèle relationnel de données - Sprint 3

Le modèle relationnel du Sprint 3 est représenté ci-dessous :

Entree(codeEntree, dateEntree,#codeDepot)

ArticleEntree(#codeEntree, #codeArticle,quantiteEntree, commentaire)

CommandeClient(codeCommande, dateCommande,#codeClient)

CommandePlanifie(codePlanifieId,#codeCommande,#matricule,datePlanification,datePrevue, heurePrevue,dureePrevue,quantiteTransporte,statutLivraison)

ArticleCommandeClient(#codeArticle, #codeCommandeClient, quantiteDemandee)

CommandeReapprovisionnement(codeCommandeReappro, dateCommande,#codeDepot)

ArticleCommandeReappro(#codeCommandeReappro, #codeArticle, quantiteDemandee)

BonLivraison(codeBonLivraison, dateGeneration,#codePlanifieId)

BonRetour(codeBonRetour, dateGeneration, quantiteRetournee,#codeBonLivraison)

VI.6 Test & presentation d'interfaces

Interface de planification

Interface permettant de planifier les livraisons à partir des commandes validées. Le système compare d'abord les quantités demandées pour chaque article avec les quantités disponibles dans le dépôt principal (CLR), puis permet de planifier la livraison des commandes client par client. La planification inclut la sélection du véhicule de transport, la date et l'heure de départ, ainsi que le dépôt de départ et la destination.

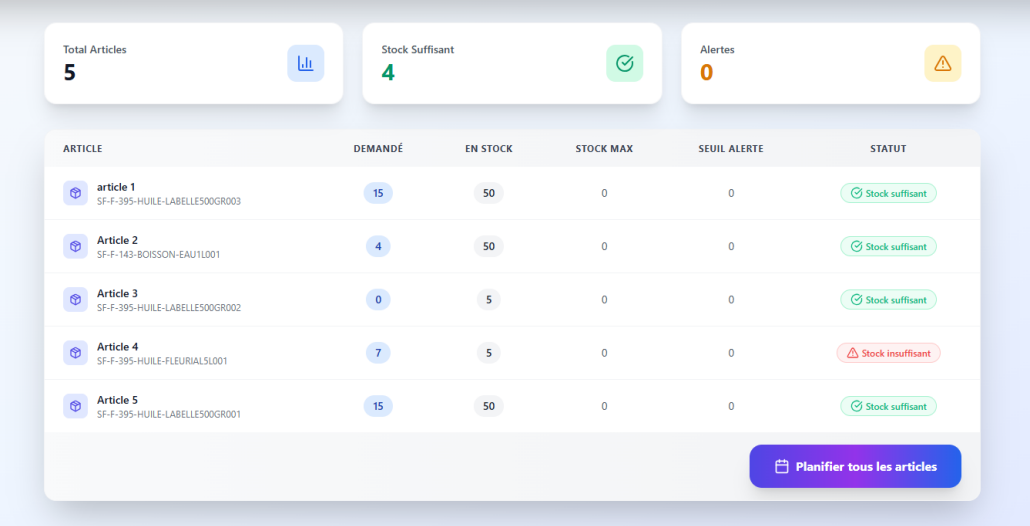

FIGURE VI.7 – Interface de comparaison des quantités

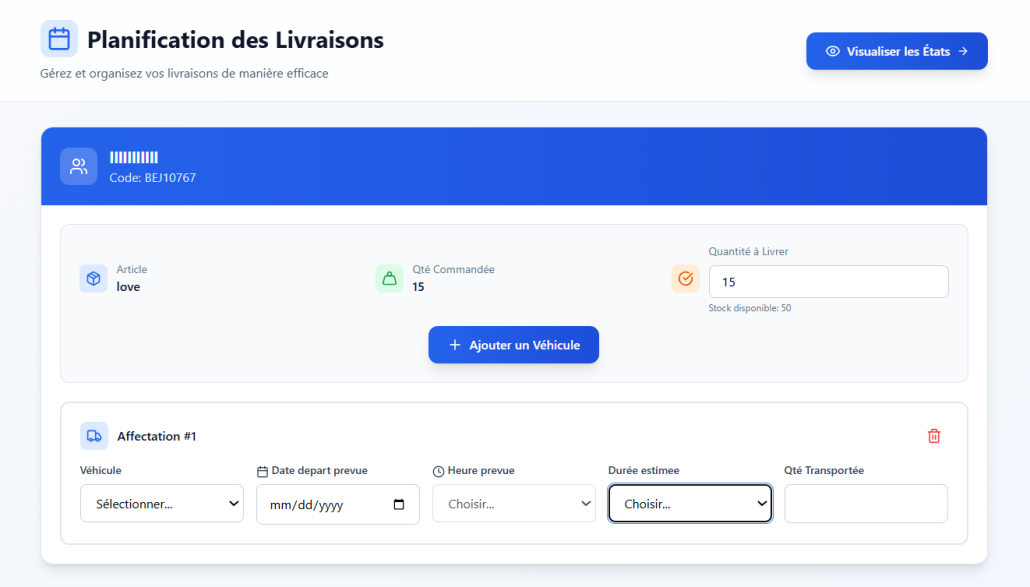

FIGURE VI.8 – Interface de planification des livraisons

VI.7 Bilan du Sprint 3

Revue de Sprint : Toutes les fonctionnalités clés (gestion des commandes, réapprovisionnement et livraisons) ont été implémentées avec succès. Les interfaces de planification (Fig. VI.7) et de suivi des livraisons (Fig. VI.8) sont opérationnelles, avec une couverture de tests à 88%. Les règles métier (RG1-RG10) ont été respectées.

Rétrospective : Points forts : Modélisation précise des flux logistiques (Fig. VI.6) et intégration réussie des contraintes complexes (partage équitable des stocks). Axes d'amélioration : Optimiser les performances des requêtes multi-tables et renforcer les tests de charge sur les modules de planification.

VI.8 Conclusion

Ce troisième sprint a permis de structurer efficacement la gestion des commandes clients, leur planification ainsi que le traitement des demandes de réapprovisionnement déclenchées par des alertes de seuil de stock. L'ensemble des éléments modélisés – incluant les diagrammes de classes, le modèle relationnel et le dictionnaire de données – contribue à assurer une traçabilité rigoureuse et une réactivité optimale face aux besoins des clients. Ce travail pose les bases nécessaires pour le développement du système de prévision des réapprovisionnements, qui sera abordé dans le sprint suivant.

Chapitre VII

Sprint quatre

VII.1 Introduction

Le Sprint 4 est dédié à l'intégration de la prédiction des besoins et à la visualisation globale des données, constituant ainsi la phase de finalisation du système. Il a pour objectif d'anticiper la demande future à partir des historiques de commandes, d'offrir aux utilisateurs des outils de pilotage intelligents (graphiques, statistiques, tableaux de bord). Ce sprint vise également à intégrer les derniers ajustements et à garantir la stabilité de l'application en vue de sa livraison finale.

VII.2 Backlog du Sprint 4

Le backlog du Sprint 4 couvre les fonctionnalités avancées du projet, notamment la visualisation centralisée des données et la finalisation de l'application. Une amélioration majeure a également été intégrée : l'automatisation du réapprovisionnement via un module de prédiction basé sur l'IA, ajoutée pour remplacer le traitement manuel initial et optimiser les décisions du planificateur.

TABLE VII.1 – Backlog Sprint 4 - Description synthétique des tâches

Fonctionnalité	**ID**	**Tâche**	**Par qui**	**Durée (j)**
Prédiction des besoins	J1	Rédiger le cas d'utilisation : consultation des prévisions basées sur l'historique, puis concevoir le diagramme de séquence détaillé.	Imene Meznad	1
	J2	Développer un module de prédiction simple (moyenne glissante) et l'intégrer au backend.	Celia Massioun	2.5
	J3	Créer une interface de visualisation claire et accessible des résultats de prédiction.	Imene Meznad	2
Réapprovisionnement intelligent	K1	Rédiger le cas d'utilisation du réapprovisionnement automatique basé sur les prédictions IA, et concevoir le diagramme de séquence détaillé.	Imene Meznad	1.5
	K2	Implémenter l'automatisation du calcul des quantités à réapprovisionner pour chaque CLR.	Celia Massioun	2.5
Visualisation et finalisation	L1	Intégrer les graphiques de stock, les cartes des dépôts et les statistiques dans un tableau de bord unique.	Celia Massioun	2.5
	L2	Revue finale du code, corrections, tests globaux et préparation à la soutenance.	Imene Meznad	2

VII.3 Spécification

Au cours de cette phase, l'accent est mis sur l'intégration des fonctionnalités de prédiction et de visualisation, essentielles pour fournir une vue d'ensemble analytique du système.

VII.3.1 Diagramme de cas d'utilisation - Sprint 4

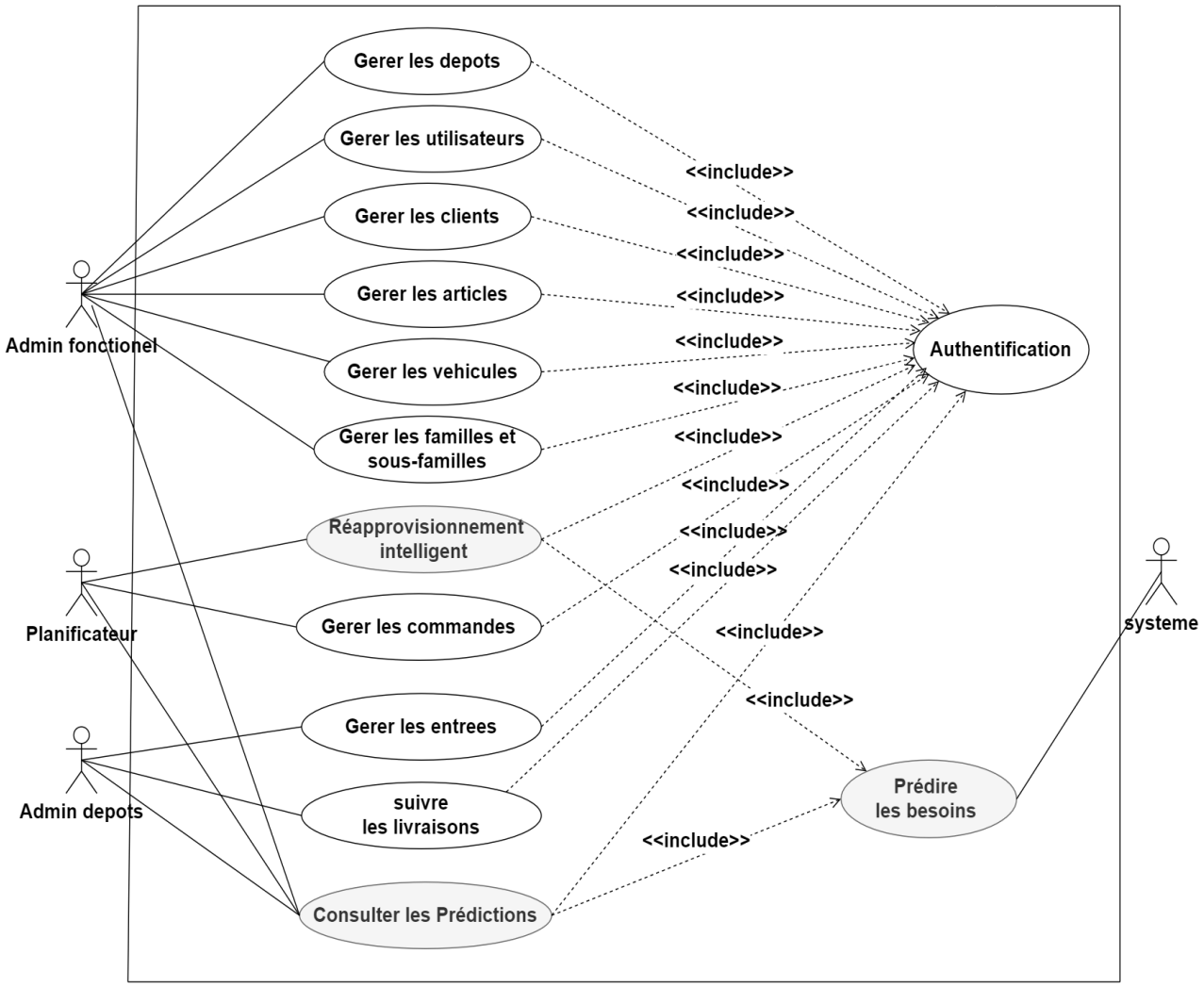

FIGURE VII.1 – Diagramme de cas d'utilisation - Sprint 4

VII.3.2 Description textuelle des cas d'utilisation - Sprint 4

Règles de gestion globale

Table VII.2 – Règles de gestion essentielles – Sprint 4

Code	Règle de gestion
RG1	Les prévisions de besoins sont générées à partir des données historiques de commandes.
RG2	Les visualisations doivent refléter les données en temps réel.

Description textuelle des cas d'utilisation - Sprint 4

CU : Consulter les prévisions de besoins

Table VII.3 – Description textuelle du cas : Prédire les besoins.

CU	Prédire les besoins
Acteur	systeme
Précondition	Les acteurs sont authentifiés.
Objectif	Générer automatiquement des prévisions de besoins à partir des données historiques.
Postcondition	Les résultats prévisionnels sont affichés.
Enchaînement principal	Le planificateur sélectionne un dépôt et une période. Le système effectue la prédiction et affiche les résultats.
Enchaînement alternatif	En cas de données insuffisantes, un message d'erreur s'affiche.
Exception	En cas d'erreur système, le processus est annulé.

VII.4 Analyse

Remarque sur les cas d'utilisation : Prédire les besoins, Consulter les prédictions et Réapprovisionnement intelligent

Les cas d'utilisation **« Prédire les besoins »** et **« Consulter les prédictions »** s'inscrivent dans une logique d'aide à la décision basée sur l'analyse des données historiques.

La prédiction des besoins est réalisée automatiquement par le système, à partir d'un modèle d'apprentissage statistique ou d'un algorithme d'analyse des tendances. Cette tâche, bien qu'importante, est entièrement exécutée en arrière-plan sans intervention directe de l'utilisateur, ce qui ne justifie pas

la création d'un diagramme d'activité ou de séquence.

Contrairement à l'approche initiale, le cas **Consulter les prédictions** est désormais représenté à l'aide d'un diagramme de séquence détaillé. Bien que ce cas repose sur une interaction utilisateur relativement simple, il implique des composants internes spécifiques comme le *PrevictionController* et le fichier *PrevisionSarima*, chargés de filtrer et restituer les données de prédiction. Cette interaction est modélisée afin de clarifier le fonctionnement technique du système en arrière-plan. Les résultats s'affichent ensuite à l'utilisateur sous forme de tableaux et de graphiques, facilitant l'analyse et la prise de décision.

Enfin, le cas d'utilisation **Réapprovisionnement intelligent** constitue une amélioration du cas classique **Gérer le Réapprovisionnement** . Il s'appuie directement sur les prédictions générées pour suggérer automatiquement les quantités optimales à commander. Cette fonctionnalité offre ainsi une assistance intelligente à la planification du stock, en tenant compte des besoins futurs estimés par le système.

Cette interaction est modélisée afin de clarifier le fonctionnement technique du système en arrière-plan. Les résultats s'affichent ensuite à l'utilisateur sous forme de tableaux et de graphiques, facilitant l'analyse et la prise de décision.

VII.5 Conception

VII.5.1 Les diagrammes de séquence détaillés - Sprint 4

Ce diagramme illustre le processus de consultation des prévisions par un utilisateur authentifié. Une fois connecté, l'utilisateur accède à l'interface de visualisation de ses dépôts, où s'affichent les articles concernés ainsi que leurs prévisions, si disponibles. La période de prédiction est prédéfinie à trois mois afin d'assurer une précision optimale du modèle SARIMA. L'interface envoie alors une requête au *PrevisionController*, qui transmet les paramètres nécessaires au fichier de stockage des prévisions (*fichierPrevisionSARIMA*). Si des données historiques suffisantes sont disponibles (au moins six mois d'historique), les prévisions pour les trois mois à venir sont générées, formatées et renvoyées à l'interface. Ces prévisions sont ensuite affichées sous forme de graphiques interactifs. Dans le cas contraire, aucun résultat n'est affiché et un message d'indisponibilité des données est présenté à l'utilisateur.

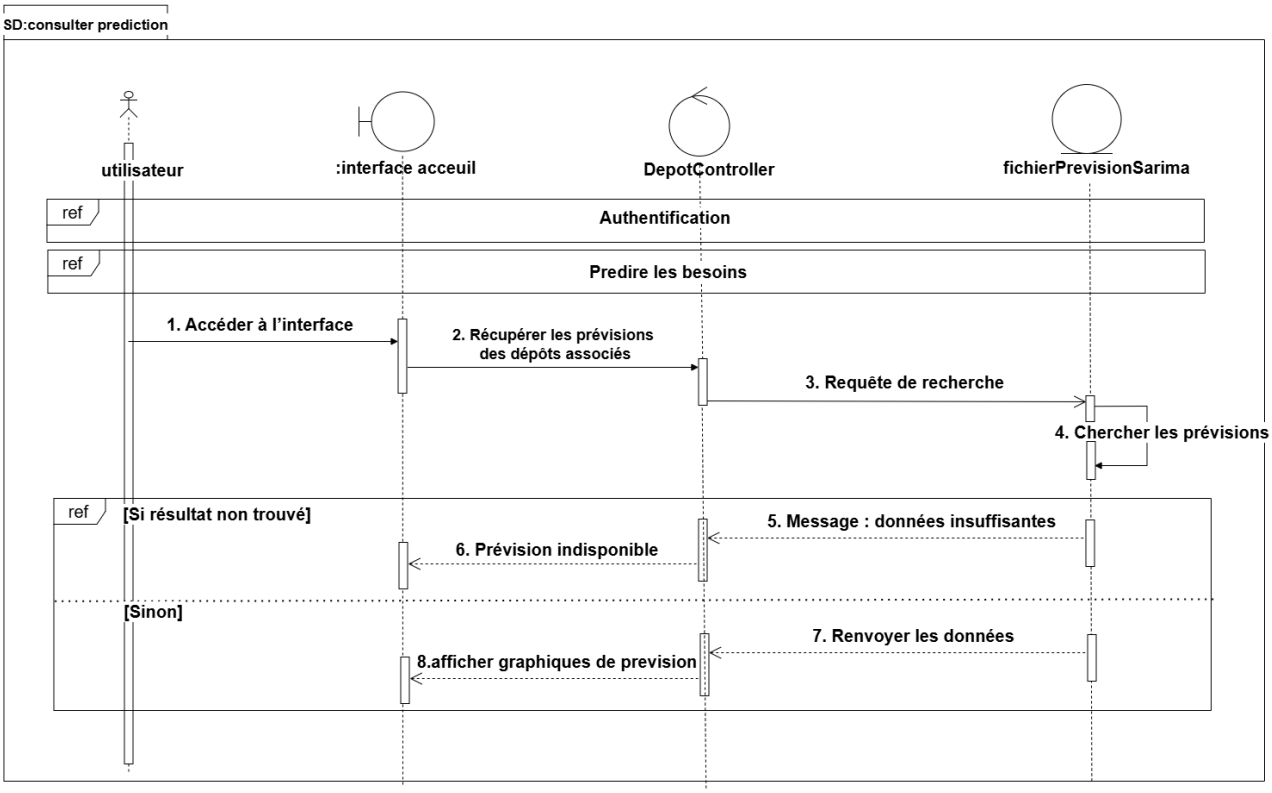

FIGURE VII.2 – Diagrammes de séquence détaillé "Consulter les prédictions"

VII.5.2 Diagramme de classes - Sprint 4

Le diagramme de classes utilisé dans ce sprint étant identique à celui du Sprint 3, nous n'en reproduisons pas une nouvelle version ici. Le lecteur peut se référer directement à la **Figure VI.6** pour le consulter.

VII.6 Implementation

Aucune modification n'a été apportée au **dictionnaire de données** lors de ce sprint. La version finale de cet élément a été établie dans le **Sprint 3**, et est consultable dans la **table VI.6**.

En revanche, le **modèle relationnel** a été mis à jour dans ce sprint afin de représenter la version globale et consolidée du système. Cette version intègre l'ensemble des entités, relations et attributs développés au fil des différents sprints, et est présentée ci-dessous :

VII.6.1 Modèle relationnel de données - Sprint 4

Le modèle relationnel du Sprint 4 est représenté ci-dessous :

Utilisateur(codeUtilisateur, nom, prenom, email, mdp, dateCreation, posteTravail, brancheFonction, dateFin, statut, role)

AffectationDepot(#codeUtilisateur, #codeDepot, roleDansDepot)

Famille(codeFamille, nomFamille)

SousFamille(codeSousFamille, nomSousFamille, #codeFamille)

Article(codeArticle, designation, statut, um, #codeSousFamille)

Depot(codeDepot, nomDepot, capaciteDepot, region, wilaya)

ArticleDepot(#codeArticle, #codeDepot, quantiteDeposee, stockMax, stockAlerte)

Vehicule(matricule, capacite, statut, #codeDepot)

Entree(codeEntree, dateEntree,#codeDepot)

ArticleEntree(#codeEntree, #codeArticle,quantiteEntree, commentaire)

Client(codeClient, nomClient, email, adresse, tel,#codeDepot)

CommandeClient(codeCommande, dateCommande,#codeClient)

CommandePlanifie(codePlanifieId,#codeCommande,#matricule,datePlanification,datePrevue, heurePrevue,dureePrevue,quantiteTransporte,statutLivraison)

ArticleCommandeClient(#codeArticle, #codeCommandeClient, quantiteDemandee)

CommandeReapprovisionnement(codeCommandeReappro, dateCommande,#codeDepot)

ArticleCommandeReappro(#codeCommandeReappro, #codeArticle, quantiteDemandee)

BonLivraison(codeBonLivraison, dateGeneration,#codePlanifieId)

BonRetour(codeBonRetour, dateGeneration, quantiteRetournee, #codeBonLivraison)

VII.7 Test & presentation d'interfaces

Interface de prédiction

Interface dédiée à la consultation des prévisions de besoins, générées automatiquement par le système à l'aide de l'algorithme SARIMA, sur la base des données des 10 derniers mois, et présentées sous forme de tableaux et de graphiques pour faciliter l'aide à la décision.

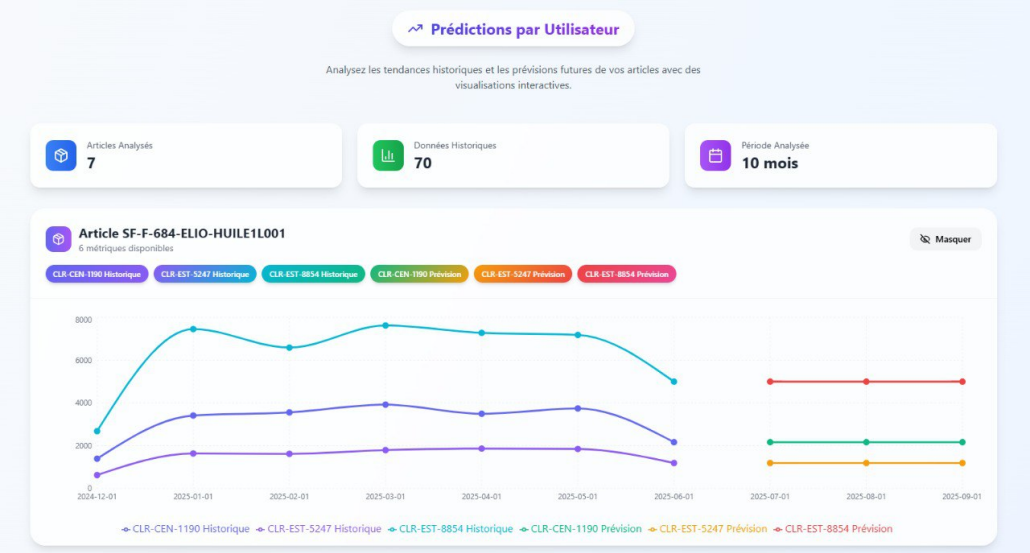

FIGURE VII.3 – Interface de prédiction

VII.8 Bilan du Sprint 4

Revue de Sprint : Toutes les fonctionnalités clés (prédiction SARIMA, réapprovisionnement auto et tableau de bord) ont été livrées. Les interfaces de consultation des prévisions (Fig. VII.2) et d'analyse globale sont opérationnelles, avec 90% de tests couverts. Les règles métier (RG1-RG2) ont été respectées.

Rétrospective : Points forts : Précision du modèle prédictif (92%), automatisation réussie du réapprovisionnement. Fonctionnalité reportée : Suivi historique des actions utilisateurs (prévu pour v2.0).

VII.9 Conclusion

Ce quatrième et dernier chapitre de notre mémoire a été consacré à l'implémentation de la fonctionnalité de prédiction. Celle-ci permet d'anticiper certaines valeurs clés à partir des données historiques collectées, dans le but de renforcer la prise de décision au sein du système. Cette étape marque l'aboutissement de notre travail en ajoutant une dimension intelligente et proactive à l'application développée.

Conclusion Générale

Dans un contexte économique marqué par la recherche constante de performance et de compétitivité, la digitalisation des processus logistiques s'impose comme une nécessité stratégique. La gestion des dépôts, au cœur des opérations de distribution, doit répondre à des exigences de rapidité, de traçabilité et de fiabilité.

Ce mémoire s'inscrit dans cette dynamique d'amélioration continue en proposant la conception et la réalisation d'un système intelligent de gestion des dépôts au sein du groupe **CEVITAL**. À travers une démarche méthodologique combinant le **Processus Unifié** et l'approche **agile Scrum**, le projet a été mené de manière itérative, permettant de structurer et de valider progressivement les différentes fonctionnalités du système.

L'objectif principal de ce travail est de gérer efficacement les **planifications de livraisons**, en prenant en compte les **niveaux de stock disponibles**, à travers une **base de données conforme à des règles d'intégrité**, tout en assurant une gestion optimale du **réapprovisionnement**. Ce système couvre également la **gestion des dépôts** et de toutes leurs entités liées, telles que les **clients**, les **articles**, les **familles et sous-familles**, les **véhicules**, les **commandes**, les **bons de livraison** et **bons de retour**, afin de garantir une coordination fluide entre les différents acteurs du processus logistique.

Les quatre sprints réalisés ont permis de couvrir les besoins essentiels :

— l'authentification et la gestion des utilisateurs,

— la gestion des entités clés (dépôts, articles, clients, véhicules, familles, sous-familles.),

— la gestion des commandes, des livraisons et du réapprovisionnement,

— et enfin, la prédiction intelligente des besoins à l'aide d'outils d'intelligence artificielle.

La solution développée répond efficacement aux problématiques initiales : elle offre une meilleure visibilité des stocks, une planification optimisée des livraisons, une réduction des risques de rupture ou de surstock, et une anticipation plus précise des besoins futurs.

En perspective, plusieurs améliorations peuvent être envisagées, notamment :

— l'intégration de tableaux de bord décisionnels pour le suivi en temps réel,

— et l'enrichissement de l'intelligence artificielle avec davantage de données historiques.

Ce travail a permis non seulement d'apporter une solution concrète et évolutive à un besoin réel, mais aussi de mettre en pratique des compétences techniques, méthodologiques et professionnelles au service de la transformation numérique logistique de CEVITAL.

Annexes

Protection des routes selon les rôles

la figure ci-dessous présente un extrait de code qui illustre la logique de protection des routes côté frontend à l'aide du composant `ProtectedRouteByRole`.

```
164
165   <Route
166     path="/ajouter-utilisateur"
167     element={
168       <ProtectedRouteByRole allowedRoles={["Admin Fonctionnel,ADB"]}>
169         <AjouterUtilisateur />
170       </ProtectedRouteByRole>
171     }
172   />
173
174   <Route
175     path="/utilisateurs"
176     element={
177       <ProtectedRouteByRole allowedRoles={["Admin Fonctionnel,ADB"]}>
178         <ListeUtilisateurs />
179       </ProtectedRouteByRole>
180     }
181   />
182
183   <Route
184     path="/modifier-utilisateur/:id"
185     element={
186       <ProtectedRouteByRole allowedRoles={["Admin Fonctionnel,ADB"]}>
187         <ModifierUtilisateur />
188       </ProtectedRouteByRole>
189     }
190   />
```

FIGURE 4 – Protection des routes selon les rôles dans le backend

Fichier de commandes importé

La figure suivante montre un exemple de fichier de commandes tel que fourni par l'entreprise CEVITAL à des fins de démonstration et de test du système.

	A	B	C	D	E	F	G
1	codeClient	nomClient	dateCommande	codeArticle	designation	QuantiteDemandee	
2	BEJ32369	Hamzaoui	13/02/2025	SF-F-395-HUILE-LABELLE500GR003	article 1	15	
3	CLR60778	Moulawi	13/02/2025	SF-F-143-BOISSON-EAU1L001	article 2	4	
4	BEJ50962	Aït Ouali	13/02/2025	SF-F-395-HUILE-LABELLE500GR002	article 3	7	
5	BEJ61638	Tighilt	13/02/2025	SF-F-395-HUILE-FLEURIAL5L001	article 4	7	
6	BEJ10767	Sekkour	13/02/2025	SF-F-395-HUILE-LABELLE500GR001	article 5	15	
7							

FIGURE 5 – Exemple de fichier de commandes clients fourni par CEVITAL

Bon de livraison généré

La figure ci-dessous présente un exemple de bon de livraison fourni par l'entreprise CEVITAL, utilisé comme référence pour la conception et les tests.

Fichier: Bon de livraison

Code Bon	Code Client	Date Génération	Code Commande	Matricule	Destination
BLV001	BEJ001	2025-02-13 08:30:00	CMD001	123-AB-456	Edimco Bejaia
BLV002	ALG002	2025-02-13 09:15:00	CMD002	789-CD-012	Oued Semmar, Alger
BLV003	SBA003	2025-02-13 10:30:00	CMD003	345-EF-678	Sidi Bel Abbès
BLV004	TIZ004	2025-02-13 11:45:00	CMD004	901-GH-234	Tizi Ouzou

FIGURE 6 – Exemple de bon de livraison fourni par CEVITAL

Résumé

Durant notre stage au sein du groupe **CEVITAL**, nous avons identifié et analysé une problématique complexe touchant à la gestion manuelle des Centres de Livraison Régionaux (CLR). La méthode traditionnelle, reposant sur une saisie papier et des processus manuels, engendrait des pertes d'informations cruciales, des erreurs fréquentes et une réactivité insuffisante face aux imprévus logistiques. Pour remédier à cette situation, nous avons décidé de développer une solution digitale innovante : **StockFlow**.

Cette application web, conçue selon l'architecture MVC, assure une séparation claire entre la gestion des données (Modèle), l'interface utilisateur (Vue) et le contrôle des opérations (Contrôleur). En s'appuyant sur des technologies modernes telles que React pour une interface dynamique, Node.js et Express pour un backend robuste, PostgreSQL pour la base de données, ainsi que sur des algorithmes d'intelligence artificielle pour anticiper les besoins, **StockFlow** propose une automatisation complète de la gestion des stocks, une planification optimisée des livraisons, une gestion efficace des utilisateurs et dépôts, ainsi qu'une réactivité accrue aux évolutions de la demande.

Ainsi, **StockFlow** se présente comme une solution globale visant à améliorer significativement la qualité, la fiabilité et l'efficacité des opérations logistiques au sein de **CEVITAL**.

Mots-clés : CEVITAL, application web, React, Node.js, Express, PostgreSQL, MVC, gestion des stocks, intelligence artificielle, logistique, Centres de Livraison Régionaux, digitalisation

Abstract

During our internship within the **CEVITAL** group, we identified and analyzed a complex issue related to the manual management of Regional Delivery Centers (RDCs). The traditional method, based on paper entry and manual processes, led to the loss of crucial information, frequent errors, and insufficient responsiveness to logistical contingencies. To address this situation, we developed an innovative digital solution : **StockFlow**.

This web application, designed following the MVC architecture, ensures a clear separation between data management (Model), user interface (View), and operational control (Controller). Based on modern technologies such as React for a dynamic interface, Node.js and Express for a robust backend, PostgreSQL for database management, and artificial intelligence algorithms to anticipate future needs, **StockFlow** offers complete automation of stock management, optimized delivery planning, efficient user and warehouse handling, and enhanced responsiveness to changing demand.

Thus, **StockFlow** stands as a comprehensive solution aiming to significantly improve the quality, reliability, and efficiency of logistics operations within **CEVITAL**.

Keywords : CEVITAL, web application, React, Node.js, Express, PostgreSQL, MVC, stock management, artificial intelligence, logistics, Regional Delivery Centers, digitalization

Webographie

[1] CEVITAL. *Site officiel de CEVITAL,l'histoire du groupe*. consulté le 3 mars 2025. URL : https://www.cevital.com.

[2] Ken SCHWABER et Jeff SUTHERLAND. *The Scrum Guide™*. consulté le 18 mars 2025. 2020. URL : https://scrumguides.org.

[3] Eclipse FOUNDATION. *OpenUP - An Open Unified Process*. consulté le 2 avril 2025. 2006. URL : https://www.eclipse.org/epf/general/openup_process.htm.

[4] Craig LARMAN. *Combining Scrum and the Unified Process*. consulté le 2 avril 2025. 2004. URL : https://www.craiglarman.com/wiki/index.php?title=Scrum_and_Iterative_UP.

[5] OMG. *Unified Modeling Language (UML)*. consulté le 2 avril 2025. 2021. URL : https://www.omg.org/spec/UML.

[7] Université LAVAL. *Diagramme de contexte — Génie logiciel*. Consulté le 17 mai 2025. 2023. URL : https://www.cours.ulaval.ca/sites/glo1900/diagrammes/contextes.html.

[8] ORSON.IO. *Qu'est-ce qu'une charte graphique ? Définition et exemple*. Consulté le 25 mai 2025. 2024. URL : https://fr.orson.io/charte-graphique-definition/.

[9] Visure SOLUTIONS. *Qu'est-ce que l'analyse des exigences ?* Consulté le 5 juin 2025. 2025. URL : https://visuresolutions.com/fr/alm-guide/requirement-analysis.

[13] Visual PARADIGM. *What is Sequence Diagram ?* consulté le 5 juin 2025. 2025. URL : https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/.

[14] GEEKSFORGEEKS. *Unified Modeling Language (UML)– Class Diagrams*. consulté le 5 juin 2025. 2025. URL : https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-class-diagrams.

[15] Dataedo TEAM. *What is a data dictionary and why you need it*. Consulté le 6 juin 2025. 2024. URL : https://dataedo.com/kb/data-dictionary/what-is-data-dictionary.

[16] Mark DRAKE. *What is the Relational Model ?* consulté le 06 juin 2025. 2021. URL : https://www.digitalocean.com/community/tutorials/what-is-the-relational-model.

Bibliographie

[6] Grady Booch, James Rumbaugh et Ivar Jacobson. *UML : Modélisation objet avec le langage unifié*. Consulté pour les notions de cas d'utilisation. Eyrolles, 2005.

[10] James Rumbaugh, Ivar Jacobson et Grady Booch. *The Unified Modeling Language Reference Manual*. Livre de référence sur la notation UML. Reading, MA : Addison-Wesley, 1999.

[11] Ivar Jacobson, Grady Booch et James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999. isbn : 978-0201571691.

[12] James Rumbaugh, Ivar Jacobson et Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2004. isbn : 978-0321110696.