

CYCLE INGENIEUR DE L'ECOLE POLYTECHNIQUE D'AGADIR
FILIERE GENIE INFORMATIQUE

STAGE TECHNICIEN

Développement de système de gestion de stock avancer (SGSA)

Réalisé par :

Abdelhamid BOULAAJOUL

Superviseur :

M. Tarik MAJID

Dédicaces

Je dédie ce rapport de stage à plusieurs personnes qui ont été d'un soutien inestimable tout au long de mon parcours académique et professionnel.

À mes parents, pour leur amour inconditionnel, leur soutien moral et financier, et pour avoir toujours cru en moi. Leur patience et leur encouragement ont été une source constante de motivation et de détermination.

À mes professeurs et mentors, qui ont partagé leur savoir et leur expérience avec passion et générosité. Leur guidance et leurs conseils ont grandement contribué à mon développement personnel et professionnel.

À mes amis, pour leur amitié sincère, leur soutien et les moments de joie partagés. Leur présence a rendu cette période de ma vie plus agréable et moins stressante.

Enfin, à toute l'équipe de l'entreprise COPAG, et en particulier à mon encadrant professionnel, pour leur accueil chaleureux, leur disponibilité, et leur confiance. Leur expertise et leurs conseils m'ont permis de tirer le meilleur parti de cette expérience de stage.

Merci à vous tous pour votre soutien et votre inspiration.

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce projet de stage.

Tout d'abord, je remercie chaleureusement toute l'équipe de COPAG pour m'avoir accueilli au sein de leur entreprise et pour m'avoir offert cette opportunité enrichissante. Un remerciement particulier à mon encadrant professionnel, M. Tarik MAJID, pour sa guidance, ses conseils avisés, et sa disponibilité tout au long de mon stage. Son expertise et sa bienveillance ont été d'une grande aide pour la réussite de ce projet.

Je souhaite également remercier mes professeurs et tuteurs académiques de l'école polytechnique agadir pour leur soutien continu et leur enseignement de qualité. Leur dévouement et leur passion pour l'enseignement ont largement contribué à mon apprentissage et à ma préparation pour ce stage.

Un grand merci à mes collègues stagiaires et à tous les membres de l'équipe projet pour leur collaboration, leur esprit d'équipe, et les précieux moments partagés. Leur soutien et leur camaraderie ont rendu ce stage non seulement productif mais également agréable.

Je tiens aussi à remercier ma famille et mes amis pour leur soutien moral indéfectible et leurs encouragements constants. Leur présence à mes côtés a été une source de réconfort et de motivation.

Enfin, je remercie tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce projet et à la réussite de mon stage. Votre aide et vos encouragements ont été précieux et je vous en suis profondément reconnaissant.

Merci à vous tous

TABLE DES MATIERES

DEDICACES	2
REMERCIEMENTS	3
TABLE DES MATIERES	4
TABLE DES ILLUSTRATIONS.....	6
INTRODUCTION GENERALE.....	9
CHAPITRE 1 :.....	10
PRESENTATION DE L'ENTREPRISE	10
I. INTRODUCTION	11
II. PRESENTATION GENERALE : COOPERATIVE COPAG.....	11
2-1- <i>Activités</i>	12
2-2- <i>Historique</i>	12
III. CONCLUSION	13
CHAPITRE 2 :.....	14
PRESENTATION DU PROJET	14
I. INTRODUCTION	15
II. CONTEXTE DU PROJET	15
2-1- <i>Problématique</i>	15
2-2- <i>Solution et Objectif</i>	15
2-3- <i>Livrable final</i>	16
2-4- <i>Périmètre fonctionnelle de projet</i>	16
III. SPECIFICATION GENERALE	16
3-1- <i>Spécification des besoins fonctionnels</i>	16
3-2- <i>Spécification de besoins non fonctionnels</i>	25
3-3- <i>Spécification technique</i>	26
IV. ÉTUDE CONCEPTUELLE	30
4-1- <i>Diagramme de flux</i>	30
4-2- <i>Diagramme Entité-Relation</i> :.....	32
4-3- <i>Diagramme de Cas d'Utilisation (Use Case Diagram)</i> :	33
4-4- <i>Diagramme d'activité</i>	37
V. CONCLUSION	42
CHAPITRE 3 :.....	43
MISSION ET TRAVAUX REALISES.....	43
I. INTRODUCTION	44
II. MISSIONS ET COORDINATION ET GANTT	44

TABLE DES MATIERES

III.	DEVELOPPEMENT FRONTEND	46
4-1-	<i>Login page</i>	46
4-2-	<i>Modèle des items</i>	47
4-3-	<i>Modèle de dimension de stockage</i>	48
4-4-	<i>Modèle de dimension de suivi</i>	50
4-5-	<i>Bon de réception</i>	51
4-6-	<i>Gère ou crée Nouveau entête</i>	52
4-7-	<i>Linges de bon de réception</i>	53
4-8-	<i>Mouvement de stock</i>	55
4-9-	<i>Gere ou crée nouveau mouvement de stock</i>	56
4-10-	<i>Lignes des mouvements de stock</i>	57
4-11-	<i>Inventory dimension</i>	59
IV.	BACKEND	59
4-1-	<i>Modèle d'article</i>	60
4-2-	<i>Avoir les informations d'un modèle d'article (getitemmodel)</i>	60
4-3-	<i>Désactiver le modèle d'article (changestateofitemmodel)</i>	61
4-4-	<i>Model de dimension de stockage</i>	62
4-5-	<i>Obtenir les informations d'un modèle de dimension de stockage</i>	63
4-6-	<i>Changer le statut d'un modèle de dimension de stockage</i>	63
4-7-	<i>Model de dimension de suivi</i>	64
4-8-	<i>Obtenir les informations d'un modèle de dimension de suivi</i>	64
4-9-	<i>Changer le statut d'un modèle de dimension de suivi</i>	65
4-10-	<i>Demande d'achat</i>	65
4-11-	<i>4. Clôture des DA avec gestion des lignes</i>	68
V.	TESTS API.....	69
4-1-	<i>Tests pour le Frontend</i>	69
4-2-	<i>Tests des API Backend</i>	71
VI.	CONCLUSION.....	73
	CONCLUSION GENERALE	74
	BIBLIOGRAPHIE	75
	ANNEXES.....	76

TABLE DES ILLUSTRATIONS

Liste des figures

Figure 1 L'historique de la COPAG	13
Figure 2 Architecture logicielle du projet	26
Figure 3 diagramme de flux	31
Figure 4 diagramme entité relation	32
Figure 5 use case Diagramme de gestion de consultation	33
Figure 6 use case Diagramme gestion journaux de transfert	33
Figure 7 use case Diagram gestion ordre de transfert	34
Figure 8 use case Diagramme gestion d'inventaire	35
Figure 9 use case Diagramme gestion de réception	37
Figure 10 Activity Diagram Gestion de réception	38
Figure 12 Activity Diagram Gestion de consultation de stock disponible	39
Figure 13 Activity diagramme Gestion d'inventaire	40
Figure 14 Activity Diagram gestion journaux transfert	41
Figure 15 Activity Diagram Gestion ordre transfert	42
Figure 16 diagramme de Gantt.....	45
Figure 17 page de login.....	46
Figure 18 page d'accueil.....	47
Figure 19 page de model d'item	47
Figure 20 Model de dimension de stockage.....	49
Figure 21 Interface de sélection et consultation d'un modèle de dimension de suivi	50
Figure 22 les entêtes des bons de réception	51
Figure 23 List des entêtes de bon de réception avec filtrage	52
Figure 24 interface d'un entête de bon de réception.....	52
Figure 25 interface des Linges de bon de réception.....	53

TABLE DES ILLUSTRATIONS

Figure 26 affichage de tous les information un linge de BR.....	54
Figure 27 affichage les informations d'article d'un linge de BR	54
Figure 28 Tableau des entêtes de mouvements de stock avec options de filtrage	55
Figure 29 Interface de gestion et création d'un mouvement de stock	56
Figure 30 Interface des lignes de mouvement de stock	57
Figure 31 Résultat de vérification de disponibilité pour les lignes de stock sélectionnées	57
Figure 32 Détails d'une ligne de mouvement de stock sélectionnée	58
Figure 33 Interface de création ou modification d'un lot d'inventaire	59
Figure 34 Message de confirmation de mise à jour d'un lot d'inventaire.....	59
Figure 35 création d'un modèle d'article dans la base de données.	60
Figure 36 récupération des informations d'un modèle d'article.	61
Figure 37 mise à jour du statut d'un modèle d'article	61
Figure 38 création d'un modèle de dimension de stockage	62
Figure 39 affichage des informations d'un modèle de dimension de stockage.....	63
Figure 40 mise à jour du statut d'un modèle de dimension de stockage	63
Figure 41 création d'un modèle de dimension de suivi	64
Figure 42 Affichage des informations d'un modèle de dimension de suivi	65
Figure 43 mise à jour du statut d'un modèle de dimension de suivi	65
Figure 44 Mise à jour de l'Enum des statuts dans le backend.....	66
Figure 45 Validation de la Demande d'Achat (DA).....	66
Figure 46 Validation des Lignes de Demande d'Achat.....	67
Figure 47 Gestion du Statut de la Demande d'Achat (DA) vers REVS	68
Figure 48 Vérification des Lignes de Demande d'Achat (DA) pour Approbation	69
Figure 49 Utilisation de Mockoon pour les tests frontend.....	70
Figure 50 Résultat de tests de l'API backend	72
Figure 51 L'organisation de la COPAG	76
Figure 52 L'organigramme de la COPAG	77

Figure 53 Maquette de la page des listes de bon de réception	78
Figure 54 Prototype de la gestion de dimension de stock	78
Figure 55 Vue de la structure de la base de données SGSA.	79
Figure 56 Vue de la structure de la base de données SGSA.	79
Figure 57 Exemple de requête SQL pour la gestion des stocks	80
Figure 58 exemple de table de modèle d'articles	80

Liste des tableaux

Tableau 1 Exemple de calcul de PMP	22
Tableau 2 Exemple de calcul de FIFO	22
Tableau 3 Exemple De Dimension De Stockage	23
Tableau 4 Exemple De Dimension De Suivi	23
Tableau 5 Fiche technique de la COPAG	76

INTRODUCTION GENERALE

Mon stage, d'une durée de deux mois, s'est déroulé au sein de la coopérative agricole COPAG, spécialisée dans le secteur agroalimentaire. COPAG est une entreprise de grande envergure qui opère principalement dans la production et la distribution de produits laitiers et autres denrées agricoles. Mon stage a eu lieu au siège social de l'entreprise, situé à Taroudant, où j'ai eu l'opportunité d'acquérir de nombreuses compétences dans le domaine de la gestion des stocks et du développement informatique.

Durant cette période, j'ai découvert en profondeur le fonctionnement de l'entreprise ainsi que le métier de développeur dans un environnement industriel. J'ai pu développer des compétences techniques et organisationnelles en participant à des projets concrets, notamment dans le cadre de la gestion avancée des stocks (SGSA). Mon travail a été encadré par un professionnel aguerri qui m'a accompagné dans l'apprentissage des outils et technologies utilisés au sein de l'entreprise.

L'objectif principal de mon stage était de participer à la conception et au développement d'un système de gestion des stocks, un projet d'envergure au sein du département informatique. Mes missions ont principalement porté sur le développement d'une application web et la gestion des bases de données, en lien avec les besoins fonctionnels de l'entreprise. Ce projet a permis de mettre en œuvre différentes technologies modernes comme NestJS, Angular, et TypeORM pour répondre aux exigences spécifiques de la gestion de stock de COPAG.

Ce rapport est structuré en plusieurs chapitres. Le premier chapitre présente le contexte général du projet ainsi que l'environnement de travail. Le deuxième chapitre détaille les spécifications fonctionnelles et techniques du projet. Ensuite, le troisième chapitre aborde l'étude conceptuelle et les différents diagrammes utilisés. Enfin, le dernier chapitre décrit la réalisation du projet et les différentes phases de développement auxquelles j'ai participé, suivie d'une conclusion présentant les perspectives du projet.

CHAPITRE 1 :

Présentation de l'entreprise

I. Introduction

Dans ce chapitre, une introduction à la coopérative COPAG est présentée, mettant en lumière son rôle significatif dans le secteur agroalimentaire au Maroc et, en particulier, dans la région du Souss Massa. COPAG, créée pour soutenir le développement rural, symbolise la synergie entre les producteurs agricoles et les exigences du marché national et international. Cette section explore les activités principales de l'entreprise, son historique, et son impact socio-économique, offrant ainsi une compréhension globale de sa contribution au secteur.

II. Présentation générale : Coopérative COPAG

COPAG est une organisation économique constituée de plusieurs services qui se coordonnent entre eux pour former un tissu bien organisé en vue d'avoir une bonne marche de travail par conséquent participer au développement économique. Elle opère dans le secteur agricole au sens le plus large du terme : productions animales (lait, viande) productions végétales (agrumes, primeurs...) agroalimentaire, etc.



COPAG est actuellement le promoteur du développement de l'élevage laitier dans le Souss Massa. Elle a été créée le 7 mai 1987 par un groupe de 39 agriculteurs de la région de Taroudant qui ont éprouvé le besoin et la nécessité de se regrouper afin de maîtriser leurs produits agricoles depuis la production jusqu'à un stade avancé de la distribution. Il s'agissait de profiter de la politique de libéralisation des exportations amorcée par l'état marocain.



Le développement de COPAG est le double résultat d'une stratégie d'intégration des stades de la filière agricole et d'une politique de porte ouverte pour augmenter l'effectif des adhérents et faire croître la quantité du lait collecté et transformé.

La COPAG étant une coopérative polyvalente, elle ne se contente pas seulement de la production, la transformation, et la commercialisation du lait et de ses dérivés. Elle s'occupe également de la production et de l'exportation des agrumes et des primeurs. La superficie exploitée atteint 11 100 ha

en total, répartis comme suit : 4000 ha d'agrumes, 1100 ha de primeurs, et 6000 ha de cultures fourragères. En ce qui concerne le cheptel bovin, il est constitué à son tour de 80 000 têtes dont 40 000 vaches laitières.

Le caractère économique de la coopérative l'oblige à assurer un ensemble de fonctions pour valoriser la production de ses membres, à travers des actions d'approvisionnement en facteurs de productions, de commercialisation, d'encadrement et de formation en collaboration étroite avec des partenaires nationaux et internationaux. Outre le conditionnement des agrumes et de transformation 11 du lait, la COPAG cherche à améliorer la productivité, la rentabilité de ses activités et par voie de conséquence garantir la croissance économique des secteurs d'activités où elle opère.

Parmi les missions que la COPAG s'est fixée :

- Assurer elle-même ou par l'intermédiaire de ses adhérents le développement socioéconomique du milieu rural de la région du Souss ;
- Offrir des produits agricoles d'origine animale et végétale de plus en plus élaborés qui peuvent satisfaire les attentes actuelles et futures des consommateurs ;
- Améliorer le revenu de la COPAG et de ses adhérents à travers des actions conjuguées à tous les stades de la production, de la transformation et de la commercialisation des produits agricoles (et les dérivés) à forte valeur ajoutée



2-1- Activités

COPAG opère principalement dans trois secteurs : la production animale (principalement les produits laitiers), la production végétale (agrumes et autres fruits), et l'agroalimentaire (transformation et commercialisation de produits agricoles). La coopérative est également active dans l'exportation de ses produits, notamment les agrumes, vers des marchés internationaux.

2-2- Historique

La coopérative COPAG a été créée par l'assemblée générale du 07 mai 1987, profitant de la politique de libéralisation des exportations amorcée par l'Etat marocain, 39 agriculteurs de la région de Taroudant ont senti le besoin et la nécessité de se grouper en coopérative pour être maîtres de leurs produits agricoles depuis la production jusqu'à un stade plus avancé de la distribution. Ainsi est née la

coopérative « COPAG », son capital social a été fixé à 4 600 000dhs divisé en 4 600 parts de 1000 DHs chacune.

La coopérative COPAG qui opérait à ses débuts dans les agrumes est aujourd'hui essentiellement connue pour ses produits laitiers (marque Jaouda). Elle revendique d'ailleurs la seconde position (en volume) des dérivés après la Centrale Laitière (Danone). Depuis 1993, la mise en place d'une unité de production du lait au sein de la coopérative a été créée dans le but de rassembler tout le lait de la région, le transformer en lait pasteurisé et ses dérivées et le commercialiser au niveau de tout le territoire national. Le choix stratégique de la création de cette unité a permis à ces adhérents de jouer le rôle de locomotive du développement dans leurs périmètres d'intervention et plus particulièrement dans le milieu rural, où la production laitière collectée par la COPAG est assurée par un réseau de 170 adhérents individuels et 67 coopératives qui regroupent près de 13 000 éleveurs

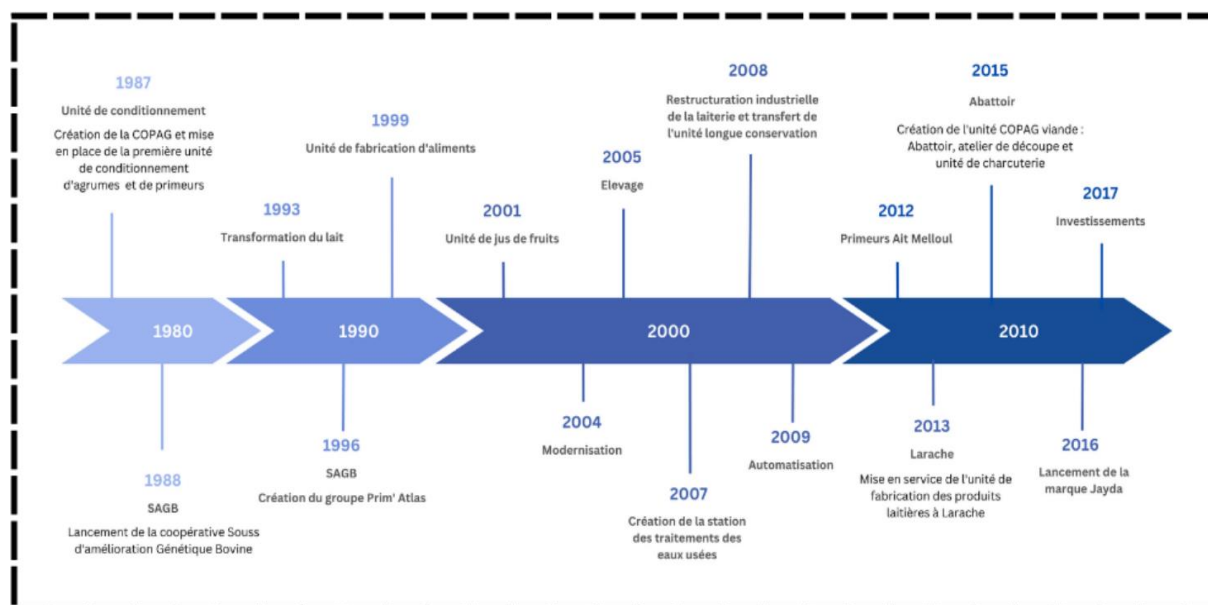


Figure 1 L'histoire de la COPAG

III. CONCLUSION

COPAG se distingue par sa capacité à intégrer et maîtriser l'ensemble des étapes de production et de distribution. Ce chapitre a permis de cerner l'importance de COPAG dans l'économie régionale et nationale, ainsi que son engagement pour le développement durable et le soutien des agriculteurs. Sa stratégie exemplaire de croissance continue et de modernisation démontre son impact sur l'économie agroalimentaire.

CHAPITRE 2 :

Présentation du projet

I. Introduction

Ce chapitre aborde le contexte du projet de Système de Gestion de Stock Avancé (SGSA) au sein de COPAG, motivé par la nécessité d'améliorer la gestion des stocks face à une croissance continue des volumes et des flux logistiques. Le manque de visibilité en temps réel, l'optimisation limitée des espaces de stockage et la gestion accrue des coûts ont conduit COPAG à entreprendre ce projet. Le SGSA vise à garantir une traçabilité précise, à optimiser les ressources et à offrir des outils analytiques pour une prise de décision éclairée.

II. Contexte du projet

La réussite de COPAG dépend fortement de l'efficacité et de la précision dans la gestion des stocks. Avec la croissance des volumes de marchandises traitées et la complexité croissante des flux logistiques, il devient crucial d'adopter un système capable de gérer les stocks de manière optimale tout en anticipant les besoins opérationnels. Actuellement, COPAG fait face à des défis majeurs tels que le manque de visibilité en temps réel sur les stocks, l'optimisation limitée des espaces de stockage et des coûts de gestion accrus, menant à des inefficacités et des dépenses supplémentaires. Dans le but de pallier ces obstacles et d'améliorer ses performances logistiques, COPAG a entrepris le développement d'un Système de Gestion de Stock Avancé (SGSA). Ce projet a pour objectif de garantir une traçabilité complète des stocks, d'optimiser l'utilisation de l'espace de stockage, de réduire les coûts de gestion et d'améliorer la précision des opérations logistiques tout en offrant des outils analytiques pour une prise de décision plus éclairée.

2-1- Problématique

COPAG bénéficie d'une vaste expérience en gestion d'entreprise, principalement grâce à son système ERP, qui orchestre les processus, opérations et finances. En tant que coopérative, COPAG collabore avec de nombreux adhérents, effectuant des transactions d'achat et de vente. Afin d'améliorer l'efficacité de ces échanges, un système d'information spécialisé est devenu nécessaire. Ainsi, COPAG a entrepris la mise en place de systèmes d'information spécifiques pour chaque coopérative, permettant des échanges électroniques fluides entre l'ERP de COPAG et ces nouveaux systèmes.

2-2- Solution et Objectif

L'objectif du projet est de concevoir, développer et déployer un Système de Gestion de Stock Avancé (SGSA) répondant aux besoins spécifiques de COPAG en matière de gestion des stocks. Le SGSA sera une solution numérique intégrée, dotée de fonctionnalités avancées visant à optimiser la gestion des stocks, améliorer la traçabilité, et réduire les coûts associés. Plus précisément, l'objectif

est de développer une application qui permettra un suivi en temps réel des mouvements de stocks, une optimisation des emplacements de stockage, une estimation précise des coûts, ainsi qu'une gestion efficace des différentes dimensions des stocks. Cette application vise à réduire les coûts de gestion des stocks, à accroître la satisfaction des clients, et à renforcer la compétitivité de COPAG sur le marché. Elle offrira également une meilleure traçabilité grâce à la mise en œuvre de méthodologies adaptées aux spécificités de l'entreprise.

2-3- Livrable final

Le livrable final de ce projet consistera en un Système de Gestion de Stock Avancé (SGSA) entièrement fonctionnel, prêt à être déployé chez COPAG. Ce SGSA sera accompagné d'une documentation complète qui décrira en détail son architecture, son fonctionnement et les procédures nécessaires à son déploiement et à son utilisation.

Avant la livraison, le système sera soumis à des tests rigoureux pour garantir sa fiabilité, sa performance et sa conformité aux besoins spécifiques de COPAG. Une fois ces tests réussis et la validation de l'équipe de projet obtenue, le SGSA sera considéré comme prêt pour une utilisation opérationnelle au sein de l'entreprise.

2-4- Périmètre fonctionnelle de projet

Le périmètre fonctionnel du projet englobe les fonctionnalités essentielles d'un Système de Gestion de Stock Avancé (SGSA), notamment la gestion complète des stocks, des commandes, des emplacements, des articles et des inventaires. Le système sera conçu pour offrir une traçabilité précise des stocks en intégrant diverses méthodologies de suivi, telles que la traçabilité par lot, par numéro de série, par date de péremption et par emplacement.

III. Spécification générale

3-1- Spécification des besoins fonctionnels

Dans ce chapitre, nous procéderons à une analyse approfondie des exigences du projet. Nous examinerons en détail les besoins fonctionnels et non fonctionnels du système, ainsi que les exigences spécifiques des utilisateurs. Cette analyse nous permettra de définir de manière précise les fonctionnalités et les caractéristiques essentielles de l'application à développer. Nous explorerons également les contraintes et les limitations qui pourraient influencer la conception et le développement du système. En fournissant une compréhension détaillée des exigences du projet, ce chapitre servira de fondement solide pour la phase de conception et de développement ultérieure.

3-1-1. Les Dimensions de Stock

a. Analyse de stock

- ♦ Analyse Temporelle :

Durée de Stockage : Suivi du temps que les produits passent en stock pour gérer les produits périssables.

Rotation des Stocks : Identifier les produits à rotation rapide et lente pour optimiser les niveaux de stock.

- ♦ Analyse Financière :

Coût de Stockage : Calcul des frais de stockage, y compris entreposage et assurance.

Valorisation des Stocks : Utilisation de méthodes telles que le PMP ou FIFO pour refléter la valeur financière des stocks.

b. Dimensions de Traçabilité

- ♦ Dimension Physique :

Localisation des Produits : Gestion des emplacements dans l'entrepôt pour un suivi efficace.

Volume et Capacité : Suivi de la capacité physique des zones de stockage et de l'optimisation de l'espace, en tenant compte des dimensions et du volume des produits stockés, ainsi que l'unité de mesure relative au stockage de l'article.

Numéros de Lots et de Séries : Attribution de numéros uniques pour une traçabilité précise.

Numéro de Palette : Suivi des palettes pour optimiser l'espace de stockage et les mouvements logistiques.

Référence de Vente au Poids : Suivi des produits vendus au poids avec des SKU spécifiques pour une gestion précise des variations de poids.

Référence d'article SKU : Le SKU (Stock Keeping Unit) est une référence unique pour identifier et suivre facilement les produits, facilitant la gestion des stocks et des commandes grâce aux codes-barres.

- ♦ Axe analytique :

- i. *Activité de Production*

Cet axe analytique permet de suivre les stocks en relation avec les différentes activités de production de l'entreprise. Il offre une vision détaillée des coûts associés à chaque étape du processus de production.

Objectifs :

- Comprendre les coûts de production.
- Optimiser les processus de fabrication.
- Identifier les inefficacités et les opportunités d'amélioration.
- Gérer les matières premières, les produits semi-finis et les produits finis de manière efficace.

ii. Business Unit.

Cet axe analytique se concentre sur l'analyse des stocks et des coûts par unité commerciale ou secteur d'activité spécifique au sein de l'entreprise. Il permet de mesurer les performances et la rentabilité des différents segments de marché ou lignes de produits.

Objectifs :

- Mesurer les performances des différentes unités commerciales.
- Évaluer la rentabilité des segments de marché.
- Prendre des décisions stratégiques basées sur la performance des différentes unités.

iii. Canal de Vente

Cet axe analytique permet de suivre les stocks et les coûts en fonction des différents canaux de vente. Il aide à optimiser les niveaux de stock selon les canaux de vente et à identifier les canaux les plus rentables.

Objectifs :

- Optimiser les niveaux de stock pour chaque canal de vente.
- Identifier les canaux de vente les plus rentables.
- Ajuster les stratégies de distribution pour maximiser les ventes et la rentabilité.

3-1-2. Gestion de Journal de Stock

La gestion du journal de stock, en particulier les aspects liés à la consommation et à la production de stock, est cruciale pour suivre l'utilisation des matériaux et la création de produits finis. Voici une explication détaillée des différents aspects de cette gestion.

a. Consommation de stock

Enregistrement : Matériaux utilisés pour la production, consommation interne, dons ou déchets.

Suivi des Quantités : Quantités utilisées et mises à jour en temps réel.

Analyse : Rapports et tendances de consommation.

b. Production de Stock

Enregistrement : Produits finis et semi-finis.

Suivi des Quantités : Quantités produites et mises à jour en temps réel.

Analyse : Rapports de production et efficacité.

c. Traçabilité et Contrôle

Journal des Transactions : Enregistrement détaillé des transactions.

Audit et Réconciliation : Traçabilité pour audits.

d. Contrôles Internes :

Vérifications Périodiques : Comparaison des enregistrements avec les quantités physiques.

Ajustements de Stock : Rectification des écarts.

3-1-3. Gestion d'Ordre de Transfert

La gestion des ordres de transfert est une fonctionnalité clé dans la gestion avancée des stocks pour coordonner et optimiser le déplacement des produits entre différents emplacements ou entrepôts.

a. Process de transfert de stock.

Initiation et Soumission :

- Demande initiée par un employé autorisé, incluant les détails nécessaires.
- Soumission pour validation par le responsable d'entrepôt.

Validation et Mise à Jour :

- Validation de la demande, mise à jour des stocks et confirmation envoyée.
- Création et validation des ordres de transfert par l'expéditeur.

Réception :

- Mise à jour des stocks lors de la réception des articles transférés.

b. Qualité de service de satisfaction des demandes de transfert.

La qualité de service de satisfaction des demandes de transfert est cruciale pour assurer une gestion efficace et fluide des stocks au sein d'une entreprise. Elle repose sur plusieurs dimensions et indicateurs clés qui permettent de mesurer et d'améliorer la performance des processus de transfert de stock. Voici les aspects essentiels à considérer :

Taux de Satisfaction : Pourcentage de demandes satisfaites en quantité et qualité.

Temps de Traitement : Durée totale pour traiter une demande de transfert.

Disponibilité des Stocks : Mesure de la disponibilité des articles pour les transferts.

Coût des Transferts : Coût total associé aux transferts de stock.

3-1-4. Gestion de Journal de Comptage.

La règle de gestion des journaux de comptage encadre le processus de comptage physique des stocks dans un entrepôt. Elle vise à garantir l'exactitude des enregistrements de stock en comparant les quantités physiques avec les données théoriques et en ajustant les écarts identifiés.

Initiation et Création : Création d'un journal pour un entrepôt spécifique, avec spécification des articles à compter.

Blocage des Articles : Verrouillage des transactions de stock pour éviter les modifications pendant l'inventaire.

Saisie Précise des Comptages : Inclusion des valeurs unitaires, quantités et dimensions de stockage.

Clôture et Ajustement : Vérification des Données : Examen des lignes de comptage et des coûts.

Clôture du Journal : Verrouillage du journal et ajustement des stocks.

Libération des Transactions : Déblocage des transactions pour reprendre les opérations normales.

3-1-5. Gestion de Qualité

La gestion de la qualité dans un système de gestion de stock avancé implique deux aspects principaux : les contrôles de qualité et la gestion des non-conformités. Voici une explication détaillée de ces éléments :

a. Contrôles de Qualité

Inspection à la Réception : Vérification minutieuse des produits reçus selon des critères définis.

Inspection en Cours de Stockage : Contrôles réguliers pour détecter toute dégradation.

b. Gestion des Non-Conformités

Enregistrement : Identification et documentation des non-conformités.

Analyse des Causes : Enquête sur les causes profondes des problèmes.

Historique et Rapports : Traçabilité des non-conformités et génération de rapports pour améliorer les processus.

3-1-6. Comptabilité de stock

La gestion des coûts est essentielle pour évaluer le stock (matières premières, produits finis, emballages). Les méthodes comme FIFO, LIFO, et la moyenne pondérée permettent de suivre la valeur des articles et d'optimiser les ressources, aidant à une gestion efficace du coût de revient.

a. Les opérations impactant la valeur de stock.

Réception de marchandises : Augmente la valeur du stock selon le coût d'achat.

Production et assemblage : Le coût des matières premières diminue, tandis que celui des produits finis augmente.

Expéditions et ventes : Réduit la valeur du stock selon la méthode d'évaluation utilisée.

Inventaire physique : Ajuste le stock pour les écarts physiques.

Transferts internes : Modifie la valeur du stock selon les emplacements.

Retours de marchandises et ajustements : Affecte la valeur en fonction des retours ou réévaluations périodiques.

Destruction : Retire les articles endommagés ou obsolètes du stock.

b. Les transactions de stock.

Transactions de consommation : Réduisent le stock (ventes, prélèvements, pertes).

Transactions de production : Augmentent le stock (réception de matières, fabrication).

c. Méthode de calcul de coût des articles en stock.

Les transactions de consommation et de production de stock peuvent avoir des impacts significatifs sur le coût des articles en stock, selon les différentes méthodes de coût utilisées. Voici une explication détaillée pour chaque méthode couramment utilisée :

♦ **Méthode PMP :**

La moyenne pondérée (PMP) est la valeur moyenne de chaque unité de stock, mise à jour à chaque nouvelle réception. Elle résulte de la multiplication de chaque transaction par son prix de revient, reflétant son importance en quantité.

Lorsque les marchandises sont reçues à la suite d'une transaction, le prix de réception est utilisé pour mettre à jour la valeur du prix moyen pondéré. Le prix moyen pondéré est calculé selon la formule suivante :

$$\frac{(\text{Stock physique existant} * \text{PMP existante}) + (\text{quantité de réception} * \text{prix de transaction})}{(\text{Stock physique existant} + \text{quantité de réception})}$$

Exemple de calcul de PMP :

Date	Transaction	Qtité	Prix U.	Qtité de stock	Valeur de stock	PMP
01/06/2024	Stock initial	10	16	10	160	16

07/06/2024	Entrée	100	19,5	$(10+100) = 110$	$(160+1950) = 2110$	$(2110/110) = 19,18$
10/06/2024	Sortie	15	19,18	$(110-15) = 95$	$(95 \times 19,18) = 1822,27$	$(1822,27/95) = 19,18$
20/06/2024	Sortie	30	19,18	$(95-30) = 65$	$(65 \times 19,18) = 1246,81$	$(1246,81/65) = 19,18$
01/07/2024	Entrée	80	22	$(80+65) = 145$	$(1246,81+1760) = 3006,81$	$(3006,81/145) = 20,73$

Tableau 1 Exemple de calcul de PMP

♦ Méthode FIFO :

La méthode FIFO (premier entré, premier sorti) valorise les sorties de stock au coût de l'article le plus ancien dans le stock. Cette méthode est généralement utilisée pour les produits périssables afin d'épuiser les lots arrivés en premier face au risque de perte de valeur par obsolescence. Le système crée des règlements où la première réception correspond à la première sortie, et ainsi de suite.

Exemple de calcul de FIFO :

Date	Transaction	Qtité	Prix U.	Montants	Qtité	Prix Unitaire	Montants
01/06	Stock Initial	80	25	2000	80	25	2000
02/06	Achat	30	32	960	80	25	2000
					30	32	960
10/06	Achat	70	28	1960	80	25	2000
					30	32	960
					70	28	1960
15/06	Vente (100)	80	25	2000	10	32	320
		20	32	640	70	28	1960
22/06	Achat	20	30	600	10	32	320
					70	28	1960
					20	30	600
25/06	Vente (50)	10	32	320	30	28	840
		40	28	1120	20	30	600

Tableau 2 Exemple de calcul de FIFO

d. Gestion de coût physique et financier.

Augmentations physiques : Ex : Réceptions de commande, fin d'ordre de fabrication.

Sorties physiques : Ex : Bon de livraison, prélèvements d'ordres de fabrication.

e. Dimensionnement de calcul de quantité et coût de stock.

Le stock, de chaque article, peut être évalué en quantité et en coût suivant deux axes dimensionnels :

- Groupe de dimension de stockage :
 - Site
 - Entrepôt
 - Emplacement

Pour la gestion des dimensions de stockage, l'administrateur système doit définir les dimensions nécessaires pour le calcul de la quantité et de la valeur du stock. Par défaut, le système utilise les dimensions site, entrepôt et emplacement pour suivre la quantité de stock.

Article	Objet de suivi	Site	Entrepôt	Emplacement
A1	Coût	X	X	-
	Quantité	X	X	X
A2	Coût	X	-	-
	Quantité	X	X	X

Tableau 3 Exemple De Dimension De Stockage

- Groupe de dimension de suivi :
 - Numéro du lot
 - Numéro de série
 - Propriétaire.
 - Palette.

Pour la gestion des dimensions de suivi, l'administrateur système doit définir les dimensions nécessaires pour le calcul de la quantité et de la valeur du stock.

Article	Objet de suivi	Lot	Série	Palette	Propriétaire
A1	Coût	X	X	-	-
	Quantité	X	X	X	-
A2	Coût	X	-	-	-
	Quantité	X	X	X	-
A3 (Consignation)	Coût	-	-	-	-
	Quantité	-	-	-	X

Tableau 4 Exemple De Dimension De Suivi

f. Modèle d'article :

Un modèle d'article est une configuration utilisée pour définir les méthodes de calcul de coût des articles et les stratégies de stockage. Voici les éléments clés d'un modèle de prix d'article :

- Méthode de valorisation de stock (PMP, FIFO)
- Stock physique négative.
- Produit stocké.

g. La Clôture de stock.

La clôture de stock permet de finaliser les ajustements et les calculs de coûts en fin de période.

3-1-7. Réception de bon de commande :

La gestion de la réception de stock passe par plusieurs étapes essentielles pour s'assurer que les produits sont correctement enregistrés et conformes à la commande. Voici un résumé du processus :

1. **Sélection et vérification des bons de commande** : Le gestionnaire sélectionne le bon de commande dans le système et vérifie les lignes de commande (quantité, dimensions). Les différences entre ce qui a été commandé et ce qui est reçu sont notées.
2. **Création de bons de réception** : Lors de la réception des articles, un bon de réception est généré automatiquement en statut brouillon, permettant des modifications avant validation.
3. **Contrôle de qualité** : Pour les articles nécessitant un contrôle, un ordre de qualité est créé. Les résultats sont enregistrés dans le système, et les lignes non conformes sont annulées.
4. **Validation et mise à jour des stocks** : Après la validation des bons de réception, les stocks sont mis à jour en termes de quantité et de valeur.
5. **Impression des bons de réception** : Une fois validé, le bon de réception est imprimé pour être archivé.
6. **Coordination gestionnaire/équipe qualité** : Collaboration entre les deux parties pour s'assurer que les quantités, dimensions, et obligations de contrôle sont correctement enregistrées.

3-1-8. Administration de la Cartographie

Les éléments clés à codifier sont les organisations, sociétés, sites, entrepôts, zones, allées, et emplacements. Chaque entité doit avoir un code unique pour garantir une identification précise et une gestion efficace des stocks.

- **Organisation** : Un code unique est attribué à chaque organisation avec des informations de contact à jour.
- **Sociétés, sites, entrepôts** : Chaque niveau hiérarchique doit être clairement codifié, en détaillant les caractéristiques comme la localisation et les conditions de stockage.
- **Zones, allées, emplacements** : Des codes uniques sont attribués à chaque zone physique de l'entrepôt avec des indications sur la capacité et l'accès.

3-1-9. Gestion Master-data

La codification des données de base inclut les comptes, fournisseurs, et articles.

- **Comptes et fournisseurs** : Chaque compte et fournisseur reçoit un code unique. Les informations doivent être mises à jour régulièrement pour éviter les erreurs.

- **Fiches articles** : Les articles sont identifiés par un code unique, avec des descriptions complètes et classifiés pour faciliter la gestion des stocks et des achats. Les données doivent être synchronisées en temps réel pour assurer la cohérence dans tous les systèmes.

3-2- Spécification de besoins non fonctionnels

L'analyse des besoins non fonctionnels est essentielle dans le développement d'une application de gestion commerciale et de vente. Elle garantit que le système répond aux attentes des utilisateurs, maintient une performance élevée et respecte les normes et réglementations.

a. Performance

L'application doit pouvoir traiter un grand volume de transactions et supporter de nombreux utilisateurs simultanés sans sacrifier la réactivité ou la vitesse d'exécution.

b. Sécurité

La protection des données sensibles est primordiale. L'application doit garantir la confidentialité, l'intégrité et la disponibilité des informations via des mesures robustes comme l'authentification multi-facteurs, le chiffrement des données, et la détection des activités suspectes.

c. Fiabilité

L'application doit offrir une disponibilité continue, avec un minimum de temps d'arrêt. Des mécanismes de sauvegarde et de récupération des données doivent être en place pour assurer la continuité du service en cas de défaillance.

d. Évolutivité

L'architecture de l'application doit permettre une croissance future, facilitant l'ajout de nouvelles fonctionnalités et l'intégration avec d'autres systèmes au fur et à mesure des besoins de l'entreprise.

e. Facilité d'utilisation

L'interface utilisateur doit être intuitive et accessible à tous les niveaux de compétence. Un support complet, comprenant des tutoriels et une documentation claire, doit être mis à disposition pour guider les utilisateurs.

f. Compatibilité

L'application doit être compatible avec une variété de navigateurs web, d'appareils mobiles, et de systèmes d'exploitation, afin de garantir une expérience fluide et cohérente pour tous les utilisateurs.

g. Disponibilité

L'application doit être disponible 24/7 avec des plans de maintenance bien planifiés pour minimiser les interruptions de service.

h. Conformité

Elle doit respecter les réglementations locales et internationales, notamment en ce qui concerne le traitement des données et la protection de la vie privée (comme le RGPD en Europe).

3-3- Spécification technique

3-3-1. Architecture logicielle

Architecture de système de gestion de stock avancé (SGSA) : frontend Angular, du backend NestJS et de la base de données PostgreSQL

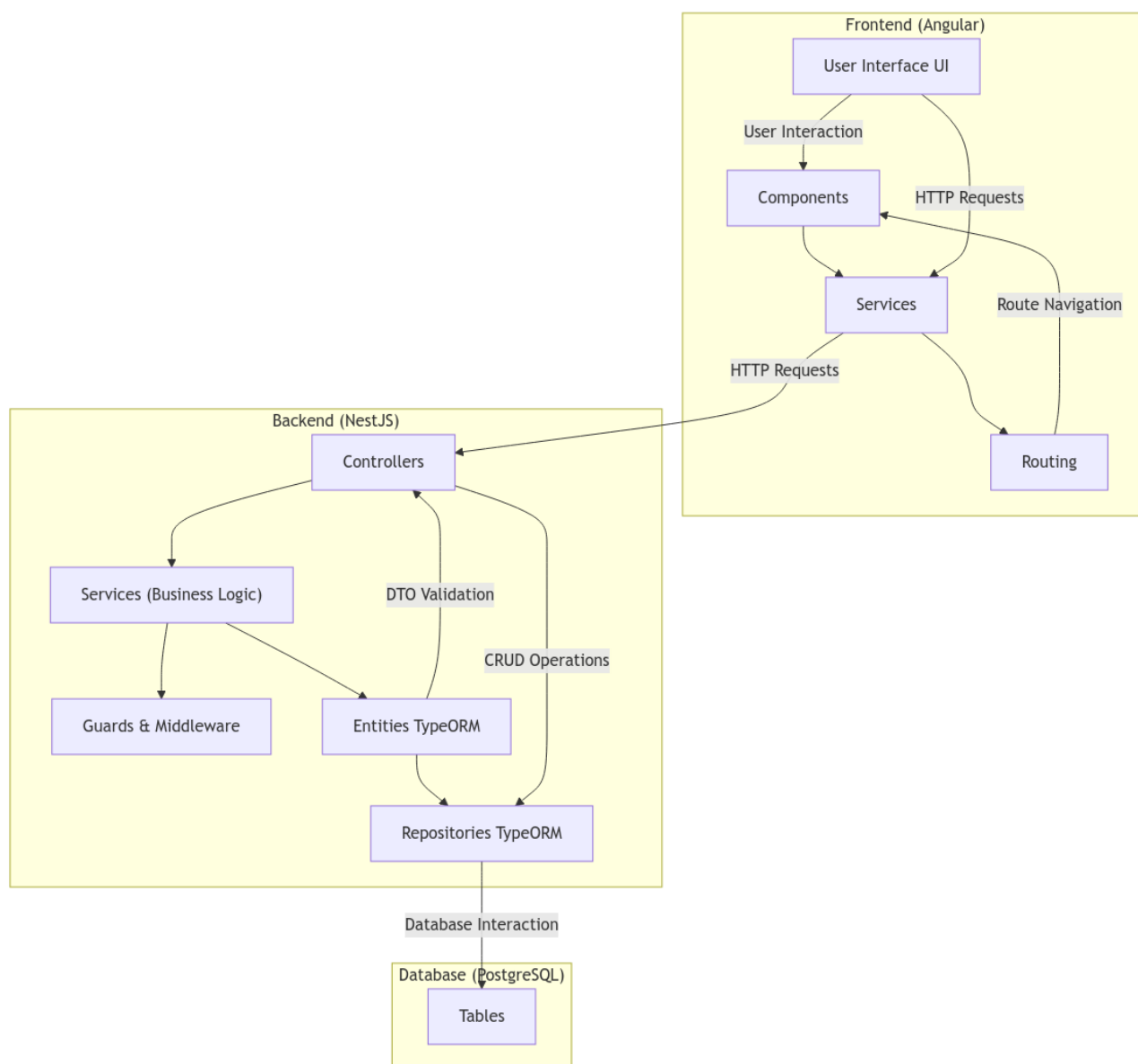


Figure 2 Architecture logicielle du projet

L'architecture de l'application repose sur une séparation claire des couches entre le frontend, le backend, et la base de données, facilitant ainsi la modularité et l'évolutivité du système. Le frontend est développé avec Angular, un framework puissant pour la création d'interfaces utilisateurs dynamiques et interactives. Il gère l'affichage des données et les interactions de l'utilisateur via des

composants, des services et un système de routage, permettant une navigation fluide entre les différentes vues de l'application.

Le backend, quant à lui, est construit avec NestJS, un framework Node.js orienté vers la création de services backend modulaires et maintenables. Ce dernier communique avec la base de données PostgreSQL à l'aide de TypeORM, un ORM qui facilite la gestion des entités, des relations entre les tables, et des requêtes à la base de données. Les DTOs (Data Transfer Objects) sont utilisés pour valider et formater les données échangées entre le frontend et le backend, garantissant la conformité des formats de données et réduisant les erreurs de communication.

3-3-2. Technologies et Outils Utilisés

a. Méthodes de conception/modélisation :

- ♦ UML :

(Unified Modeling Language ou Langage de modélisation unifiée en français) est un langage graphique de modélisation informatique. Ce langage est désormais la référence en modélisation objet, ou programmation orientée objet.

- ♦ BDR (Base de Données Relationnelle) :

Une base de données relationnelle est un ensemble d'informations qui organise les données dans des relations prédéfinies et les stockent dans une ou plusieurs tables (ou "relations") de colonnes et de lignes, ce qui permet de les consulter et de comprendre les relations entre différentes structures de données.

b. Versioning

- ♦ GitHub

Plateforme de collaboration basée sur Git, utilisée pour héberger le code source du projet. Grâce à GitHub, l'équipe de développement a pu travailler de manière collaborative et centralisée, tout en assurant la sécurité et la disponibilité du code.



c. Technologies

- ♦ Node.js :

Un environnement d'exécution JavaScript côté serveur utilisé pour développer le backend du projet. Node.js permet de créer des applications web évolutives et performantes grâce à son architecture non-bloquante et orientée événements.



d. Environnement de développement intégré (EDI)

- ♦ VS Code



Pour le développement de ce projet, j'ai utilisé Visual Studio Code (VS Code), un EDI open-source et performant. Grâce à ses nombreuses fonctionnalités comme le support des extensions, les outils de débogage intégrés, l'intégration de Git, le développement a été plus efficace et la gestion du code..

e. Logiciels

- ♦ DBeaver :



Utilisé pour la gestion et l'administration des bases de données, DBeaver permet de concevoir, interroger, et analyser les bases de données de manière efficace. Il prend en charge de nombreux systèmes de gestion de bases de données, facilitant ainsi l'interaction avec PostgreSQL utilisé dans ce projet.

f. Serveurs de bases de données

- ♦ PostgreSQL :



Utilisé comme système de gestion de base de données relationnelle (SGBDR) pour stocker et organiser les données du projet. PostgreSQL est un système de gestion de base de données relationnelle orienté objet puissant et open source qui est capable de prendre en charge en toute sécurité les charges de travail de données les plus complexes.

g. Framework

- ♦ Angular



Est un framework open-source de développement web créé par Google. Il est conçu pour faciliter la création d'applications web dynamiques et réactives. Angular utilise TypeScript comme langage principal, ce qui permet de bénéficier des avantages de la typage statique et d'une meilleure maintenance du code. Grâce à ses puissants outils et fonctionnalités, Angular nous a permis de développer une interface utilisateur riche et interactive, assurant une expérience utilisateur fluide et réactive.

- ♦ NestJS



Est un framework backend progressif construit sur Node.js et TypeScript. Il est fortement inspiré par les concepts d'architecture de modules de Angular, ce qui facilite une transition et une intégration harmonieuse entre le frontend et le backend. NestJS adopte une approche modulaire et fournit une architecture solide et évolutive, idéale pour la création d'applications

backend robustes et maintenables. NestJS nous a permis de construire une API backend sécurisée et performante, intégrant des fonctionnalités de gestion de la base de données, d'authentification et d'autorisation, ainsi que de communication en temps réel.

h. Librairies

- ♦ TypeORM :



Un ORM (Object-Relational Mapping) pour Type Script et JavaScript, utilisé avec PostgreSQL pour la gestion des bases de données relationnelles dans le projet. TypeORM simplifie les opérations de base de données en permettant une interaction avec celles-ci via des objets JavaScript/Type Script.

- ♦ PrimeNG :



PrimeNG est une bibliothèque de composants d'interface utilisateur pour Angular, offrant une vaste collection d'éléments visuels riches et personnalisables. Elle permet aux développeurs de créer des applications web interactives et réactives avec facilité.

- ♦ CoreUI pour Angular :



Est une bibliothèque d'interface utilisateur open source qui offre des composants prêts à l'emploi pour créer des applications web dynamiques et réactives. Elle facilite la conception de tableaux de bord et d'interfaces utilisateur administratives grâce à ses éléments flexibles et modulaires. CoreUI s'intègre parfaitement avec Angular, permettant aux développeurs de gagner du temps tout en maintenant des standards de qualité élevés.

i. Tests API

- ♦ Postman



Postman est un outil de développement d'API qui permet aux utilisateurs de concevoir, tester et documenter leurs interfaces de programmation d'applications (API). Il offre une interface utilisateur intuitive pour envoyer des requêtes, inspecter les réponses et automatiser les tests.

- ♦ Mockoon



Mockoon a été utilisé pour simuler des API en créant des serveurs moque. Cet outil a permis de tester le frontend de manière indépendante, même lorsque le backend n'était pas encore entièrement développé ou disponible. En configurant des réponses personnalisées dans Mockoon, nous avons pu reproduire différents scénarios d'utilisation et s'assurer

que le frontend se comportait correctement dans diverses conditions sans avoir besoin d'accéder aux serveurs de production ou de développement. Cela a grandement accéléré les phases de développement et de test.

j. Langages de programmation

- ♦ Type Script

Est un sur ensemble de JavaScript qui introduit des types statiques facultatifs, facilitant ainsi le développement d'applications robustes et maintenables. Conçu par Microsoft, il améliore la productivité des développeurs en offrant des fonctionnalités avancées comme l'auto-complétions, la vérification de types et les interfaces. En permettant de détecter les erreurs au moment de la compilation plutôt qu'à l'exécution, Type Script contribue à une meilleure qualité du code. C'est un outil essentiel pour les projets de grande envergure nécessitant une structure solide.



- ♦ HTML

HTML (HyperText Markup Language) est le langage standard utilisé pour créer et structurer les pages web. Il permet de définir les éléments de contenu, comme les titres, les paragraphes et les liens hypertextes, formant ainsi la base de toute page web.



- ♦ SCSS

SCSS (Sassy Cascading Style Sheets) est une extension de CSS qui introduit des fonctionnalités avancées telles que les variables, facilitant ainsi la gestion et la maintenance des feuilles de style. Utilisé par les développeurs web, SCSS améliore la productivité et la modularité du code CSS.



k. Outils de prototypage

- ♦ Balsamiq

Pour la création de maquettes et de prototypes, nous avons utilisé Balsamiq. Cet outil permet de concevoir rapidement et efficacement des wireframes pour les interfaces utilisateur, facilitant ainsi la visualisation et la validation des concepts avant de passer à la phase de développement. Balsamiq offre une interface intuitive et des éléments préconçus qui simplifient le processus de création de maquettes, contribuant ainsi à une meilleure communication des idées et des exigences entre les membres de l'équipe.



IV. Étude Conceptuelle

4-1- Diagramme de flux

Ce diagramme de flux offre une visualisation claire et structurée des étapes nécessaires pour gérer efficacement les réceptions de stock, incluant les contrôles de qualité et les mises à jour des inventaires. Il permet de s'assurer que chaque étape est suivie de manière rigoureuse et que les stocks sont maintenus avec précision et transparence.

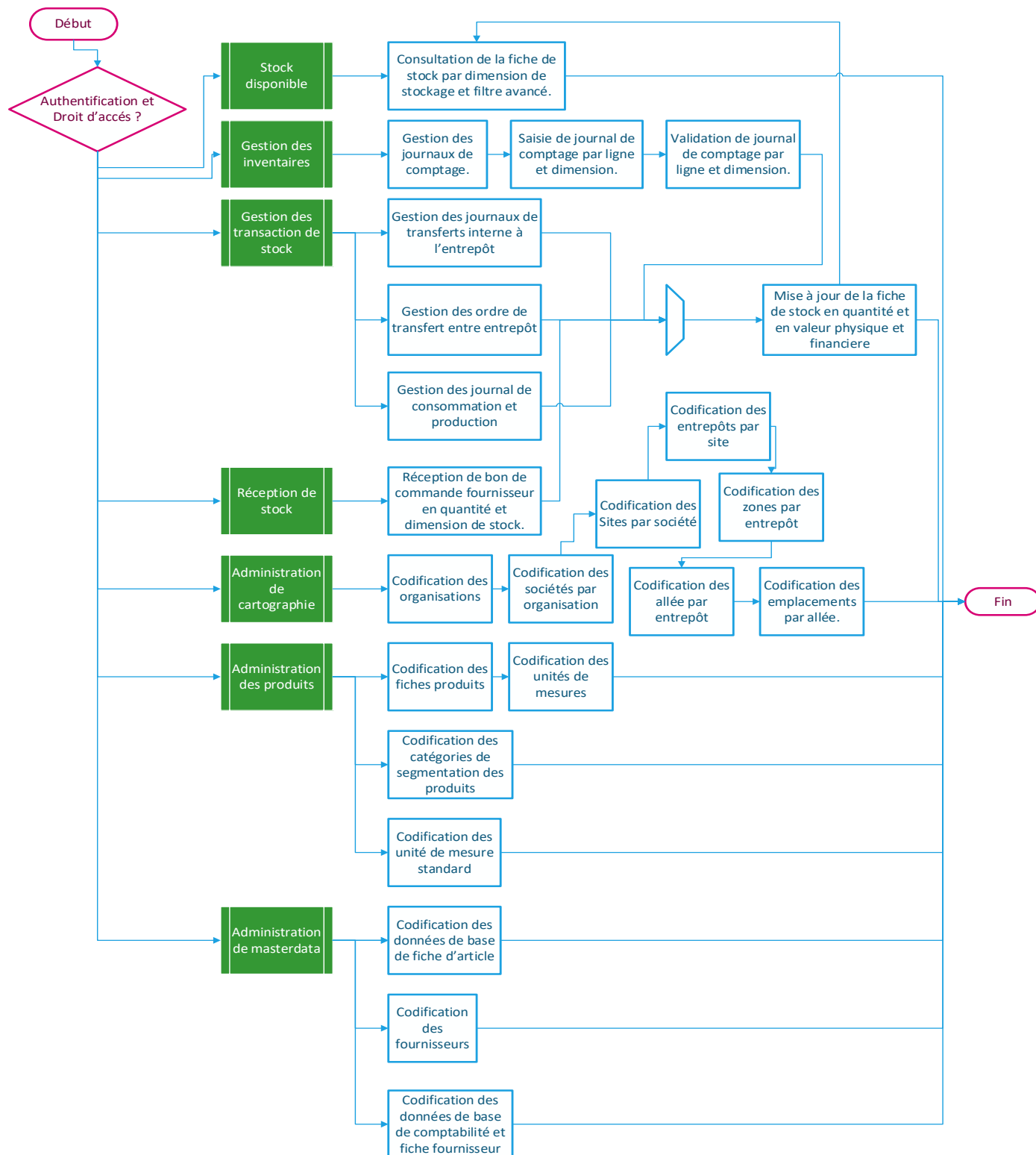


Figure 3 diagramme de flux

4-2- Diagramme Entité-Relation :

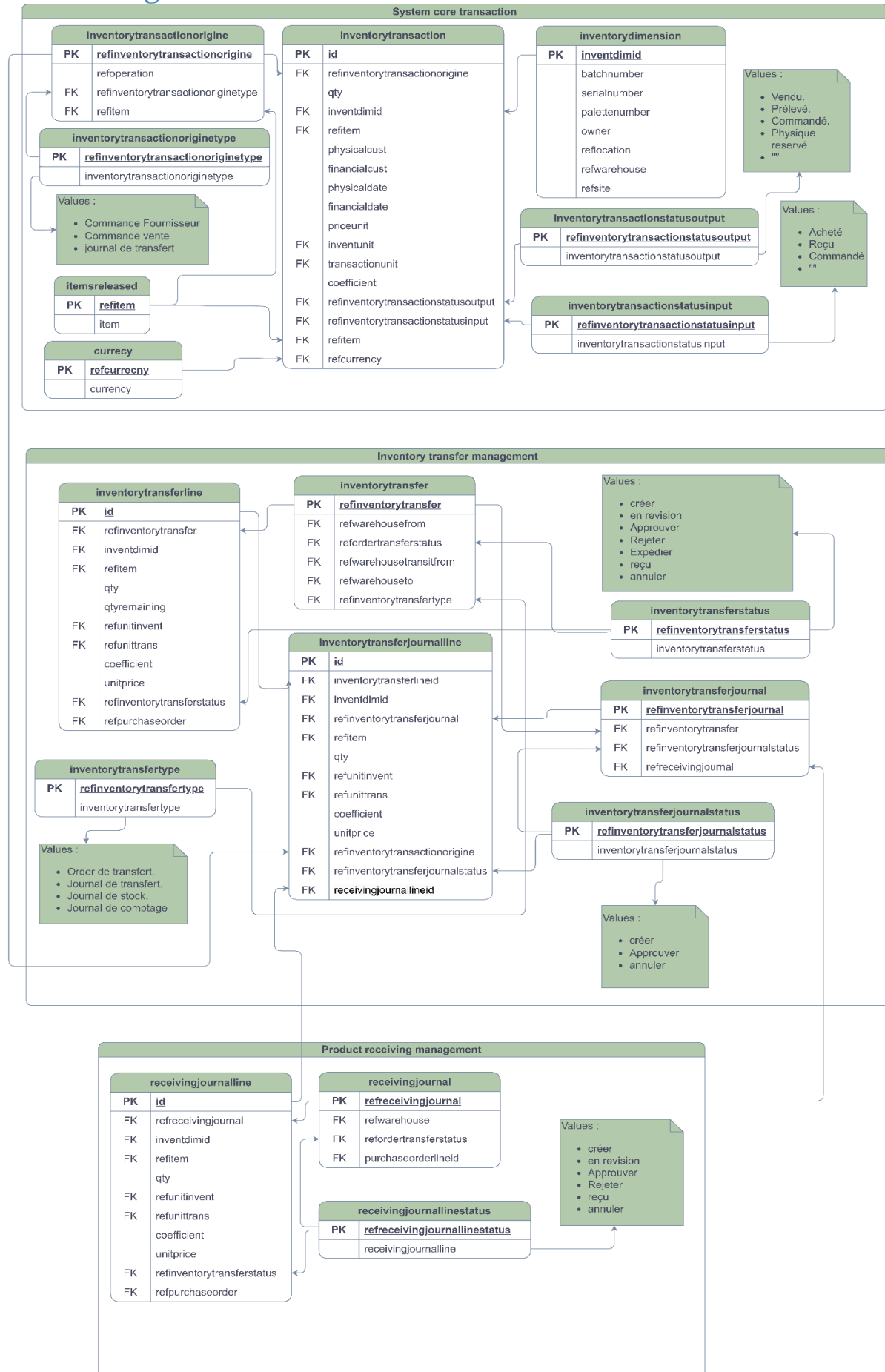


Figure 4 diagramme entité relation

4-3- Diagramme de Cas d'Utilisation (Use Case Diagram) :

4-3-1. Diagramme de gestion de consultation

Ce diagramme illustre le processus de consultation des stocks disponibles en termes de quantité et de valeur. Le magasinier est l'acteur principal qui accède aux informations de stock. Deux sous-processus sont inclus :

- Spécifier les dimensions du stock à inclure dans l'image de stock.
- Spécifier le statut du stock à inclure dans l'image de stock. Ces sous-processus permettent au magasinier de personnaliser la vue des stocks selon les besoins spécifiques de l'entreprise.

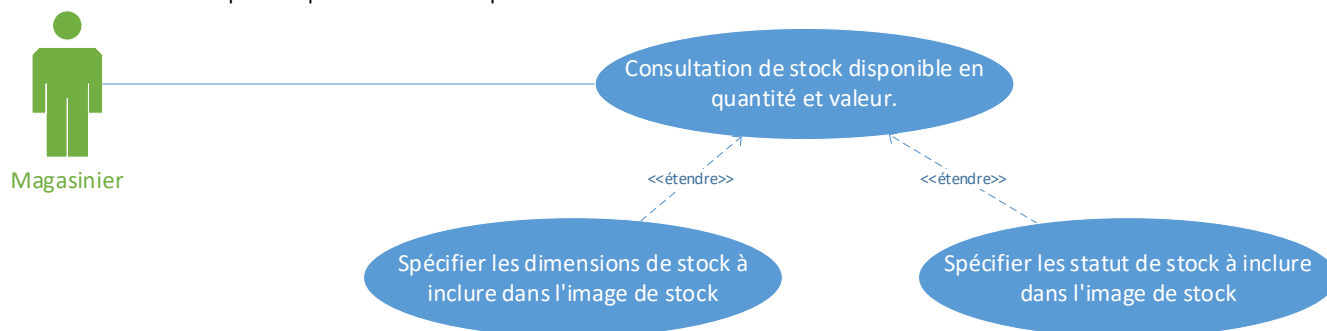


Figure 5 use case Diagramme de gestion de consultation

4-3-2. Diagramme gestion journaux de transfert

Ce diagramme décrit le processus de création et de gestion des journaux de transfert internes. Le magasinier crée d'abord l'entête du journal de transfert. Ensuite, les lignes de journal sont saisies par articles et dimensions. Une fois le journal de transfert créé, il peut être reçu par le service de gestion des stocks, qui vérifie la disponibilité des stocks par dimension et met à jour les stocks en quantité et en valeur. Ce processus assure une traçabilité et une gestion efficace des mouvements de stock internes.

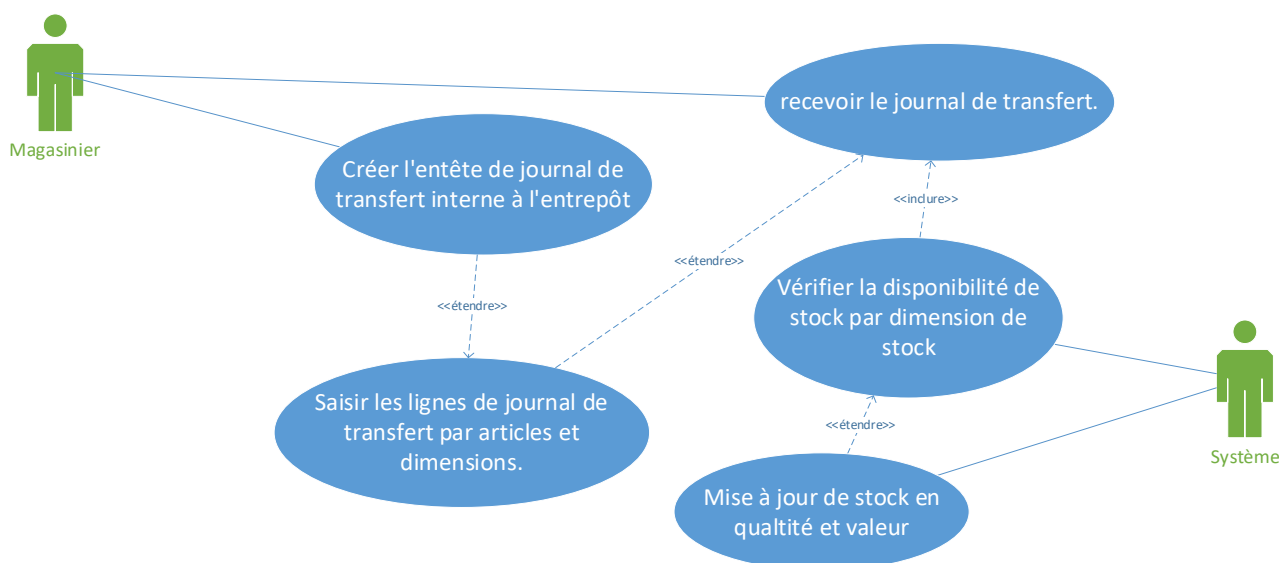


Figure 6 use case Diagramme gestion journaux de transfert

4-3-3. Diagramme gestion ordre de transfert

Ce diagramme représente un processus complet de gestion des ordres de transfert, divisé en trois parties :

1. Réception d'ordre de transfert :
2. Demande de transfert :
3. Expédition d'ordre de transfert :

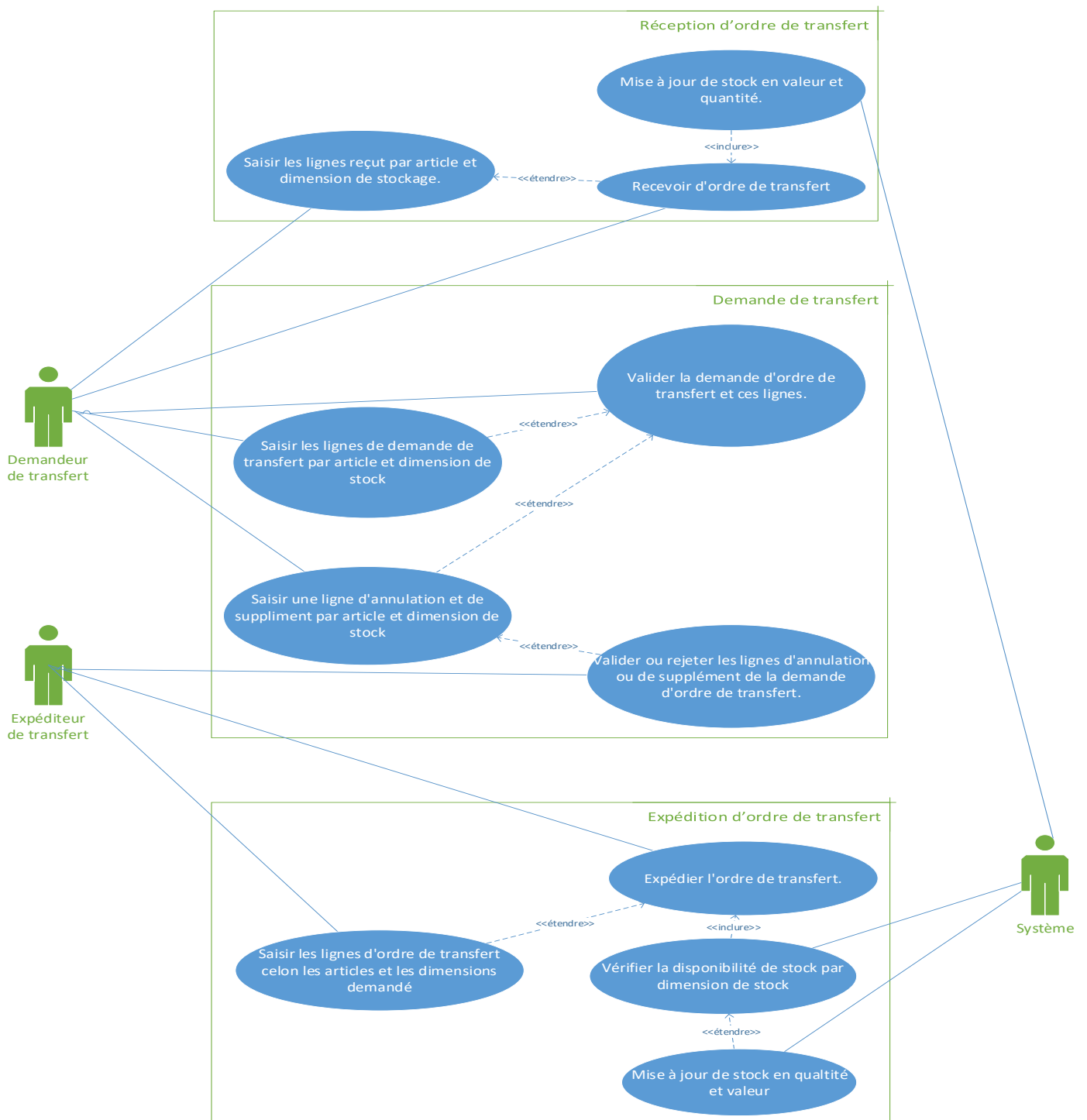


Figure 7 use case Diagram gestion ordre de transfert

4-3-4. Diagramme gestion d'inventaire

Ce diagramme de cas d'utilisation représente les différentes étapes impliquées dans le processus de comptage de stock. Il détaille les interactions entre le Magasinier et le Système pour la gestion des stocks. Voici les principaux cas d'utilisation et leurs relations :

Création d'un journal de comptage par dimension cartographique : Le Magasinier crée un journal de comptage en spécifiant des dimensions telles que l'entrepôt, la zone, l'allée et l'emplacement.

Spécifier les articles à compter : Le Magasinier sélectionne les articles spécifiques qui doivent être comptés.

Lancer le comptage : Une fois les articles spécifiés, le Magasinier lance le processus de comptage.

Comptage des articles à compter par dimension de stockage : L'Agent de comptage effectue le comptage physique des articles selon les dimensions spécifiées.

Validation de journal de comptage : Le Magasinier valide les résultats du comptage.

Calcul des écarts de stock par dimension de stock : Le Magasinier calcule les écarts entre le stock théorique et le stock physique.

Ajustement de stock en quantité et valeur : Le Magasinier ajuste les stocks en fonction des écarts constatés.

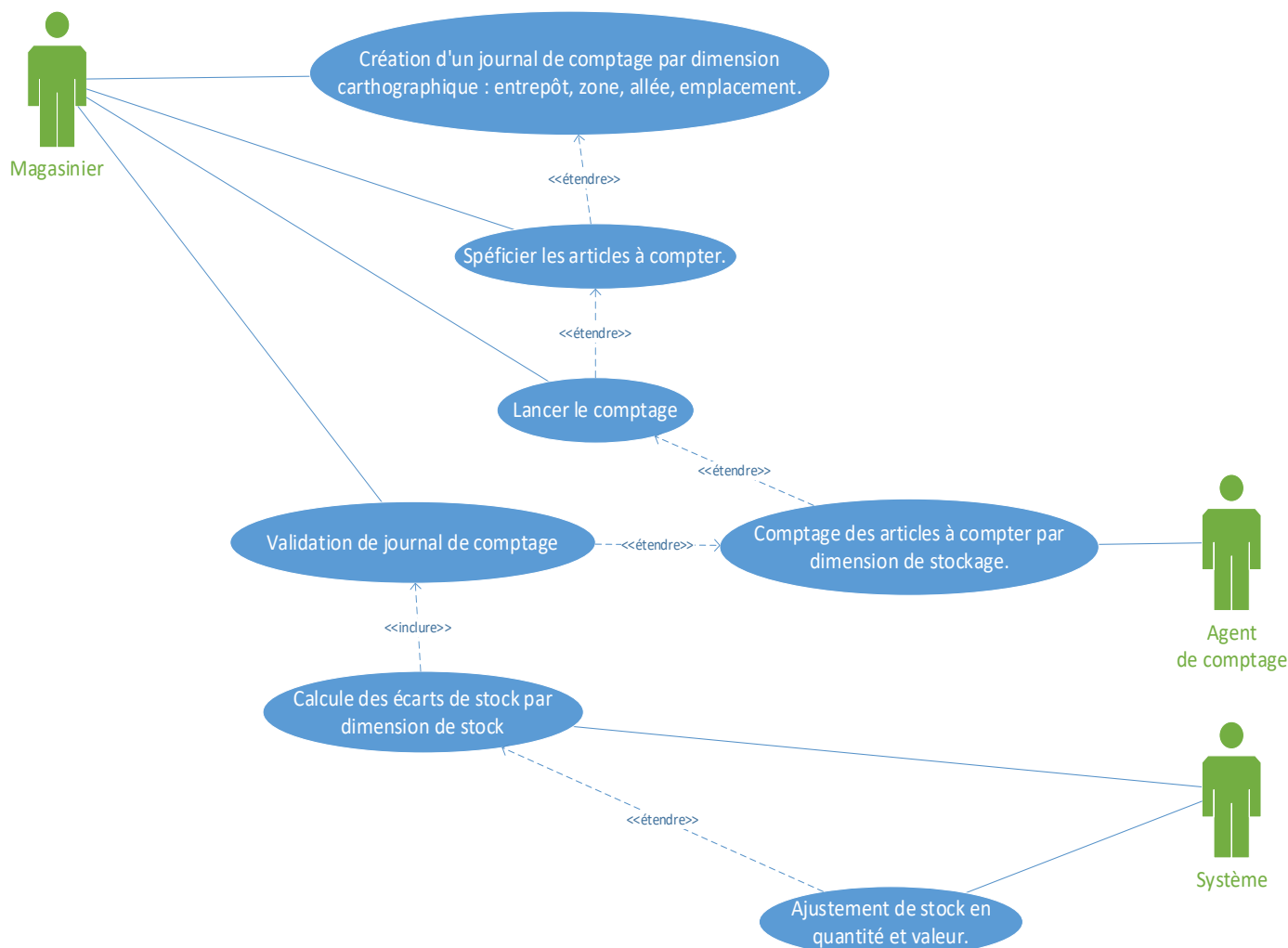


Figure 8 use case Diagramme gestion d'inventaire

4-3-5. Diagramme gestion de réception

Ce diagramme de cas d'utilisation illustre le processus de réception de stock, depuis la sélection du bon de commande jusqu'à la mise à jour des stocks après réception. Les principaux acteurs sont le Gestionnaire des stocks et le Qualiticien. Voici les principaux cas d'utilisation et leurs relations :

Sélectionner le bon de commande à recevoir : Le Gestionnaire des stocks choisit le bon de commande à traiter.

Saisir les lignes de bon de réception en quantité et dimension de stock : Le Gestionnaire des stocks enregistre les détails des articles reçus.

Création de bon de réception en statut Brouillon : Un bon de réception provisoire est créé pour la validation.

Vérification de quantité restante au bon de commande : Le Gestionnaire des stocks vérifie les quantités reçues par rapport au bon de commande.

Vérification d'obligation de contrôle de qualité par ligne de BC : Le Qualiticien contrôle la qualité des articles reçus.

Création des ordres de qualité par article : Des ordres de qualité sont créés pour chaque article nécessitant un contrôle.

Remplir les résultats de contrôle de qualité : Le Qualiticien enregistre les résultats du contrôle de qualité.

Validation de bon de réception : Le Gestionnaire des stocks valide le bon de réception après vérification.

Mise à jour de stock en quantité et valeur : Les stocks sont mis à jour en fonction des réceptions validées.

Annulation des lignes de BC non conforme : Les lignes de bon de commande non conformes sont annulées.

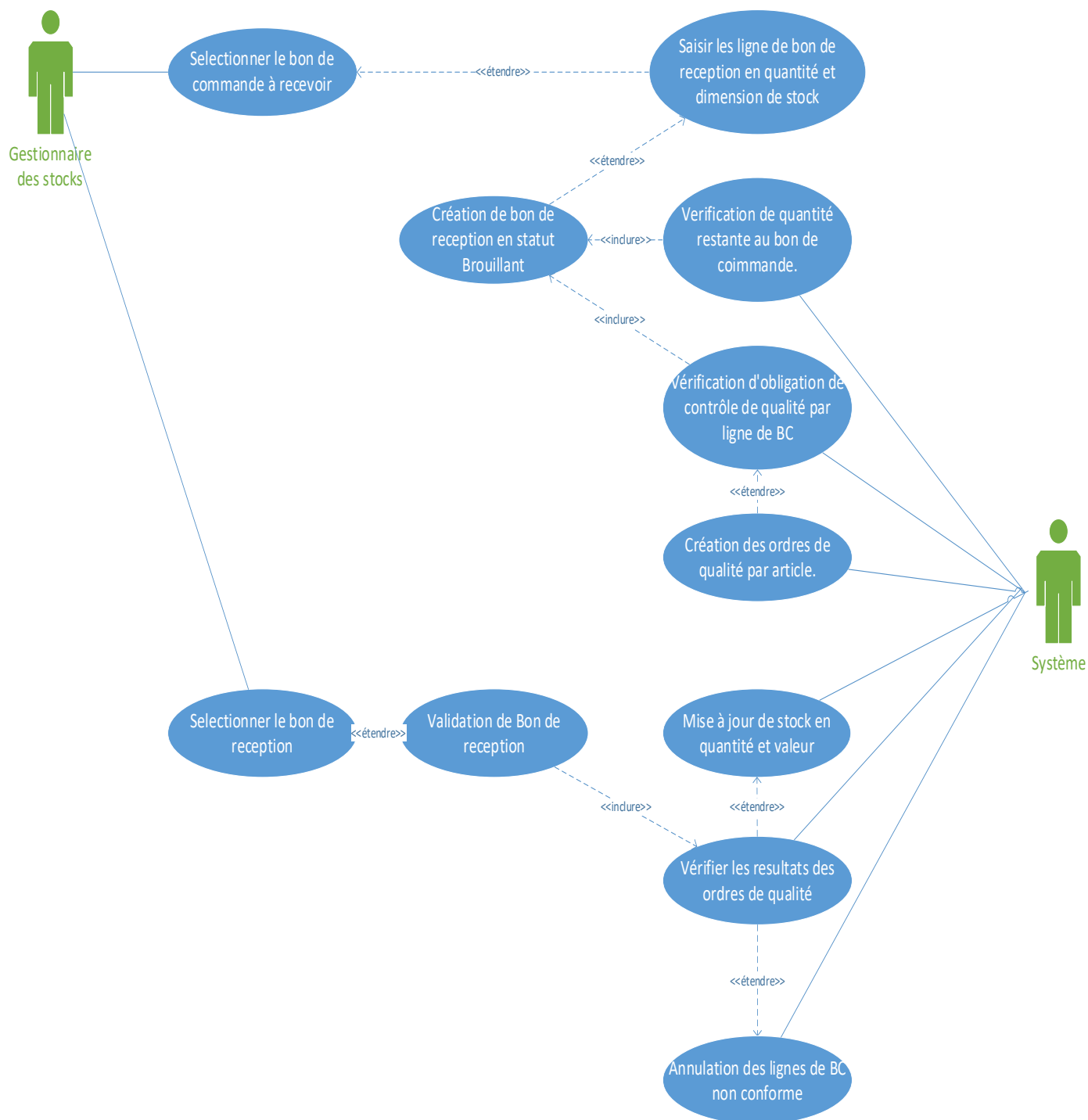


Figure 9 use case Diagramme gestion de réception

4-4- Diagramme d'activité

4-4-1. Gestion de réception

Le diagramme d'activité ci-dessus illustre le processus de réception des commandes Ce diagramme fournit une vue détaillée des étapes impliquées dans la réception, la vérification, et la mise à jour des stocks suite à une commande.

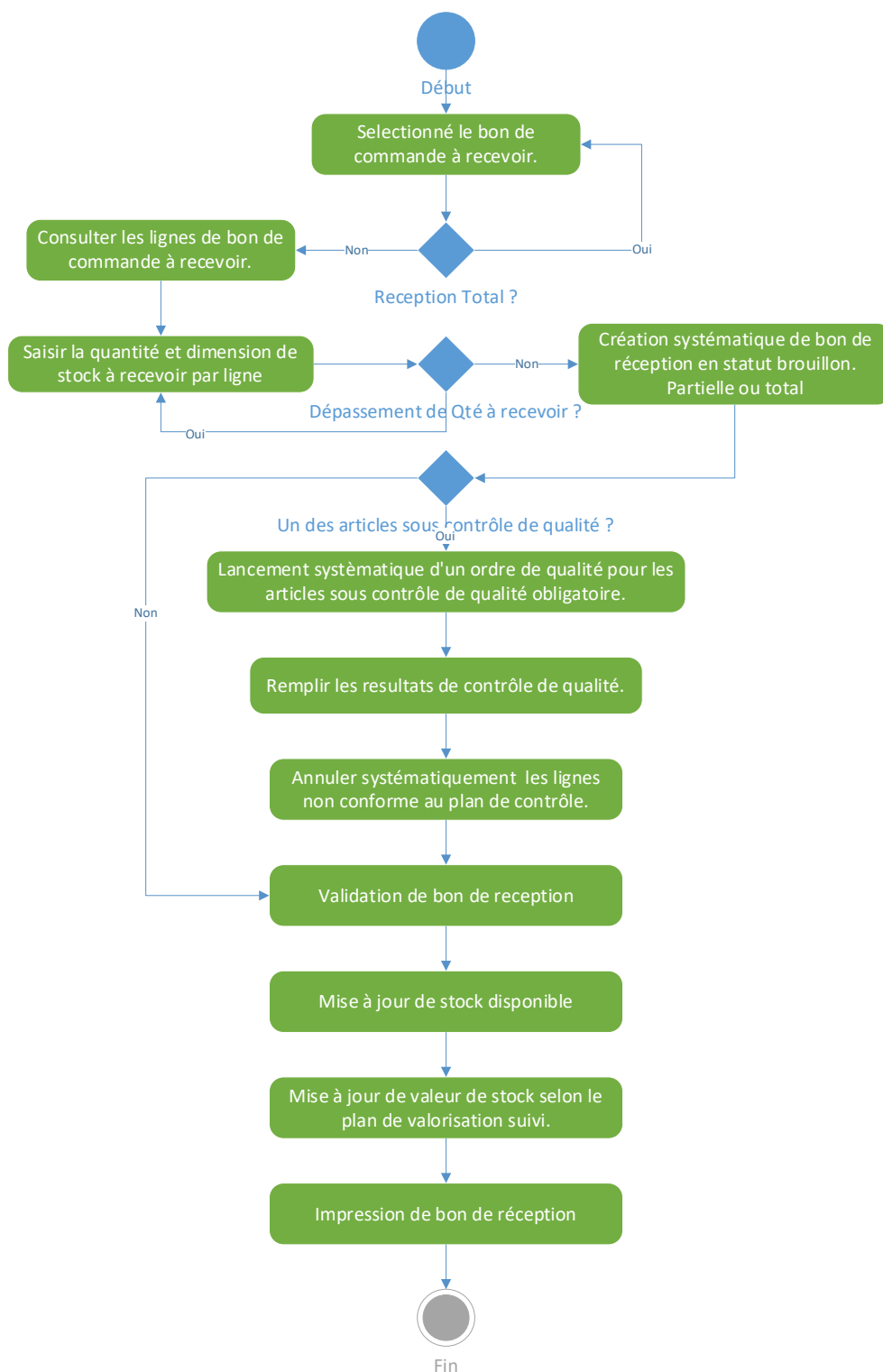


Figure 10 Activity Diagram Gestion de réception

4-4-2. Gestion de consultation de stock disponible

Le diagramme de consultation de stock décrit de manière claire et concise le processus permettant de consulter les informations de stock dans un système de gestion des stocks. Ce diagramme est structuré pour illustrer les étapes successives nécessaires à la réalisation d'une demande de consultation de stock, depuis l'initiation de la demande jusqu'à l'affichage des résultats.

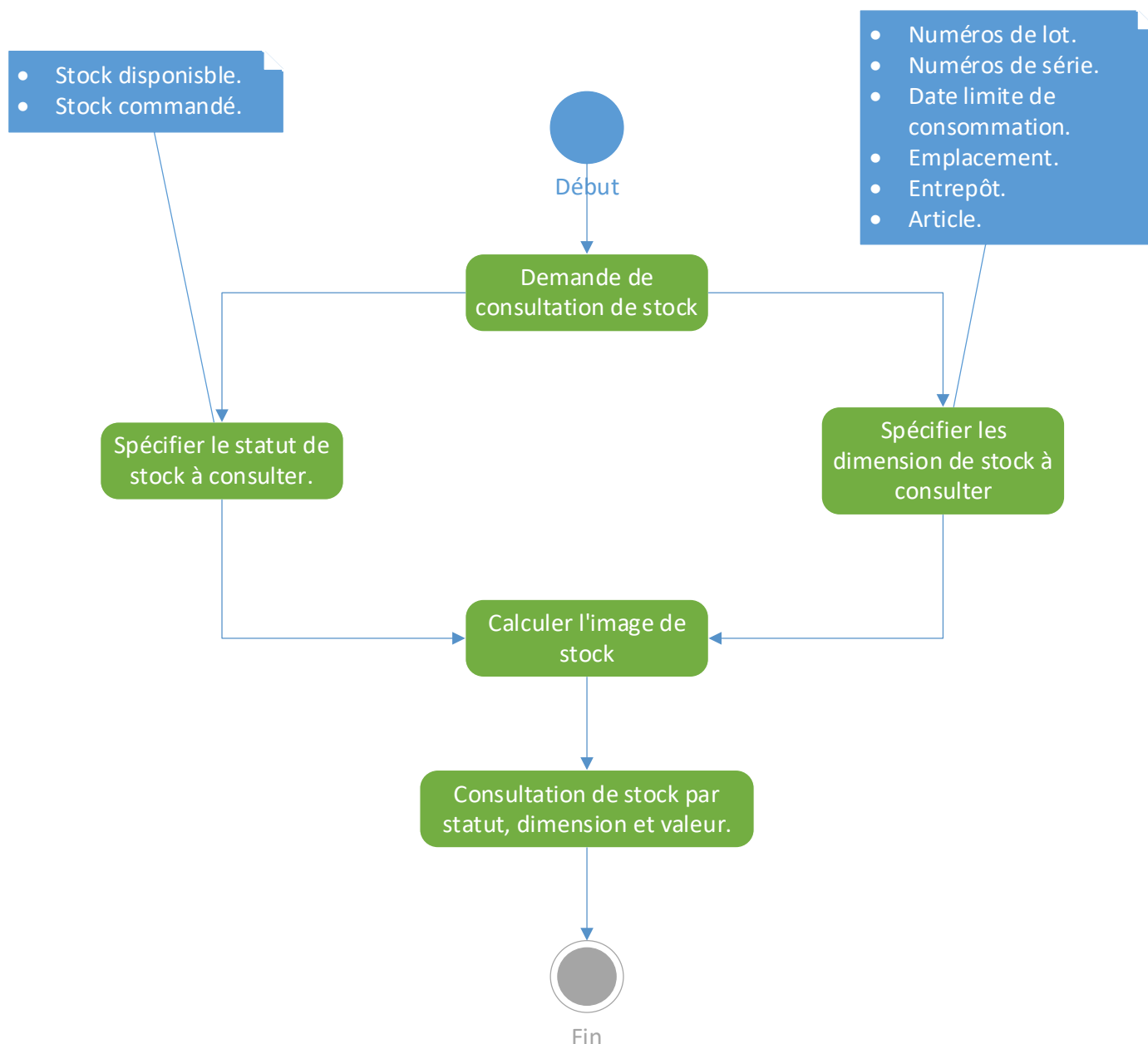


Figure 11 Activity Diagram Gestion de consultation de stock disponible

4-4-3. Gestion d'inventaire

Ce diagramme d'activité illustre le processus de comptage de stock dans un entrepôt. Le processus commence par la création d'un journal de comptage, où les détails comme l'entrepôt, la zone et l'allée sont spécifiés. Ensuite, les articles à compter sont sélectionnés. Si le journal est ajusté, les transactions de stock pour ces articles sont bloquées. Les lignes de comptage, incluant la valeur unitaire, la quantité et les dimensions de stock des articles, sont saisies. Après la clôture du journal de comptage, les écarts entre le stock physique et théorique sont ajustés. Enfin, les transactions de stock pour les articles bloqués sont libérées.

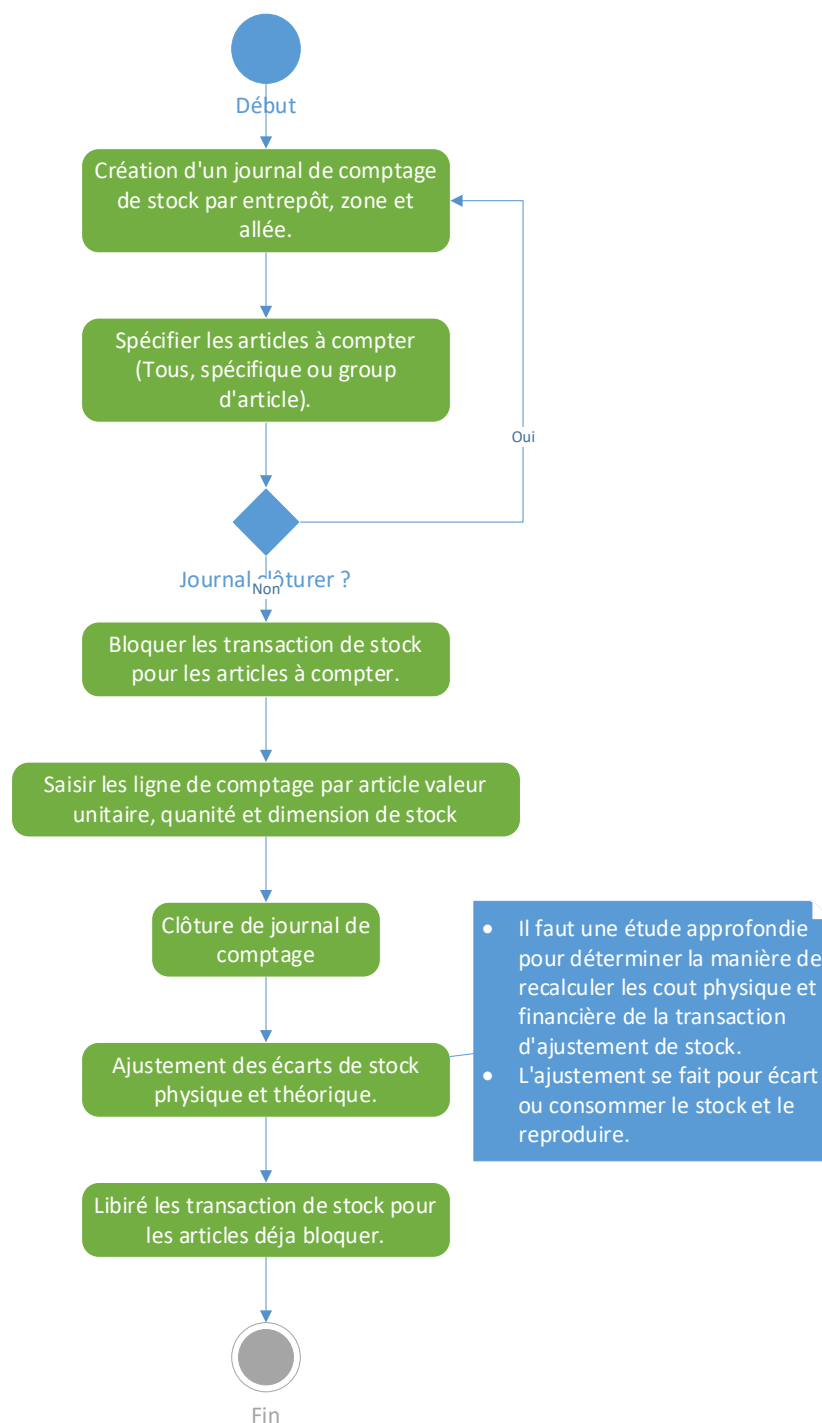


Figure 12 Activity diagramme Gestion d'inventaire

4-4-4. Gestion journaux de transfert

Ce diagramme d'activité montre le processus de transfert de stock au sein du même entrepôt. Le processus commence par la création d'un journal de transfert. Si la demande de transfert n'est pas encore clôturée, les lignes de journal, y compris les articles, les quantités et les dimensions de stockage, sont saisies. Une fois le journal de transfert clôturé, la disponibilité du stock est vérifiée. Si le stock est disponible, les valeurs de stock et la disponibilité du stock sont mises à jour. Si le stock n'est pas disponible, le processus de transfert doit être réévalué.

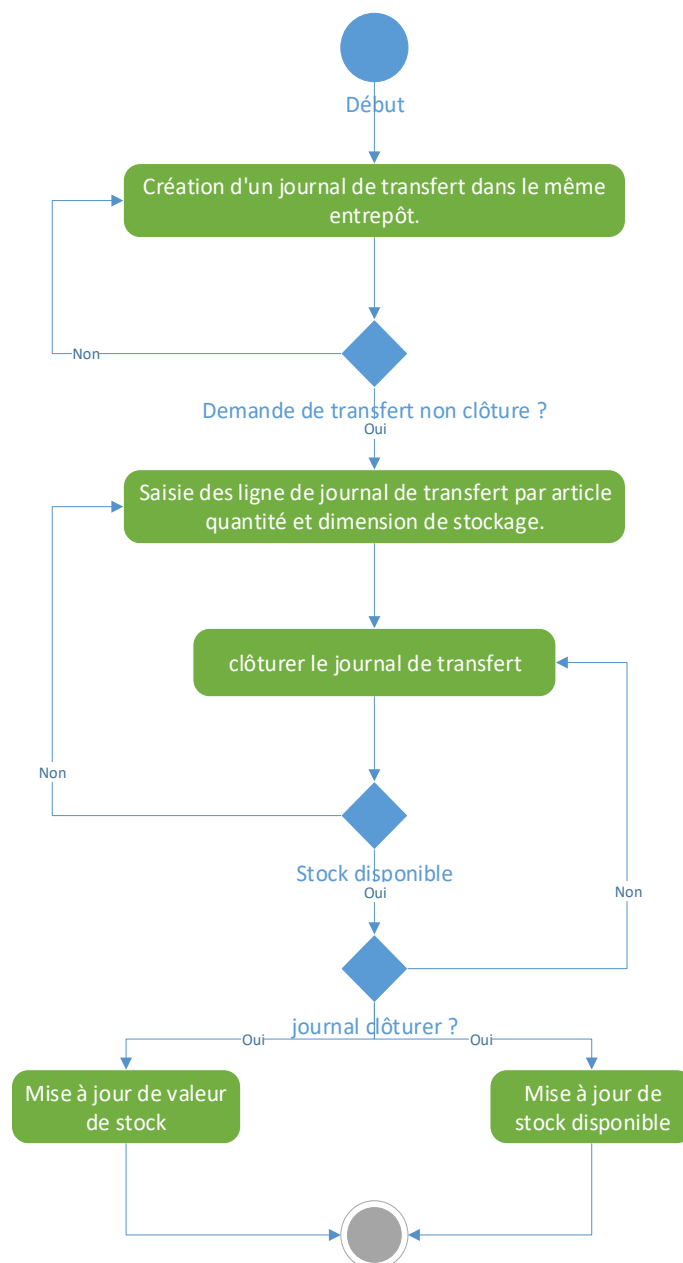


Figure 13 Activity Diagram gestion journaux transfert

4-4-5. Gestion ordre de transfert

Ce diagramme d'activité décrit le processus de gestion des ordres de transfert de stock entre deux entrepôts. Le processus débute par la création d'une demande de transfert d'un entrepôt demandeur à un entrepôt préparateur. Si la demande de transfert n'est pas encore soumise, les lignes de la demande, incluant les articles et les quantités, sont saisies. Une fois la demande soumise, les lignes de demande sont consultées. Si la ligne de demande est satisfaite à 100% et que le stock est disponible, la demande de transfert est partiellement ou totalement satisfaite selon les articles, les quantités et les dimensions de stock. Enfin, la mise à jour de la valeur du stock et de la disponibilité du stock est effectuée, et l'ordre de transfert est clôturé.

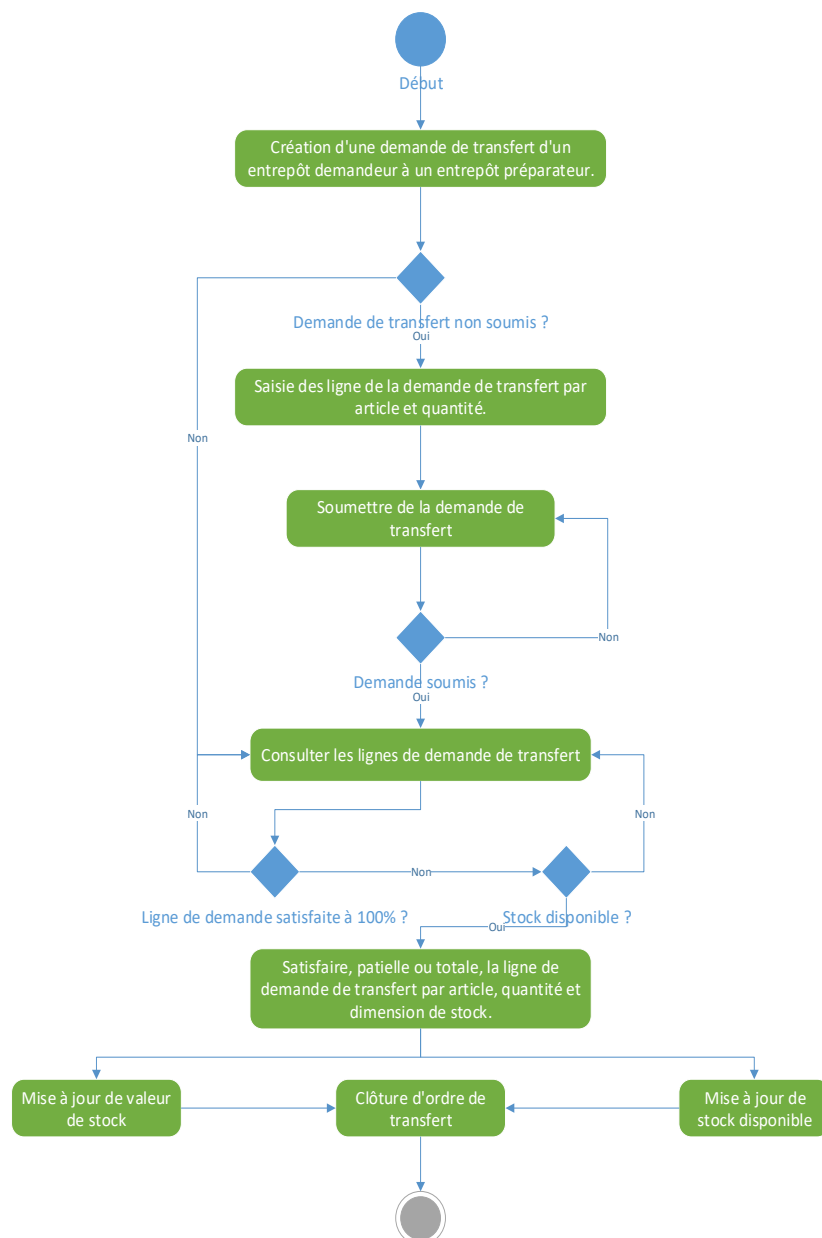


Figure 14 Activity Diagram Gestion ordre transfert

V. Conclusion

La mise en place du SGSA répond aux défis de COPAG en termes d'efficacité et de précision dans la gestion des stocks. Ce projet s'inscrit dans une démarche d'amélioration continue des opérations logistiques, permettant à COPAG d'augmenter sa compétitivité tout en minimisant les coûts. Les objectifs de traçabilité, de gestion optimisée et de réduction des coûts établis dans cette section posent les bases pour une exécution réussie du projet.

CHAPITRE 3 :

Mission et travaux réalisés

I. Introduction

Dans ce chapitre, je présente les missions et travaux réalisés au cours de mon stage au sein de COPAG, où j'ai été chargé de développer des modules liés à la gestion des stocks dans le cadre du SGSA. Mes responsabilités comprenaient notamment le développement des fonctionnalités de gestion des modèles d'articles, de dimension de stockage, et de réception des stocks. De plus, j'ai assuré l'implémentation des mouvements de stock et des tests pour garantir la cohérence des données et l'efficacité des processus.

II. Missions et coordination et Gantt

Au cours de mon stage, j'ai été principalement responsable de la gestion des modules liés aux données de base et à la réception des produits dans le **Système de Gestion des Stocks Avancé (SGSA)**. Mes principales missions incluaient le développement des fonctionnalités pour la gestion des modèles de prix, de dimension de stockage et de suivi. J'ai également travaillé sur la gestion des bons de réception et des mouvements de stock, en implémentant des fonctionnalités permettant la consultation, l'édition et la validation des informations liées aux réceptions et mouvements de stock.

• Diagramme de Gantt

Le diagramme de Gantt ci-dessous représente la chronologie des tâches effectuées pendant le projet, réparties en trois catégories principales : apprentissage, frontend et backend.

Apprentissage :

- Apprendre Angular et se connecter aux API (01/07 - 08/07)
- Apprendre NestJS (07/08 - 14/08)

Frontend :

- Paramétrer modèle de prix (09/07 - 11/07)
- Paramétrer modèle de stockage (10/07 - 11/07)
- Paramétrer modèle de suivi (11/07 - 12/07)
- Consulter bon de réception (12/07 - 13/07)
- Modifier bon de réception (13/07 - 17/07)
- Consulter ligne bonne de réception (17/07 - 18/07)
- Consulter lignes bonnes de commande (18/07 - 29/07)
- Consulter infos générales bon de réception (19/07 - 29/07)
- Affecter lot à bon de réception (21/07 - 29/07)
- Créer nouveau lot avec DLC (23/07 - 29/07)
- Spécifier filtres pour image de stock (24/07 - 29/07)
- Calculer image de stock (25/07 - 29/07)
- Consulter mouvement de stock (26/07 - 30/07)

- Modifier mouvement de stock (26/07 - 30/07)
- Saisie des emplacements selon type (29/07 - 30/07)
- Saisir quantité à expédier (29/07 - 30/07)
- Vérifier quantité à expédier (29/07 - 30/07)
- Contrôler disponibilité de stock (31/07 - 05/08)
- Valider lignes de mouvement de stock (31/07 - 05/08)

Backend :

- Mise à niveau de fiche article (15/08 - 29/08)
- Dév. Modèles de prix d'article (16/08 - 19/08)
- Dév. Modèles de dimension de stockage (16/08 - 19/08)
- Dév. Modèles de dimension de suivi (19/08 - 26/08)
- Suppression composante PRICEMO (19/08 - 26/08)
- Ajout statut VALIDER (26/08 - 28/08)
- Gestion statuts DA (27/08 - 28/08)
- Bouton passage statut VALIDER (27/08 - 28/08)
- Clôture des lignes de DA (29/08 - 29/08)

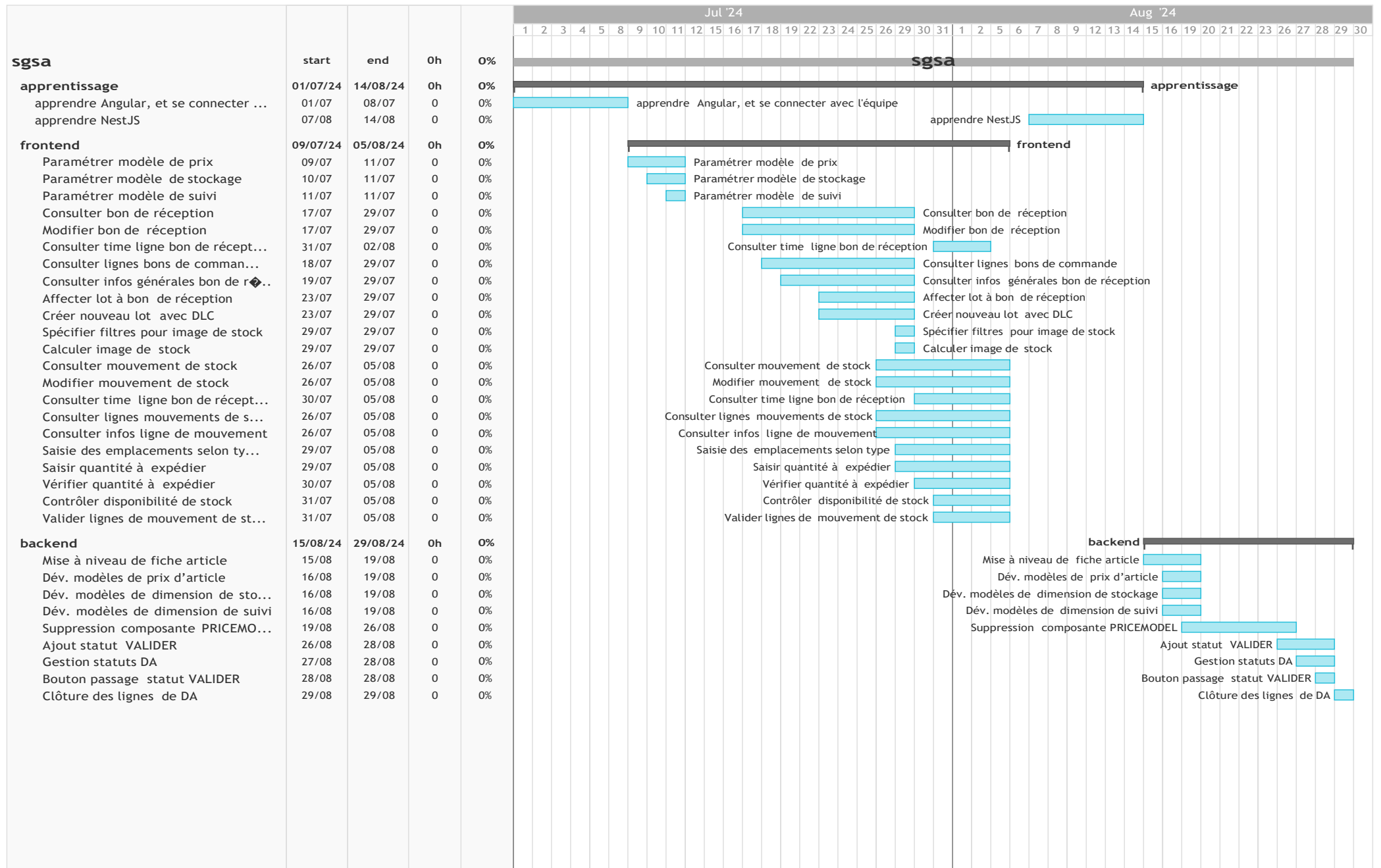


Figure 15 diagramme de Gantt

III. Développement Frontend

4-1- Login page

Login
Sign In to your account

WINSRV

300497

•••••

Login

Forgot password?

Sign up
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Register Now!

Figure 16 page de login

Sur cette page, l'utilisateur doit saisir la référence de la société ainsi que leur nom d'utilisateur et mot de passe. À cette étape, notre encadrant fournit un Token¹ à placer dans le logiciel Mockoon afin de se connecter à l'application depuis la partie frontend.

¹ Token : Il s'agit d'un message envoyé par un serveur à un client et stocké temporairement par ce dernier. Le client inclut une copie du jeton dans les demandes ultérieures envoyées au serveur pour confirmer l'état d'authentification du client.

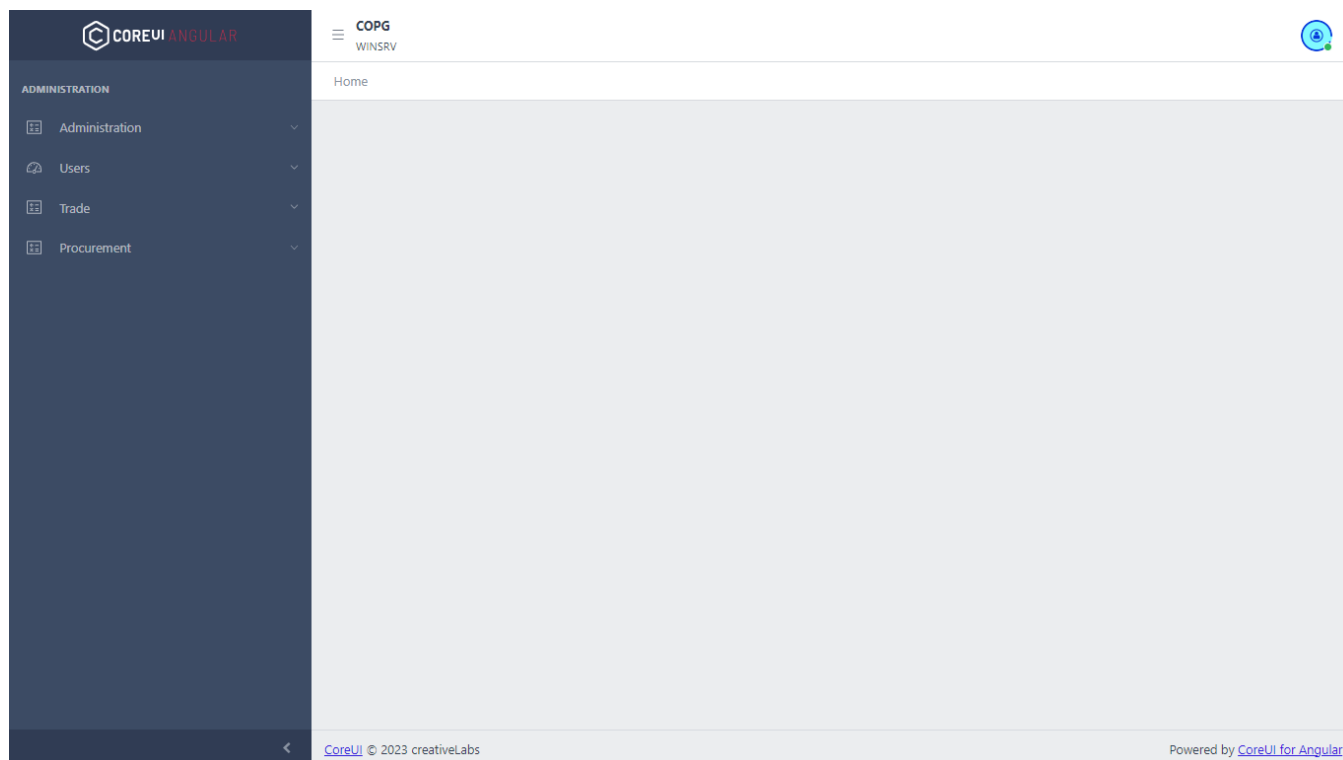


Figure 17 page d'accueil

La page d'accueil comporte les éléments essentiels de l'application, à savoir :

- Barre latérale (Sidebar) : pour organiser les différents liens de navigation d'administration
- En-tête (Header)
- Pied de page (Footer)

4-2- Modèle des items

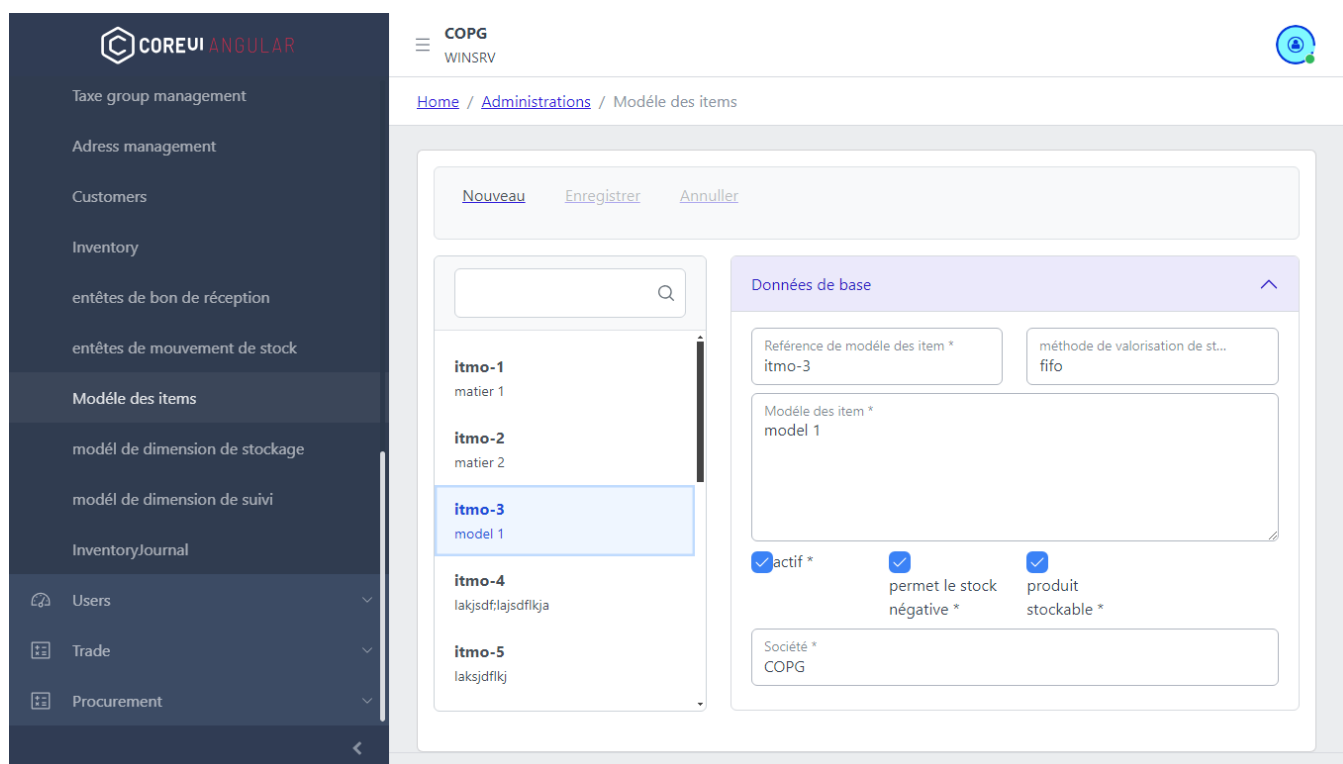
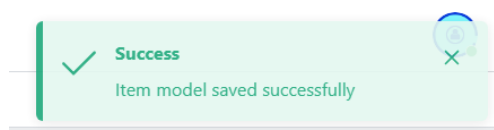


Figure 18 page de model d'item

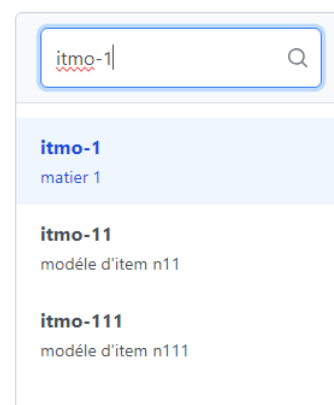
Dans cette interface on peut gérer les modèles des items en haut de l'écran on trouve un Megamenu² qui consiste en trois boutons :

- **Nouveau** : pour entrer dans le mode de création d'un nouveau modèle d'item, dans ce mode on donne le droit de saisir les données de référence de modèle d'item, méthode de valorisation de stock, modèle d'item et de cliquer sur les boutons de choix même pas de changer la valeur de société car il a une relation avec l'utilisateur actuelle
- **Enregistrer** : pour soumettre le formulaire pour être enregistré dans la base de données et afficher ici l'enregistrement et faire avec succès ou pas
- **Annuler** : pour annuler la saisie sortir de mode de création dans ce mode l'utilisateur n'a pas le droit de saisir ou modifier les données dans le formulaire la même chose se produit en cliquant sur un modèle d'article dans la liste



À gauche on a une liste qui affiche les modèles des items qui déjà existent dans la base de données avec la possibilité de rechercher un item grâce à la barre de recherche, la recherche se déroule en temps réel.

Lorsque l'utilisateur clique sur un modèle d'article, le formulaire à droite sera rempli avec les informations sur ce modèle d'article comme indiqué avec l'itmo-3 dans la figure précédente



4-3- Modèle de dimension de stockage

² Megamenu : Un menu de navigation élargi permettant de regrouper plusieurs options et sous-options pour faciliter l'accès aux différentes fonctionnalités d'une application.

The screenshot displays the 'modél de dimension de stockage' (Storage Dimension Model) interface. On the left, a dark sidebar contains a megamenu with categories like 'Taxe management', 'Inventory', and 'Users'. The main area shows a breadcrumb trail: 'Home / Administrations / modél de dimension de stockage'. Below this are buttons for 'Nouveau', 'Enregistrer', and 'Annuler'. A search bar is positioned above a list of storage dimension models, including 'itmo-5', 'item-6', 'itmo-7', 'item-15' (selected), 'itm 43', and 'imt 88'. To the right, the 'Données de base' form contains fields for 'Référence de modèle de dime...', 'modèle de dimension de stock...', 'active *', 'site géographique *', 'Référence d'objet de suivi *', 'warehouse *', 'emplacement *', and 'Société *'. The footer indicates 'CoreUI © 2023 creativeLabs' and 'Powered by CoreUI for Angular'.

Figure 19 Model de dimension de stockage

Dans cette interface en a la même chose que modèle d'article : un Megamenu avec les mêmes boutons qui faire le même rôle et une liste des modèles de dimension de stockage qui a aussi la possibilité de la recherche et sélectionner un modèle pour l'afficher. La différence c'est le formulaire de données de base :

Dans ce modèle en a le référence de modèle, Le référence d'objet de suivi et la société, le modèle, des choix :

- S'il est actif
- Si a un site géographique
- Si a un entrepôt
- Si a un emplacement

4-4- Modèle de dimension de suivi

Figure 20 Interface de sélection et consultation d'un modèle de dimension de suivi

Dans ce cas, l'utilisateur recherche un modèle qui porte le numéro 1 dans sa référence et sélectionne le modèle track-41 pour voir ses informations.

Dans cette interface on a la même chose que modèle d'article et modèle de dimension de stockage : un Megamenu avec les mêmes boutons (nouveau, enregistrer, et annuler) qui font le même rôle et une liste des modèles de dimension de suivi qui a aussi la possibilité de la recherche et sélectionner un modèle pour l'afficher. La différence c'est le formulaire de données de base :

Dans ce modèle on a la référence de modèle, le modèle, des choix :

- S'il est actif
- Si a un lot
- Si a un serial
- Si a un palet
- Si a un propriétaire

Le référence d'objet de suivi et la société

4-5- Bon de réception

4-5-1. Liste des entêtes des bon de réception

Home / Administrations / entêtes de bon de réception

[Nouveau](#)

Ref. Receiving Journal ↑↓	Warehouse ↑↓	Receiving Journal Status ↑↓	Purchase Order ↑↓	Site Geographic ↑↓
Ref. Receiving Journal	Warehouse	Receiving Journal Status	Purchase Order	Site Geographic
EBR-294381	Margarettaworth	Rejeter	PO-289139	Port Laurettaworth
EBR-368800	Edina	annuler	PO-637310	West Georgiana
EBR-564013	North Ophellachester	reçu	PO-837877	Tevinfort
EBR-688201	Lake Odessa	annuler	PO-180856	Bednarside
EBR-662630	Trujillo Alto	Approuver	PO-228274	Dickview
EBR-701493	Greenwood	annuler	PO-163953	Lake Nicoletown
EBR-579017	Sengerbury	annuler	PO-54808	Goldenmouth
EBR-774141	Dorcascester	annuler	PO-303614	Eden Prairie
EBR-347115	Redondo Beach	reçu	PO-48552	Sebastianside
EBR-530988	South Shaniaside	créer	PO-224458	Schultzfort

« < 3 4 5 6 7 > » 10

Figure 21 les entêtes des bons de réception

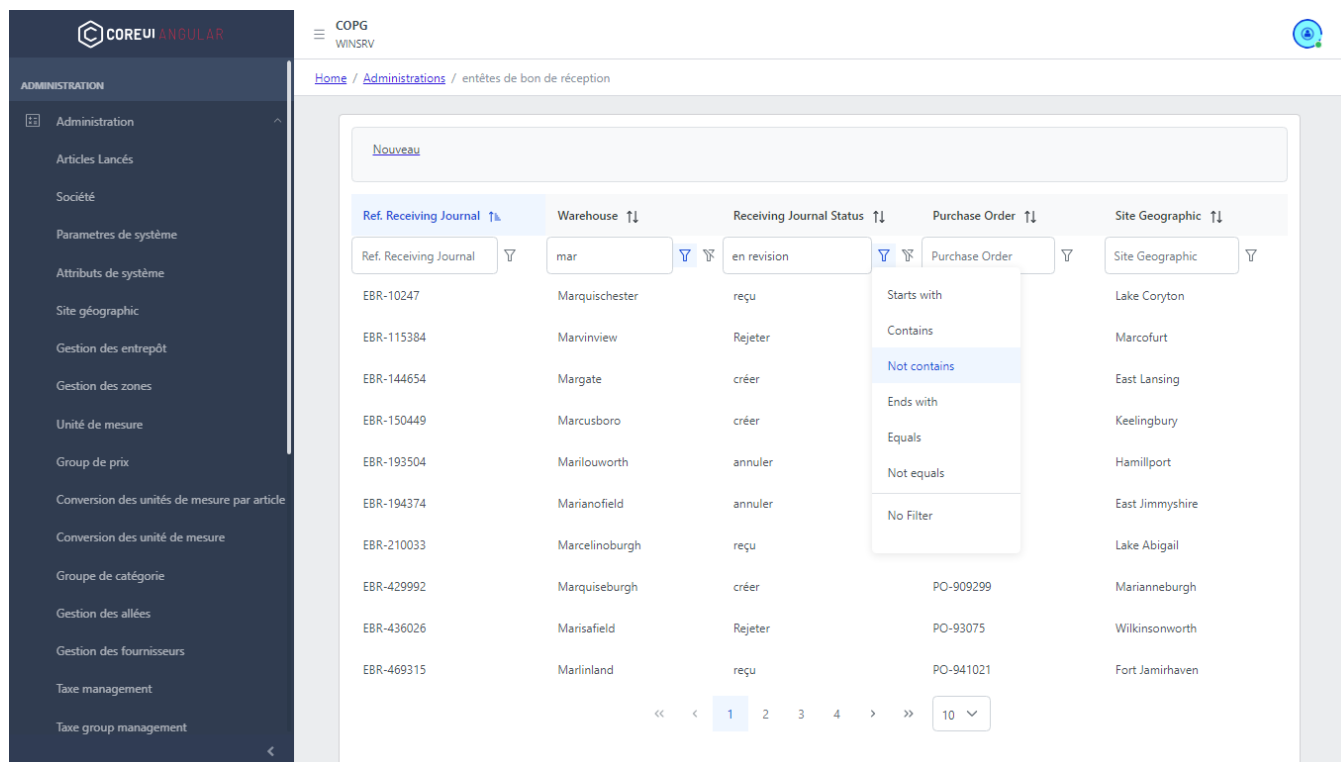
Dans cette interface on a la liste des entêtes des bons de réception.

En haut on a un Megamenu avec un seul bouton qui va rediriger vers une interface pour ajouter un nouvel entête de bon de réception. En bas on a un tableau qui affiche la liste des entêtes avec des informations :

- Référence de l'entête de bon de réception
- L'entrepôt
- Le statut
- Bon de commande
- Le site géographique

Chacune de ces colonnes peut être triée par ordre croissant ou décroissant si l'on clique sur le label d'une colonne.

Nous avons également la possibilité de rechercher dans la liste en utilisant différents types de filtrage pour chaque colonne (Commence par, Contient, Ne contient pas, Se termine par, Égal à, et Pas gal) avec le bar de recherche en haut de chaque colonne.



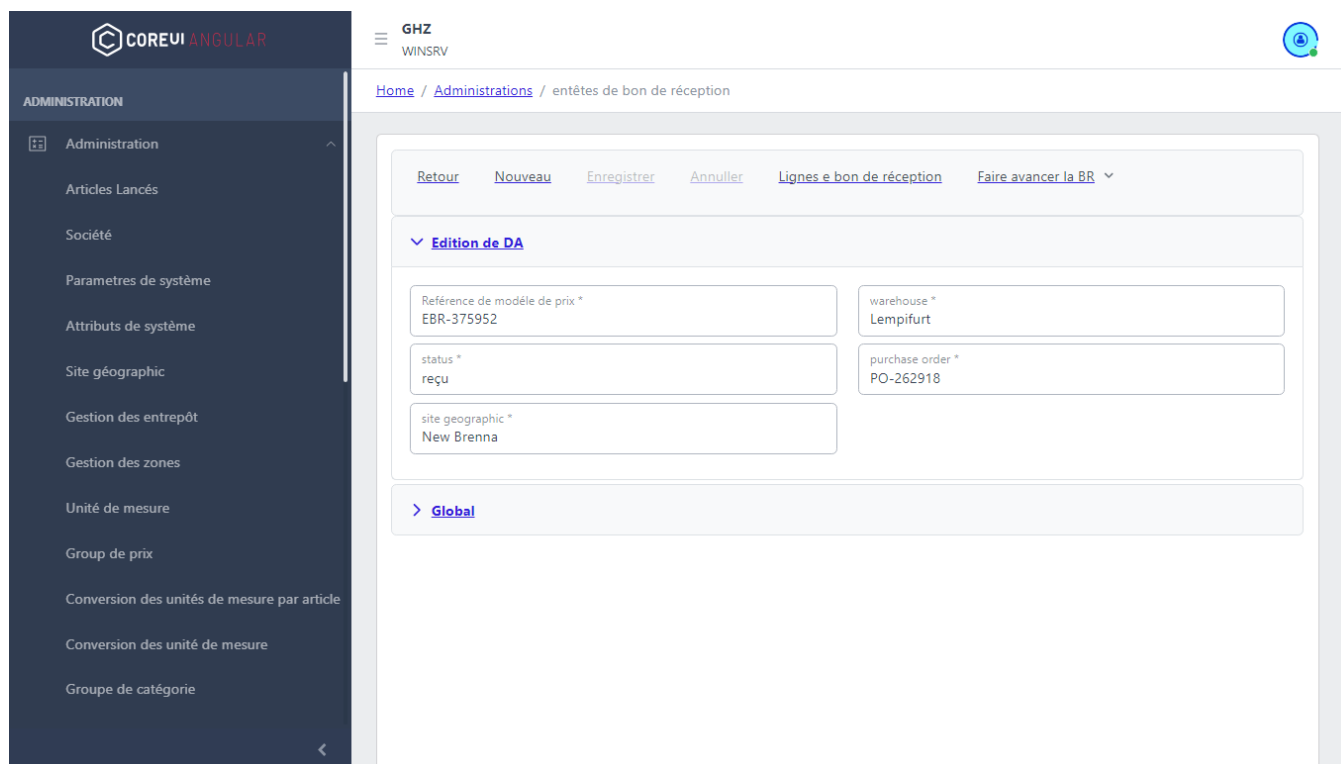
The screenshot shows the 'entêtes de bon de réception' (Receiving Journal Headers) page in the COREUI Angular application. The left sidebar contains the 'ADMINISTRATION' menu. The main content area displays a table with columns: Ref. Receiving Journal, Warehouse, Receiving Journal Status, Purchase Order, and Site Geographic. A filter dropdown menu is open over the 'Receiving Journal Status' column, showing options: Starts with, Contains, Not contains (selected), Ends with, Equals, Not equals, and No Filter. The table data is as follows:

Ref. Receiving Journal	Warehouse	Receiving Journal Status	Purchase Order	Site Geographic
EBR-10247	Marquischester	reçu		Lake Coryton
EBR-115384	Marvinview	Rejeter		Marcofurt
EBR-144654	Margate	créer		East Lansing
EBR-150449	Marcusboro	créer		Keelingbury
EBR-193504	Marilouworth	annuler		Hamillport
EBR-194374	Marianofield	annuler		East Jimmyshire
EBR-210033	Marcelinoburgh	reçu		Lake Abigail
EBR-429992	Marquiseburgh	créer	PO-909299	Marianneburgh
EBR-436026	Marisafeld	Rejeter	PO-93075	Wilkinsonworth
EBR-469315	Marlinland	reçu	PO-941021	Fort Jamirhaven

Figure 22 List des entêtes de bon de réception avec filtrage

Dans cet exemple on a trié la liste avec ordre croissant dans les colonnes de référence et on a fait un filtrage pour afficher les entêtes avec un entrepôt qui commence par MAR et un statut qui n'a pas en révision

4-6- Gère ou crée Nouveau entête



The screenshot shows the 'Edition de DA' (Edit Data) form for a receiving journal header. The form contains the following fields:

- Retour
- Nouveau
- Enregistrer
- Annuler
- Lignes e bon de réception
- Faire avancer la BR

The form is titled 'Edition de DA' and contains the following fields:

- Reférence de modèle de prix *: EBR-375952
- warehouse *: Lempifurt
- status *: reçu
- purchase order *: PO-262918
- site geographic *: New Brenna

Below the form is a 'Global' button.

Figure 23 interface d'un entête de bon de réception

Après que l'utilisateur clique sur le bouton nouveau dans la liste des entêtes des bons de réception, il sera redirigé vers cette interface mais le formulaire sera vide. L'alternative pour obtenir cette interface est de cliquer sur un entête d'un BR³, il sera redirigé vers cette interface avec les informations sur cet entête de BR

Megamenu dans cette interface a les boutons suivants :

- Retour : bouton qui redirige à l'interface précédant (interface des entêtes de BR)
- Nouveau : bouton pour entre dans le mode de création
- Enregistre : pour valider et soumettre le formulaire
- Annuler : pour sortir le mode de création
- Lignes de bon de réception : bouton pour accéder à une interface d'une liste des lignes de ce BR
- Faire avancer le BR : une liste déroulante avec le rôle de changer le statut de ce BR après être traité par le server de backend



4-7- Lignes de bon de réception

receivingjournallineid ↑↓	quantity ↑↓	remainingQty ↑↓	commandQty ↑↓	inventoryDimensionId ↑↓	article ↑
ID886954	4600	411	2701	ID151162	itm67219
ID273147	9033	6979	3795	ID518178	itm55116
ID566570	8872	5163	8969	ID630606	itm20547
ID633578	5750	4913	2517	ID234800	itm46246
ID409078	6797	2278	6723	ID930167	itm84309
ID497364	1141	9910	309	ID768143	itm99966
ID302977	9252	5910	1280	ID109775	itm18328
ID489474	1698	6478	9038	ID117402	itm77038
ID767824	7234	4616	1868	ID373285	itm24578
ID973337	9448	9827	7609	ID838031	itm59929

Figure 24 interface des Lignes de bon de réception

Après cliquer sur le bouton : Lignes de bon de réception, l'utilisateur va être redirigé vers cette interface avec un seule bouton dans la Megamenu qui et Mode d'affichage pour le moment ne fait

³ BR : bon de réception

rient. Après on a un tableau avec une liste des lings de BR le tableau peut également trier et filtrer comme le tableau précédent des entêtes des BR

The screenshot displays the COREUI Angular application interface. On the left is a dark sidebar with a menu containing various management options like 'Gestion des allées', 'Gestion des fournisseurs', 'Taxe management', etc. The main content area shows a table of receiving journal lines. The table has columns for ID, quantity, and item code. Below the table, there's a 'Détails' section with tabs for 'Global', 'Catégorie d'approvisionnement', and 'Axe Analytique'. The 'Global' tab is active, showing a form for a selected line with fields for 'receivingjournalid', 'RefReceivingJournal', 'ID dimension', 'Réf. article', 'Quantité', 'quantity restante', 'Unité de stock', 'Unité de transformation', 'Coefficient', 'Prix unitaire', 'Statut de la journal de réception', 'ID ligne de commande', and 'article'.

Figure 25 affichage de tous les information un linge de BR

Si l'utilisateur sélectionner un linge de BR un formulaire va être aperçu en bas dans les détails dans catégorie globale avec toutes les informations de ce linge car dans le tableau en n'afficher pas que les informations importantes

This screenshot shows the 'article' details form within the application. It features a grid of input fields for various parameters. The top row includes checkboxes for 'Stopped Sales' (checked), 'Stopped Purchases' (unchecked), and 'Stopped Inventory' (checked), along with a 'Sales Price Unit' field. Subsequent rows contain fields for 'Sales Price', 'Safety Stock', 'Removal Time', 'Reference Unit Sales', 'Reference Unit Purchase', 'Reference Unit Order', 'Reference Unit Inventory', 'Reference Tax Sales', 'Reference Tax Purchase', 'Reference Price Model', 'Reference Organisation', 'Reference Item Tracking', 'Reference Item Group', 'Reference Item', 'Reference Company', 'Purchase Price Unit', 'Purchase Price', 'ID Header Parameter', 'Expiration Date', 'Date Time Last Update', 'Date Time Creation', 'Best Before Time', 'Alert Time', and 'Date Time Creation'. A blue button with a calendar icon is visible next to the 'Date Time Creation' field.

Figure 26 affichage les informations d'article d'un linge de BR

4-8- Mouvement de stock

4-8-1. List des mouvements de stock

The screenshot displays the 'List des mouvements de stock' interface. The sidebar on the left contains navigation links such as 'Groupe de catégorie', 'Gestion des allées', 'Gestion des fournisseurs', 'Taxe management', 'Taxe group management', 'Adress management', 'Customers', 'Inventory', 'entêtes de bon de réception', 'entêtes de mouvement de stock', 'Modèle des items', 'modél de dimension de stockage', 'modél de dimension de suivi', 'InventoryJournal', 'Users', 'Trade', and 'Document'. The top header shows the user 'GHZ WINSRV' and a 'Nouveau' button. The main table has the following columns: 'Ref. Transfert', 'Statut du transfert', 'Entrepôt transit', 'Entrepôt de départ', 'Entrepôt d'arrivée', 'Type de transfert', and 'Localisation de'. The table contains 8 rows of data. A pagination bar at the bottom shows page 1 of 10.

Ref. Transfert	Statut du transfert	Entrepôt transit	Entrepôt de départ	Entrepôt d'arrivée	Type de transfert	Localisation de
IT46	Statut du trans	Entrepôt trc	Entrepôt de dé	Entrepôt d'arr	Type de trans	Localisation de c
IT46471	Statu-30	WTF17367	WF14832	WT51888	type-20	LF28937
IT46811	Statu-40	WTF50002	WF21515	WT80087	type-20	LF68572
IT46252	Statu-40	WTF16633	WF41657	WT11612	type-30	LF84398
IT46252	Statu-10	WTF56002	WF78539	WT22737	type-40	LF75826
IT46259	Statu-20	WTF92902	WF80538	WT29720	type-20	LF52033
IT46113	Statu-10	WTF95633	WF96318	WT56746	type-10	LF75396
IT46982	Statu-30	WTF34593	WF98116	WT41770	type-20	LF44590

Figure 27 Tableau des entêtes de mouvements de stock avec options de filtrage

Dans cette interface on a le tableau des entêtes des mouvement de stock comme interface des entêtes des bon s de réception

Dans ce cas on à faire un filtrage par les références qui début par « IT45 » et u triage pour l'entrepôt de départ.

4-9- Gere ou crée nouveau mouvement de stock

Figure 28 Interface de gestion et création d'un mouvement de stock

Pour l'interface pour gère l'entête des mouvements de stock on la Megamenu comme l'interface de bon de réception et dans le formulaire l'utilisateur-il que l'utilisateur sélectionner un statut qui va affecter la saisie des informations, en a quarts statuts :

- Ordre de transfert : dans ce cas l'utilisateur ne doit pas entrer les information (entrepôt de départ, entrepôt d'arrivée, localisation de départ, et localisation de destination)
- Journal de transfert : dans ce cas l'utilisateur doit saisir toutes les informations
- Journal de stock : dans ce cas l'utilisateur doit saisir juste l'entrepôt de départ et localisation de départ
- Journal de comptage : ce cas et comme le cas d'ordre de transfert

Pour avancer le MS⁴ on a 7 statuts

⁴ MS : mouvement de stock

4-10-Lignes des mouvements de stock

quantity	inventorydimensionid	refitem	refunittrans	qtyremaining	date
2975	ID17768	188318	UT64482	1862	
1665	ID48818	160425	UT59398	4363	
315	ID30678	148027	UT56579	1142	
7810	ID31440	117438	UT34376	1908	
9479	ID14606	174173	UT73975	6583	
4975	ID13624	161498	UT51436	6671	
6207	ID10049	142382	UT24375	8049	
7511	ID54356	110827	UT81094	5274	
676	ID50076	140366	UT78594	8491	
8876	ID87712	150450	UT24310	8079	

Figure 29 Interface des lignes de mouvement de stock

Dans cette interface, nous avons les lignes de mouvement de stock. Le Megamenu comporte deux boutons :

- **Mode d'affichage** : actuellement sans fonction.
- **Disponibilité de stock** : pour vérifier la disponibilité de stock après avoir sélectionné les lignes dans le tableau afin de voir s'il y a une disponibilité ou non.

quantity	inventorydimensionid	refitem	refunittrans	qtyremaining	date
7117	ID56680	199126	UT82853	7672	
8286	ID88352	194972	UT15773	1	
482	ID32904	198926	UT35313	8136	
321	ID19888	127772	UT72848	2667	
2466	ID53402	188438	UT62174	2493	
4853	ID59289	125412	UT55187	4820	
5287	ID24290	153126	UT25275	4061	
116	ID53130	145484	UT39496	2584	
9482	ID76455	190501	UT95032	5044	
7297	ID66972	142156	UT21808	9268	

Figure 30 Résultat de vérification de disponibilité pour les lignes de stock sélectionnées

Dans ce cas, nous avons sélectionné des lignes pour vérifier la disponibilité de stock et reçu une réponse indiquant le succès. Sur l'image, l'interface utilisateur affiche un tableau contenant des informations telles que la quantité, l'ID de dimension de l'inventaire, la référence de l'article, l'unité de transport de référence, la quantité restante et la date. Certaines lignes sont mises en surbrillance, montrant qu'elles ont été sélectionnées pour la vérification. En haut à droite, une notification verte indique "inventory disponible", confirmant la disponibilité des articles sélectionnés.

The screenshot displays the COREUI INDOAR application interface. On the left is a dark sidebar with a menu including 'Administration', 'Articles Lancés', 'Société', 'Paramètres de système', 'Attributs de système', 'Site géographique', 'Gestion des entrepôt', 'Gestion des zones', 'Unité de mesure', 'Group de prix', 'Conversion des unités de mesure par article', 'Conversion des unité de mesure', 'Groupe de catégorie', 'Gestion des allées', 'Gestion des fournisseurs', 'Taxe management', 'Taxe group management', 'Adress management', and 'Customers'. The main content area shows two forms: 'Ligne de mouvement de stock' and 'article'.

Ligne de mouvement de stock

inventorytransferlineid ITL49282	reference de mouvement de stock IT66394	inventorydimensionid ID29191	refitem 188293
quantity 564	qtyremaining 8720	reference de unité inventair UI99125	refunitrans UT69842
coefficient 56	unit price 84178	refinventorytransferstatus ITS73090	

article

Stopped Sales <input checked="" type="checkbox"/>	Stopped Purchases <input type="checkbox"/>	Stopped Inventory <input checked="" type="checkbox"/>	Sales Price Unit 19.99
Sales Price 199.99	Safety Stock 50	Removal Time 30	Reference Unit Sales unit123
Reference Unit Purchase unit1456	Reference Unit Order unit789	Reference Unit Inventory unit012	Reference Tax Sales tax123
Reference Tax Purchase tax456	Reference Price Model modelA	Reference Organisation org789	Reference Item Tracking tracking456
Reference Item Group group123	Reference Item Item456	Reference Company company789	Purchase Price Unit 15.75
Purchase Price 157.5	ID Header Parameter 1001	Expiration Date 20240815	Date Time Last Update 2024-07-21
Date Time Creation 2024-07-20	Best Before Time 7	Alert Time 3	

Figure 31 Détails d'une ligne de mouvement de stock sélectionnée

Après la sélection d'une ligne dans le tableau, deux formulaires apparaissent avec des informations relatives à la ligne choisie.

- **Ligne de mouvement de stock** : inclut des détails tels que l'ID du transfert, la quantité, le coefficient, et plus encore.
- **Article** : affiche des paramètres comme le prix de vente, le stock de sécurité, les dates d'expiration et de création, ainsi que d'autres informations pertinentes pour l'article.

4-11-Inventory dimension

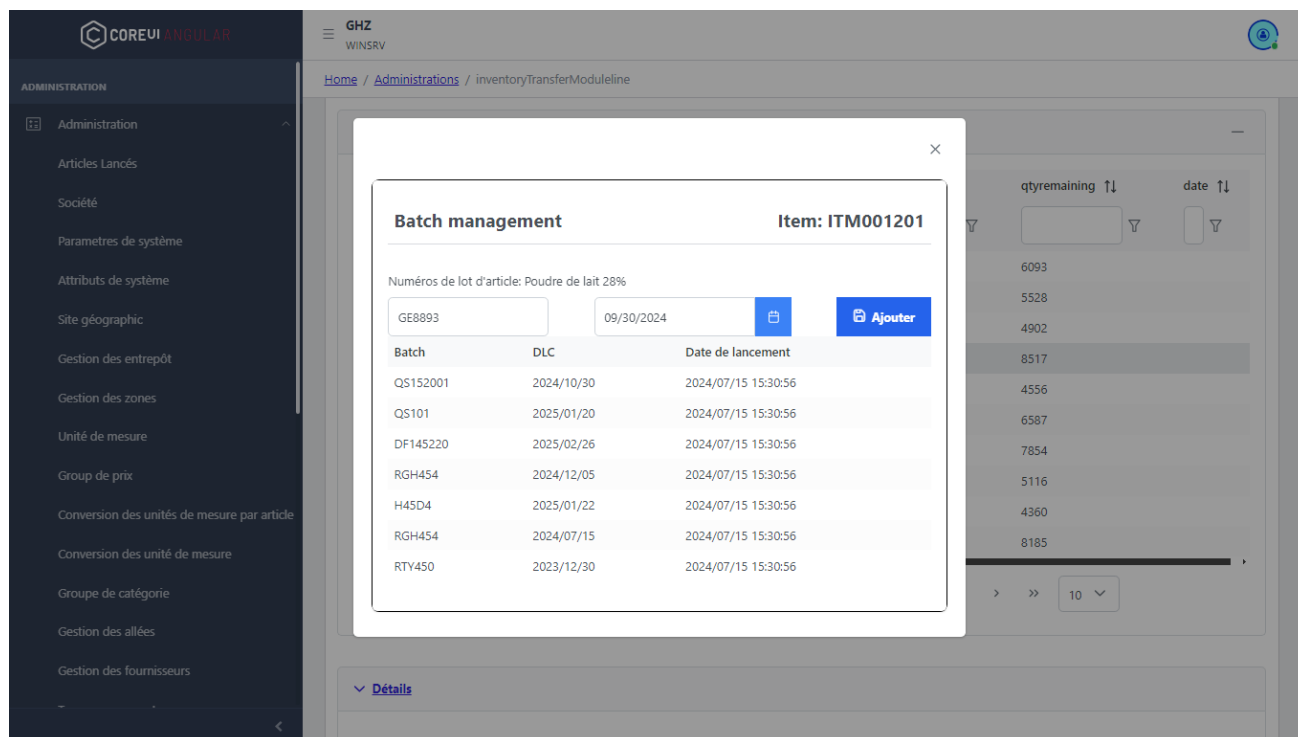



Figure 32 Interface de création ou modification d'un lot d'inventaire

Pour modifier ou créer un nouveau lot pour une ligne, j'ai conçu un nouveau composant accessible en cliquant sur ce bouton . Ce bouton se trouve dans la colonne "ID de dimension d'inventaire" du tableau des lignes.

Dans ce composant, je peux sélectionner une dimension de suivi existante ou en créer une nouvelle, et attribuer une date limite de consommation (DLC⁵).

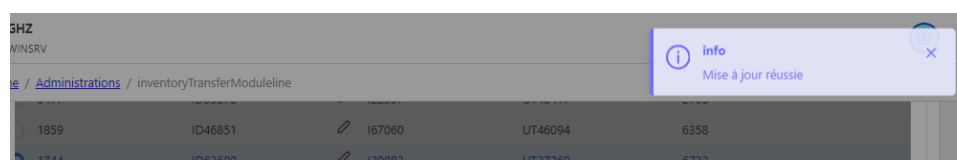


Figure 33 Message de confirmation de mise à jour d'un lot d'inventaire.

Après la sélection ou la création d'un lot, un message de confirmation apparaît. Sur l'image, on peut voir que l'interface affiche un message en haut à droite indiquant "Mise à jour réussie," confirmant que l'opération a été réalisée avec succès.

IV. Backend

Pour moderniser la gestion des articles, j'ai supprimé l'ancien modèle de prix et ajouté une API⁶ pour les modèles d'articles, de dimensions de stockage et de dimensions de suivi. Tout d'abord, j'ai

⁵ DLC : Date Limite de Consommation

⁶ API (Application Programming Interface) : Ensemble de fonctions permettant la communication entre différentes applications, facilitant l'interopérabilité

ajouté deux DTO⁷ à chacun pour afficher et créer des données. Ensuite, j'ai ajouté trois API au contrôleur d'articles pour la création de nouveaux articles, la récupération d'informations et la modification du statut.

4-1- Modèle d'article

Pour gérer un modèle d'article, la méthode du contrôleur appelle une méthode qui effectue le traitement dans le service.

4-1-1. Créer un modèle d'article (saveitemmodel)

```
async createItemModel(itemmodelDto: ItemModelCreateDto) {
  // Check if the item model already exists
  const count = await this.itemmodelRepository.countBy({
    refcompany: itemmodelDto.refcompany,
    reforganisation: itemmodelDto.reforganisation,
    refitemmodel: itemmodelDto.refitemmodel,
  });
  if (count > 0) {
    throw new BadRequestException({
      error: `Le modèle d'article avec refitemmodel : ${itemmodelDto.refitemmodel}
existe déjà.`
    });
  }
  // If it doesn't exist, create and save the new item model
  const itemmodel = this.itemmodelRepository.create(itemmodelDto);
  return await this.itemmodelRepository.save(itemmodel)
    .then((res) => {
      return res;
    })
    .catch((err) => {
      throw new BadRequestException(err.message, { cause: err, description:
err.query });
    });
}
```

Figure 34 création d'un modèle d'article dans la base de données.

Dans cette méthode, je vérifie si un modèle d'article avec la même société, organisation et référence existe déjà. Si oui, je retourne une erreur avec un message. Sinon, le modèle d'article est créé et le résultat de la création est retourné.

4-2- Avoir les informations d'un modèle d'article (getitemmodel)

⁷ DTO (Data Transfer Object) : Modèle utilisé pour transférer des données entre différentes parties d'une application sans exposer les détails d'implémentation

```

async getItemmodels(itemmodelDto: ItemModelShowDto){
  const query = this.itemmodelRepository
    .createQueryBuilder('itemmodel')
    .innerJoinAndSelect(
      'itemmodel.inventoryvaluationmethode',
      'inventoryvaluationmethode'
    )
    .where(
      'itemmodel.refcompany = :refcompany and itemmodel.reforganisation = :reforganisation',
      {refcompany: itemmodelDto.refcompany, reforganisation: itemmodelDto.reforganisation}
    )
    if(itemmodelDto.refitemmodel){
      query.andWhere('itemmodel.refitemmodel = :refitemmodel',
        {refitemmodel: itemmodelDto.refitemmodel}
      )
    }
    // Execute the query and return the results
    return await query.getMany()
    .then((res) => {
      return res;
    })
    .catch((err) => {
      throw new BadRequestException(err.message, { cause: err, description: err.query });
    });
}

```

Figure 35 récupération des informations d'un modèle d'article.

Dans cette méthode, je réalise une recherche sur les modèles d'article en fonction de la société et de l'organisation. Si une valeur facultative est présente dans le DTO, elle est également prise en compte dans la recherche. Les résultats sont ensuite transmis au frontend.

4-3- Désactiver le modèle d'article (changestateofitemmodel)

```

async changeStateOfItemmodel(itemmodelDto: ItemModelCreateDto) {
  await this.itemmodelRepository
    .createQueryBuilder()
    .update(ItemModelEntity)
    .set({ actif: itemmodelDto.actif })
    .where(
      'refitemmodel = :refitemmodel and refcompany = :refcompany and reforganisation = :reforganisation',
      { refitemmodel: itemmodelDto.refitemmodel, refcompany: itemmodelDto.refcompany, reforganisation: itemmodelDto.reforganisation }
    )
    .execute()
    .then(async (res) => {
      return res;
    })
    .catch((err) => {
      throw new BadRequestException(err.message, { cause: err, description: err.query });
    });
}

```

Figure 36 mise à jour du statut d'un modèle d'article

Dans cette application, il ne faut pas supprimer un modèle d'article. On peut juste le désactiver. J'ai donc créé cette méthode qui change le statut d'un modèle d'article donné.

4-4- Model de dimension de stockage

4-4-1. Créer un modèle de dimension de stockage

```
async savestoragedimensionmodel(trackingdimensionmodelDto:
itemstoragedimensionmodelcreateDto){
    // Check if the item model already exists
    const count = await this.itemstoragedimensionRepository.countBy({
        refcompany: trackingdimensionmodelDto.refcompany,
        reforganisation: trackingdimensionmodelDto.reforganisation,
        refitemstoragedimensionmodel:
trackingdimensionmodelDto.refitemstoragedimensionmodel,
    });

    if (count > 0) {
        throw new BadRequestException({
            error: `Le modèle de dimension de stockage avec refitemstoragedimensionmodel
: ${trackingdimensionmodelDto.refitemstoragedimensionmodel} existe déjà.`
        });
    }
    // If it doesn't exist, create and save the new item model
    const itemmodel =
this.itemstoragedimensionRepository.create(trackingdimensionmodelDto);
    return await this.itemstoragedimensionRepository.save(itemmodel)
        .then((res) => {
            return res;
        })
        .catch((err) => {
            throw new BadRequestException(err.message, { cause: err, description: err.query
        });
    });
}
```

Figure 37 création d'un modèle de dimension de stockage

Cette méthode permet de vérifier si un modèle de dimension de stockage existe déjà pour une société et une organisation donnée. Si le modèle existe, une exception est levée avec un message d'erreur. Si le modèle n'existe pas, il est créé et sauvegardé dans le dépôt.

4-5- Obtenir les informations d'un modèle de dimension de stockage

```

async getItemStorageDimension(storageDimensionDto: itemStorageDimensionModelShowDto){
  const query = this.itemStorageDimensionRepository
    .createQueryBuilder('itemStorageDimensionModel')
    .innerJoinAndSelect(
      'itemStorageDimensionModel.inventorytrackingobject',
      'inventorytrackingobject'
    )
    .where(
      'itemStorageDimensionModel.refcompany = :refcompany and
      itemStorageDimensionModel.reforganisation = :reforganisation',
      {refcompany: storageDimensionDto.refcompany, reforganisation:
      storageDimensionDto.reforganisation}
    )
    if(storageDimensionDto.refitemStorageDimensionModel){
      query.andWhere('itemStorageDimensionModel.refitemStorageDimensionModel =
      :refitemStorageDimensionModel',
      {refitemStorageDimensionModel:
      storageDimensionDto.refitemStorageDimensionModel}
    )
    }
    // Execute the query and return the results
    return await query.getMany()
    .then((res) => {
      return res;
    })
    .catch((err) => {
      throw new BadRequestException(err.message, { cause: err, description: err.query });
    });
}

```

Figure 38 affichage des informations d'un modèle de dimension de stockage

Cette méthode effectue une recherche sur les modèles de dimension de stockage en fonction de la société et de l'organisation. Si une référence de modèle de dimension de stockage est fournie, elle est également utilisée dans la recherche. Les résultats sont ensuite renvoyés au frontend.

4-6- Changer le statut d'un modèle de dimension de stockage

```

async changeItemStorageDimensionStatus(storageDimensionModel:
itemStorageDimensionModelCreateDto) {
  await this.itemStorageDimensionRepository
    .createQueryBuilder()
    .update(ItemStorageDimensionModelEntity)
    .set({ actif: storageDimensionModel.actif })
    .where(
      `refitemStorageDimensionModel = :refitemStorageDimensionModel
      and refcompany = :refcompany
      and reforganisation = :reforganisation`,
      { refitemStorageDimensionModel:
      storageDimensionModel.refitemStorageDimensionModel,
      refcompany: storageDimensionModel.refcompany,
      reforganisation: storageDimensionModel.reforganisation }
    )
    .execute()
    .then(async (res) => {
      return res;
    })
    .catch((err) => {
      throw new BadRequestException(err.message, { cause: err, description:
      err.query,});
    });
}

```

Figure 39 mise à jour du statut d'un modèle de dimension de stockage

Pour désactiver un modèle de dimension de stockage, cette méthode met à jour le statut du modèle dans le dépôt. Le statut est changé en fonction des critères de la société, de l'organisation et de la référence du modèle. Une fois la mise à jour effectuée, le résultat est renvoyé.

4-7- Model de dimension de suivi

4-7-1. Créer un modèle de dimension de suivi

```

async savestoragedimensionmodel(trackingdimensionmodelDto:
itemstoragedimensionmodelcreateDto){
    // Check if the item model already exists
    const count = await this.itemstoragedimensionRepository.countBy({
        refcompany: trackingdimensionmodelDto.refcompany,
        reforganisation: trackingdimensionmodelDto.reforganisation,
        refitemstoragedimensionmodel:
trackingdimensionmodelDto.refitemstoragedimensionmodel,
    });

    if (count > 0) {
        throw new BadRequestException({
            error: `Le modèle de dimension de stockage avec refitemstoragedimensionmodel
: ${trackingdimensionmodelDto.refitemstoragedimensionmodel} existe déjà.`
        });
    }
    // If it doesn't exist, create and save the new item model
    const itemmodel =
this.itemstoragedimensionRepository.create(trackingdimensionmodelDto);
    return await this.itemstoragedimensionRepository.save(itemmodel)
        .then((res) => {
            return res;
        })
        .catch((err) => {
            throw new BadRequestException(err.message, { cause: err, description: err.query
        });
    });
}

```

Figure 40 création d'un modèle de dimension de suivi

Cette méthode vérifie si un modèle de dimension de suivi existe déjà pour une société et une organisation donnée. Si le modèle existe, une exception est levée avec un message d'erreur. Si le modèle n'existe pas, il est créé et sauvegardé dans le dépôt.

4-8- Obtenir les informations d'un modèle de dimension de suivi


```

async getItemStorageDimension(storageDimensionDto: itemStorageDimensionModelShowDto){
  const query = this.itemStorageDimensionRepository
    .createQueryBuilder('itemStorageDimensionModel')
    .innerJoinAndSelect(
      'itemStorageDimensionModel.inventorytrackingobject',
      'inventorytrackingobject'
    )
    .where(
      'itemStorageDimensionModel.refcompany = :refcompany and
      itemStorageDimensionModel.reforganisation = :reforganisation',
      {refcompany: storageDimensionDto.refcompany, reforganisation:
      storageDimensionDto.reforganisation}
    )
    if(storageDimensionDto.refitemStorageDimensionModel){
      query.andWhere('itemStorageDimensionModel.refitemStorageDimensionModel =
      :refitemStorageDimensionModel',
      {refitemStorageDimensionModel:
      storageDimensionDto.refitemStorageDimensionModel}
    )
  }
  // Execute the query and return the results
  return await query.getMany()
  .then((res) => {
    return res;
  })
  .catch((err) => {
    throw new BadRequestException(err.message, { cause: err, description: err.query });
  });
}

```

Figure 41 Affichage des informations d'un modèle de dimension de suivi

Cette méthode effectue une recherche sur les modèles de dimension de suivi en fonction de la société et de l'organisation. Si une référence de modèle de dimension de suivi est fournie, elle est également utilisée dans la recherche. Les résultats sont ensuite renvoyés au frontend.

4-9- Changer le statut d'un modèle de dimension de suivi

```

async changeItemStorageDimensionStatus(storageDimensionModel:
itemStorageDimensionModelCreateDto) {
  await this.itemStorageDimensionRepository
    .createQueryBuilder()
    .update(ItemStorageDimensionModelEntity)
    .set({ actif: storageDimensionModel.actif })
    .where(
      'refitemStorageDimensionModel = :refitemStorageDimensionModel
      and refcompany = :refcompany
      and reforganisation = :reforganisation',
      { refitemStorageDimensionModel:
      storageDimensionModel.refitemStorageDimensionModel,
      refcompany: storageDimensionModel.refcompany,
      reforganisation: storageDimensionModel.reforganisation }
    )
    .execute()
    .then(async (res) => {
      return res;
    })
    .catch((err) => {
      throw new BadRequestException(err.message, { cause: err, description:
      err.query, });
    });
}

```

Figure 42 mise à jour du statut d'un modèle de dimension de suivi

Pour désactiver un modèle de dimension de suivi, cette méthode met à jour le statut du modèle dans le dépôt. Le statut est changé en fonction des critères de la société, de l'organisation et de la référence du modèle. Une fois la mise à jour effectuée, le résultat est renvoyé.

4-10-Demande d'achat

4-10-1. Ajout du statut "VALIDER" entre "BROUILLON" et "EN COURS DE RÉVISION"

a. Contexte :

Dans la gestion des workflows de validation, il est souvent nécessaire de définir plusieurs étapes de contrôle pour assurer la qualité et la cohérence des données avant qu'un élément ne soit officiellement accepté ou modifié. Le processus initial comportait les statuts « BROUILLON » et « EN COURS DE RÉVISION », mais il manquait une étape intermédiaire pour valider les données avant révision.

b. Implémentation technique :

- **Mise à jour de l'Enum des statuts** : dans le code backend, j'ai ajouté une nouvelle valeur dans l'Enum ou la table des statuts.

```
export enum Purchaserequisitionstatuts {
  REVS = 'REVS',
  VALID = 'VALID',
  APRV = 'APRV',
  CLTR = 'CLTR',
  BRLN = 'BRLN',
  RJCT = 'RJCT',
}
```

Figure 43 Mise à jour de l'Enum des statuts dans le backend

- **Déplacement des vérifications métier** : Les vérifications métier, qui étaient auparavant exécutées entre les statuts « BROUILLON » et « EN COURS DE RÉVISION », ont été déplacées entre « BROUILLON » et « VALIDER ». Cela garantit que seules les données validées passeront à l'étape suivante.

```
else if (purchaserequisitionChangeStatutDto.refpurchaserequisitionstatuts
=== Purchaserequisitionstatuts.VALID.toString()){
  // ----->>> Valider la DA.
  if
  (![Purchaserequisitionstatuts.BRLN.toString()].includes(purchreq.refpurchaserequisitionstatuts))
  {
    errorMessage = 'le statut '+purchreq.refpurchaserequisitionstatuts+'ne permet pas
de Valider la DA!';
  } else {
    await this.verifyPurchReqLinesToValider({
      refpurchaserequisition:
      purchaserequisitionChangeStatutDto.refpurchaserequisition,
      refcompany: purchaserequisitionChangeStatutDto.refcompany,
      refoorganisation: purchaserequisitionChangeStatutDto.refoorganisation,
      refvendor: undefined,
      id: undefined})
    purchreq.refpurchaserequisitionstatuts =
    purchaserequisitionChangeStatutDto.refpurchaserequisitionstatuts;
    purchreq.datesubmission = new Date();
    purchreq.submittedby = purchaserequisitionChangeStatutDto.matricule;
  }
}
```

Figure 44 Validation de la Demande d'Achat (DA)

Cette méthode vérifie et met à jour le statut de la demande d'achat en validant certaines conditions spécifiques, telles que le statut actuel et les informations nécessaires, avant de permettre la soumission de la demande.

```

async verifyPurchReqLinesToValider(purchaserequisitionlinesFindDto:
PurchaserequisitionLinesFindDto){
    return await this.purchreqlinesRepository.findBy({
        refcompany: purchaserequisitionlinesFindDto.refcompany,
        refororganisation: purchaserequisitionlinesFindDto.reforganisation,
        repurchaserequisition: purchaserequisitionlinesFindDto.repurchaserequisition
    })
    .then(async (linesPurchReq) => {
        const alreadytraiteddata : {refitem: string}[] = [];
        let message = '';
        for(let i = 0; i< linesPurchReq.length; i++) {
            if ([undefined, null, ''].includes(linesPurchReq[i].refitem)) {
                message = 'Il faut spécifier un item pour la ligne achat ' +
                linesPurchReq[i].id + ' !'
                throw new BadRequestException(message, {cause: message, description:
                message,});
            }

            if (linesPurchReq[i].quantity <= 0) {
                message = 'Quantité invalide pour la ligne achat ' +
                linesPurchReq[i].id + ' !'
                throw new BadRequestException(message, {cause: message, description:
                message,});
            }

            if (alreadytraiteddata.find(prline => prline.refitem ===
            linesPurchReq[i].refitem ) == undefined) {
                alreadytraiteddata.push({refitem: linesPurchReq[i].refitem})
                await this.itemService.isItemValid({
                    refcompany: linesPurchReq[i].refcompany,
                    refitem: linesPurchReq[i].refitem,
                    refororganisation: linesPurchReq[i].reforganisation,
                })
            }
        }
    })
    .catch((err) => {
        throw new BadRequestException(err.message, { cause: err, description:
        err.query,});
    });
}

```

Figure 45 Validation des Lignes de Demande d'Achat

Cette méthode s'assure que chaque ligne de la demande d'achat contient des items valides et des quantités appropriées. Elle effectue des vérifications détaillées avant d'approuver la demande.

- **Logique de changement de statut** : Lors du passage de « VALIDER » à « EN COURS DE REVISION », il n'y a pas de vérification métier. J'ai donc implémenté une logique conditionnelle pour vérifier si un statut est modifié et exécuter ou non des validations en fonction de l'état.

```

async purchReqStatutsManagementToREVS(purchaserequisitionChangeStatutDto:
PurchaserequisitionChangeStatutDto){
    const purchreq = await this.purchreqRepository.findOneBy({
        refpurchaserequisition: purchaserequisitionChangeStatutDto.refpurchaserequisition,
        refcompany: purchaserequisitionChangeStatutDto.refcompany,
        refororganisation: purchaserequisitionChangeStatutDto.reforganisation,
    });

    let errormessage = '';
    if (purchaserequisitionChangeStatutDto.refpurchaserequisitionstatuts
=== Purchaserequisitionstatuts.REVS.toString()) {
        // ----->>> lancer la révision pour la DA.
        if
        (![Purchaserequisitionstatuts.VALID.toString()].includes(purchreq.refpurchaserequisitionstatuts
)) {
            errormessage = 'le statut '+purchreq.refpurchaserequisitionstatuts+'ne permet
pas de lancer la révision!';
        } else {
            purchreq.refpurchaserequisitionstatuts =
purchaserequisitionChangeStatutDto.refpurchaserequisitionstatuts;
            purchreq.datesubmission = new Date();
            purchreq.submittedby = purchaserequisitionChangeStatutDto.matricule;
        }
    }else{
        errormessage = 'le statut
'+purchaserequisitionChangeStatutDto.refpurchaserequisitionstatuts+'ne permet pas de lancer la
révision!';
    }

    if(!['', null, undefined].includes(errormessage)) {
        throw new BadRequestException(errormessage, { cause: errormessage, description:
errormessage,});
    }

    return await this.purchreqRepository
        .save(purchreq)
        .then(async (res) => {
            return res;
        })
        .catch((err) => {
            throw new BadRequestException(err.message, { cause: err, description: err.query,});
        });
}

```

Figure 46 Gestion du Statut de la Demande d'Achat (DA) vers REVS

Cette méthode gère le changement de statut d'une demande d'achat vers le statut "révision" (REVS). Elle vérifie d'abord le statut actuel de la demande d'achat pour s'assurer qu'il permet la révision. Si les conditions sont remplies, la demande d'achat est mise à jour avec le nouveau statut, la date de soumission et l'identifiant de l'utilisateur ayant soumis la demande. En cas de conditions non remplies, une exception est levée avec un message d'erreur.

4-11- 4. Clôture des DA avec gestion des lignes

a. Contexte :

Une DA ne peut être clôturée que si toutes ses lignes sont soit clôturées, soit rejetées. Cela garantit que le processus est complet avant la finalisation.

b. Implémentation technique :

Vérification des lignes : Avant de permettre la clôture d'une DA, j'ai implémenté une vérification qui s'assure que toutes les lignes sont soit en statut « CLÔTURÉ » soit en statut « REJETÉ ».

```

async verifyPurchReqLinesToApprove(purchaserequisitionlinesFindDto:
PurchaserequisitionLinesFindDto) {
    await this.purchreqLinesRepository
        .createQueryBuilder('purchaserequisitionlines')
        .where(
            'refpurchaserequisition = :refpurchaserequisition and refcompany = :refcompany
and refororganisation = :reforganisation',
            { refcompany: purchaserequisitionlinesFindDto.refcompany, refpurchaserequisition
:purchaserequisitionlinesFindDto.refpurchaserequisition, refororganisation:
purchaserequisitionlinesFindDto.reforganisation })
        .andWhere('(coalesce(quantity, 0) <= 0 or coalesce(price, 0) <= 0 or refvendeur is
null or reftaxegroup is null or refcurrency is null or reftaxe is null or refitem is null)')
        .getCount()
        .then(async (res) => {
            if (res > 0) {
                const message = 'Merci de saisir les quantités, les prix et les fournisseurs
de toutes les lignes de DA'
                throw new BadRequestException(message, { cause: message, description:
message, });
            }
        })
        .catch((err) => {
            throw new BadRequestException(err.message, { cause: err, description:
err.query, });
        });
}

```

Figure 47 Vérification des Lignes de Demande d'Achat (DA) pour Approbation

Cette méthode vérifie les lignes de la demande d'achat pour s'assurer que toutes les quantités, prix et fournisseurs sont renseignés avant l'approbation. Elle filtre les lignes qui ne respectent pas ces critères et lève une exception avec un message d'erreur le cas échéant.

V. Tests API

4-1- Tests pour le Frontend

Afin de faciliter le développement et le test du frontend de notre application sans dépendre d'un backend en cours de développement, nous avons opté pour l'utilisation de Mockoon, un outil de simulation de serveurs REST API. Mockoon nous a permis de créer des environnements de moque avec des endpoints qui renvoient des données factices (fake data), ce qui s'est avéré essentiel pour valider le bon fonctionnement des appels API dans le frontend sans nécessiter un backend fonctionnel.

Processus de création de l'environnement Mockoon

1. **Création des endpoints** : Dans Mockoon, nous avons configuré plusieurs routes simulant les différentes API du backend, telles que l'authentification des utilisateurs, la gestion des stocks, etc. Chaque route renvoyait une réponse JSON pré-configurée correspondant à des scénarios typiques (succès, erreurs).
2. **Configuration des données factices** : Les données retournées par Mockoon ont été générées de manière à représenter des cas réels, par exemple des informations sur des produits, les

niveaux de stock, ou encore les détails des utilisateurs. Ces données ont été spécifiquement formatées pour imiter les résultats renvoyés par notre API réelle.

3. **Tests avec le frontend Angular** : Une fois les API simulées configurées, nous avons intégré Mockoon au processus de développement du frontend. Grâce à l'intégration des endpoints moqué, nous avons pu tester l'intégralité des composants du frontend en simulant les requêtes API. Cela a permis de vérifier la gestion des données reçues, le traitement des erreurs, et l'affichage des informations sans attendre que le backend soit complètement implémenté.

L'utilisation de Mockoon s'est avérée très utile pour :

- **Accélérer le développement du frontend** : en simulant des API fonctionnelles, il a été possible de développer des fonctionnalités frontales de manière indépendante du backend.
- **Tests unitaires et de bout en bout** : les données moquées nous ont permis de réaliser des tests unitaires et des tests end-to-end sur les composants du frontend.

Exemple de configuration de Mockoon

Voici un exemple de configuration d'un endpoint de Mockoon pour simuler l'obtention d'une liste de produits :

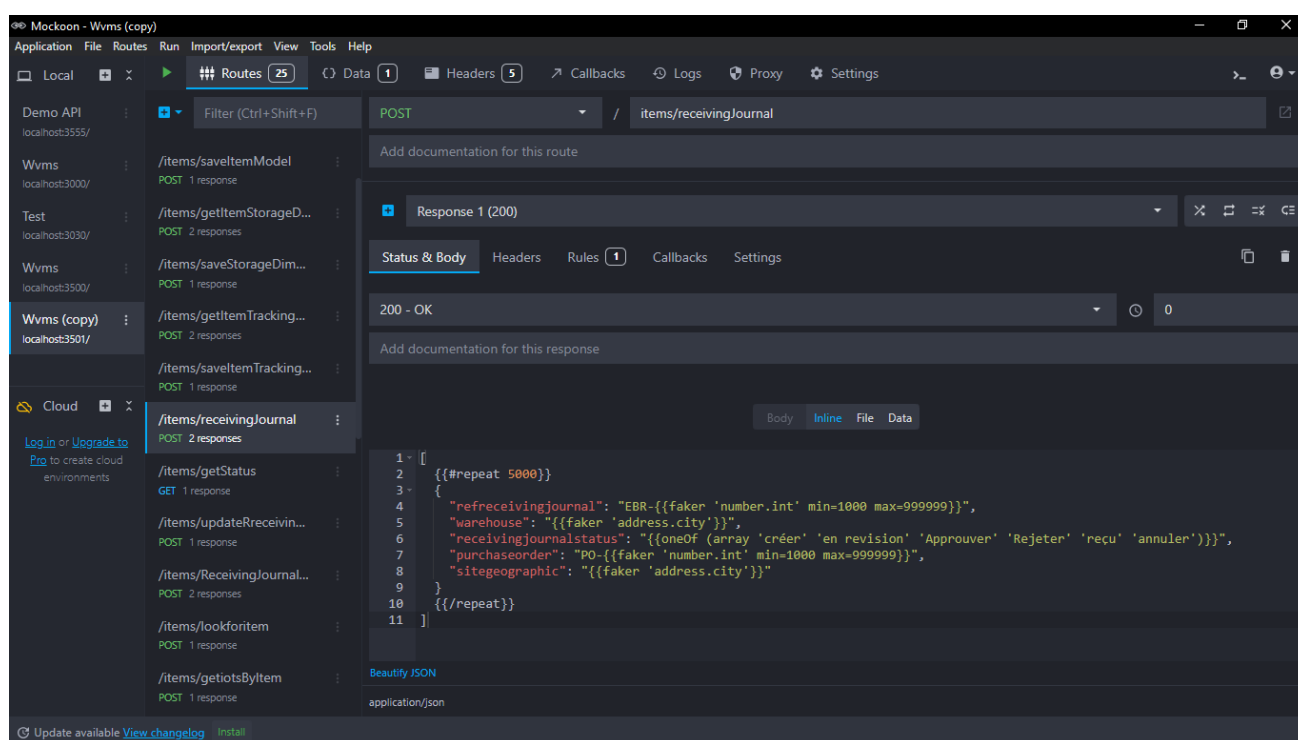


Figure 48 Utilisation de Mockoon pour les tests frontend

Cette capture d'écran montre l'interface de **Mockoon**, un outil utilisé pour simuler des serveurs REST API localement. Le route affichée est **/items/receivingJournal** avec une méthode **POST**.

- **Route /items/receivingJournal** :

- **Statut de réponse :** La réponse HTTP est configurée pour renvoyer un statut 200 OK, ce qui signifie que la requête POST est réussie.
- **Corps de la réponse :** Le corps de la réponse est formaté en JSON et utilise des modèles pour générer des données factices dynamiques.
- La structure utilise la syntaxe de Mockoon avec la fonctionnalité `{{repeat 5000}}`, qui permet de générer 5000 entrées dans la réponse.
- **Champs générés dynamiquement :**

Refreceivingjournal : Ce champ représente un identifiant unique pour chaque bon de réception. Il est généré dynamiquement à l'aide de la fonction « **faker** », avec une plage de valeurs numériques allant de 1000 à 999999.

Warehouse : Ce champ représente l'entrepôt lié au bon de réception. Il utilise la fonction « **faker.address.city** » pour générer un nom de ville aléatoire.

Receivingjournalstatus : Ce champ représente le statut du journal de réception. Il utilise la fonction « **oneOf** » pour choisir aléatoirement un statut parmi : créer, en révision, Approuver, Rejeter, reçu, ou annuler.

Purchaseorder : Il s'agit du numéro de commande d'achat, généré dynamiquement avec la même plage de valeurs que `refreceivingjournal`.

Sitegeographic : Il représente la localisation géographique du site, également générée à l'aide de la fonction « **faker.address.city** ».

Simulation de données massives :

La route retourne 5000 enregistrements, simulant ainsi une large liste de bons de réception. Cette fonctionnalité permet de tester la capacité du frontend à gérer des volumes importants de données.

4-2- Tests des API Backend

Afin de vérifier le bon fonctionnement des API développées avec NestJS, j'ai utilisé **Postman** pour tester et valider les différentes requêtes avant de les passer au frontend. Cette étape a permis de simuler des appels API et de s'assurer que le backend fonctionnait correctement. Voici les principales étapes que j'ai suivies pour réaliser ces tests :

Configuration des requêtes API : J'ai défini plusieurs requêtes dans Postman pour interagir avec les Endpoint de l'API NestJS. Chaque requête était configurée avec la méthode HTTP appropriée (POST, GET, etc.) et les paramètres requis, par exemple :

- Pour rechercher un article spécifique, j'ai utilisé une requête POST avec les paramètres refcompany, reorganisation, et refitem afin d'obtenir les détails de l'article.
- Pour ajouter un nouveau modèle d'article, j'ai utilisé une autre requête POST en spécifiant des informations telles que refitemmodel, itemmodel, et d'autres attributs dans le corps de la requête.
-

Simulation des requêtes et validation des réponses : Après avoir configuré les requêtes, j'ai exécuté ces dernières dans Postman et analysé les réponses renvoyées par l'API. Cela m'a permis de valider que les données étaient correctement traitées et que les réponses respectaient les spécifications.

Gestion des erreurs et ajustements : En cas d'erreurs (par exemple, des réponses avec un code d'état HTTP 400 ou 500), j'ai pu rapidement identifier les problèmes dans la logique de traitement du backend et ajuster le code NestJS en conséquence.

Passage au Frontend : Une fois que les API ont été testées et validées, j'ai fourni les détails des Endpoint et les résultats attendus à l'équipe frontend. Cela leur a permis d'intégrer facilement les API dans leur logique de développement, en ayant déjà la certitude que le backend fonctionnait correctement.

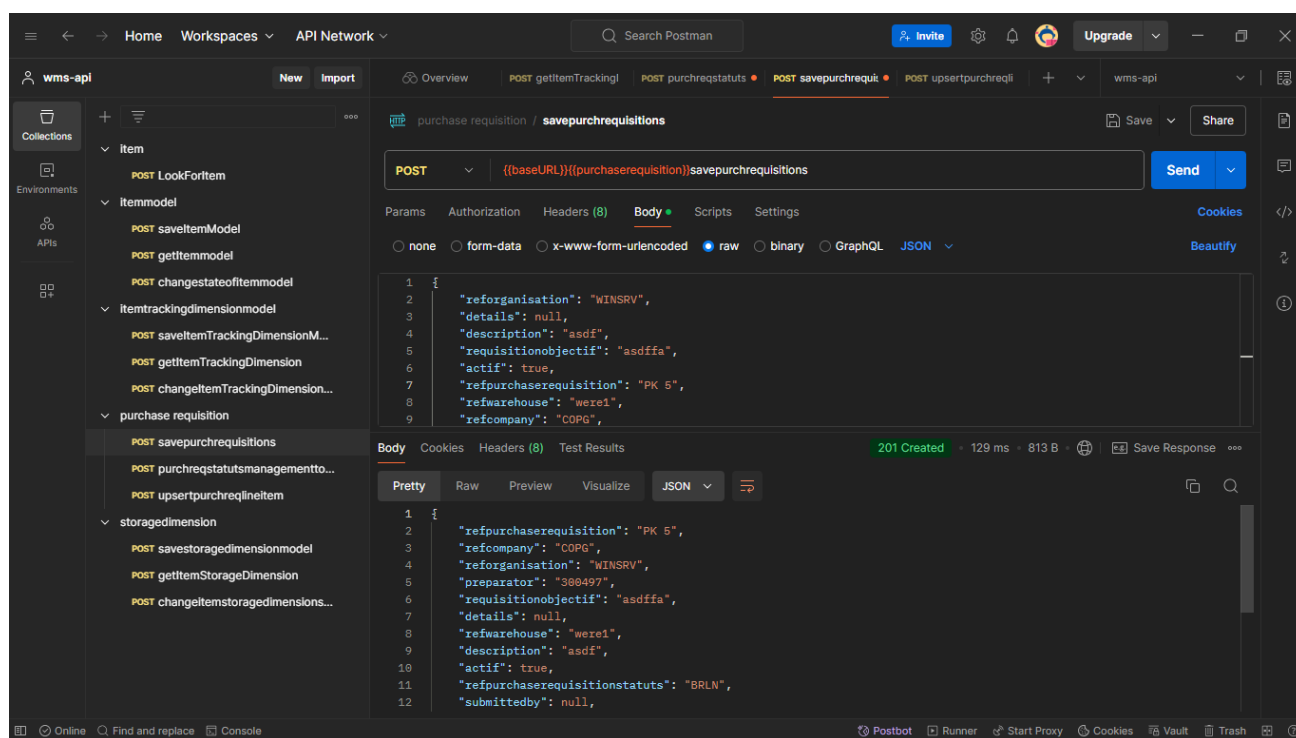


Figure 49 Résultat de tests de l'API backend

Cette image montre un exemple de test d'API backend effectué avec Postman. On voit une collection nommée "**wms-api**" avec divers endpoints organisés sous des catégories telles que "item", "itemmodel", "itemtrackingdimensionmodel", "purchase requisition" et "storagedimension". L'Endpoint sélectionné est "savepurchrequisitions" sous la catégorie "purchase requisition". La méthode de requête est POST, et l'URL est construite en utilisant des variables {{baseUrl}} et {{purchaserequisition}}.

L'onglet "Body" est sélectionné, montrant une charge utile JSON montre une réponse réussie avec un statut de « 201 Created ».

Cette image montre le processus de test des API backend en utilisant Postman, incluant l'envoi d'une requête avec un payload JSON et la réception d'une réponse JSON.

VI. Conclusion

Les missions réalisées ont permis de renforcer la gestion des stocks de COPAG, répondant aux objectifs de traçabilité et de précision fixés en début de stage. La réalisation de ces tâches a été une expérience enrichissante, tant sur le plan technique qu'organisationnel, et a contribué de manière significative à l'avancement du SGSA. Les compétences développées et les solutions apportées démontrent une capacité à répondre aux besoins concrets d'une entreprise dans un secteur dynamique.

CONCLUSION GENERALE

Mon stage s'est déroulé au sein de la coopérative agricole COPAG, spécialisée dans la production agroalimentaire, plus particulièrement dans les produits laitiers. Ce stage, d'une durée de deux mois, m'a permis d'acquérir une expérience pratique en développement informatique, en particulier dans la gestion des stocks. J'ai principalement travaillé au sein du département informatique, où j'ai pu contribuer au projet de développement d'un Système de Gestion des Stocks Avancé (SGSA).

Durant ce stage, j'ai participé à diverses missions, dont la principale était le développement de l'interface frontend du SGSA. J'ai également travaillé sur la conception de l'architecture backend et l'intégration des bases de données, en utilisant des technologies modernes telles qu'Angular et NestJS. Parmi les réalisations notables, on compte la mise en place de modules permettant d'améliorer la traçabilité et la gestion des stocks, ainsi que des outils pour optimiser les opérations logistiques de COPAG.

Sur le plan technique, cette expérience m'a permis de renforcer mes compétences en développement web et en gestion de bases de données. J'ai appris à maîtriser des outils essentiels, tels que TypeORM et PostgreSQL, et à m'adapter aux contraintes d'un environnement industriel. D'un point de vue personnel, j'ai également développé des compétences relationnelles en travaillant au sein d'une équipe multidisciplinaire, ce qui m'a permis d'améliorer ma communication et ma capacité à collaborer efficacement.

Je tiens à remercier l'équipe de COPAG, notamment mon encadrant, M. Tarik MAJID, pour son soutien et ses précieux conseils. Mes remerciements vont également à mes professeurs de l'École Polytechnique d'Agadir pour leur encadrement académique. Enfin, je remercie mes collègues stagiaires et tous ceux qui ont contribué à rendre cette expérience enrichissante.

BIBLIOGRAPHIE

- [Ang22] : Angular Developers, *Angular Documentation*, Google Developers, 2022. [Disponible en ligne : <https://angular.io/docs>].
- [COP87] : COPAG, *Coopérative agricole COPAG*, 1987. [Disponible en ligne : <https://www.copag.ma/>].
- [CUI22] : CoreUI, *CoreUI for Angular Documentation*, 2022. [Disponible en ligne : <https://coreui.io/angular/docs/>].
- [FAK22] : Faker, *Faker.js Documentation*, 2022. [Disponible en ligne : <https://fakerjs.dev/>].
- [MDN22] : Mozilla Developers, *MDN Web Docs*, Mozilla, 2022. [Disponible en ligne : <https://developer.mozilla.org/>].
- [MKN22] : Mockoon, *Mockoon Documentation*, 2022. [Disponible en ligne : <https://mockoon.com/>].
- [NJS22] : NestJS Framework, *NestJS Documentation*, 2022. [Disponible en ligne : <https://docs.nestjs.com/>].
- [Orm22] : TypeORM, *TypeORM Documentation*, 2022. [Disponible en ligne : <https://typeorm.io/>].
- [PNG22] : PrimeNG, *PrimeNG Documentation*, 2022. [Disponible en ligne : <https://primeng.org/>].
- [PST22] : Postman, *Postman Documentation*, 2022. [Disponible en ligne : <https://learning.postman.com/>].
- [SOF22] : Stack Overflow, *Stack Overflow Documentation*, Stack Overflow, 2022. [Disponible en ligne : <https://stackoverflow.com/>].
- [TS22] : Microsoft, *TypeScript Documentation*, 2022. [Disponible en ligne : <https://www.typescriptlang.org/docs/>].

ANNEXES

Annexe 1 : Fiche technique de la COPAG

Raison sociale	Coopérative agricole COPAG Taroudant
Date de création	07 mai 1987
Président	Moulay Mohamed LOULTITY
Forme juridique	Coopérative agricole
Nombre des adhérents	14 000
Effectif	Plus de 9 500 des employés directs
Secteurs d'activité	Agriculture, industrie agroalimentaire et conditionnement
Capacité de production	Pour les agrumes : 8 000 tonnes par an, et pour les primeurs : 10 000 tonnes par an
Chiffre d'affaires	7 MMDH
Siège social	BP 1001 FREIJA - 83200 - TAROUDANT MAROC
Tél	(05) 28 53 61 71 / 82 / 11
Fax	(05) 28 53 61 39
Email	mmloultiti@copag.ma

Tableau 5 Fiche technique de la COPAG

Annexe 2 : Fiche technique de la COPAG

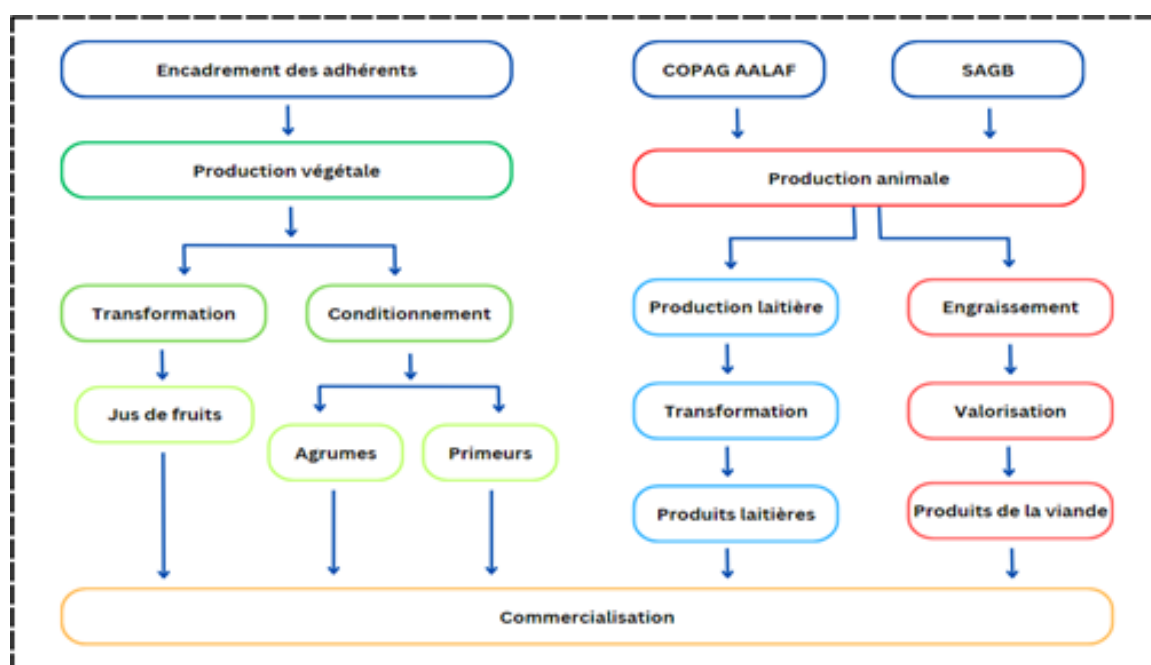


Figure 50 L'organisation de la COPAG

Annexe 3 : Fiche technique de la COPAG

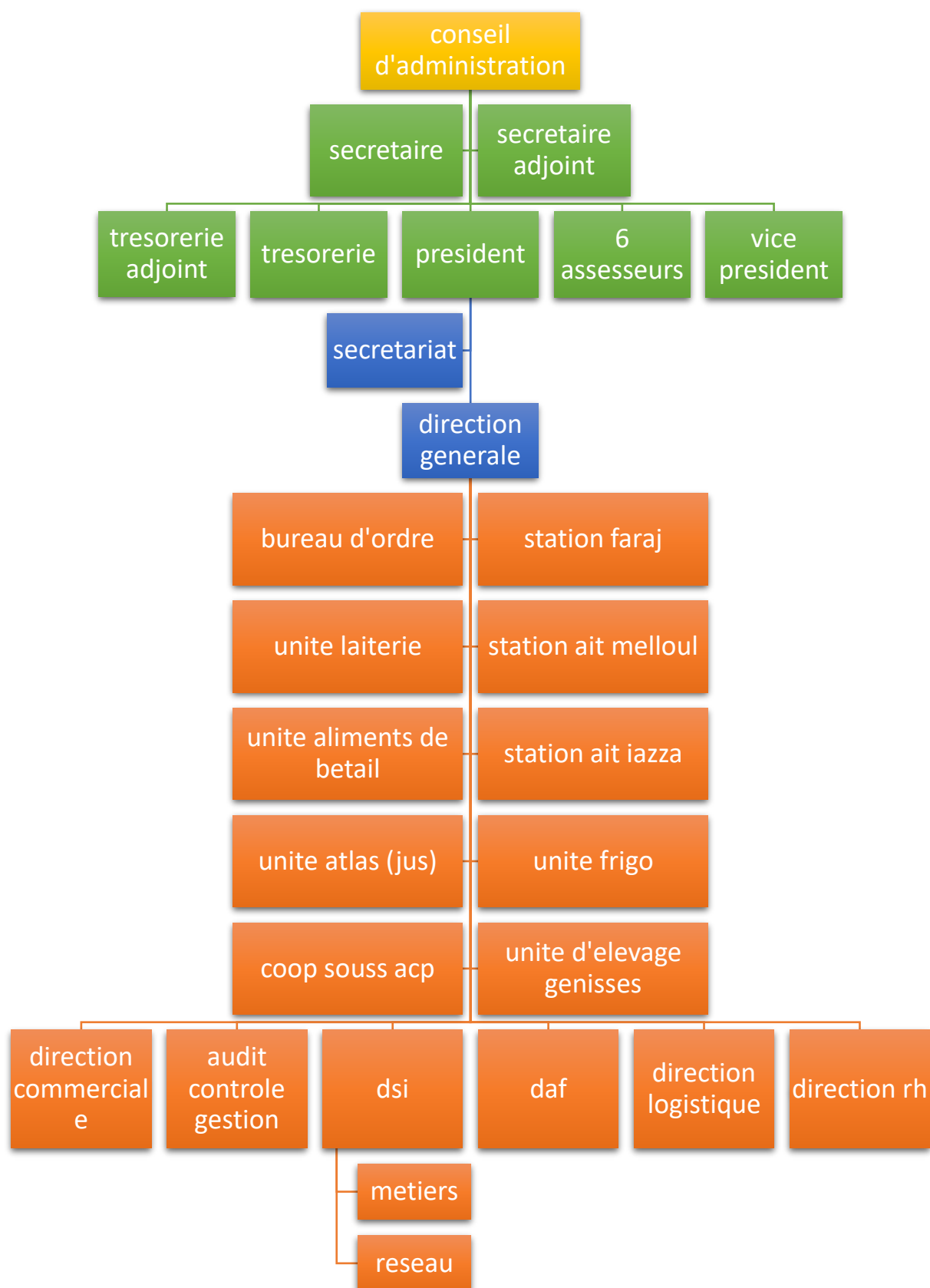


Figure 51 L'organigramme de la COPAG

Annexe 4 : Utilisation de Balsamiq pour le prototypage

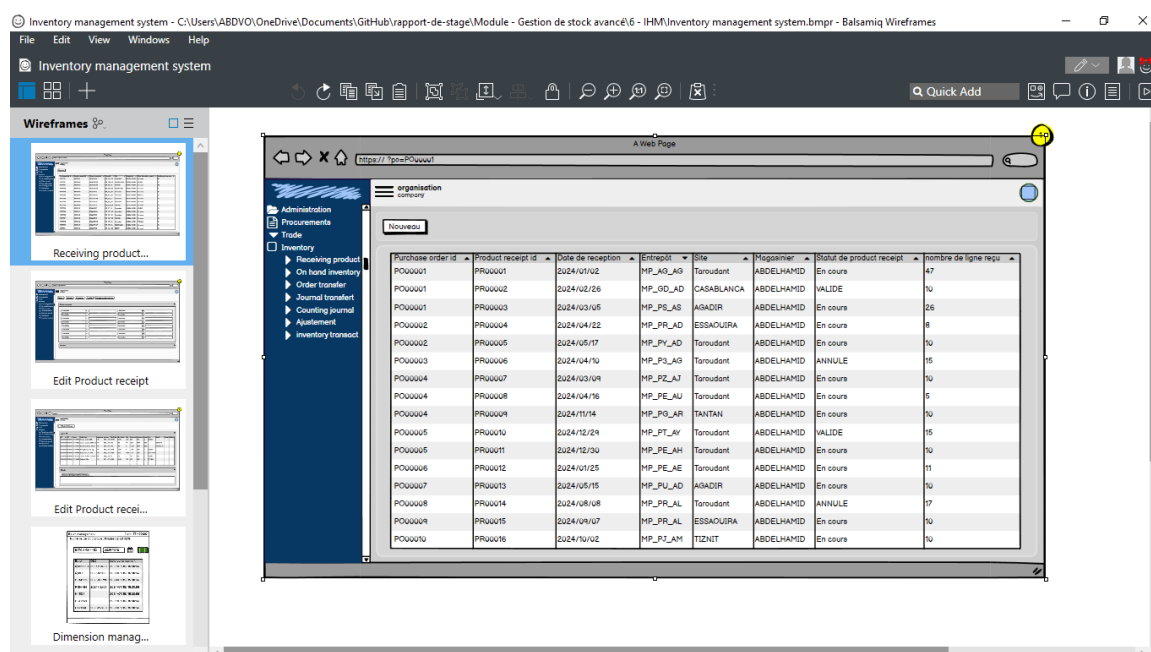


Figure 52 Maquette de la page des listes de bon de réception

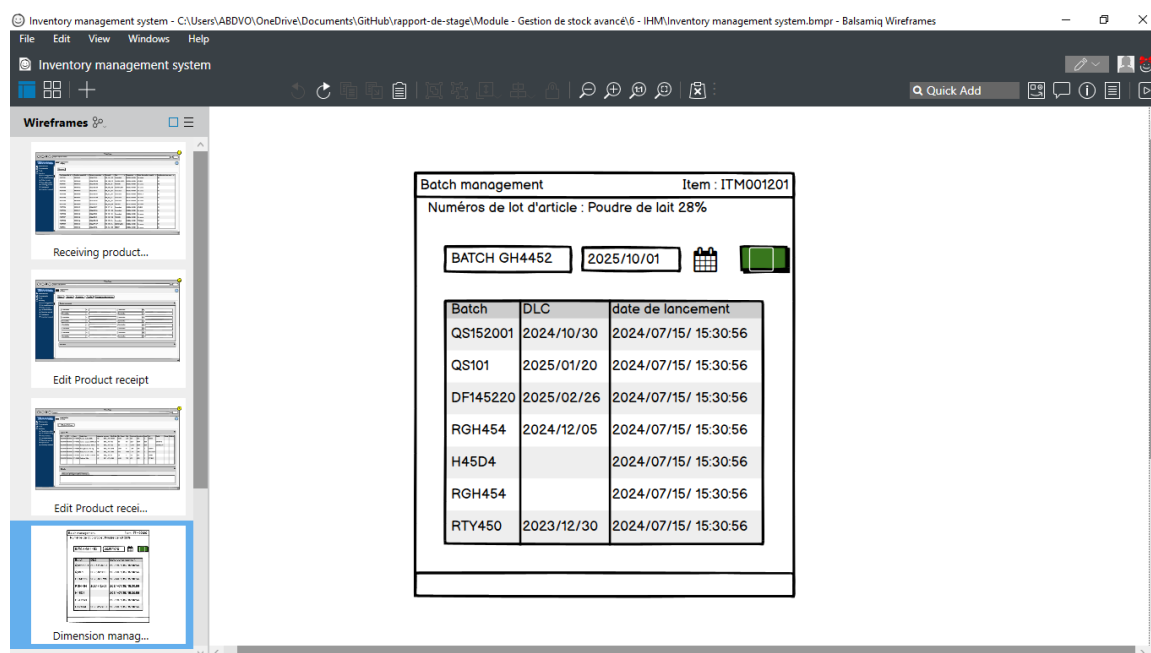


Figure 53 Prototype de la gestion de dimension de stock

Annexe 5 : Utilisation de DBeaver pour la gestion des bases de données

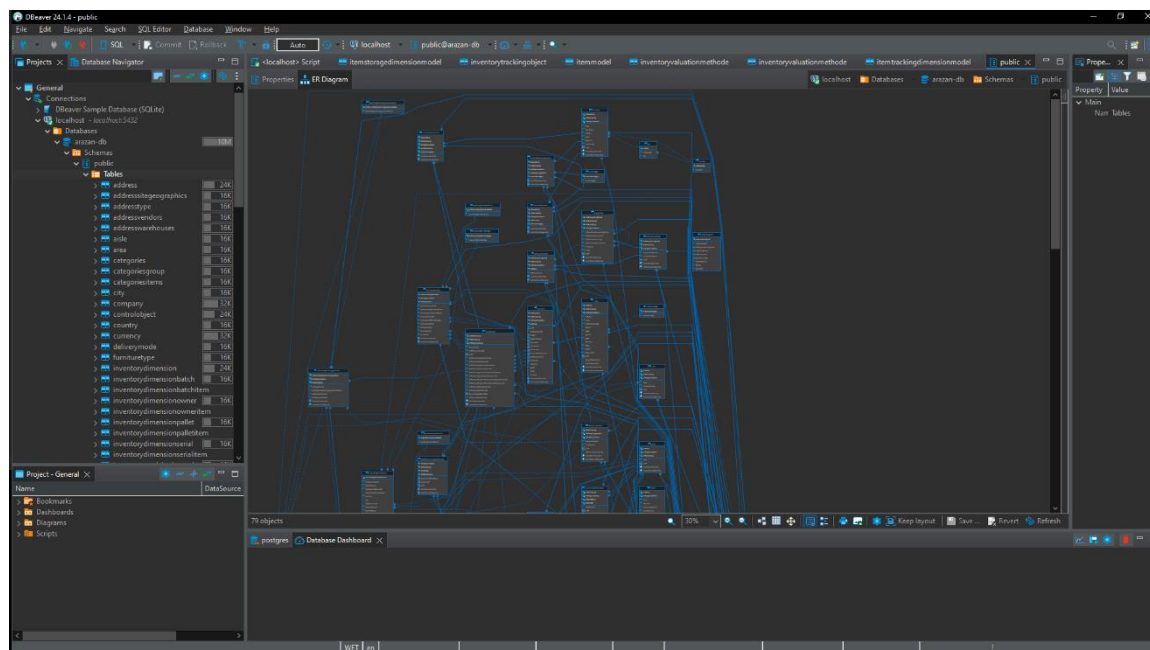


Figure 54 Vue de la structure de la base de données SGSA.

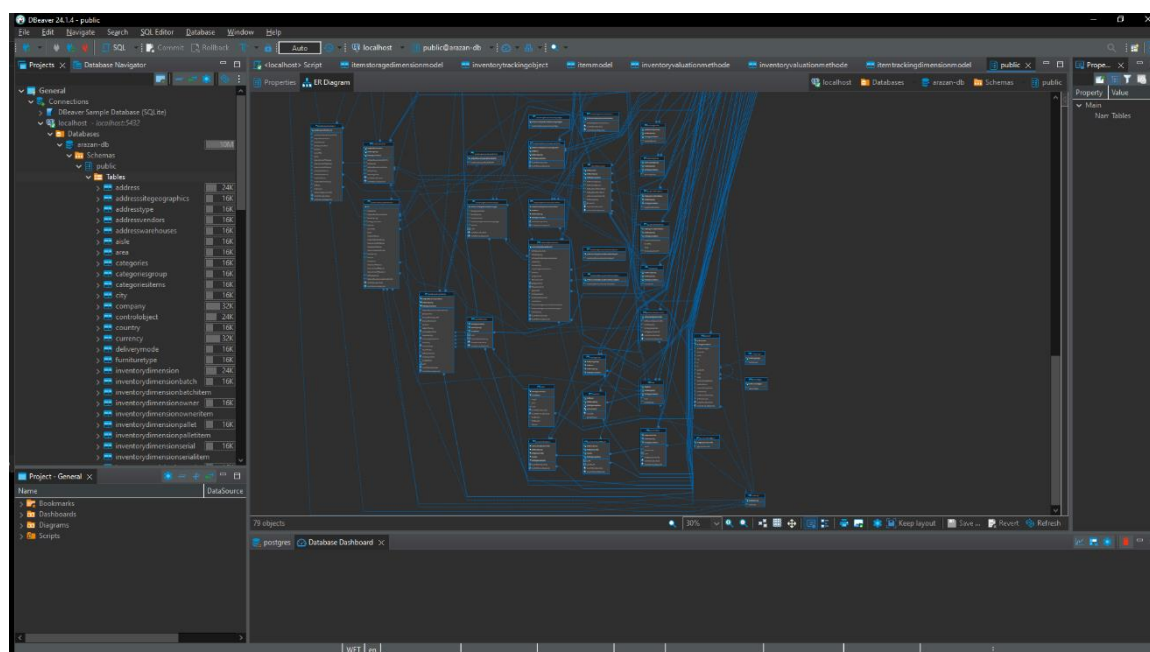


Figure 55 Vue de la structure de la base de données SGSA.

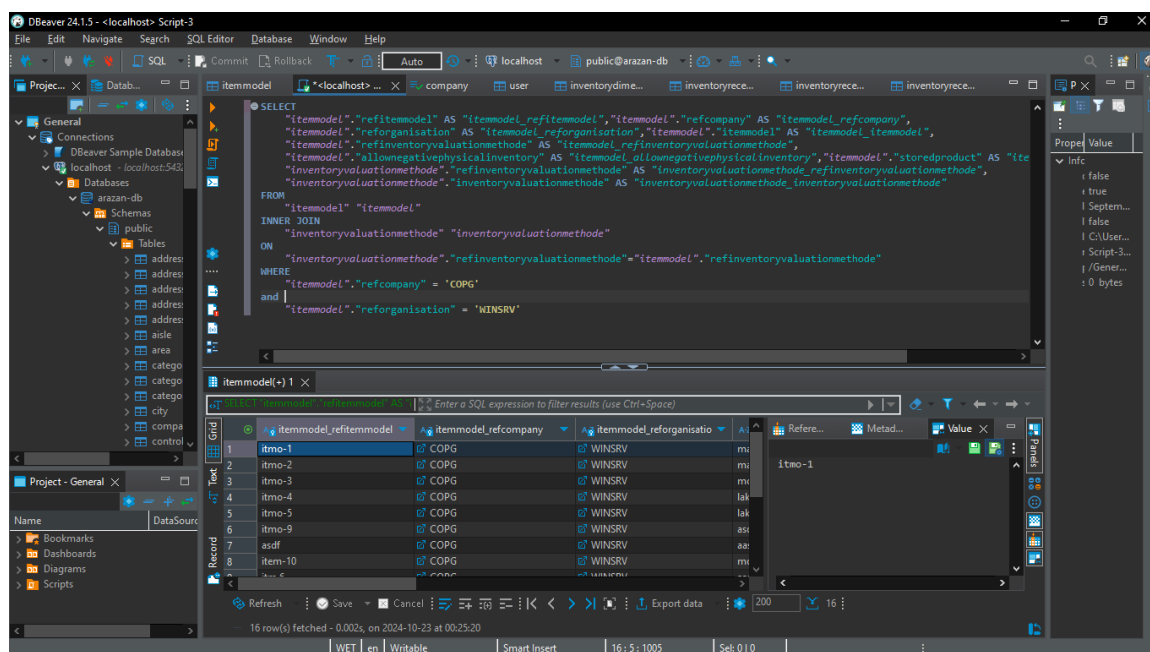


Figure 56 Exemple de requête SQL pour la gestion des stocks

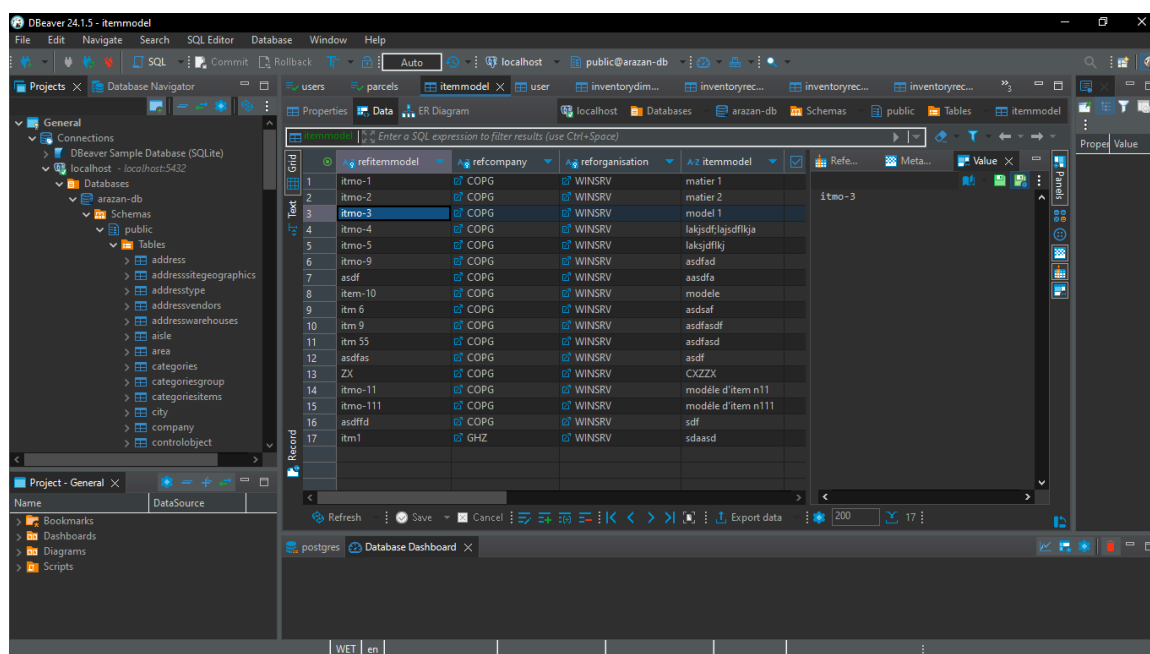


Figure 57 exemple de table de modèle d'articles