République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane MIRA – BEJAIA
Faculté des Sciences Exactes
Département d'Informatique

Mémoire Fin de Cycle

En vue de l'obtention du diplôme de Master Professionnel en Informatique

OPTION : Génie Logiciel

THEME

Conception et Réalisation d'une Application Web de Gestion Du processus d'achat avec Integration de l'Intelligence Artificielle

Réalisé par :

- Amrouche Melissa
- Ait Ahmed Meriem

Devant le jury composé de :

- Présidente : Mme.Yessad Nawal
- Encadrante : Mme. Bouadem Nassima
- Examinateur : M.Akilal Karim
- Examinatrice : Mme.Hamza Lamia
- Examinatrice : Mme.Tassoult Nadia

Année universitaire 2024-2025

Remerciements

Au nom d'Allah, le Tout Miséricordieux, le Très Miséricordieux,
Nous exprimons notre profonde gratitude à Dieu Tout-Puissant, qui nous a accordé la force, la patience et la persévérance nécessaires pour mener à bien ce modeste travail.

Nous tenons, dans un premier temps, à exprimer nos sincères remerciements à **Madame Bouadem Nassima**, notre encadrante universitaire, pour sa disponibilité, ses conseils avisés, ainsi que son accompagnement tout au long de ce projet. Son soutien pédagogique et son implication ont grandement contribué à l'orientation de notre travail.

Nous adressons également notre profonde reconnaissance à **Monsieur Samah Yassine**, notre encadrant au sein de **Cevital Agro-Alimentaire**, pour sa confiance, ses orientations claires et son suivi rigoureux durant les différentes phases de conception et de développement. Sa vision pratique et professionnelle a enrichi considérablement notre apprentissage.

Nos remerciements vont aussi à **Monsieur Boussad Ait Ahmed**, pour son implication sur le terrain et pour nous avoir permis de découvrir concrètement les processus de développement et de maintenance au sein de l'entreprise Cevital. Sa disponibilité et son partage d'expérience ont été d'une grande valeur et aide pour nous.

Nos remerciements s'adressent également à l'ensemble des membres de Cevital qui ont su se montrer disponibles, que ce soit pour répondre à nos questions, suivre l'évolution de notre travail, ou simplement nous apporter leurs encouragements. Leur implication a grandement contribué à la réussite de ce projet.

Nous remercions chaleureusement l'ensemble des membres du jury pour avoir accepté d'évaluer notre travail, ainsi que pour leurs remarques constructives et enrichissantes.

Un grand merci également à tous les enseignants et personnels administratifs du **Département d'Informatique** de l'**Université Abderrahmane Mira de Béjaïa**, pour la qualité de la formation dispensée et leur soutien tout au long de notre parcours universitaire.

Nous exprimons toute notre reconnaissance et gratitude à nos familles et à nos amis pour leur soutien moral et affectif tout au long de ce projet. Leur patience, leurs encouragements et leur confiance ont été un pilier essentiel dans la réussite de notre parcours.

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

Dédicaces

Je dédie ce travail, fruit d'un long parcours d'apprentissage, à toutes celles et ceux qui ont contribué à ma réussite et à l'accomplissement de ce projet.

À la mémoire de mon père, dont l'amour, les valeurs et les enseignements continuent de me guider chaque jour, j'aurais tant aimé qu'il soit là pour me voir réussir. Puisse ce travail lui rendre hommage.

À la mémoire de mon grand-père, dont la bienveillance et la sagesse continuent d'éclairer mon chemin. Je lui rends ici un hommage empreint d'admiration et de gratitude.

À ma mère, pour son amour inconditionnel, sa patience, ses prières silencieuses et sa présence constante, qui ont été une source inestimable de force et de motivation.

À mon frère, pour sa présence rassurante, ses encouragements et son soutien discret mais essentiel dans les moments de doute.

À ma famille, à ma grand-mère, mes tantes, mes oncles et mes cousines qui ont toujours cru en moi, m'ont soutenue avec tendresse et confiance, et ont célébré chacun de mes pas vers la réussite.

À moi-même, pour avoir persévéré malgré les obstacles, pour avoir cru en mes capacités, et pour ne jamais avoir abandonné.

À ma binôme, pour son engagement sérieux, sa collaboration rigoureuse et l'esprit d'équipe dont elle a fait preuve tout au long de ce travail.

À mes amis, pour leurs échanges stimulants, leurs remarques constructives et leur amitié précieuse, qui m'ont permis d'élargir ma réflexion et de grandir sur le plan personnel et académique.

À toutes les personnes, proches ou moins visibles, qui ont joué un rôle dans cette réussite, de près ou de loin — que ce soit par un geste, un mot, un regard ou un simple encouragement.

Que chacun d'entre vous trouve ici l'expression de ma reconnaissance et de mon affection sincère.

Amrouche Melissa

Dédicaces

Je dédie ce travail à ceux qui m'ont toujours soutenue et cru en moi

À mes très chers parents, pour qui je ne saurais trouver de mots assez forts pour vous exprimer mon amour infaillible, et ma profonde gratitude pour tous les sacrifices que vous avez fait pour moi. Votre amour inconditionnel, votre confiance en moi et vos encouragements durant toute ces années, font de moi la personne que je suis aujourd'hui.

À mes deux frères adorés, Yanis et Ouassim, qui ont toujours su me redonner le sourire, même dans les moments les plus difficiles. Merci d'être toujours là pour moi.

À ma binôme, pour notre collaboration sincère et notre soutien mutuel tout au long de ce travail.

À moi-même, pour la persévérance et la détermination dont j'ai su faire preuve pour mener ce projet à terme.

À mes amis, pour leur amitié précieuse, leur soutien constant et leurs nombreux encouragements.

À la mémoire de ma très chère grand-mère, que je porterai pour toujours et à jamais dans mon cœur.

Et à toutes les personnes qui, de près ou de loin, m'ont apporté leur aide, leurs encouragements ou simplement une pensée bienveillante tout au long de ce parcours.

Merci à tous pour vos encouragements, votre amour et votre confiance.

AIT AHMED Meriem

Table des matières

Liste des tableaux

Liste des figures

Liste des abréviations

IA	Intelligence Artificielle
LLM	Large Language Model
FAISS	Facebook AI Similarity Search
JWT	JSON Web Token
CSS	Cascading Style Sheets
JS	JavaScript
HTML	HyperText Markup Language
XSS	Cross-Site Scripting
ORM	Object-Relational Mapping
MVC	Model-View-Controller
MTV	Model-Template-View
KPI	Key Performance Indicator

Introduction Générale

Les entreprises de production jouent un rôle essentiel dans l'économie moderne. Elles transforment des matières premières en biens finis destinés à la consommation ou à d'autres secteurs industriels. Pour rester compétitives et assurer la pérennité de leurs activités, elles doivent maîtriser l'ensemble des flux liés à leurs opérations, depuis l'approvisionnement jusqu'à la livraison des produits finis aux clients.

Dans ce contexte, la gestion efficace de la chaîne d'approvisionnement (ou *supply chain*), qui désigne l'ensemble des fonctions et processus impliqués dans l'optimisation des flux de produits, d'informations et de flux financiers, devient un enjeu stratégique majeur. [1].

Parmi les processus clés de la *supply chain*, celui qui nous intéresse particulièrement est le **processus d'achat**,qui englobe toutes les activités visant à acquérir des biens ou des services nécessaires au fonctionnement de l'entreprise, depuis l'identification du besoin jusqu'à la réception et le paiement du produit ou service[2]. Plus précisément, notre attention se porte sur l'achat des **pièces de rechange**. Ces dernières conditionne directement la performance de la production, car elles sont essentielles pour assurer la maintenance et le bon de fonctionnement des équipements industriels.

Néanmoins, le processus d'achat demeure complexe, en particulier dans un environnement industriel comme Cevital, où le nombre élevé de références, dépassant parfois les 60 000 pièces de rechange, complique considérablement l'identification des composants requis. Cette diversité rend la gestion des approvisionnements à la fois délicate et chronophage.

C'est dans ce cadre que s'inscrit notre projet, qui vise à gérer l'intégralité du processus d'achat des pièces de rechange, depuis l'identification du besoin jusqu'à l'établissement du contrat et réception des produits, nous intégrons l'intelligence artificielle dans la première étape, via un chatbot conversationnel capable de faciliter la recherche de pièces à partir d'une description textuelle.

Pour accomplir ce travail, nous avons structurer notre memoire en quatre chapitres, comme suit :

— Le premier chapitre, nous presentons l'entreprise d'acceuil *Cevital* et nous decrivant le stage.

— Le second chapitre, introduit la méthodologie de conception adoptée : une approche hybride combinant les méthodes Cascade et Scrum. Nous expliquons ces deux approches séparément, justifions le choix de cette hybridation et illustrons un exemple concret afin d'appuyer notre choix. Ce chapitre présente également le langage de modélisation UML et le processus d'analyse mis en œuvre.

— Le troisieme chapitre orienté par le modèle en cascade, est dédié à l'analyse et la conception. Il commence par une une étude préalable, il expose la problématique, les objectifs

et la solution envisagée et il définit les différents acteurs du système ainsi que les besoins fonctionnels et non fonctionnels.

— Le quatrieme chapitre, concerne l'application de la méthode Scrum avec l'élaboration des users stories, le découpage en sprints, et la présentation des diagrammes associés à chaque itération.

— Le dernier chapitre est consacré aux phases de test et de déploiement de notre application.

Ce mémoire vise à offrir une vue d'ensemble complète de la conception et de la réalisation d'une application de gestion du processus d'achat avec integration IA en illustrant chaque étape du processus, depuis l'analyse initiale jusqu'à la mise en oeuvre finale.

Chapitre 1

Présentation de l'entreprise et description du stage

1 Définitions des concepts clés

— **Chaine d'approvisionnement** : La chaîne d'approvisionnement désigne l'ensemble des processus qui ont pour objectif d'optimiser les flux des produits, des informations et des finances depuis l'achat des matières premières jusqu'à la mise à disposition du produit fini aux consommateurs. [1]

— **Processus d'achat** C'est l'un des processus de la chaine d'approvisionnement et il regroupe toutes les activités visant à acquérir des biens ou des services nécessaires au fonctionnement de l'entreprise, depuis l'identification du besoin, en passant par la selection des founrnisseurs, la negociation des conditions contractueles, jusqu'à la réception et le paiement du produit ou service.[2]

— **Entreprise** est une unité de production juridiquement autonome dont l'objectif est de produire des biens et/ou des services à destination de personnes physiques ou morales afin d'en tirer un bénéfice, sa categorie varie selon son nombre d'employés et son chiffre d'affaire. [3]

— **Intelligence artificielle** : est une technologie qui repose sur la création et l'application d'algorithme et permet aux ordinateurs et aux machines d'imiter l'intelligence humaine tels que l'apprentissage, la résolution de problèmes, la prise de décision, ou encore la créativité et l'autonomie. [4]

— **Chatbot** : Un chatbot est un programme informatique qui simule une conversation humaine avec un utilisateur. La plupart des chatbots modernes utilisent des techniques d'IA conversationnelle telles que le traitement automatique du langage naturel (NLP) pour comprendre les questions des utilisateurs et automatiser les réponses qu'ils leur donnent. [5]

2 Historique de Cevital

2.0.1 Création et développement

Le Groupe Cevital a été fondé en 1998, marquant le début de son aventure en tant qu'acteur clé de l'industrie algérienne. Initialement axé sur le secteur agro-alimentaire, Cevital s'est rapidement développé pour devenir un conglomérat diversifié, présent dans plusieurs secteurs, notamment l'agro-industrie, la distribution, l'immobilier, et les services.[6]

Au fil des ans, l'entreprise a misé sur des investissements stratégiques et une vision entrepreneuriale forte, favorisant l'innovation et la création d'emplois. Cette stratégie a permis à Cevital de s'imposer comme un leader économique en Algérie et un acteur incontournable dans le bassin méditerranéen.

2.0.2 Évolution des secteurs d'activité

1. **1998 : Fondation et Lancement des Premières Activités**
   - Création de CEVITAL SPA Agro-Industrie
   - Développement initial axé sur les huiles alimentaires et le sucre.
   - Construction des premières infrastructures industrielles pour répondre aux besoins locaux.
2. **2005-2006 : Diversification et Développement Stratégique**
   - Création de filiales stratégiques : Numidis (distribution), Cevico (immobilier), et Imobis, élargissant les secteurs d'intervention de Cevital.
   - Lancement de LLK (activité industrielle) pour renforcer la capacité de production locale.
   - Acquisition de Cojek, élargissant les capacités industrielles du groupe.
3. **2007-2008 : Expansion Internationale**
   - En 2007 : Création de MFG, SAMHA et Aquisition BATICOMPOS
   - En 2008 : Création de MFG Europe, COGETP, CEVIAGRO ET NOLIS.
   - Consolidation de sa présence en Europe grâce à des investissements stratégiques dans l'industrie.
4. **2013-2014 : Renforcement en Europe**
   - En 2013 : Création de OXXO en France, ET ALAS en Espagne.
   - En 2014 : Acquisition de Brandt en France, ouvrant la voie à l'exportation de produits de haute technologie, et Aferpi rejoint le portefeuille du groupe, renforçant sa position dans le secteur manufacturier.
5. **2018-2023 : Transformation et Modernisation**
   - Intégration de nouvelles technologies, notamment dans la gestion des données et l'automatisation des processus.
   - Développement de projets axés sur l'analyse prédictive et la logistique pour optimiser les performances.

2.0.3 Produits alimentaires

- **De 1998 à 2007**
  - Huile alimentaire : Elio, Margarine : Fleurial et Matina.
  - Sucre : Skor 1kg, sucre liquide, Boissons aux Jus : Tchina, Eau minérale : Lalla Khedidja.
- **De 2013 à 2023**
  - La Chaux Calcique, Gaz CO2 alimentaire, Sucre Roux, Sauces et Condiments, La Production Plasturgie, La Trituration des Graines Oléagineuses.

2.0.4 Les filiales du groupe Cevital

1. **Cevital Agro-Alimentaire**
   - Cette branche regroupe plusieurs unités de production et est répartie sur sept sites physiques, et quinze unités de production. Elle est spécialisée dans plusieurs secteurs industriels de l'agroalimentaire.
   - Notre stage se deroule au niveau de ce complexe.
2. **Cevital Automotive**
   Cette branche englobe plusieurs entités liées à l'industrie automobile :
   - **Hyundai Motor Algérie :** Distributeur officiel de véhicules Hyundai en Algérie (particuliers, poids lourds, bus, véhicules commerciaux).
   - **Cevital MTP :** Location d'équipements pour travaux publics, transport et construction.
3. **Cevital Industry**
   Cette branche comprend plusieurs entités industrielles :
   - **MFG (Mediterranean Float Glass) :** Principal producteur de verre plat en Afrique, avec une capacité de production de 600 tonnes par jour.
   - **Cevital Minerals :** Exploration et exploitation de substances minérales, notamment pour la production de matières premières pour les huiles.
   - **Djawhara :** Usine de trituration pour la production d'huile brute à partir de graines oléagineuses (tournesol, soja, colza).
   - **MS (Metal Structure) :** Fabrication de profils métalliques pour l'industrie.
   - **Metal Sider :** Industrie spécialisée dans la production de produits métallurgiques.
   - **Prainsa Cevico Algeria :** Fabrication de composants en béton préfabriqué pour la construction.
   - **Baticompos :** Fabrication de composants industriels, panneaux sandwich et tôles métalliques nervurées.
4. **Cevital Distribution**
   Cette branche regroupe des entités dédiées à la distribution et au commerce de détail :
   - **Numidis :** Spécialisée dans la grande distribution, incluant les enseignes Unocity et Uno.
   - **Sierra Cevital :** Coentreprise entre Cevital et Sonae Sierra, spécialisée dans la gestion et le développement de centres commerciaux.
5. **Cevital Services**
   Offrant des services divers dans plusieurs domaines :
   - **Numilog :** Services logistiques incluant le transport, le stockage, l'emballage, et le transit douanier.
   - **Future Media :** Spécialisée dans la communication multimédia, design graphique et affichage publicitaire.
   - **Immobis :** Développement immobilier et gestion de projets à travers l'Algérie.
6. **Cevital Group - Divers**
   Offrant des services divers dans plusieurs domaines :
   - **Brandt :** Fabrication et vente d'électroménager, avec une forte présence en Algérie.
   - **GTP (Groupe de Travaux Publics) :** Spécialisé dans les travaux publics et la construction.
   - **OXXO :** Fabrication de fenêtres et portes préfabriquées avec des propriétés d'isolation thermique et acoustique.

FIGURE 1.1 – Les diferentes entreprises du Groupe Cevital

2.1 Secteur d'activité et spécialisation

Le secteur agro-alimentaire est le cœur historique et stratégique du Groupe Cevital. À travers ses infrastructures modernes et ses produits diversifiés, Cevital Agro-Alimentaire répond aux besoins des ménages algériens tout en se positionnant comme un acteur compétitif sur les marchés internationaux.

2.1.1 Unité de Production

Cevital Agro-Alimentaire s'organise autour de trois pôles majeurs :

1. **Pôle Gras**
   - **Production d'huiles alimentaires variées :**
     - **Fleurial :** gamme premium, reconnue pour sa qualité supérieure, incluant l'huile de tournesol, l'huile de colza, et l'huile spéciale friture.
     - **ELIO :** huile subventionnée par l'État, destinée à un usage quotidien accessible.
   - **Fabrication de margarines et beurres :**
     - **MATINA :** mélange matière grasse.
     - **Fleurial :** margarine de table et margarine de feuilletage.
     - **Beurre Tendre Gourmand :** un produit phare adapté à la consommation domestique.
2. **Pôle Sucre**
   - **Production de sucres variés, adaptés aux besoins des ménages et des industriels :**
     - **Sucre raffiné :** produit phare pour la consommation quotidienne.
     - **Sucre glace :** destiné aux usages culinaires et pâtissiers.
     - **Sucre roux cristallisé :** plus proche de l'état naturel, moins transformé.
     - sucre dilué pour le B2B (DAnone, HAmoud...)
3. **Pôle Boissons**
   - **Une large gamme de boissons sous la marque :**
     - **Jus en verre :** citronnade, orange-mandarine, orange-pêche, orange-abricot.
     - **Boissons gazéifiées :** TCHINA Pep's.
   - **Production d'eau minérale naturelle :**
     - **Lalla Khedidja,** disponible en version classique ou gazéifiée, en plusieurs formats (33 cl, 1L, 2L).

2.1.2 Unité de Support

L'unité de support joue un rôle transversal en assurant les fonctions essentielles au bon fonctionnement de Cevital Agro-Alimentaire. Elle est composée des départements suivants :

- **Direction des Ressources Humaines (DRH) :** Gestion du personnel, recrutement, et développement des compétences.
- **Direction Financière et Comptable (DFC) :** Gestion des budgets, analyses financières, et suivi économique.
- **Direction des Systèmes d'Information (DSI) :** Support technologique et digitalisation des processus.
- **Supply Chain :** Planification et optimisation des flux logistiques et des transports.
- **Commerciale :** Gestion des ventes et relations avec les clients B2C et B2B.
- **Qualité :** Contrôle des produits pour garantir des normes élevées et la satisfaction des clients.
- **Marketing :** Élaboration des stratégies de marque et analyse du marché.
- **Achats et Approvisionnements :** Gestion des relations avec les fournisseurs et et des flux d'approvisionnement.

2.1.3 Innovations dans le Secteur Agro-Alimentaire

Cevital Agro-Alimentaire se distingue par ses innovations technologiques et stratégiques, visant à améliorer la qualité et la durabilité de ses produits :

1. **Construction de l'usine Djawhara**
   - Permet la trituration des graines oléagineuses (colza, soja, tournesol), assurant une production d'huile brute 100 % locale
   - Cette avancée stratégique réduit la dépendance aux importations et valorise les ressources agricoles nationales.
2. **Segmentation de la Gamme d'Huiles**
   - Introduction d'une gamme publique subventionnée pour garantir l'accessibilité.
   - Développement d'une gamme premium (Fleurial) répondant aux attentes des consommateurs exigeants.
3. **Modernisation des Chaînes de Production**
   - Adoption de technologies avancées pour garantir une qualité constante et optimiser les coûts.
   - Intégration de processus automatisés pour la production et l'emballage, réduisant les pertes et améliorant l'efficacité.
4. **Engagement pour des Produits Naturels**
   - Développement de sucres roux et confitures à base de fruits locaux, répondant à une demande croissante pour des produits plus naturels et moins transformés.

2.2 Marchés

1. **Marché local**
   - **Origine et implantation :**
     Dès sa création en 1998, Cevital s'est concentré sur le marché algérien, répondant aux besoins alimentaires locaux grâce à une production axée sur les huiles alimentaires, le sucre, et d'autres produits essentiels.

— **Infrastructures de production :**
La mise en place des premières infrastructures industrielles a permis à l'entreprise de satisfaire une demande locale importante, en développant des unités de production adaptées aux spécificités du marché national.

2. **Marché régional**

— **Diversification et extension territoriale :**
Avec le lancement de filiales telles que Cevico, Numidis et Imobis entre 2005 et 2006, le groupe a commencé à étendre son empreinte au-delà des frontières immédiates, en ciblant des marchés régionaux dans le Maghreb.

— **Adaptation de l'offre :**
Cette phase d'expansion a nécessité une adaptation des produits et des processus logistiques pour répondre aux particularités des différentes régions, tout en maintenant une qualité constante et une distribution efficace.

3. **Marché international**

— **Expansion stratégique en Europe et au-delà :**
À partir de 2007, Cevital a amorcé une véritable expansion internationale. La création de MFG Europe, ainsi que d'autres initiatives comme l'acquisition de Brandt en France et la création d'OXXO en France et d'ALAS en Espagne, témoignent d'une volonté affirmée de conquérir des marchés étrangers.

— **Positionnement dans le bassin méditerranéen et en Afrique :**
La vision stratégique du groupe vise à devenir un acteur clé non seulement sur le marché européen, mais également dans le bassin méditerranéen et sur le continent africain. Cette orientation permet à Cevital de tirer parti de synergies économiques, de bénéficier d'économies d'échelle et d'adapter son offre aux exigences des marchés internationaux.

3 Organisation générale de Cevital

L'organisation de Cevital est structurée autour d'une **Direction Générale**, qui supervise plusieurs directions stratégiques et opérationnelles. Cette structure hiérarchique garantit une coordination efficace entre les différentes entités et permet a l'entreprise de répondre aux exigences de ses activités diversifiées.

1. Direction Générale

Au sommet de la structure, la Direction Générale définit les orientations stratégiques et supervise l'ensemble des activités de l'entreprise. Elle veille à l'exécution des objectifs à long terme et assure la cohérence des opérations.

2. Directions stratégiques et opérationnelles

Sous l'autorité de la Direction Générale, plusieurs directions assurent la gestion des fonctions clés de Cevital :

a) **Direction Achats et Approvisionnements**
1. Gère les relations avec les fournisseurs et assure un approvisionnement continu en matières premières, équipements, et services.
2. Optimise les coûts et garantit la qualité des produits et services acquis.

b) **Direction Supply Chain**
1. Planifie, coordonne et supervise les flux logistiques pour garantir une distribution rapide et efficace des produits.
2. Gère les infrastructures de transport et les systèmes de gestion des stocks.

c) **Direction Marketing**
1. Développe et met en œuvre des stratégies de promotion des produits et des marques du groupe.
2. Analyse les tendances du marché pour anticiper les attentes des consommateurs.

d) **Direction Qualité**
1. Garantit la conformité des produits aux normes nationales et internationales.
2. Met en place des processus d'amélioration continue pour maintenir les standards de qualité.

e) **Direction Commerciale B2C**
1. Responsable des ventes directes aux consommateurs via les points de distribution et les canaux numériques.
2. Analyse les performances de vente et adapte les offres en fonction des retours clients.

f) **Direction Commerciale B2B**
1. Développe les relations commerciales avec les entreprises et partenaires stratégiques.
2. Gère les contrats, les négociations, et les solutions adaptées aux besoins spécifiques des clients professionnels.

g) **Direction Finances et Comptabilité**
1. Assure la gestion financière et la comptabilité de toutes les entités du groupe.
2. Supervise les budgets, les prévisions, et les analyses de performance économique.

h) **Direction des Ressources Humaines (DRH)**
1. Recrute, forme et développe les talents pour répondre aux besoins des différentes entités.
2. Met en œuvre des politiques de gestion des compétences et assure un environnement de travail motivant.

i) **Direction des Systèmes d'Information (DSI)**
1. Soutient les autres directions en fournissant des outils technologiques modernes pour la digitalisation des processus.
2. Gère les infrastructures informatiques, la cybersécurité, et les projets de transformation digitale.

Représentation visuelle

une figure représentant l'organigramme général sous forme de graphique.

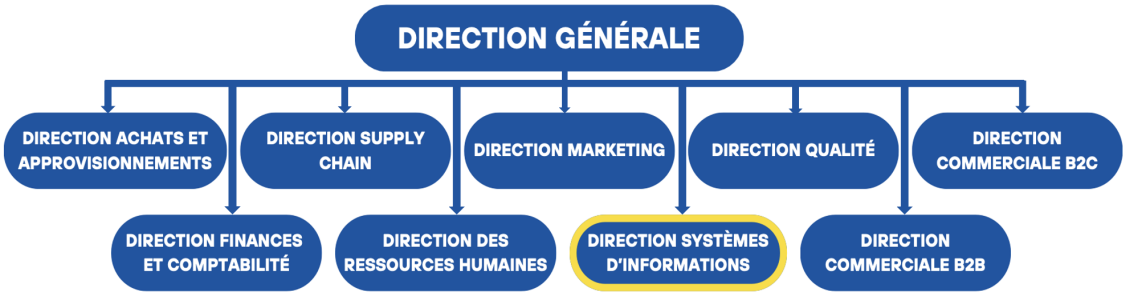

FIGURE 1.2 – Organigramme Général

4 Présentation de la DSI

4.1 Organisation générale de la DSI

La Direction des Systèmes d'Information (DSI) de Cevital est un organe stratégique qui pilote les technologies de l'information au sein du groupe. Ses responsabilités principales incluent :

4.1.1 Rôles et Responsabilités

La Direction des Systèmes d'Information (DSI) de Cevital est un organe stratégique qui pilote les technologies de l'information au sein du groupe. Ses responsabilités principales incluent :

1. **Gestion des applications métiers :** Fournir des outils adaptés aux besoins des départements clés (Supply Chain, Qualité, Vente, etc.).
2. **Transformation digitale :** Mettre en œuvre des solutions innovantes pour optimiser les processus métiers et la prise de décision.
3. **Sécurité des Systèmes d'information :** Assurer la protection des systèmes d'information et garantir leur résilience face aux menaces cybernétiques.
4. **Support technique :** Assister les utilisateurs dans la résolution des problèmes informatiques et la maintenance des infrastructures.

4.1.2 Organisation et Directions

Le Directeur SI supervise l'ensemble des activités de la DSI et veille à aligner les stratégies informatiques avec les objectifs globaux de l'entreprise.
La Direction des Systèmes d'Information est structurée en plusieurs directions spécialisées :

1. **Direction IT Technique**
   - Gère le **Support IT**, **les réseaux**, et **l'administration des systèmes**.
   - Responsable de la virtualisation des serveurs, de la sauvegarde des données et de la configuration des réseaux téléphoniques.
2. **Direction Sécurité des Systèmes d'Information**
   - Définit les normes et les démarches de gouvernance en matière de sécurité.
   - Assure la surveillance des systèmes et la réponse rapide aux incidents.
3. **Direction Transformation Digitale et BI**
   - Crée des rapports et des tableaux de bord en utilisant des outils comme Power BI et SQL Server.
   - Déploie des solutions d'extraction et de transformation des données via des ETL tels que Pentaho et Talend.
4. **Direction Applications Métiers**
   Cette direction couvre plusieurs domaines :
   - **Ressources Humaines** : Gestion des paies, des processus administratifs et des accès.
   - **Supply Chain** : Planification logistique et gestion des flux.
   - **Qualité Produit et Achats** : Supervision de la conformité produit et des fournisseurs.
   - **Finance et Comptabilité (FICO)** : Contrôle des budgets, des factures et des rapports financiers.

— **Vente et Distribution** : Coordination des ventes et suivi des performances commerciales.
— **GMAO (Gestion de Maintenance Assistée par Ordinateur)** : Planification et suivi des interventions de maintenance.

5 Description du stage

5.1 Présentation du sujet de stage

Le sujet de stage s'est concentré sur l'optimisation du processus d'achat au sein de l'entreprise Cevital, en adoptant une approche hybride combinant des méthodes en cascade et Agile pour répondre aux défis spécifiques du domaine des achats.

Ce projet se décline en trois volets principaux :

— **Gestion du Processus d'Achat :** Développement d'une application intégrée au service Supply Chain pour assurer la disponibilité des ressources essentielles :
  — Les PDR (pièces de rechange pour les machines).
— **Projet d'Achat :** Mise en place d'un processus complet allant du sourcing à la commande d'achat, comprenant :
  — L'avant-projet d'achat, où sont identifiés les fournisseurs.
  — La formulation des demandes d'achat initiées par les utilisateurs,
  — La validation et la finalisation de la commande d'achat par un expert.
— **Intégration de l'Intelligence Artificielle :** Déploiement d'une solution IA pour permettre la reconnaissance automatique des pièces et fournir leurs références correspondantes.

L'objectif global de ce stage est de concevoir et développer une solution qui permet la gestion des achats, améliore la traçabilité des références et facilite le choix des fournisseurs. Le projet vise à renforcer l'efficacité opérationnelle de Cevital en combinant rigueur planifiée et flexibilité itérative grâce à une démarche Agile adaptée à un environnement industriel complexe.

5.1.1 Apport du projet pour Cevital

Ce projet répond aux besoins stratégiques et opérationnels de Cevital en apportant plusieurs bénéfices concrets :

— **Gain d'efficacité :** L'automatisation du processus d'achat permettent de réduire le temps nécessaire à la gestion des commandes, améliorant ainsi la réactivité de l'entreprise et permettant une meilleure coordination entre les départements.
— **Amélioration de la productivité :** La simplification des processus et l'identification rapide des pièces permettent de diminuer les interruptions de production, conduisant à une meilleure gestion des stocks et à une augmentation de la productivité globale.
— **Avantage compétitif :** En modernisant le système de gestion des achats et en intégrant des technologies innovantes comme l'IA, Cevital se positionne comme un acteur agile et réactif dans un environnement concurrentiel, capable de s'adapter rapidement aux évolutions du marché.

Ce projet, en alignant les objectifs techniques et fonctionnels avec les besoins stratégiques de l'entreprise, offre ainsi une solution complète pour la gestion des achats et améliorer la performance opérationnelle de Cevital.

6 Conclusion

Dans ce premier chapitre, nous avons présenté l'entreprise Cevital ainsi que le cadre de notre stage. La suite de ce mémoire débutera par une étude préalable afin d'identifier les interactions et les besoins spécifiques du processus d'achat, dans le but de mieux cerner les objectifs de notre projet.

Pour sa réalisation, nous avons opté pour une méthodologie de conception hybride, qui sera détaillée dans le troisième chapitre.

Chapitre 2

Méthodologie de développement

1 Introduction

Dans ce chapitre, nous commençons par expliquer les méthodes Cascade et Agile, en décrivant leurs spécificités respectives. Nous illustrons ensuite leur complémentarité à travers un exemple concret d'intégration dans un projet réel.

Nous introduisons par la suite la méthodologie hybride, qui combine les deux approches mentionnées précédemment, et nous justifions ce choix dans le cadre de notre projet. Enfin, nous présentons le langage UML pour la modélisation des diagrammes ainsi que les outils technologiques retenus pour le développement.

2 Le modèle en Cascade

2.1 Présentation du modèle en Cascade

2.1.1 Définition

La modèle en cascade, ou **Waterfall**, est un modèle de gestion de projet linéaire, principalement utilisé dans le développement de logiciels. Il divise le processus de développement en phases successives et distinctes [7]. Chaque phase est réalisée une seule fois et doit être achevée avant de passer à la suivante, car ses sorties servent d'entrées à la phase suivante.

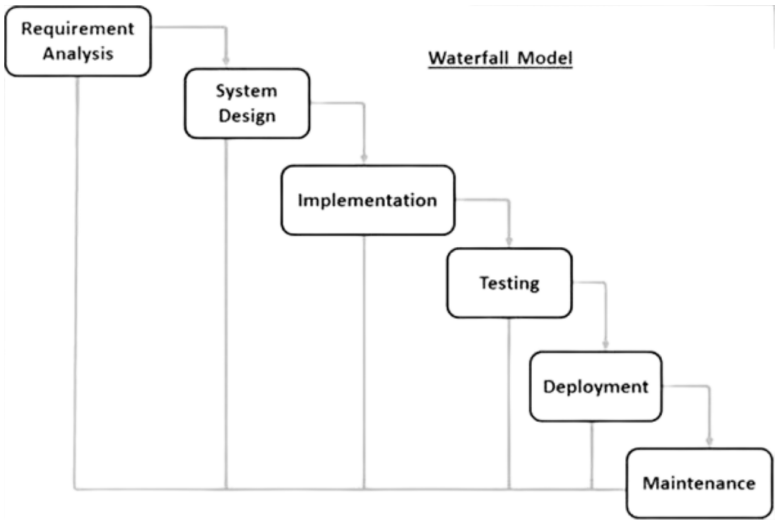

FIGURE 2.1 – Le modèle en Cascade

2.1.2 Phases du cycle de vie

Le modèle en cascade repose sur une série de phases bien définies, exécutées de manière séquentielle. Ces phases sont généralement les suivantes :

— **Phase 1 – Expression des besoins** En amont du projet, la définition des besoins et exigences est cruciale pour établir une base solide et un plan détaillés. Cette étape implique le receuil et la documentation précise des attentes des parties prenantes et des utilisateurs.

— **Phase 2 – Analyse des besoins**
Cette phase est essentielle pour clarifier la compréhension des attentes du projet, en analysant en profondeur la liste des exigences spécifiées lors de la phase précédente, afin de formuler les fonctionnalités du système à développer. Elle permet d'établir une base claire et conforme aux attentes des parties prenantes, tout en anticipant les principales contraintes de réalisation : budget, délais et faisabilité.

— **Phase 3 – Conception** : Cette phase consiste à transformer les besoins identifiés lors de l'analyse en une architecture technique et fonctionnelle du système. Elle comprend la définition de l'architecture logicielle, le choix des technologies (frameworks, outils), ainsi que la modélisation des données. C'est également à ce stade que sont élaborés les modèles UML tels que le diagramme de classe.

— **Phase 4 – Implémentation** : Cette phase consiste à traduire les fonctionnalités définie précédemment en langage de programmation, tout en respectant les choix techniques décidés lors de la phase de conception. L'objectif principal de cette étape, est l'obtention d'une version fonctionnelle du logiciel.

— **Phase 5 – Tests** : Cette phase a pour objectif de s'assurer que le logiciel répond aux exigences définies lors de la phase d'analyse. Pour cela, plusieurs types de tests sont réalisés afin de vérifier son bon fonctionnement, tels que : les tests unitaires, les tests d'intégration et les tests d'acceptation. Toute anomalie détectée est corrigée avant le déploiement.

— **Phase 6 – Déploiement** : Une fois les tests passés et le logiciel validé, il est livré à l'utilisateur final et installé dans l'environnement réel. Cette phase comprend également la formation des utilisateurs, la configuration finale, ainsi que la rédaction de la documentation d'utilisation.

— **Phase 7 – Maintenance** : Après le déploiement, le logiciel peut nécessiter des ajustements. Cette phase assure la correction des éventuels bugs détectés, l'adaptation aux évolutions de l'environnement, ainsi que l'ajout de nouvelles fonctionnalités selon les besoins exprimés par les utilisateurs.

3 La méthode Scrum

3.1 Présentation de la méthode Scrum

3.1.1 Définition

Scrum est un framework de gestion de projet Agile [8] et constitue aujourd'hui l'approche Agile la plus utilisée car elle aide les équipes de développeurs à structurer et à gérer leur travail selon des principes, des pratiques et l'ensemble des valeurs du **Agile Manifesto** : la collaboration avec le client, l'acceptation du changement, l'interaction avec les personnes et des logiciels opérationnels [9].

Son objectif est d'établir un cadre de travail clair et structuré basé sur des itérations courtes,

facilitant ainsi la gestion de projets complexes. Cette approche vise à améliorer la productivité des équipes agiles, même en télétravail, tout en optimisant le produit grâce à des retours réguliers des utilisateurs finaux.

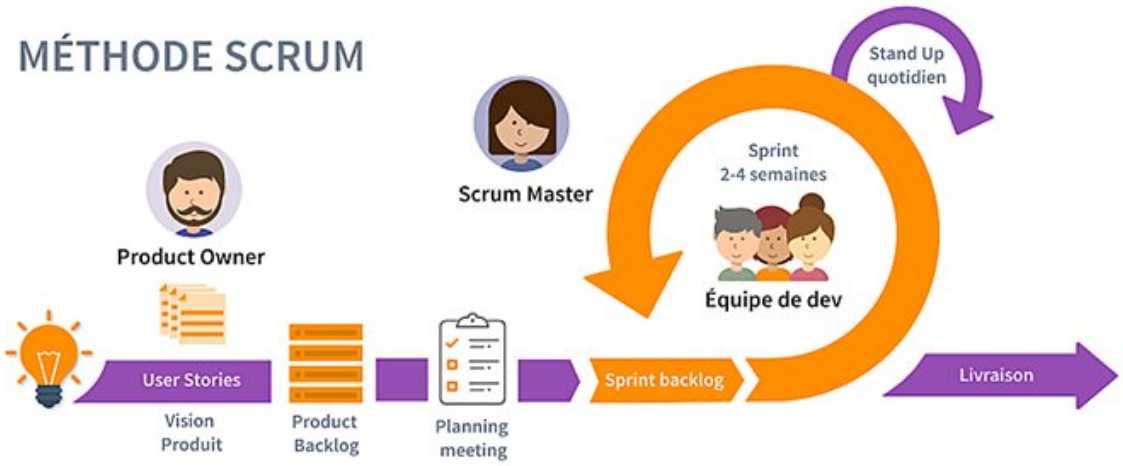

FIGURE 2.2 – représentation de la methode Scrum

3.1.2 Les rôles dans Scrum

Les équipes Scrum sont auto-organisées et pluridisciplinaires. Elles choisissent la meilleure façon d'accomplir leur travail, au lieu d'être dirigées par des personnes externes à l'équipe, ce qui favorise la flexibilité, la créativité et la productivité. Chaque équipe est organisé comme suit :

- **Le Scrum Master :** est le garant de la mise en application et le respect de la méthode.
- **Le Product Owner :** responsable du produit, donne les retours sur ce dernier, définit le backlog, donne les retours des utilisateurs
- **L'equipe de développement :** Est chargée de transformer les besoins exprimés en fonctionnalités utilisables. L'équipe peut être pluridisciplinaire et composer de plusieurs profils : développeurs, architectes logiciel, graphistes, ergonomes,etc.

3.1.3 Les artefacts de Scrum

Les artefacts de Scrum représentent la valeur créée et sont conçus pour garantir la transparence des informations essentielles. Ainsi, chaque artefact intègre un engagement visant à fournir des informations qui renforcent cette transparence qui permet à tous de disposer d'une base commune pour ajuster le processus.[10]

1. **Product Backlog :** Un carnet de produit constitué d'une liste ordonnée et évolutive regroupant tous les éléments susceptibles d'améliorer le produit. Il sert à recueillir les attentes des clients sous forme de *User Stories*, qui seront utilisées lors de la planification des sprints. Ce backlog est affiné par le Product Owner en collaboration avec l'équipe de développement afin de clarifier et prioriser les éléments.Ce carnet est affiné régulièrement (*Product Backlog Refinement*). Bien qu'il soit géré par le Product Owner, il doit rester facilement accessible et partagé avec l'ensemble de l'équipe de développement.

2. **Sprint Backlog :** Un carnet de sprint qui regroupe l'ensemble des éléments sélectionnés dans le Product Backlog pour être réalisés durant un Sprint donné. Il inclut non seulement ces éléments : user stories, tâches, etc. , mais aussi un plan détaillé permettant à l'équipe de développement d'atteindre l'objectif du Sprint (*Sprint Goal*). Ce backlog est défini et ajusté par l'équipe de développement et mis à jour en temps réel, offrant ainsi une vision claire de l'avancement du travail pendant le Sprint.

3. **Le Sprint :** Un Sprint ou itération est un intervalle de temps fixe, généralement compris entre 1 à 4 semaines, durant lequel l'équipe de développement conçoit, réalise et teste de nouvelles fonctionnalités afin de produire un incrément du produit. Chaque Sprint démarre par une réunion de planification (*Sprint Planning*) et se termine par une revue (*Sprint Review*) suivie d'une rétrospective (*Sprint Retrospective*). L'objectif du Sprint est défini par le **Sprint Goal**, qui oriente le travail de l'équipe dans le but de produire un incrément du produit qui soit potentiellement livrable.

4. **Increment :** L'incrément représente la somme de tous les éléments terminés du Product Backlog à la fin d'un Sprint. Il s'agit d'une version du produit qui intègre les nouvelles fonctionnalités développées. Chaque incrément doit être :fonctionnel, testable et conforme à la "Definition of Done" avant d'être considéré comme terminé. Plusieurs incréments peuvent être livrés dans un même Sprint, mais ils doivent tous être utilisables. L'incrément est présenté par la suite aux parties prenantes lors de la Sprint Review.

5. **Definition of Done (DoD) :** Un ensemble de critères définie par l'équipe Scrum qui déterminent quand un élément du Product Backlog ou un Increment est considéré comme terminé et garantir que le travail livré est de qualité et répond aux attentes des parties prenantes.

3.1.4 Les événements de Scrum

1. **Planification du Sprint (Sprint Planning) :** Le Sprint Planning est l'événement qui démarre un sprint, il a une durée d'environ 4 heures pour un sprint de 2 semaines et 8 heures pour un sprint d'un mois. Toute l'équipe Scrum se réunit et collabore pour définir ce qui se fera durant le sprint en sélectionnant des éléments du Product Backlog, comment le travail sera réalisé en élaboration d'un plan d'action) et determiner l'objectif du Sprint (Sprint Goal) qui guide l'équipe tout au long de l'itération.
2. **Daily Scrum :** Une réunion courte d'environ 15 minutes et quotidienne permettant d'inspecter la progression du travail et d'adapter le plan si nécessaire. Son objectif est de suivre l'avancement du projet, identifier les éventuels obstacles et ajuster le plan en conséquent.
3. **Revue de Sprint (Sprint Review) :** Une réunion interactive avec les parties prenantes, pour leurs présenter le résultat du Sprint et d'adapter le Product Backlog si nécessaire en cas de changements. A une durée maximale de 4 heures pour un Sprint d'un mois.
4. **Rétrospective du Sprint (Sprint Retrospective) :** Le dernier événement du Sprint, la rétrospective permet à l'équipe d'dentifier les points positifs et les difficultés rencontrées, et définir des améliorations à mettre en place dès le prochain Sprint. Il peut avoir une durée maximale de 3 heures pour un Sprint d'un mois.[10]

   Bien que Scrum fournisse un cadre complet pour l'organisation et la gestion du projet, nous avons toutefois eu recours à quelques pratiques de l'Extreme Programming (XP), qui est également une méthodologie qui applique à l'extrême les principes du développement

agile [11]. Ces pratiques, appliquées tout au long des événements Scrum, ont permis un développement plus efficace et une meilleure collaboration au sein de l'équipe. Les voici :

(a) **Programmation en binôme :** Deux développeurs travaillent ensemble sur le même code, ce qui favorise la qualité de ce dérnier, la communication et le partage des connaissances.

(b) **Refactoring :** Restructurer et nettoyer le code existant sans modifier son comportement externe. Le refactoring permet d'améliorer la lisibilité et la maintenabilité.

(c) **Petites releases fréquentes :**Livrer réguliérement des versions fonctionnelles du logiciel,même si elles ne contiennent qu'un petit nombre de fonctionnalités. Cela permet d'obtenir rapidement des retours d'expérience des utilisateurs et de s'adapter aux changements de maniére agile.

4 Intégration des deux approches

L'intégration des approches Cascade et Agile permet de combiner les avantages de chacune pour maximiser l'efficacité du projet. La méthode Cascade apporte une structure rigoureuse, avec une planification détaillée, une documentation exhaustive et des jalons bien définis pour les phases critiques, telles que l'analyse des besoins, la conception, les tests finaux et le déploiement. Cette approche assure une stabilité et une cohérence tout au long du projet. En parallèle, la méthode Agile introduit une flexibilité essentielle, en permettant des développements itératifs et rapides, facilitant ainsi l'adaptation aux feedbacks des utilisateurs et aux évolutions des besoins métiers.

En combinant ces deux approches, notre projet peut démarrer sur une définition claire des exigences et une conception solide (Cascade), puis évoluer vers un développement itératif qui intègre en continu les ajustements nécessaires (Agile). Cette complémentarité est particulièrement adaptée aux projets complexes, comme la mise en place de notre solution chez Cevital, où il est essentiel de concilier prévisibilité et agilité pour répondre aux exigences opérationnelles tout en s'adaptant aux changements.

5 La méthodologie Hybride

5.1 Définition de la méthodologie Hybride

La méthodologie hybride est une approche de gestion de projet qui combine des éléments des méthodes en cascade et Agile. Elle vise à profiter de la rigueur, de la planification détaillée et de la documentation structurée de la méthode cascade pour les phases critiques (analyse des besoins, conception, tests finaux, déploiement), tout en intégrant la flexibilité et l'itération rapide offertes par la méthode Agile durant le développement. Cette approche permet de mieux répondre aux évolutions fréquentes des exigences tout en assurant la stabilité et la qualité du produit final [12, 13].

5.2 Justification du choix

Le choix de recourir à une méthodologie hybride repose sur plusieurs facteurs clés :

— **Planification Rigoureuse et Flexibilité :** La méthode cascade offre une structure claire et une planification précise indispensable pour les phases d'analyse, de conception et de tests finaux, tandis que la méthode Agile permet d'adapter rapidement le développement aux retours des utilisateurs et aux évolutions des besoins métiers [14, 12].

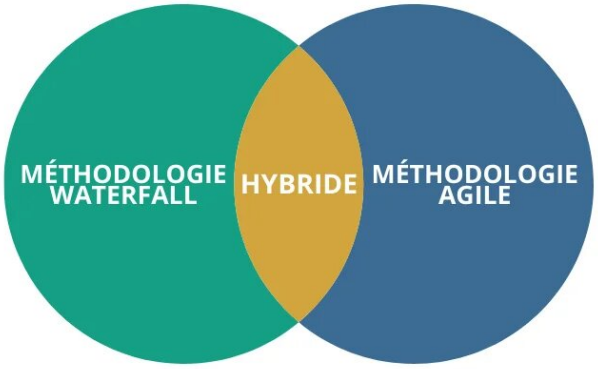

FIGURE 2.3 – représentation de la gestion de projet hybride

— **Gestion de Projets Complexes :** Dans des environnements complexes, tels que le développement d'ERP, il est crucial de combiner la prévisibilité des projets planifiés avec la capacité d'adaptation rapide aux changements. Cette approche est notamment utilisée chez Cevital, où la complexité des projets exige une coordination fine entre planification et agilité [13].

— **Optimisation des Processus Métiers :** En intégrant des cycles itératifs Agile, il est possible d'améliorer en continu les processus métiers et de répondre efficacement aux exigences spécifiques des utilisateurs finaux [14].

— **Expérience Organisationnelle :** De nombreuses entreprises, dont Cevital, adoptent déjà des approches hybrides pour maximiser l'efficacité des projets d'envergure en assurant à la fois la stabilité et l'innovation [12].

5.3 Avantages et inconvénients de la méthode Hybride

5.3.1 Avantages :

— **Combinaison des Forces :** Permet de bénéficier de la planification et de la documentation détaillée de la cascade tout en intégrant la flexibilité et l'adaptabilité d'Agile [13].

— **Adaptabilité aux Changements :** Facilite l'ajustement rapide aux évolutions des besoins métiers grâce à des cycles de développement itératifs [14].

— **Visibilité et Contrôle :** Offre une meilleure visibilité sur l'ensemble du projet grâce à une structure planifiée, tout en permettant des révisions régulières et des ajustements basés sur les retours utilisateurs.

— **Implication des Utilisateurs :** Encourage une forte collaboration avec les utilisateurs finaux, améliorant ainsi l'adéquation du produit aux attentes métiers.

5.3.2 Inconvénients :

— **Complexité de Gestion :** La coexistence de deux approches peut engendrer une complexité supplémentaire dans la gestion du projet et nécessiter une coordination étroite entre les équipes.

— **Synchronisation Difficile :** Les cycles itératifs Agile doivent être bien alignés avec les jalons planifiés de la cascade, ce qui peut poser des problèmes de synchronisation et de priorisation.

— **Culture Organisationnelle :** La réussite de cette approche hybride requiert une culture d'entreprise capable d'adopter et de gérer simultanément deux méthodologies différentes, ce qui peut représenter un challenge dans certaines organisations.
— **Risques de Conflits :** Des conflits de priorisation et des retards peuvent survenir si les processus et les équipes ne sont pas parfaitement intégrés et alignés sur les objectifs du projet.

5.4 Exemple réel d'intégration Hybride (Cascade + Agile) : La transformation numérique chez ING Bank

La banque ING a utilisé une approche hybride combinant Cascade et Agile pour moderniser son infrastructure IT et accélérer le développement de ses services numériques. Ce projet visait à digitaliser la banque en améliorant l'expérience utilisateur et en optimisant l'efficacité opérationnelle [15, 16].

5.4.1 Analyse des Besoins et Etude préalable(Cascade)

Dans un premier temps, ING a suivi une approche traditionnelle pour structurer le projet durant laquelle :

— Un cahier des charges détaillé a été élaboré afin de structurer le projet et aligner les équipes IT et métiers.

Cette phase a permis d'établir une roadmap avec des objectifs précis avant de lancer le développement

5.4.2 Conception de l'Architecture IT (Casacade)

Par la suite, ING a conçu un modèle d'architecture système robuste, garantissant l'intégration avec les systèmes existants et assurant la conformité réglementaire :

— Un modèle robuste a été défini pour garantir la sécurité et la scalabilité du système.
— Des exigences en cybersécurité ont été définies pour renforcer la protection des données clients et éviter les risques liés aux cyberattaques.

5.4.3 Développement Itératif (Agile - Scrum)

Après la phase de conception, ING a adopté Scrum pour accélérer le développement :

— Les équipes ont été orgnanisé en Squads et Tribes, chacune étant responsable d'un domaine spécifique (ex. paiements, gestion des comptes, sécurité IT).
— Le projet a été découpé en sprints de 2 à 4 semaines, chaque sprint permettant de développer et tester des fonctionnalités de manière itérative.
— Des outils DevOps ont été intégrés pour automatiser les déploiements et optimiser la gestion du code [17].

5.4.4 Tests et Validation (Cascade & Agile)

Pour la phase final, afin de garantir la fiabilité du système, une approche mixte a été adoptée :

— Tests unitaires et fonctionnels réalisés à chaque sprint pour assurer la qualité du code.
— Tests d'intégration et de sécurité en suivant une approche cascade pour valider l'ensemble du système avant son déploiement.
— Une stratégie de test en continu a été mis en place pour garantir la stabilité des nouvelles fonctionnalités avant leur mise en production.

5.4.5 Déploiement Progressif et Mise en Production (Cascade - Canary Release)

Le déploiement a été effectué progressivement selon une approche Canary Release :

— Les nouvelles fonctionnalités ont d'abord été testées sur un petit groupe d'utilisateurs, avant un déploiement généralisé.

— Un plan de formation a été mis en place pour accompagner les employés dans la transition vers les nouveaux outils digitaux.

5.4.6 Amélioration Continue (Agile)

Même après la mise en production, ING a maintenu une dynamique d'amélioration continue :

— Prise en compte des retours utilisateurs dans de nouveaux sprints Agile.

— Planification régulière de mises à jour pour optimiser l'expérience utilisateur et la performance du système, tout en s'adaptant aux nouveaux besoins[18].

6 Langage de modélisation UML

6.1 Définition

Le langage de modélisation UML est un langage visuel créé en janvier 1997 par la fusion de plusieurs langages de modélisation objet tels que BOOCH, OMT et OOSE. Cette fusion a été réalisée par James Rumbaugh, Grady Booch et Ivar Jacobson, qui ont collaboré chez Rational Software pour créer un langage de modélisation objet standardisé et prometteur.

UML, acronyme de 'Unified Modeling Language', qui signifie en français 'Langage de Modélisation Unifié, permet de représenter graphiquement et abstraitement à travers differents diagrammes les multiples aspect d'un système informatique : sa structure, son comportement, ses interactions et ses processus [19]. UML propose 14 types de diagrammes, classés en deux catégories principales :

— Les diagrammes structurels, qui décrivent la composition statique du système, notamment ses classes, objets et relations.

— Les diagrammes comportementaux, , qui représentent la dynamique du système, en mettant en évidence les interactions et les processus en cours d'exécution.

Comme illustré dans la figure ci-dessous, ces diagrammes permettent aux développeurs de visualiser, spécifier, construire et documenter les systèmes logiciels. Cette approche facilite la compréhension et la communication entre les différentes parties prenantes du projet.

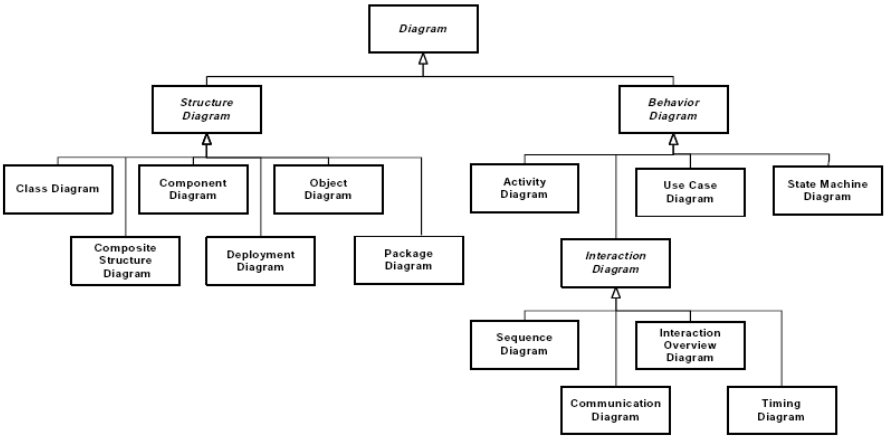

FIGURE 2.4 – Les diagrammes UML

6.2 Justification du choix du langage UML

Nous avons opté pour l'UML pour la modélisation de notre projet car c'est est un langage standardisé et en raison de son efficacité à représenter les différentes dimensions de notre système et à capturer de manière exhaustive ses exigences fonctionnelles et ses aspects structurels.

Son utilisation présente plusieurs avantages :

- **Langage standardisé et universel :** UML est largement utilisé dans l'industrie du développement logiciel, permettant une communication claire entre toutes les parties prenantes (développeurs clients,...etc).
- **Clarification des éxigences :** Grâce à ses différents diagrammes, UML aide à formaliser et visualiser les exigences du système avant même de commencer le développement.
- **Facilite la conception :** Il permet d'organiser les composants du système, de définir leurs interactions et de structurer les différentes phases du projet.
- **Facilite les communications :** Il favorise l'échange entre les équipes techniques et fonctionnelles, réduisant ainsi les risques d'erreurs.
- **Réutilisation et Maintenance :** Une modélisation claire assure la maintenabilité du système et de favorise la réutilisation des concepts dans de futurs projets.

Pour notre projet, nous allons utilisé quantre types de diagrammes UML pour modéliser les différents aspects du système :

- Le diagramme de cas d'utilisation : est un diagramme UML qui permet de décrire les interactions entre les utilisateurs et le système. Son objectif est de représenter les différentes façons dont un utilisateur peut interagir avec un système. [20]
- Le diagramme de séquence : est un diagramme UML qui offre une modélisation dynamique, il comprend un groupe d'objets représentés par des lignes de vie et la séquence de messages echangé entre eux durant une interaction. Il illustre le déroulement des scénarios d'utilisation. [21]
- Le diagramme de classes : est un diagramme de type structurel qui est fondamental le processus de modélisation des objets, car il décrit de maniere claire la structure statique du système en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets. [22]
- Le diagramme d'états-transitions : est un diagramme UML de type comportemental, il représente les différents états d'un objet et les événements qui les déclenchent tout au long de son cycle de vie. [23]

Cette approche nous a permis d'avoir une vision claire du fonctionnement du système et de garantir une conception cohérente et optimisée.

7 Les Processus d'analyse et de conception

Deux grands processus peuvent guider l'analyse et la conception d'un système :

— Le processus dirigé par les classes

— Le processus dirigé par les cas d'utilisation

Chacun d'eux suit un flux logique distinct, comme illustré dans l'image ci-dessous :

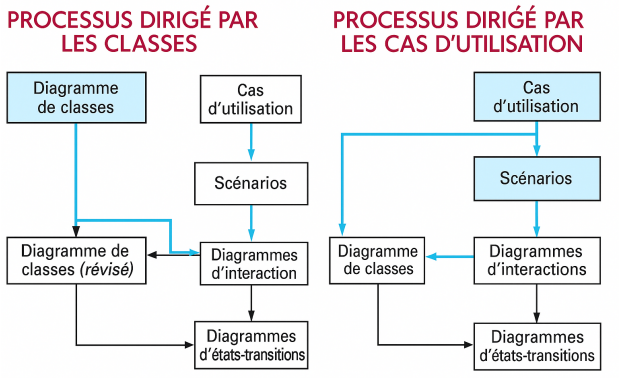

FIGURE 2.5 – Les Processus d'analyse et de conception

7.1 Processus dirigé par les classes

Description

Ce processus commence par l'identification et la modélisation des **classes principales** du domaine. Ensuite, les cas d'utilisation, scénarios et interactions, viennent enrichir ou réviser le diagramme de classes.

Avantages

— Favorise une **modélisation rigoureuse du domaine**.

— Convient bien aux projets **structurés ou fortement orientés données**.

— Facilite la **réutilisabilité du modèle objet**.

Inconvénients

— Moins adapté aux **besoins évolutifs ou itératifs**.

— Ne met pas assez l'accent sur le **point de vue utilisateur ou métier**.

7.2 Processus dirigé par les cas d'utilisation

1. Description

Ce processus commence par l'identification des **cas d'utilisation**, qui expriment les fonctionnalités attendues par les utilisateurs. Par la suite, les scénarios, les diagrammes d'interaction sont élaborés, puis les classes nécessaires sont déduites.

2. Avantages

— Fortement **orienté utilisateur / métier**.
— Très adapté aux méthodes **agiles** et aux contextes où les exigences évoluent.
— Permet une **traduction directe des besoins fonctionnels en architecture technique**.

3. Inconvénients

— Peut entraîner une **modélisation désorganisée du domaine** si les classes sont négligées.
— Parfois **moins rigoureux** sur le plan de la structure objet.

7.3 Choix du processus dirigé par les cas d'utilisation

Pour notre projet de **gestion du processus d'achat avec intégration d'IA**, nous avons opté pour le **processus dirigé par les cas d'utilisation**, pour les raisons suivantes :

Justifications

1. **Méthodologie hybride (Cascade et Agile Scrum)** :
   — Nous utilisons **Scrum pour les itérations fonctionnelles**, donc le découpage en **cas d'utilisation** est naturel et permet une planification par **sprint**.
   — Le **Cascade initial (spécifications, modélisation)** est assuré par les cas d'utilisation comme base formelle.
2. **Projet centré sur les utilisateurs et les workflows** :
   — Notre applciation implique plusieurs rôles (ex. : acheteur, responsable, magasinier, etc.), chacun se voit associés des **actions spécifiques** (ex. : créer commande, valider contrat, réceptionner, etc.), ce qui facilite la modélisation sous forme de cas d'utilisation.
3. **Évolution possible des besoins** :
   — L'intégration de l'intelligence artificielle implique de possibles adaptations en cours de projet.
   — L'IA intervient souvent dans des cas d'utilisation ciblés, notamment dans notre cas via une recherche intelligente de pièces de rechange assistée par un chatbot .
   — Le processus basé sur les cas d'utilisation permet de de s'adapter plus facilement à ces évolutions, contrairement à une modélisation rigide basée sur les classes.
4. **Meilleure collaboration équipe/client** :
   — Les cas d'utilisation sont **faciles à comprendre pour les parties prenantes**.
   — Ils permettent des revues fonctionnelles avant de coder, ce qui réduit le risque de non-conformité.

8 Conclusion

En conclusion, ce chapitre a permis de définir et de présenter les différentes méthodologies utilisées, ainsi que les langages et outils sur lesquels nous nous sommes appuyés pour la réalisation de notre projet. Nous avons détaillé les phases clés de la méthodologie hybride adoptée, définie le langage de modélisation UML et exposé les outils techniques qui seront mobilisés. Les chapitres suivants seront consacrés à la mise en œuvre de ces éléments, en illustrant leur application concrète à travers le développement et l'intégration progressive de notre solution.

Chapitre 3

Étude Préalable et Analyse

1 Introduction

Dans ce troisième chapitre, nous réalisons une étude préalable afin d'identifier l'étendue du projet et d'en cerner les principaux enjeux. Nous commençons par exposer la problématique puis l'objectif, avant de présenter la solution envisagée. Enfin, nous aborderons l'étape d'analyse des besoins, ou nous identifant les besoins fonctionnels et non fonctionnels ainsi que les acteurs de notre application.

2 Étude préalable

L'étude préalable est une étapes fondamentale dans la mise en place d'un projet informatique, elle garantit une compréhension approfondie du contexte, délimiter le champ d'étude et l'étude de l'éxistant.

2.1 Flux d'informations et processus métier

Dans le cadre de tout projet, la compréhension des flux d'informations et des processus métier constitue une étape clé. Elle permet d'identifier les intéractions entre les différentes entités, les données échangées ainsi que les étapes opérationnelles à automatiser. Le processus métier que nous étudions ici l'approvisionnement et la gestion des pièces de rechange (PDR), qui sont essentielles à la disponibilité des équipements et a continuité de la production.

2.1.1 Gestion Amont : Approvisionnement et Stockage

En amont, avant le processus de production, la Supply Chain joue un rôle crucial dans la planification et l'approvisionnement des pieces de rechange. Ainsi La gestion des stocks est essentielle pour assurer une production fluide et continue et doit donc prendre en compte la capacité de stockage qui est limitée, l'optimisation de l'espace disponible devient nécessaire.

La gestion des stock repose sur le suivi des niveaux de stock :

- **Niveau de sécurité** : Quantité minimale nécessaire pour éviter toute rupture.
- **Limite maximale et minimale** : Définition des seuils optimaux de stockage.
- **Couverture de stock** : Capacité de l'entreprise à fonctionner sans réapprovisionnement en fonction de la consommation.

La production repose sur deux flux :

- **Production continue** : Flux constant nécessitant une gestion rigoureuse des stocks.

— **Production par campagne** : Tous les matériaux nécessaires doivent être disponibles avant le lancement.

2.1.2 Gestion de la Production

Pendant la production, l'ordonnancement permet d'atteindre les volumes prévus grâce à une planification efficace. Cela implique :

— **Allocation d'un budget annuel** dédié à la production.

— **Conversion du budget en quantités à produire**, réparties par mois.

— **Suivi hebdomadaire et journalier**, avec ajustements en fonction des besoins du marché et de la demande.

La planification est directement influencée par le marché et la demande des clients. Cela conduit à la gestion du **MRP (Material Requirements Planning)**, aussi appelé en français **CBN (Calcul des Besoins Nets)**, permettant d'identifier les produits à acheter et d'établir le **Plan Directeur d'Approvisionnement (PDA)**.

2.1.3 Gestion en Aval : Distribution et Livraison

Une fois les produits fabriqués, la Supply Chain gère la logistique de leur distribution. Lorsque le client passe une commande, l'entreprise doit :

— **Acheminer les produits** vers le point de distribution le plus proche.

— **Optimiser la chaîne de distribution** pour garantir rapidité et efficacité.

2.1.4 Avant-Projet d'Achat

Le processus d'achat démarre dès qu'un employé soumet une demande :

1. **Vérification des stocks** :
   — **Si le produit est disponible**, il est expédié immédiatement.
   — **Si le produit est indisponible**, une expression des besoins est réalisée sous forme d'un cahier des charges.
2. **Prospection des fournisseurs** : Un appel d'offres est lancé pour identifier les meilleurs fournisseurs.
3. **Commande et contrat** : Une fois le fournisseur sélectionné, une commande d'achat est passée et un contrat est établi pour formaliser l'engagement.

2.2 Etude de l'existant

Dans le cadre de notre étude préalable, nous avons recueilli des informations auprès de l'entreprise d'accueil concernant les outils utilisés pour la gestion du processus d'achat. Deux solutions logicielles sont à retenir : Sage X3 et SAP. Afin de mieux comprendre leurs spécificités et d'évaluer leurs avantages respectifs, nous avons réalisé un tableau comparatif basé sur plusieurs critères, comme le montre la figure 3.1 ci dessous :

Logiciels	**Sage X3**	**SAP**
Type de déploiement	on-premise	Cloud
Personnalisation	Élevée	Difficile
Facilité d'utilisation	Moyenne	Bonne
Intégration avec d'autres systèmes	Élevée	Élevée

Fonctionnalités clés	Gestion des stocks, flux logistiques et des ventes et de la comptabilité	Création et gestion des bons de commande, des contrats
Mise à jour et maintenance	Gérée par Sage pour Cloud, par l'entreprise pour on-premise	Gérée par SAP
Modèle de tarification	Payant	Payant

TABLE 3.1: Tableau comparatif des solutions logicielles

2.3 Délimitation du champ d'étude

Le champ d'étude du projet est limité à la Direction Métier, plus précisément au service Supply Chain, Achat et Approvisionnement de Cevital. L'objectif est d'optimiser le processus d'achat en améliorant la gestion des flux internes liés à l'approvisionnement et aux commandes des pièces de rechange pour les machines.

Le projet se concentre sur certaines activités internes de la Supply Chain, notamment :

- **La gestion des stocks :** permet d'assurer la disponibilité des pieces de rechange.
- **La gestion des demandes d'achat :** Automatisation du processus de soumission, validation et suivi des demandes.
- **L'amélioration de la traçabilité des achats :** Suivi des commandes et des niveaux de stock pour garantir une disponibilité optimale des ressources.
- **La gestion des fournisseurs :** Mise en place d'une gestion efficace des fournisseurs.
- **L'intégration de l'intelligence artificielle :** Automatisation de l'identification des références des pièces de rechange pour faciliter la gestion des approvisionnements.

Exclusions du périmètre

Le projet n'inclut pas :

- La gestion comptable et financière des achats.
- Les processus logistiques externes (livraison, entreposage hors site).
- Les interactions avec des fournisseurs externes hors du réseau validé de Cevital.
- La Planification et Ordonnancement de la Production sera aussi exclu.

Cette limitation au service Supply Chain garantit que le projet reste aligné avec les besoins métiers spécifiques, en apportant des améliorations ciblées sur l'efficacité des achats, la réduction des délais et l'optimisation des ressources.

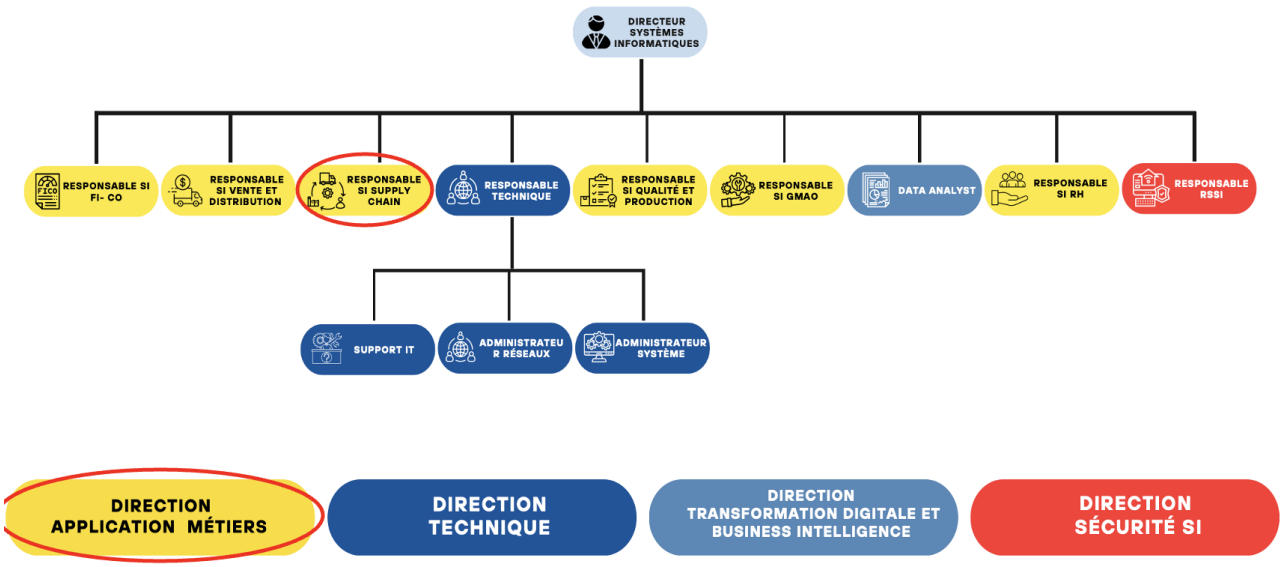

FIGURE 3.1 – Délimitation du champ d'étude

2.4 Problématique

Dans un environnement industriel aussi exigeant que celui de Cevital, le processus d'achat constitue un maillon essentiel de la chaîne d'approvisionnement. Il joue un rôle crucial dans la garantie de la disponibilité des ressources, notamment des pièces de rechange, afin de garantir la continuité de la production. Cependant, ce processus est confronté à plusieurs défis majeurs.

D'une part, la gestion d'environ douze mille références de pièces de rechange complique fortement l'identification rapide et fiable des éléments requis. Cette difficulté notamment lors des interventions de maintenance, car les méthodes traditionnelles de recherche, reposant principalement sur des filtres rigides ou la saisie de références exactes, se révèlent souvent inefficaces lorsqu'un utilisateur ne dispose que d'une description partielle ou imprécise.

D'autre part, le processus d'achat implique une collaboration avec un vaste réseau de fournisseurs, tant nationaux qu'internationaux. Cette diversité accroît la complexité des références des disponibles, chaque fournisseur ayant ses propres normes, délais de livraison et conditions contractuelles. Cette hétérogénéité rend difficile la comparaison objective des performances fournisseurs et complique la sélection des partenaires les plus fiables.

Enfin, la validation des différentes demandes de ce processus, qu'il s'agisse des demandes d'achat, de réservation ou des commandes, repose sur un circuit de validation hiérarchique mobilisant plusieurs services et étapes successives d'approbation. Ce fonctionnement rigide entraîne souvent des délais de traitement importants.

La problématique à laquelle ce travail cherche à répondre est donc la suivante :

Comment centraliser et automatiser le processus d'achat au sein de Cevital, en optimisant la sélection des fournisseurs, en fluidifiant les circuits de validation, et en s'appuyant sur des technologies d'intelligence artificielle pour faciliter l'identification des pièces ?

2.5 Objectif

L'objectif de ce travail est de concevoir une solution permettant d'automatiser et de centraliser le processus d'achat, tout en améliorant son efficacité, ainsi que la gestion des ressources, des fournisseurs et l'identification des piéces de rechange.

Plus précisément, cette solution s'articule autour des sous-objectifs suivants :

— **Réduire les délais de traitement** des demandes d'achat, de réservation et des commandes, en automatisant et fluidifiant les circuits de validation.

— **Améliorer la gestion des fournisseurs** à travers des indicateurs de performance mesurables, afin d'identifier les partenaires les plus fiables et de faciliter la prise de décision.

— **Assurer la disponibilité des pièces de rechange**en optimisant les approvisionnements et en anticipant les ruptures de stock grâce à des déclenchements automatiques d'alertes ou de demandes d'achat dès qu'un seuil critique est atteint.

— **Automatiser l'identification des pièces**, à l'aide d'une intelligence artificielle capable d'assister les utilisateurs dans la recherche des références, et d'accélérer la création des demandes.

2.6 Solution

La solution proposée repose sur la mise en place d'un logiciel de gestion intégré, conçu pour centraliser et automatiser l'ensemble du processus d'achat. Elle vise à fluidifier les opérations, et améliorer prise de décision. Elle s'articule autour des fonctionnalités suivantes :

— **Gestion des stocks :** L'application assurera un suivi en temps réel des pièces de rechange, intégrant des seuils minimum et maximum. Des alertes seront générées en cas de stock critique, et des demandes d'approvisionnement pourront être déclenchées automatiquement pour éviter les ruptures.

— **Automatisation du processus d'achat :**de la demande jusqu'à la commande, en passant par les sorties et les réceptions, l'ensemble du cycle d'achat sera automatisé à travers des circuits de validation fluides, un systéme de notifications et un suivi réel des états.

— **Gestion des fournisseurs :** l'application regroupera toutes les informations relatives aux fournisseurs tels que l'historique des commandes et les indicateurs de performance. L'évaluation se fera basée sur des critères spécifiques comme le délai, la qualité, et le coût, afin de faciliter la sélection des partenaires les plus performants.

— **Identification intelligente des pièces :** Un module d'intelligence artificielle sera intégré pour analyser les descriptions textuelles saisies par les utilisateurs et proposer les piéces qui correspondent le mieux aux textes fournis.

Pour structurer efficacement l'implémentation de notre solution, nous avons modélisé un flux retraçant les différentes étapes du processus d'achat. Celui-ci se déroule comme suit :

1. **Expression du besoin :** tout commence par l'identification d'un besoin en pièces, formulé par un employé (souvent un agent de maintenance) via une demande de réservation.
   — Si le produit est disponible en stock, il est immédiatement remis à l'employé.
   — Dans le cas contraire, une demande d'achat est générée.

   Si la référence exacte de la pièce est inconnue, une recherche peut être effectuée via le chatbot. Celui-ci analyse la description textuelle saisie par l'utilisateur afin de proposer les pièces les plus pertinentes.
2. **Sélection des fournisseurs** : après validation de la demande, une analyse du marché est menée pour identifier les fournisseurs potentiels.
3. **Négociation et contractualisation** : le fournisseur retenu est sélectionné, et un contrat peut être établi pour formaliser les engagements (prix, délais, conditions).
4. **Passation de commande** : une commande d'achat est émise basé sur la demande d'achat .

5. **Réception et contrôle** : les produits sont réceptionnés puis contrôlés afin de vérifier leur conformité (quantité, qualité, spécifications techniques).
6. **Mise en stock** : les pièces conformes sont enregistrées dans le système et stockées en tenant compte de la capacité des magasins et des exigences de traçabilité.

Voici le schéma correspondant au processus :

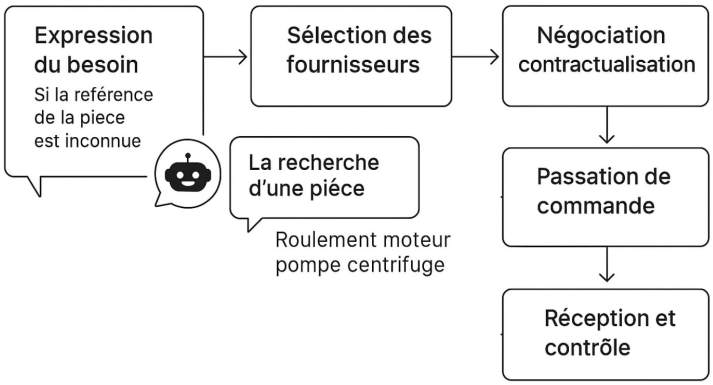

FIGURE 3.2 – schéma du processus d'achat

3 Analyse des besoins

L'analyse des besoins constitue une étape fondamentale dans tout projet informatique, car elle permet de poser les bases solides d'une solution pertinente, cohérente et adaptée aux attentes des utilisateurs finaux. Dans le cadre du présent projet, il s'agit de concevoir un système de gestion du processus d'achat pour le groupe Cevital, intégrant des fonctionnalités d'intelligence artificielle afin d'optimiser les prises de décision, de fluidifier les procédures internes et d'assurer un meilleur suivi des approvisionnements.

3.1 Les besoins fonctionnels

Les besoins fonctionnels décrivent ce qu'un produit ou service est censé faire, c'est-à-dire ses fonctions principales [26]. Voici un aperçu des fonctionnalités clés de notre application :

1. **Gestion du Processus d'Achat**
   - **Demande de réservation**
     - Permettre a un utilisateur de soumettre une demande de réservation sur une ou plusieurs pièces.
     - Permettre la consultation des demandes de réservation avec possibilités de filtrage par statut : en attente, validée, rejetée.
     - Permettre le suivi de l'évolution de l'état de la demande : en attente, validée, rejetée.
     - Mise en place d'un processus de validation à deux niveaux des demandes de réservation.
     - Envoi de notifications au parties prenantes à chaque étape de validation.
     - Enregistrer les sorties suite à une demande de réservation validée.
   - **Demande d'achats**

- Générer des demandes d'achats à partir de demande de réservation validée.
- Permettre aux employés de soumettre une demande d'achat.
- Consulter la liste des demandes d'achats et filtrer par statut : en attente, validée, rejetée.
- Mise en place d'un processus de validation hiérarchisé selon le type de la demande (urgente ou normale).
- Permettre le suivi de l'état de validation : en attente, validée, rejetée.
- Envoi de notifications au parties prenantes à chaque étape de validation.
- Déclencher automatiquement des demandes d'achat en fonction des seuils de stock et des fréquences d'utilisation.

- **Commande d'achats**
  - Permettre à un utilisateur de soumettre une commande d'achat suite à une demande d'achat validée.
  - Mise en place d'un processus de validation hiérarchisé selon type de la commande (urgente ou normale).
  - Permettre le suivi des états des commandes :en cours, livrée, annulée.
  - Consulter la liste des commandes avec possibilités de filtrage : en attente, validée, rejetée.
  - Vérifier la conformité des commandes en termes de piéces et quantités commandées.
  - Enregistrer les arrivages suite à la reception d'une commande.

2. **Gestion des stock**
   - Gérer les pièces de rechange (ajout, modification, suppression) ainsi que les types, familles et groupes associés.
   - Gérer les magasins (ajout, modification, suppression).
   - Suivre les niveaux de stock en temps réel.
   - Consulter la liste des pièces disponibles.
   - Gérer les sorties de stock et générer automatiquement les bons de sortie à partir d'une demande de réservation validée.
   - Gérer les entrées de stock et générer les bons d'entrée à partir d'une commande d'achat livrée.
   - Recevoir des notifications en cas de dépassement ou d'atteinte des seuils critiques de stock.
3. **Gestion des fournisseurs et contrats**
   - Gérer les fournisseurs : ajout, modification, suppression, et affectation des commandes.
   - Visualiser les performances d'un fournisseur et accéder à l'historique des évaluations.
   - Gérer les contrats d'achat : ajout, modification, suppression.
   - Vérifier la conformité des contrats par rapport aux engagements définis.
   - Être notifié de l'avancement des vérifications contractuelles.
4. **Gestion des Pièces de Rechange avec Integration de l'IA**
   - Utilisation de l'IA pour identifier les pièces et retourner leurs référence à partir d'une description fournie par l'utilisateur.
   - Développement d'un moteur de recherche sémantique permettant aux utilisateurs de décrire les pièces en langage naturel afin d'affiner les résultats.

3.2 Les besoins non fonctionnels

Un besoin non fonctionnel ne décrit pas directement l'activité ou la fonctionnalité du produit, mais plutôt comment cette fonctionnalité devrait être [27]. Autrement dit ils sont des indicateurs de qualité de l'exécution des besoins fonctionnels.

1. **Performance :**
   - Le système doit pouvoir traiter plusieurs requêtes simultanées sans latence notable.
2. **Utilisabilité et convivialité :**
   - L'interface doit être intuitive et optimisée pour une bonne expérience utilisateur
3. **Maintenance et Évolutivité :**
   - Le code doit suivre les bonnes pratiques de développement pour faciliter la maintenance.
   - L'architecture doit permettre l'ajout facile de nouvelles fonctionnalités sans impacter les performances.
4. **Compatibilité :**
   - L'application doit être accessible via navigateur web (Chrome, Firefox, Edge) et compatible aux différents systemes d'exploitation

3.3 Identifications des acteurs

Un acteur représente un rôle joué par un utilisateur qui interagit avec le système modélisé. Il peut s'agir d'une personne, d'une organisation, d'une machine ou encore d'un système externe [28]. Le tableau 3.2 ci-dessous présente les principaux acteurs ainsi que les cas d'utilisation associés dans notre application.

Acteur	**Rôle**
Administrateur	— S'authentifier — Gérer les utilisateurs (ajout, modification, suppression) — Gérer les groupes et acces
Maintenancier	— S'authentifier — Soumettre une demande de réservation — Consulter ses demandes et voir leur état — Interagir avec le chatbot
Magasinier	— S'authentifier — Soumettre une demande d'achat — Valider une demande de réservation — Suivre l'état de validation de ses demandes — Gérer les PDR et magasins — Suivre le niveau de stock — Suivi des sortie et reception des Arrivages
Responsable Approvisionnement	— S'authentifier — Soumettre une demande d'achat — Valider une demande de réservation et achat — Valider une commande d'achat — Gérer les PDR et magasins — Suivre le niveau de stock — Consulter la liste des magasiniers et suivi des actions
Responsable Achat	— S'authentifier — Soumettre une demande d'achat — Valider une demande et commande d'achat — Consulter la liste des acheteurs et suivi des actions
Directeur Général	— S'authentifier — Soumettre une demande et commande d'achat — Valider une demande et commande d'achat — Suivre le niveau de stock et les coûts — Consulter les listes demandes et commandes
Responsable Contrats	— S'authentifier — Consulter les contrats et vérifier leurs conformités.
Acheteur	— S'authentifier — Soumettre une commande d'achat — Gérer les fournisseurs (ajout, modification, suppression, affectation de commandes) — Gérer les contrats (ajout, modification, suppression) — Suivi du statut des contrats

TABLE 3.2 – Représentation des acteurs et cas d'utilisations

3.4 Diagramme de contexte

Le diagramme de contexte permet de donner une vue d'ensemble simple et claire du système et de ses interactions avec les acteurs externes : utilisateurs, autres systèmes, bases de données, etc. Il définit les principales entrées et sorties du système sans entrer dans les détails internes de son fonctionnement. Ce type de diagramme aide à comprendre rapidement les limites du système et ses points de contact avec d'autres systèmes ou acteurs.[29]

La figure ci-dessous représente le diagramme de contexte du projet, mettant en évidence les entités externes interagissant avec le système et les flux d'informations échangés.

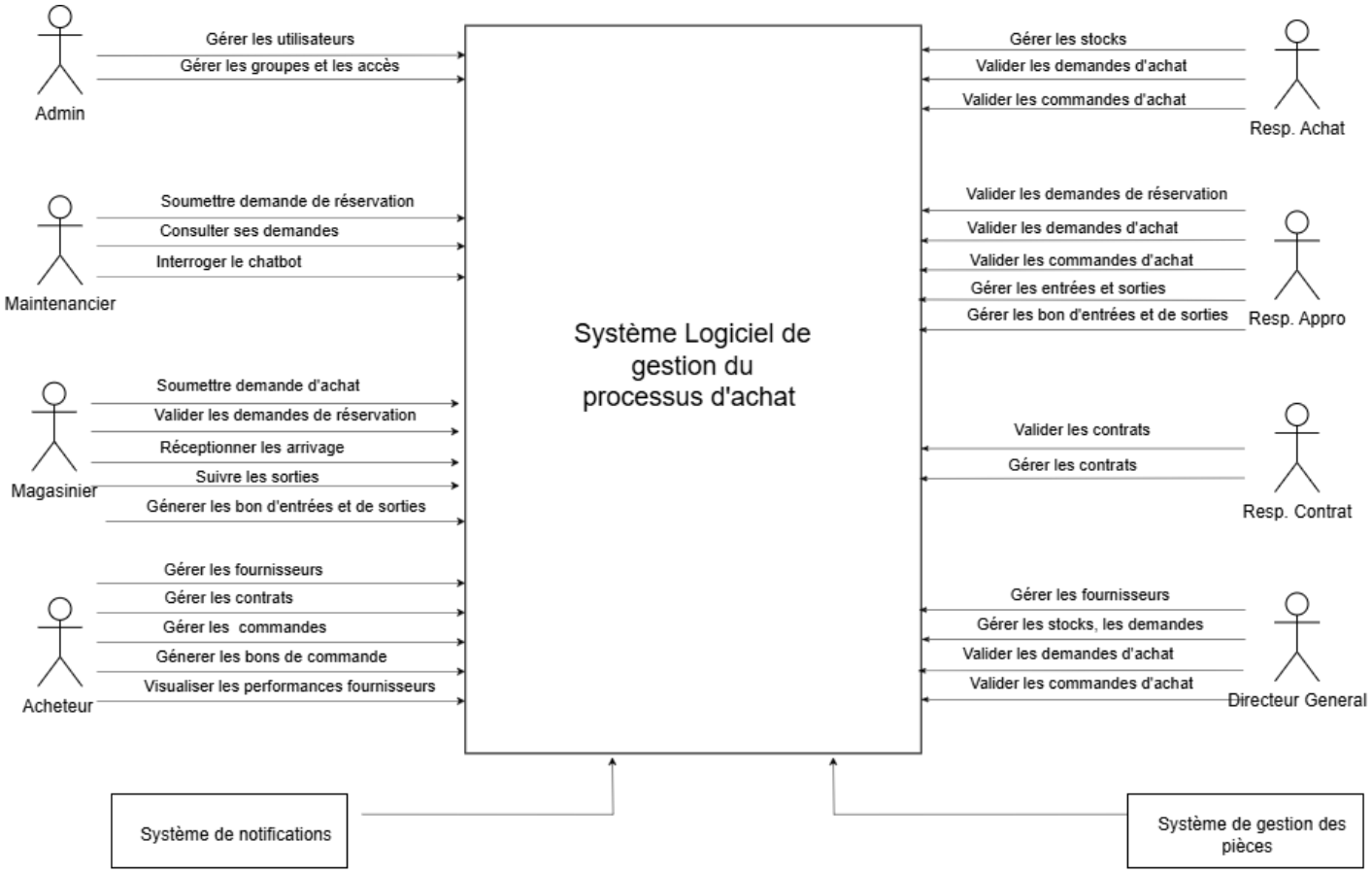

FIGURE 3.3 – Diagramme de contexte

4 Conclusion

En résumé, cette phase d'étude préalable et d'analyse des besoins, menée selon une approche en cascade, a permis d'identifier clairement les exigences fonctionnelles et techniques du système. Elle a posé les bases solides pour la suite du projet. Elle a également permis de structurer les processus métier, d'analyser les flux d'information et de spécifier les acteurs, leurs rôles ainsi que les interactions attendues avec le système. Cette analyse exhaustive servira de fondement pour le chapitre suivant, « Conception et mise en œuvre », qui adoptera cette fois une méthodologie agile, favorisant l'adaptabilité, les itérations courtes et l'implication continue des utilisateurs.

Chapitre 4

Conception, développement et mise en œuvre

1 Introduction

Dans ce chapitre, nous développons notre projet en suivant la méthodologie Scrum. La première étape de ce processus est souvent appelée "Sprint 0", une phase de planification au cours de laquelle nous avons constitué l'équipe, défini les user stories et élaboré le backlog produit. Ensuite, nous avons découpé le projet en plusieurs sprints, chacun étant consacré à l'implémentation de fonctionnalités spécifiques. Grâce à cette approche itérative et incrémentale, nous avons pu structurer notre progression de manière agile et adaptative, garantissant ainsi une meilleure gestion du projet et une optimisation continue du processus de développement.

2 Sprint 0 : Initialisation du projet

2.1 Identification des rôles de l'équipe

La méthode Scrum, est une approche qui se repose sur une répartition des responsabilités au sein de l'équipe afin d'assurer une coordination efficace et une bonne gestion du projet.Le tableau ci-dessous présente les rôles définis ainsi que les personnes assignées à chaque fonction :

Rôle	**Personne (s) assignée (s)**
Product Owner	Représentant de Cevital
Scrum Master	Mme Bouadem
Équipe de Développement	Mlle Ait Ahmed Meriem Mlle Amrouche Melissa

Table 4.1 – Tableau de la répartition des rôles Scrum

2.2 Liste des User Stories

Afin de garantir que le système réponde aux attentes des utilisateurs et aux besoins métier identifiés, nous avons défini une série de User Stories en attribuant une priorité à chacune d'elles. Chaque User Story exprime une fonctionnalité du point de vue de l'utilisateur final, en précisant son objectif. Ces récits utilisateur permettent d'assurer une conception centrée sur l'expérience utilisateur et facilitent la priorisation des développements. Voici une liste de User Storiespriorisées basées sur les besoins mentionnés par le client :

ID	En tant que	Je veux	Afin de	Priorité
US01	Admin	Gérer les utilisateurs, groupes et acces	Créer, modifier ou désactiver des comptes selon les besoins	Haute
US02	Resp. Appro / Magasinier	Gérer les PDR et les stocks	Maintenir à jour la BDD des pieces ainsi que les quantités disponibles	Haute
US03	Resp. Appro / Magasinier	Suivre le niveau des stocks et être alerté des seuils	Réagir rapidement aux baisses de stock	Moyenne
US04	Maintenancier	Rechercher une pièce de rechange en interrogeant le chatbot.	Trouver rapidement la pièce nécessaire	Haute
US05	Maintenancier	Obtenir une référence qui correspond à la piéce	Optimiser le processus d'achat.	Haute
US06	Maintenancier	Soumettre une demande de réservation	Réserver des pièces pour un besoin	Haute
US07	Maintenancier	Consulter ses demandes de réservation	Suivre ses demandes précédentes	Moyenne
US08	Maintenancier	Suivre l'état de validation de ses demandes	Être informé des validations	Moyenne
US09	Utilisateur	Consulter son tableau de bord	Avoir une vue d'ensemble sur ses actions	Haute
US10	Utilisateur	Soumettre une demande d'achat	Obtenir les pièces nécessaires à son besoin	Haute
US11	Utilisateur	Consulter ses demandes d'achat	Suivre l'avancement des demandes	Moyenne
US12	Utilisateur	Suivre l'état de validation de ses demandes d'achat	Être informé du statut à chaque étape	Moyenne
US13	Resp. Approvisionnement	Évaluer les demandes et commandes	Optimiser les achats et les stocks	Haute
US14	Resp. Achat / Directeur Général	Évaluer les commandes et demandes d'achat	Garantir le respect des procédures	Haute
US15	Magasinier	Évaluer les demandes de réservation	Organiser la sortie des pièces	Haute
US16	Resp. Achat / Acheteur	Consulter l'historique des achats	Éviter les redondances et optimiser les coûts	Moyenne
US17	Acheteur	Passer une commande d'achat	Réaliser l'achat d'un produit	Haute
US18	Acheteur	Générer les bons de commande	Automatiser le processus d'achat	Moyenne
US19	Acheteur	Gérer les fournisseurs et l'affectation des commandes	Répartir la charge d'achat équitablement	Haute
US20	Acheteur	Visualiser les performances des fournisseurs	Évaluer les fournisseurs	Haute
US21	Acheteur	Gérer les contrats et suivre leur statut	Vérifier l'avancement des contrats	Haute
US22	Responsable Contrat	Consulter les contrats et verifier leurs conformités	Suivre la conformité et les engagements contractuels	Haute
US23	Directeur Général	Suivre les actions des utilisateurs	Avoir une traçabilité complète	Haute
US24	Magasinier	Gérer les sorties de stock et les bons de sortie	Suivre les pièces remises aux utilisateurs	Haute

ID	En tant que	Je veux	Afin de	Priorité
US25	Magasinier	Gérer les entrées de stock et les bons d'entrée	Enregistrer les arrivages et mettre à jour le stock	Haute
US26	Utilisateur	Recevoir des notifications de suivi	Suivre l'avancement de ses demandes	Moyenne

TABLE 4.2: Liste des User Stories du projet

2.3 Rédaction du Product Backlog

Chaque User Story se traduit en une ou plusieurs fonctionnalités concrètes qui seront développées prochainement. Ces fonctionnalités définissent précisément les actions que l'application doit permettre aux utilisateurs d'effectuer afin de répondre aux besoins exprimés. En associant chaque User Story à des fonctionnalités correspondantes, nous facilitons la planification du développement.

ID Fonctionnalité	Description	User Stories associées	Sprint associé
F1	Gestion des stocks (ajout, retrait, mise à jour, suivi de niveau, alertes seuil)	US02, US03	2
F2	Gestion des utilisateurs et de leurs permissions	US01	2
F3	Recherche intelligente de pièces de rechange	US04, US05	1
F4	Gestion des demandes de réservation (soumission, suivi, consultation)	US06, US07, US08, US15	3
F5	Évaluation des demandes de réservation selon un workflow	US13, US15	3
F6	Gestion des demandes d'achat (soumission, suivi, consultation)	US10, US11, US12	3
F7	Évaluation des demandes d'achat selon un circuit de validation	US13, US14	3
F8	Historique et consultation des achats passés	US16	4
F9	Gestion des commandes d'achat	US17	4
F10	Génération automatique des bons de commande	US18	4
F11	Gestion des fournisseurs et affectation équitable des commandes	US19	4
F12	Visualisation et évaluation des performances des fournisseurs	US20	4
F13	Gestion des contrats (création, suivi, statut)	US21, US22	4
F14	Tableau de bord personnalisé pour chaque type d'utilisateur	US09	3,4

ID Fonctionnalité	Description	User Stories associées	Sprint associé
F15	Suivi global des actions utilisateurs par la direction	US23	3
F16	Gestion des bons d'entrée (réception, mise à jour stock)	US24	4
F17	Gestion des bons de sortie (remise de pièces, mise à jour stock)	US25	4
F18	Système de notifications	US26	3, 4

TABLE 4.3: Définition du Product Backlog fonctionnel

2.4 Technologies et outils utilisés

2.4.1 Technologies

Pour la réalisation de notre application web de gestion du processus d'achat intégrant de l'intelligence artificielle, nous avons opté pour un ensemble de technologies modernes, robustes et adaptées aux besoins fonctionnels et techniques du projet.

Langage principal : Python L'ensemble de la logique applicative est développé en **Python**, il s'agit d'un langage de haut niveau qui dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet.Il est idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines [30]. Il facilite également, l'intégration de bibliothèques tierces pour le traitement de données ou l'intelligence artificielle et il esttrès bien intégré avec l'écosystème Django.

Django est un framework web puissant, modulaire et open-source écrit en `Python`, qui suit le principe **MTV** (Model - Template - View), une variante du modèle MVC (Model - View - Controller).Il intègre également un ORM (Object-Relational Mapping), qui permet de manipuler avec les bases de données sans écrire de requêtes SQL. Ce framework permet aisni développer des applications rapidement, de manière modulaire, tout en assurant une séparation claire des responsabilités [31]. Parmi ses principales caractéristiques, on peut citer :

- **La Modularité** : Django repose sur une architecture en *applications* réutilisables et découplées, facilitant l'organisation du code, la maintenance et l'extension du projet.
- **La Sécurité et bonnes pratiques** : Django intègre nativement des protections contre les failles courantes telles que XSS, CSRF ou les injections SQL.
- **Les bonnes pratiques** : Django encourage une structure de code claire, propre et maintenable.

Parmi les composants fondamentaux du modèle MTV, on retrouve les **modèles** et les **vues**, qui jouent un rôle central dans la gestion des données et de la logique métier :

Les modèles et vues Django :

- Les **modèles** permettent de définir la structure des données via des classes Python, directement reliées à la base de données grâce à l'**ORM Django** (Object-Relational Mapping). Cela nous permet de manipuler les données comme des objets Python, sans écrire directement de requêtes SQL.

— Les **vues** gèrent la logique de traitement côté serveur. Elles reçoivent les requêtes utilisateur, interagissent avec les modèles si nécessaire, et renvoient une réponse (souvent sous forme de page HTML rendue à partir d'un *template*).

 **PostgreSQL** est un système de gestion de base de données relationnelle puissant, robuste et open source. Il prend en charge le langage **SQL** ainsi que de nombreuses fonctionnalités avancées telles que les transactions, les vues, les procédures stockées et les types personnalisés. PostgreSQL respecte les propriétés **ACID**, ce qui garantit la fiabilité, la cohérence et la sécurité des opérations sur les données [32]. Il est particulièrement adapté aux environnements de production, et Django offre une intégration native avec PostgreSQL, permettant une configuration fluide et des performances optimales.

 **Front-end : HTML, CSS et JavaScript** Pour la partie interface utilisateur :

— **HTML** (HyperText Markup Langage) est un langage de base standard utilisé dans la création des pages Web. Il fonctionne en utilisant des balises qui indiquent au navigateur comment structurer et afficher le contenu d'une page.[33]

— **CSS** (Cascading Style Sheets) est utilisé pour styliser et mettre en forme les pages Web créées avec HTML.il permet de créer des designs attrayants, visuellement cohérents et responsifs pour les pages Web, en séparant la présentation du contenu, ce qui facilite la maintenance et la personnalisation du style d'un site Web.[33]

— **JavaScript** est un langage de programmation de haut niveau basé sur un prototype simple, qui permet de mettre en place des mécanismes logiques et des calculs, et dynamiser les interactions côté client.[34]

 **Chart.js** est une bibliothèque JavaScript open source qui permet de créer et d'afficher des **graphiques interactifs** (courbes, camemberts, barres, etc.) à partir des données du système [35]. Cette fonctionnalité est particulièrement utile pour le suivi des indicateurs de performance (KPI).

2.4.2 Initialisation du projet Django

Avant de démarrer le développement, il est recommandé d'isoler l'environnement de travail pour éviter les conflits de dépendances. Pour cela, nous avons utilisé un **environnement virtuel** Python. Voici les étapes que nous avons suivies :

1. **Création d'un environnement virtuel :**

```
python -m venv env
source env/bin/activate   # Sur Linux/Mac
env\Scripts\activate.bat  # Sur Windows
```

2. **Installation de Django :**

```
pip install django
```

3. **Création du projet principal :**

```
django-admin startproject mon_projet
cd mon_projet
python manage.py startapp gestion_achat
```

La structure d'un projet Django : Une fois le projet initialisé, Django automatiquement une structure de dossiers et de fichiers. Celle-ci permet une organisation claire du code. Voici un de cette structure :

Après l'initialisation, Django génère automatiquement une structure de dossiers et fichier pour le projet. Elle permet une organisation claire. Voici un aperçu de cette structure :

```
mon_projet/
|-- manage.py
|-- mon_projet/
|    |-- __init__.py
|    |-- settings.py
|    |-- urls.py
|    |-- asgi.py
|    |-- wsgi.py
|-- store/
|    |-- migrations/
|    |-- __init__.py
|    |-- admin.py
|    |-- apps.py
|    |-- models.py
|    |-- tests.py
|    `-- views.py
```

— `manage.py` : est le script de gestion du projet (exécution du serveur, migrations, etc.).

— `settings.py` : est le fichier ou se trouve la configuration globale (base de données, applications installées, templates, etc.).

— `urls.py` : pour la gestion des routes de l'application.

— `models.py` : permet la définition des modèles (structure des données).

— `views.py` : ici se trouve toute la logique métier et le traitement des requêtes.

— `templates/` et `static/` : sont des dossiers pour contenir respictivement les fichiers HTML, CSS et JS.

Déclaration des applications Django : Django utilise le fichier `settings.py` et plus précisément la liste `INSTALLED_APPS` pour recenser les applications (créés ou installées)à charger. Cette configuration lui permet d'initialiser les composants nécessaires à chaque application comme les modèles et les fichiers.

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
```

```
    'widget_tweaks',

    # Applications locales
    'store.apps.StoreConfig',
    'users.apps.UsersConfig',
    'demandeAchat.apps.DemandeachatConfig',
    'bonSortie.apps.BonsortieConfig',
    'demandeReservation',
    'purchase_orders',
    'historical_performance',
    'arrivage',
    'notifications',
    'chatbot',
]
```

— Les applications natives de Django (`admin`, `auth`, etc.) permettent de gérer l'administration, les utilisateurs, les sessions, les messages, etc.

— L'application `widget_tweaks` est une bibliothèque externe qui permet de personnaliser facilement les formulaires HTML dans les templates.

— Les applications locales (`store`, `users`, `demandeAchat`, etc.) correspondent aux modules fonctionnels spécifiques de notre projet, tel que :

  — `store` : pour la gestion des piéces de rechange et du stock.

  — `users` : pour la gestion des utilisateurs, des rôles et des droits d'accés.

  — `demandeAchat`, `bonSortie`, `demandeReservation` : englobe la gestion des processus métier liés aux achats et sorties.

  — `purchase_orders`, `arrivage` : pour la gestion des commandes et des arrivages.

  — `historical_performance` : pour assurer la traçabilité et performance des fournisseurs.

  — `notifications` : contient le système d'alertes et de notifications.

  — `chatbot` : il s'agit du module optionnel pour l'intégration d'un assistant intelligent (actuellement désactivé).

La Configuration de la base de données :
Par défaut, Django utilise une base de données légère, idéale pour le développement rapide : Postgresql. Cette configuration peut être modifiée dans le fichier `settings.py`.

```
from dotenv import load_dotenv

load_dotenv()  # Charge les variables du .env

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.getenv('DB_NAME'),
        'USER': os.getenv('DB_USER'),
        'PASSWORD': os.getenv('DB_PASSWORD'),
        'HOST': os.getenv('DB_HOST'),
        'PORT': os.getenv('DB_PORT', '5432'),
    }
}
```

```
SECRET_KEY = os.getenv("SECRET_KEY")
```

Les Migrations et l'exécution : Une fois les modèles définis dans les applications, Django une étape de migration pour les tables en base de données. Ensuite, le serveur

Une fois les modèles définis dans les applications, Django nécessite une étape de migration pour pouvoir créer les tables en base de données. Ensuite, le serveur de développement peut être lancé. Cela se fait comme suit :

```
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
```

Ces lignes de code, permettent d'appliquer les migratiode et lancer le serveur local, accessible à l'adresse : `http://127.0.0.1:8000/`.

2.4.3 Outils

Le bon déroulement d'un projet logiciel repose autant sur les compétences techniques que sur l'utilisation d'outils adaptés. Dans le cadre de ce projet, plusieurs outils ont été utilisés pour organiser le travail en équipe, développer les fonctionnalités, versionner le code et documenter le tout de manière structurée et collaborative.

Jira est un outil de gestion de projet et de suivi des tâches développé par Atlassian. Il est particulièrement utilisé pour la gestion des projets Agile (Scrum, Kanban) et permet d'organiser les sprints, suivre les bugs, et gérer l'avancement des développements de manière collaborative. [36]

Miro est une plateforme en ligne de collaboration visuelle qui facilite le brainstorming, la planification et la conception en équipe. Grâce à ses tableaux blancs interactifs, il permet de structurer les idées, réaliser des diagrammes et organiser les workflows de manière dynamique, il offre divers fonctionnalités pour la collaboration telle que le chat vidéo. [37]

GitHub est une plateforme d'hébergement de code basée sur le cloud qui offre des outils et fonctionnalités pour la gestion de projets, la collaboration et le contrôle de versions.[38]

PlantUML : est un outil très polyvalent et open-source qui facilite la création d'un large éventail de diagrammes UML à partir de texte, en utilisant une langage simple et intuitif. Il s'intègre parfaitement dans les projets versionnés avec Git, ce qui facilite la collaboration, la traçabilité des modifications et l'automatisation de la documentation technique. [39]

LucidChart : est un outil en ligne basée sur le Cloud qui permet de créer des diagramme UML, il facilite la collaboration au sein des équipes grâce à des fonctionnalités telles que la co-création en temps réel [40].

PyCharm : environnement de développement intégré (IDE) puissant dédié à Python, avec un excellent support natif pour Django, les tests, le debug et la gestion des environnements virtuels.[41]

VisualStudio Code : est un environnement de développement intégré (IDE) complet et puissant,il offre une grande flexibilité et des fonctionnalités avancées pour les développeurs, telle que le débogage pour identifier et résoudre les erreurs, et la gestion de version intégrée pour faciliter le contrôle des modifications du code source. [42]

Postman : est une application qui permet d'interroger ou tester des API's, il offre une interface conviviale ainsi que des outils adaptés, aussi bien dans sa version de bureau qu'en ligne.[43]

2.5 Planification des Sprints

Dans le cadre de la méthodologie Scrum, le projet a été divisé en quatre sprints afin de favoriser une approche itérative et incrémentale. Chaque sprint a été planifié avec une priorisation des tâches à réaliser. Cette organisation a permis une gestion plus efficace et claire du projet en suivant une progression continue, facilitant l'adaptation aux éventuels ajustements et l'améliorations si nécessaire au cours du développement.

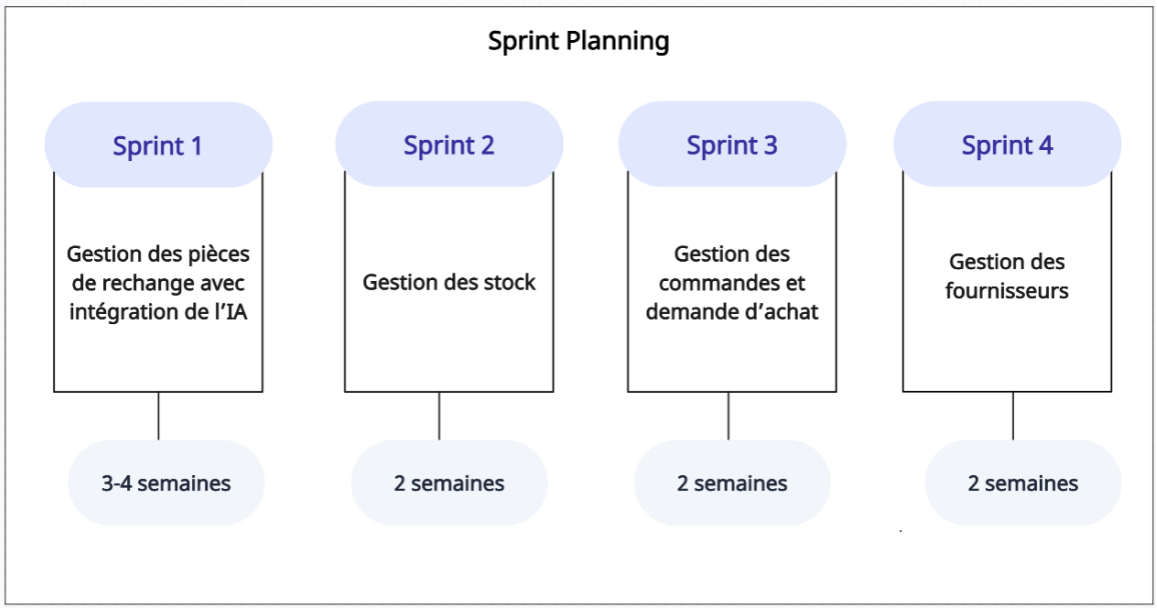

FIGURE 4.1 – Organisation du projet en Sprint

La figure 4.2 ci-dessous représente un diagramme de Gantt de plannification. Il montre la répartition des tâches sous forme de backlog, avec des sprints planifiés sur plusieurs mois. Chaque section correspond à un module spécifique du projet, permettant de visualiser l'avancemment des différentes fonctionnalités à développer

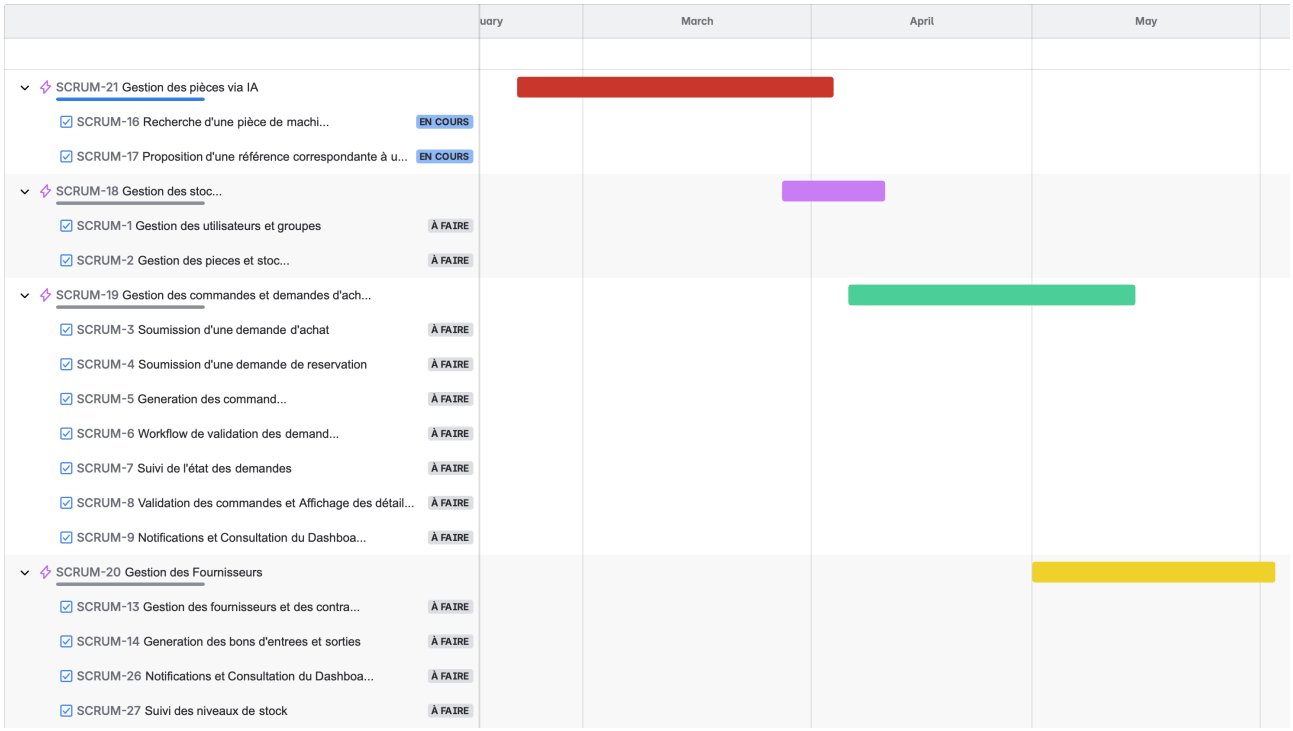

FIGURE 4.2 – Planification des Sprints

2.6 Identité visuelle

Dans le cadre du Sprint 0, nous avons amorcé la phase de conception visuelle du projet en définissant les éléments clés de son identité graphique. Cette étape préliminaire vise à assurer une cohérence visuelle tout au long du développement. Ainsi, nous avons procédé à la création du logo. Parallèlement, une palette de couleurs a été sélectionnée avec soin, en s'appuyant sur des principes d'ergonomie, d'accessibilité et d'esthétique. Ce socle graphique constitue la base du design UI/UX à venir.

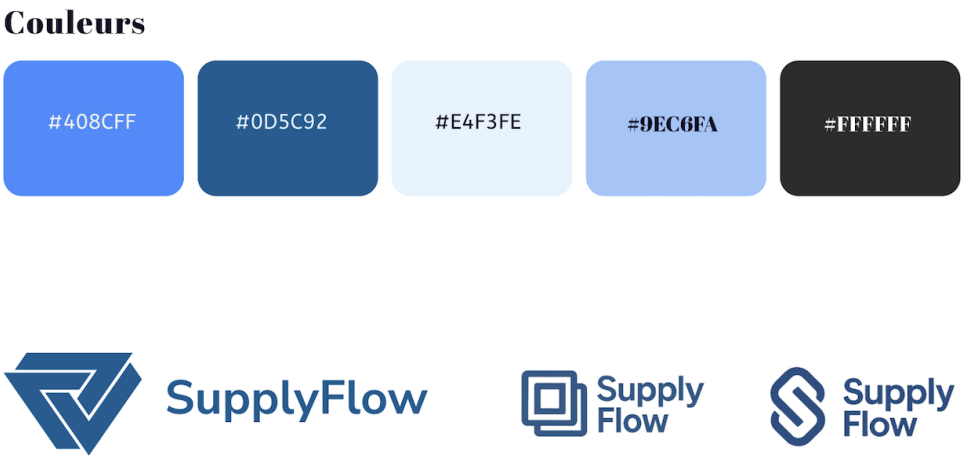

FIGURE 4.3 – Identité Visuelle

3 Déroulement des Sprints 1 à 4

Une fois le Sprint 0, consacré à la préparation et à la planification initiale du projet, achevé, nous avons entamé les quatre sprints qui ont structuré le développement de notre application. Chacun de ces sprint a été conçu de sorte à assurer une progression continue et une adaptation fluide aux besoins exprimés par les parties prenantes.

Pour chaque sprint, un objectif a été défini, accompagné d'un **Sprint Backlog** permettant de cadrer les fonctionnalités à développer. Des prototypes ont été réalisés pour illustrer les interfaces utilisateurs.Une partie « conception » regroupe les différents diagrammes UML, tandis qu'une autre section recense le développement effectué ainsi que les ajustements réalisés au fil de l'implémentation.

Cette démarche nous a permis de garder une vision claire et structurée de chaque étape du projet.

3.1 Sprint 1 : Recherche intelligente de pièces (IA)

3.1.1 Objectif :

L'objectif de ce Sprint 1 est la mise en place d'un chatbot conversationnel intelligent pour faciliter la recherche des pièces de rechange et réduire les erreurs humaines. L'utilisateur fournit une description d'une pièce, même partielle, et le système utilise des techniques d'intelligence artificielle pour lui proposer les pièces les plus pertinentes. Ce sprint vise à optimiser le temps de recherche dans la base de données.

Le tableau suivant 4.4 décrit le Sprint Backlog de cette première itération :

Fonctionnalités	**Description**	**User Stories associées**
S1-F01	Ajout des pièces dans les stocks	US02
S1-F02	Recherche intelligente de pièces de rechange	US04
S1-F03	Création, modification (renommage) et suppression d'un chat	US04
S1-F04	Accès à l'historique des discussions	US04
S1-F05	Obtenir une référence qui correspond à la description	US05

Table 4.4: Sprint Backlog du Sprint 1

3.1.2 Prototype

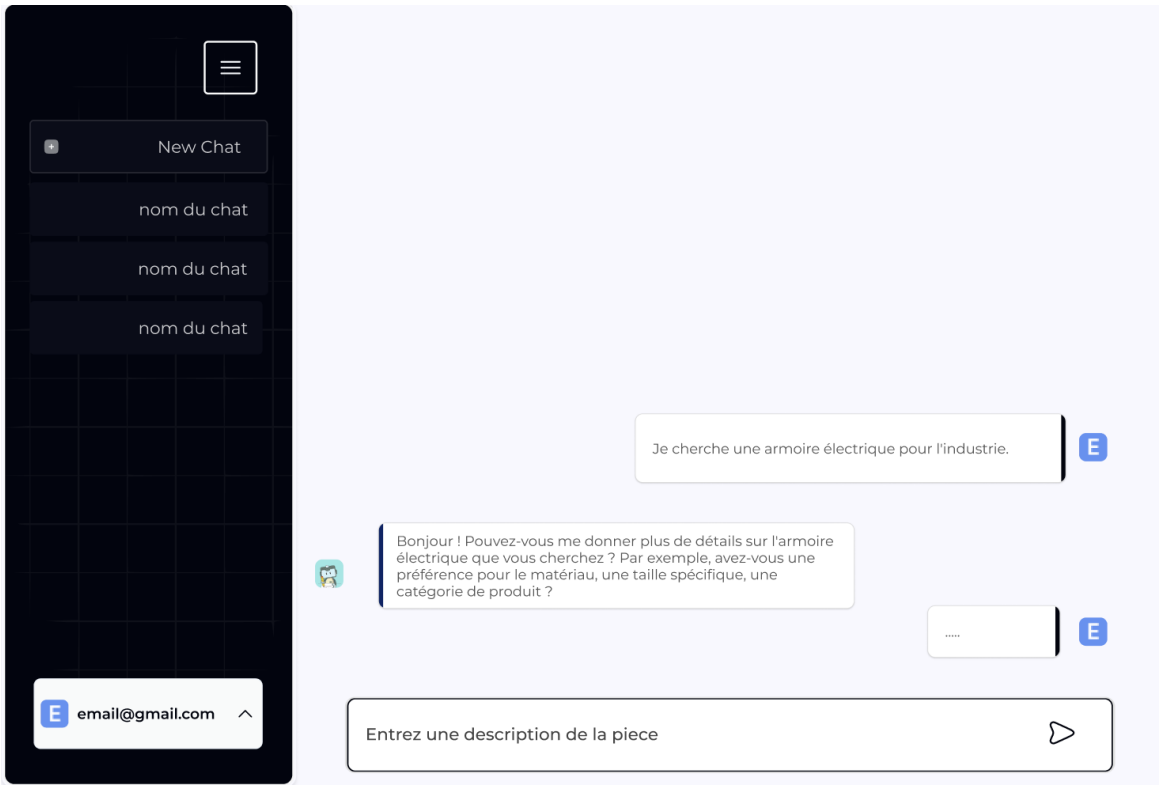

FIGURE 4.4 – Design du Chatbot

3.1.3 Conception

Les diagrammes UML présentés dans cette partie permettent de décrire les aspects fonctionnels et structurels développés pendant le Sprint 1.

1. **Diagramme de cas d'utilisation** La figure suivante 4.5 présente le diagramme de cas d'utilisation du sprint 1.

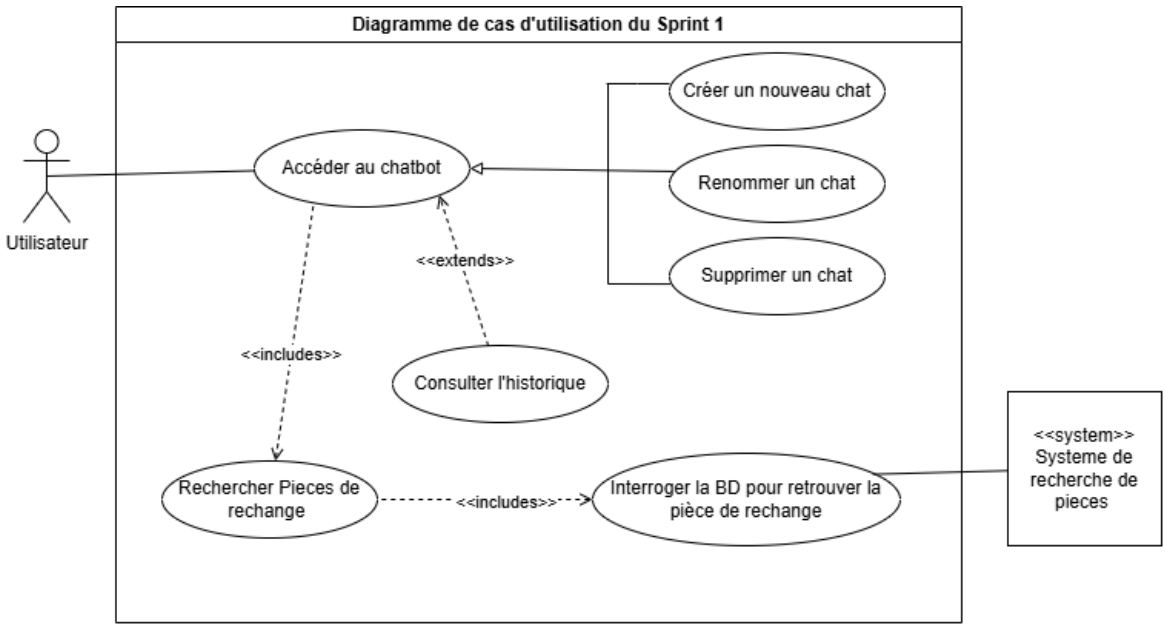

FIGURE 4.5 – Diagramme de cas d'utilisation du sprint 1

Le tableau 4.5 ci-dessous, montre la description textuelle faite pour ce premier Sprint.

Nom du cas : Recherche intelligente de pièces de rechange
But : Permettre à un maintenancier de retrouver rapidement une pièce de rechange en utilisant une description textuelle libre analysée par un système d'intelligence artificielle.
Acteur principal : Maintenancier
Acteur secondaire : Module d'intelligence artificielle.
Date de création : 05/04/2025
Date de mise à jour : 01/07/2025
Responsable : Èquipe technique de développement
Version : 1.0
Séquencement : Le cas d'utilisation commence lorsqu'un maintenancier souhaite retrouver une pièce en utilisant l'interface de recherche intelligente.
Préconditions : — Le maintenancier est connecté et le module IA de recherche est disponible et opérationnel.
Enchaînement nominal : (a) Le maintenancier accède à l'interface de recherche intelligente. (b) Il saisit une description textuelle de la pièce (forme, fonction, matériau, etc.). (c) Le système interprète la requête à l'aide d'un modèle OpenAI. (d) Le système recherche dans la base les références les plus proches. (e) Une ou plusieurs pièces compatibles sont proposées avec leur details. (f) Le maintenancier peut sélectionner une référence pour l'utiliser dans une demande.
Enchaînements alternatifs : **A1 : Aucune correspondance trouvée** — Le système informe l'utilisateur qu'aucune pièce ne correspond. — Il propose de reformuler ou d'élargir la description. — La séquence peut reprendre au point 2. **A2 : Plusieurs résultats similaires** — Le système affiche une liste classée avec des taux de correspondance. — L'utilisateur peut affiner la recherche ou en sélectionner une.
Enchaînements d'exception : **E1 : Erreur d'analyse** — Le système affiche un message d'erreur technique et la recherche est annulée. — L'utilisateur est invité à réessayer ou à contacter le support.
Postconditions : — Une ou plusieurs références ont été proposées au maintenancier. — Les résultats sont enregistrés dans l'historique de recherche. — Une référence peut être utilisée pour pré-remplir une demande de réservation.

TABLE 4.5 – Description textuelle du cas d'utilisation "Recherche intelligente de pièces de rechange"

2. **Diagramme de séquence** La figure 4.6 ci-dessous,illustre le diagramme de séquence du cas d'utilisation "Rechercher une piéce de rechange".

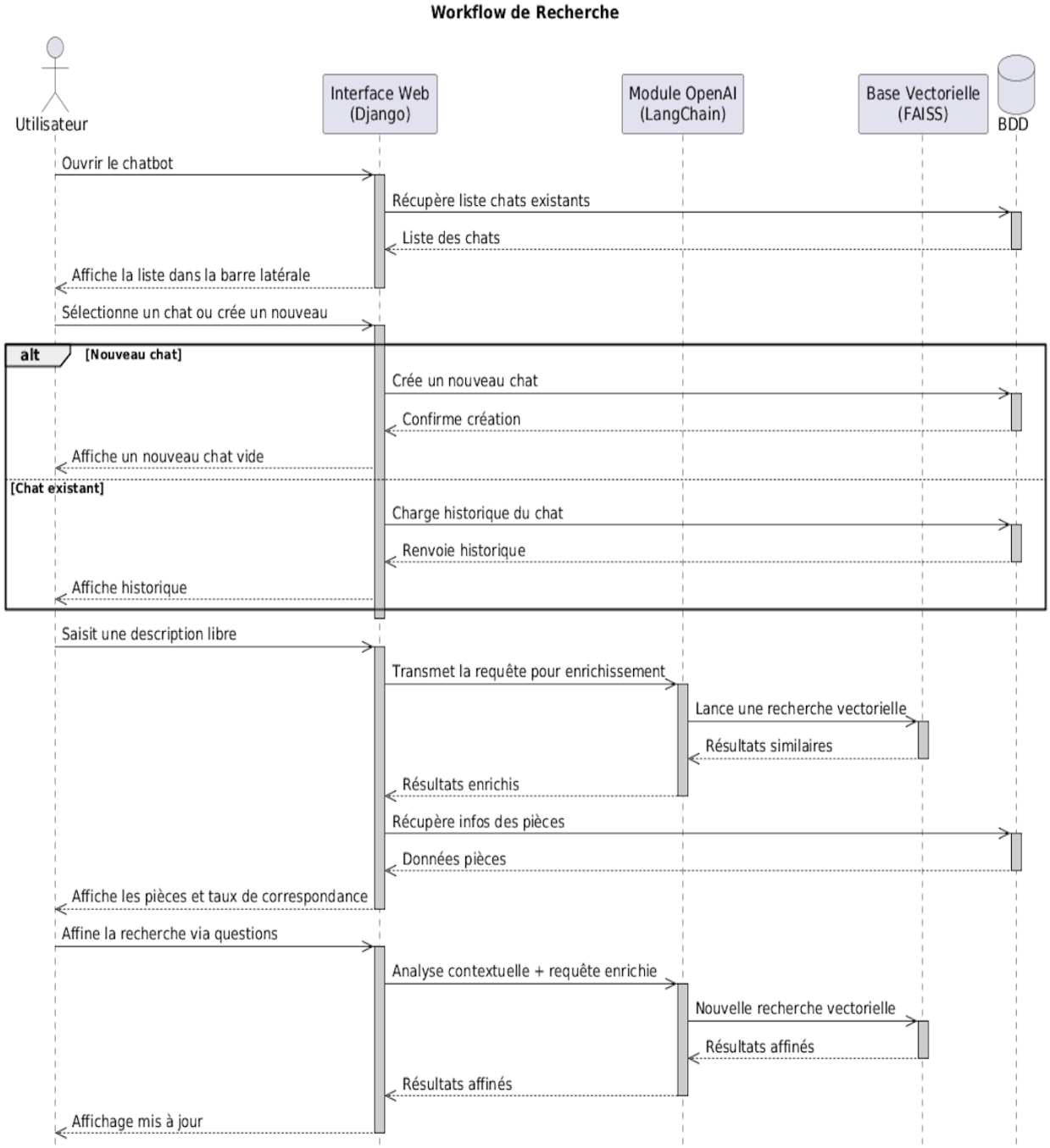

FIGURE 4.6 – Diagramme de séquence du cas d'utilisation "Rechercher piéce de rechange

3. **Diagramme de classe** La figure suivante 4.7, décrit la structure statique du système pour le Sprint 1.

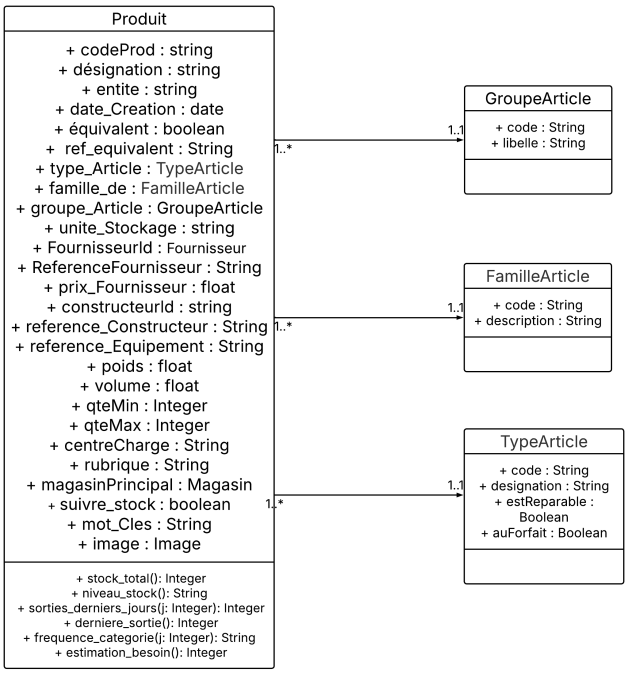

FIGURE 4.7 – Diagramme de classe du Sprint 1

3.1.4 Développement

Dans cette phase de développement, nous présentons les principales technologies et techniques utilisées pour réaliser ce premier sprint. Nous nous relayons en mode *driver/navigator* pour concevoir le chatbot. Ce binômage, réexaminé chaque jour lors du *Daily Scrum*, garantit une compréhension partagée du code, réduit les défauts dès l'écriture et accélère la résolution des obstacles.

1. Définitions préliminaires

Avant d'aborder les modules implémentés, il est important de définir certains concepts clés liés à l'intelligence artificielle, essentiels pour comprendre le fonctionnement du système de reconnaissance intelligente des pièces, que voici :

— **Large Language Model (LLM) :**
Un Large Language Model (LLM), ou modèle de langage de grande taille, est un type de programme d'intelligence artificielle basé sur l'apprentissage automatique, et plus précisément sur l'apprentissage profond, conçu pour comprendre, générer et manipuler du langage naturel. Ces modèles sont entraînés sur d'immenses volumes de textes provenant de livres, d'articles, de sites web ou d'autre source, d'où l'usage du terme « large ». Cela leur permet d'apprendre les structures, la grammaire, les concepts, et les relations sémantiques du langage humain. [44]
Les LLM récents reposent généralement sur l'architecture *Transformer*, qui leur permet de prendre en compte le contexte global d'une séquence de texte de manière efficace. Parmi les modèles les plus connus figurent **GPT** (OpenAI), **BERT** (Google) ou encore **LLaMA** (Meta).

— **Retrieval-Augmented Generation (RAG) :**
Le RAG est une technique d'intelligence artificielle qui vise à améliorer la qualité des

réponses générées par les grands modèles de langage (LLM), en leur permettant d'exploiter des ressources de données supplémentaires sans nécessiter de réentraînement. Cette approche combine deux étapes :la recherche d'information *(retrieval)* et la génération de texte *(generation)*. Elle commence par récupérer des documents pertinents, par exemple la description d'une pièce, puis utilisie un modèle génératif tel que GPT pour formuler une réponse contextualisée à partir de ces informations.[45]

— **Embeddings :**
Les embeddings sont des représentations numériques de mots, de phrases ou de documents sous forme de vecteurs dans un espace de dimension réduite. Le but est de capturer le sens d'un objet textuel en le traduisant en une suite de nombres, tels que les objets similaire aient des vecteurs similaires. cette technique permet par exemple de mesurer la similarité sémantique entre deux descriptions de pièce, ce qui est essentiel pour la recherche par similarité dans les bases vectorielles comme FAISS. [46]

— **Agent de type ReAct :**
Est un système d'IA qui combine deux capacités essentielles : *raisonnement* (Reasoning) et *action* (Acting) pour interagir de manière intelligente avec son environnement. Cette approche permet à l'agent de résoudre des problèmes complexes en alternant entre raisonnement et action, similaire à la façon dont les humains résolvent des problèmes.[47]
Cet agent est basé sur un *LLM* (comme `GPT`) qui :

- raisonne à voix haute pour comprendre un problème (*reasoning*) ;
- agit en choisissant et en appelant un outil ou une action (*acting*) ;
- répète ces étapes en boucle jusqu'à obtenir une réponse finale.

— **API (Application Programming Interface) :**
Une API est un ensemble de fonctions et de protocoles permettant à des logiciels de communiquer entre eux et échanger des données [48]. Par exemple, l'API OpenAI permet d'interagir avec les modèles GPT pour générer des réponses.

— **LangChain :**
est une bibliothèque Python conçue pour la création d'applications alimentées par les LLM, comme les chatbots et les agents conversationnels. Elle fournit des composants modulaires pour gérer les *prompts*, la mémoire, les chaînes d'exécution et l'intégration avec des outils externes (API, bases de données, navigateurs...).[49]

— **LangGraph :**
Crée par LangChain, c'est une extension qui permet de structurer les agents sous forme de **graphes d'états cycliques**. Chaque nœud du graphe peut exécuter une étape du raisonnement, permettant à l'agent de garder une mémoire et de revenir en arrière si besoin.[50]

2. Mise en contexte

L'integration de l'intelligence Artificielle se fera via un chatbot intelligent conçu pour aider les utilisateurs à trouver des pièces de rechange industrielles spécifiques à partir de descriptions en langage naturel.

Le backend est construit sous la forme d'une API performante, destinée à alimenter une interface web. Il s'appuie sur une pile technologique moderne afin d'offrir une authentification robuste, la persistance des historiques de conversation, et une expérience conversationnelle avancée basée sur une architecture RAG (Retrieval-Augmented Generation).

Au coeur du système se trouve un agent de type ReAct, développé avec LangChain et LangGraph, capable de raisonner sur la requête d'un utilisateur, d'utiliser des outils pour

interroger une base de données spécialisée en pièces détachées, et de générer des réponses précises et adaptées au contexte.

3. Objectif principal

Développer un point de terminaison API conversationnel intelligent capable de :

— Comprendre une description libre d'une pièce industrielle formulée par l'utilisateur.

— Gérer et conserver des conversations multi-tours pour chaque utilisateur.

— Utiliser une architecture RAG pour interroger une base de données vectorielle (FAISS) contenant des informations sur les pièces.

— Fournir des réponses en temps réel, diffusées en continu à l'utilisateur.

4. Pile technologique

Le backend repose sur les technologies clés suivantes :

— **Framework Backend : FastAPI**
Est un framework web moderne, basé sur Python, conçu pour créer des API rapides, robustes et faciles à maintenir [51]. Nous l'avons choisi pour ses hautes performances, son support asynchrone (essentiel pour le streaming), sa documentation automatique via OpenAPI, ainsi que la validation des données basée sur les annotations de types Python.

— **Base de données : Supabase**

Est une plateforme open-source qui fournit une base de données managée, utilisée pour stocker les données des utilisateurs et des conversations. Elle intègre également un système d'authentification sécurisé, permettant de gérer facilement les connexions, inscriptions et sessions.[52]

— **ORM / Modélisation des données : SQLModel**
Est une bibliothèque permettant d'interagir avec des bases de données SQL en utilisant du code Python orienté objet. Conçue pour être intuitive, facile à utiliser, compatible et robuste, elle est alimentée par Pydantic et SQLAlchemy. Elle permet de définir un modèle de données unifié, utilisé à la fois pour la création des tables et la validation des schémas de l'API. Cela réduit la duplication de code et facilite la maintenabilité. [53]

— **Authentification : Supabase Auth (JWT)**
Il permet de gèrer de manière sécurisée l'inscription, la connexion et les sessions des utilisateurs. Le backend valide les JWT sans état émis par Supabase, garantissant que les points de terminaison sont protégés et que les données sont bien associées à l'utilisateur authentifié.

— **Framework d'agent IA : LangChain & LangGraph**
LangChain fournit les composants de base pour interagir avec les modèles de langage (LLM) et les outils. LangGraph permet de construire un graphe cyclique avec état, donnant naissance à un agent ReAct robuste, capable d'utiliser des outils et de maintenir une mémoire.

— **Base de données vectorielle : FAISS**
Est une bibliothèque de recherche vectorielle haute performance, utilisée pour indexer et interroger les représentations vectorielles (embeddings) des pièces mécaniques. Elle permet de retrouver rapidement les éléments les plus proches d'une requête, en se basant sur la similarité sémantique dans un espace vectoriel dense. [54]

— **Fournisseur LLM : API OpenAI**
Il donne accès à des modèles de langage de pointe (par exemple, GPT-4o) qui alimentent les capacités de raisonnement, d'utilisation d'outils et de génération de réponses de l'agent.

5. Architecture du système

Le système est conçu selon une séparation claire des responsabilités, en suivant les principes modernes de conception d'API.

Le cheminement typique d'une requête dans le système se déroule comme suit :

1. **Requête HTTP** : Le client de l'utilisateur envoie une requête HTTP authentifiée vers un point de terminaison FastAPI (par exemple, `POST /api/chats/{id}/messages`).
2. **Authentification et validation** : La dépendance d'authentification intercepte la requête, valide le JWT et extrait l'`user_id`. FastAPI valide ensuite le corps de la requête en s'appuyant sur le schéma défini via Pydantic/SQLModel. La logique du point de terminaison récupère l'historique de conversation pertinent depuis la base de données.
3. **Invocation de l'agent** : La requête validée, accompagnée de l'historique de conversation, est transmise à l'agent RAG.
4. **Boucle de l'agent (ReAct)** :
   (4a) Utilisation d'outil : S'il doit rechercher des pièces, il appelle l'outil `search_vectorstore`, qui interroge l'index FAISS.
   (4b) Appel au LLM : Il envoie le contexte (requête originale, historique de chat, résultats d'outil) à l'API OpenAI pour raisonner ou générer une réponse.
   (4c) Réponse finale : Une fois la réponse formulée, il la prépare pour l'envoi.
5. **Réponse en streaming** : L'agent transmet la réponse en continu via le point de terminaison. Les jetons de réponse et les événements liés aux appels d'outils sont envoyés en temps réel au client en utilisant le protocole SSE (Server-Sent Events). Les messages finaux sont ensuite enregistrés dans la base de données.

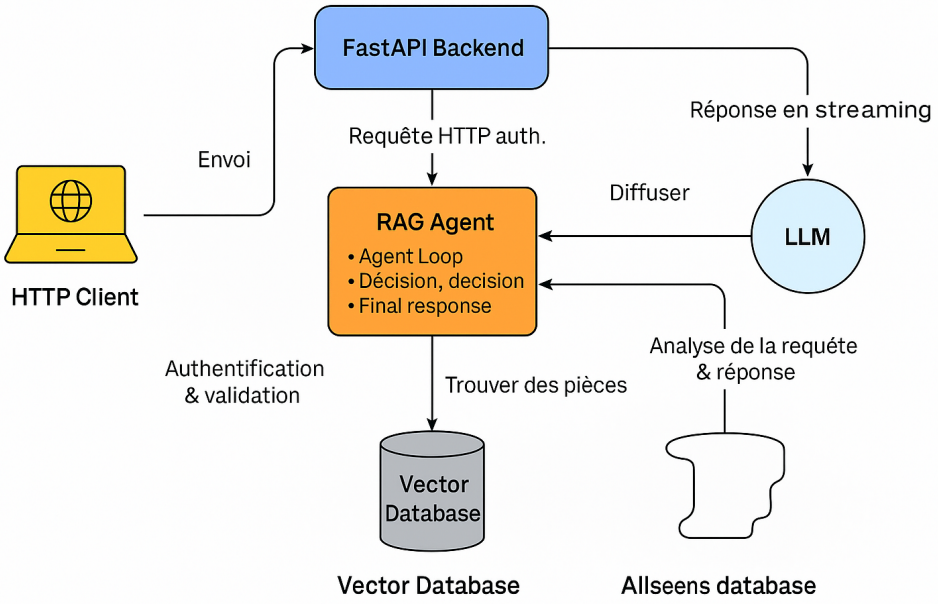

FIGURE 4.8 – Architecture globale

6. Les Points de Terminaison API

L'API expose les points de terminaison suivants (avec le préfixe `api/`). Tous les points de terminaison nécessitent un jeton JWT valide.

Point de terminaison & Méthode	**Description**	**Corps de Requête (JSON)**	**Réponse en cas de succès (Statut & Corps)**
`POST /chats/`	Crée un nouveau chat vide pour l'utilisateur authentifié.	Aucun	`201 Created` avec le nouvel objet chat : { `"id"` : "...", `"title"` : "New Chat", ... }
`GET /chats/`	Récupère la liste de tous les chats pour l'utilisateur, triée par dernière mise à jour.	Aucun	`200 OK` avec un tableau d'objets chat : [{ `"id"` : "...", `"title"` : "...", ... }]
`GET /chats/{chat_id}`	Récupère un seul chat incluant tous ses messages.	Aucun	`200 OK` avec l'objet chat complet et son tableau de messages imbriqués.
`PATCH /chats/{chat_id}`	Met à jour le titre d'un chat spécifique.	{ `"title"` : "New Title" }	`200 OK` avec l'objet chat mis à jour.
`DELETE /chats/{chat_id}`	Supprime un chat et tous ses messages associés.	Aucun	`204 No Content`.
`POST /chats/{chat_id} /messages`	**(Streaming)** Envoie un message utilisateur et diffuse en continu la réponse de l'agent.	{ `"content"` : "User message" }	`200 OK` avec un corps `text/event-stream`. Les événements sont envoyés au format SSE : — `event: tool` pour les appels d'outils. — `data: {"type": "token", "content": ...}` pour les jetons de réponse LLM.

TABLE 4.6: Méthodes HTTP

7. Suivi de l'utilisation de l'API OpenAI

Pour contrôler et analyser l'utilisation de l'API OpenAI dans le cadre du projet, une interface de monitoring est disponible via le tableau de bord fourni par OpenAI. Cette interface permet de visualiser en temps réel :

— Le nombre total de requêtes effectuées (ici 133),

— Le volume total de tokens consommés.

— La répartition des requêtes par jour sur une période donnée,

— La consommation budgétaire

— L'identification des utilisateurs ayant effectué les requêtes.

Ce type de suivi est indispensable pour maîtriser les coûts d'utilisation de l'API, optimiser les performances, et détecter d'éventuelles surconsommations.

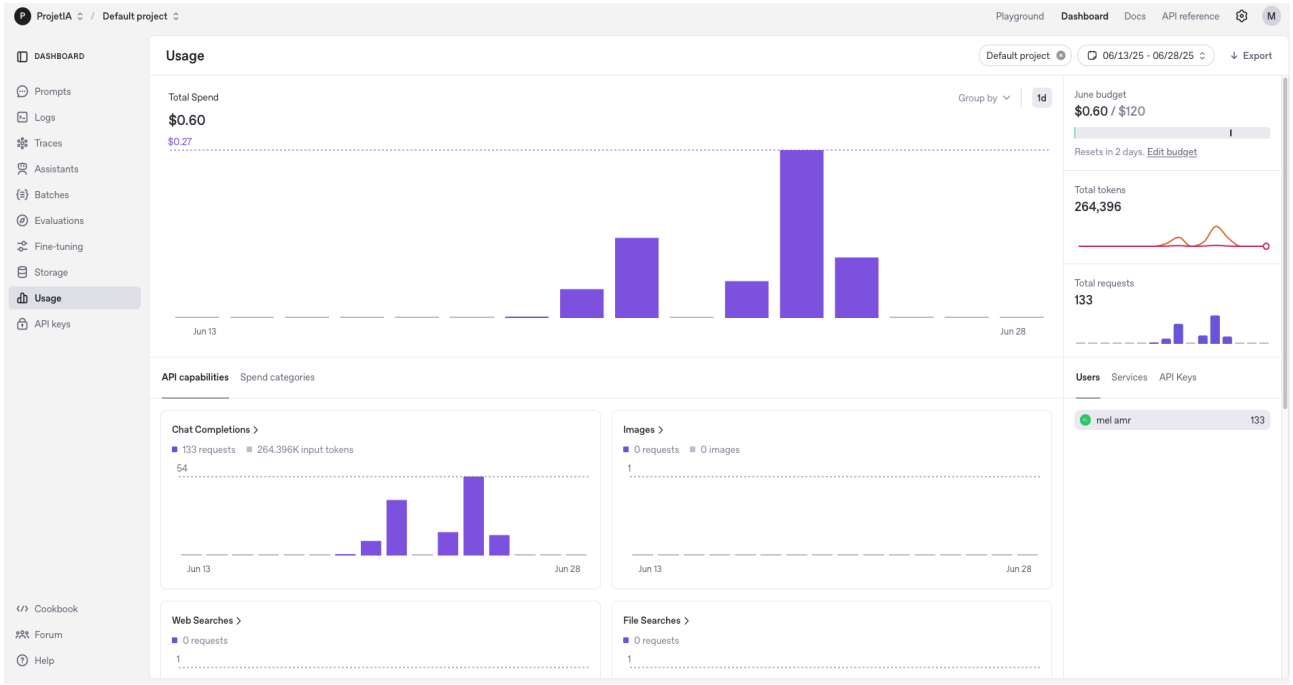

FIGURE 4.9 – Interface de gestion de l'usage de l'API OpenAI (Dashboard Usage)

L'interface de suivi d'usage d'OpenAI permet également une ventilation détaillée des coûts par modèle et par type d'opération (entrée, sortie, cache, embeddings, etc.). Cela permet d'identifier les composantes les plus coûteuses et d'optimiser leur usage.

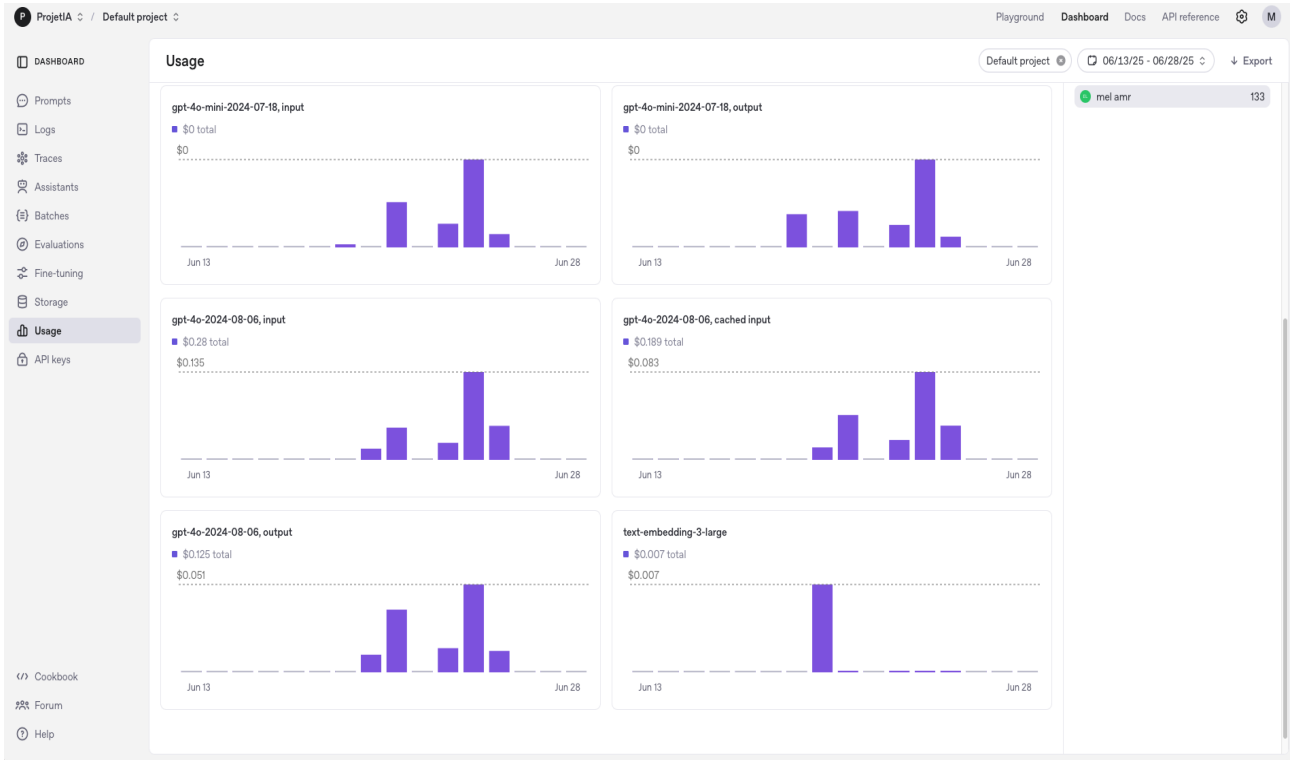

FIGURE 4.10 – Répartition de la consommation par modèle et type d'opération (OpenAI Dashboard)

Ces informations permettent de mieux ajuster les appels API en fonction du coût, par exemple en misant davantage sur les modèles plus légers si la tâche le permet.

3.1.5 Réalisation

1. Interface Chatbot

Les captures d'écran ci-dessous illustrent le fonctionnement de l'interface conversationnelle utilisée par les utilisateurs pour interagir avec l'assistant.

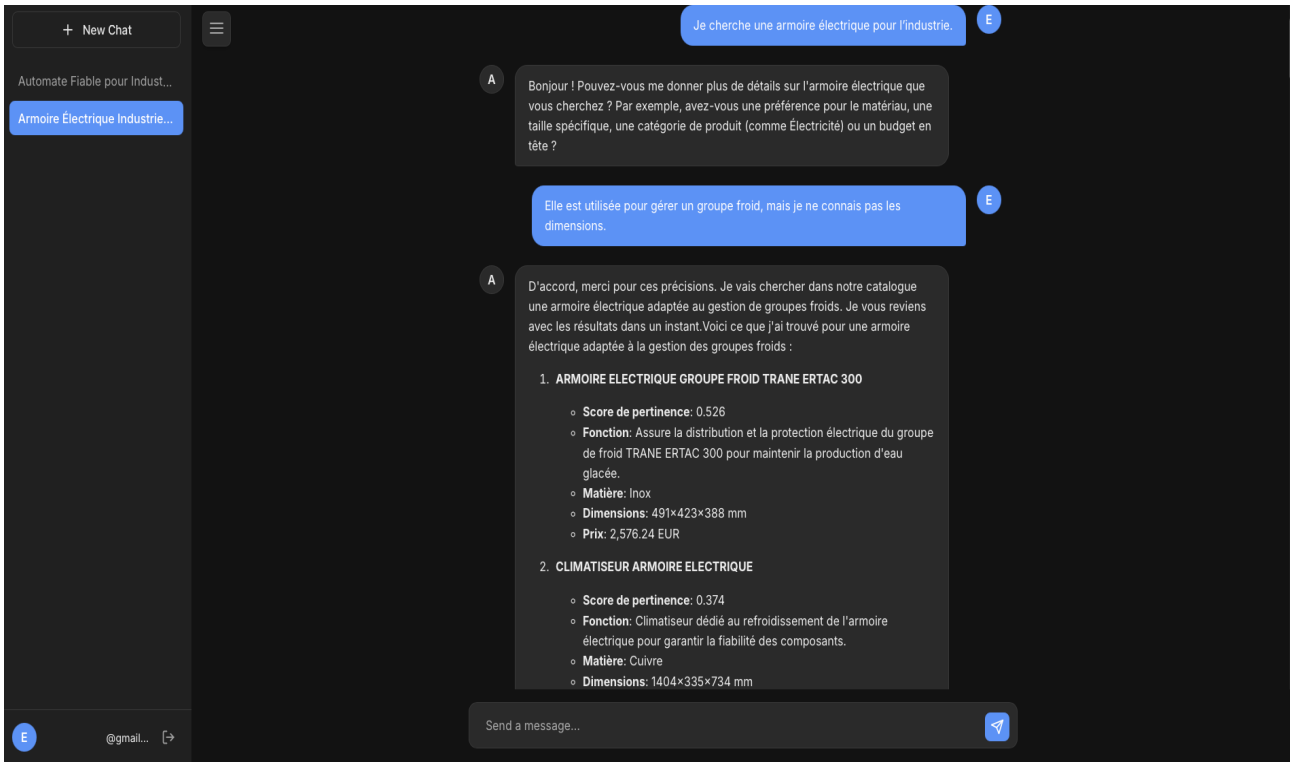

FIGURE 4.11 – Interface utilisateur : début de conversation avec l'assistant

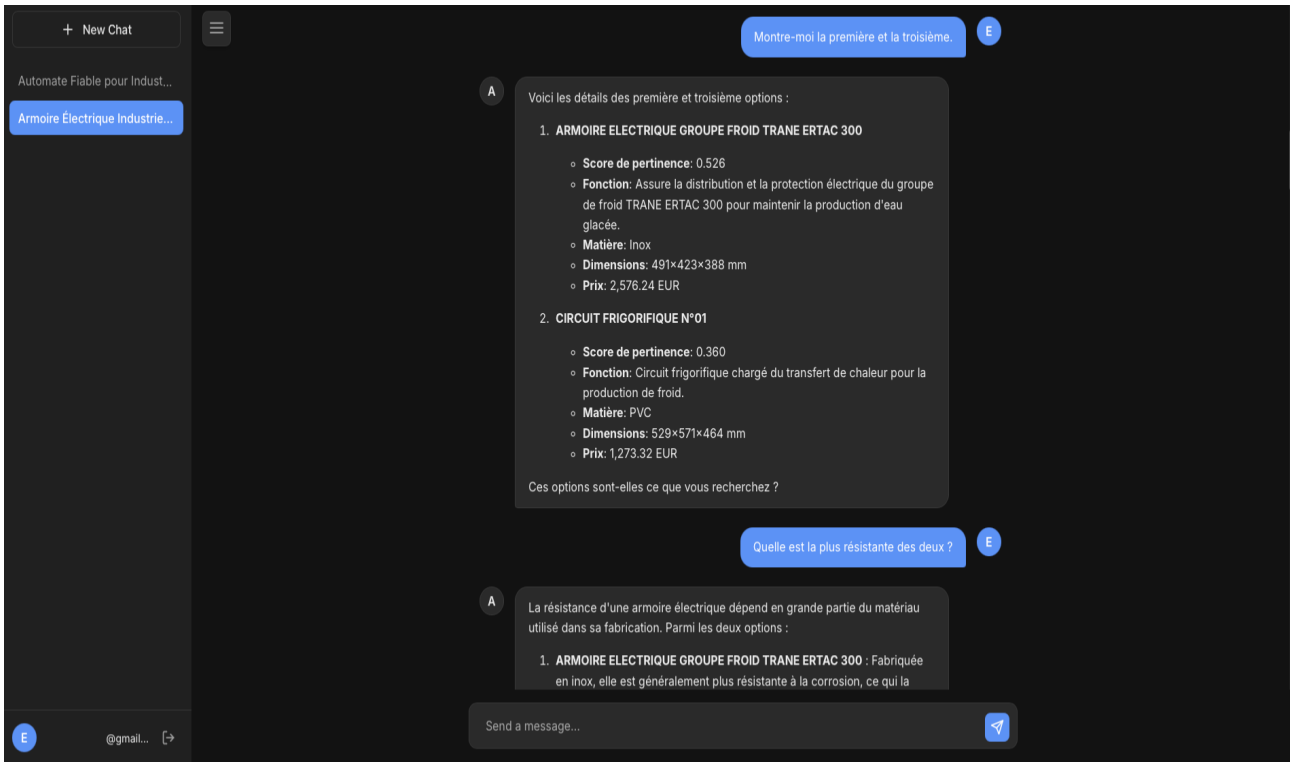

FIGURE 4.12 – Suggestions de'une piece selon les besoins exprimés

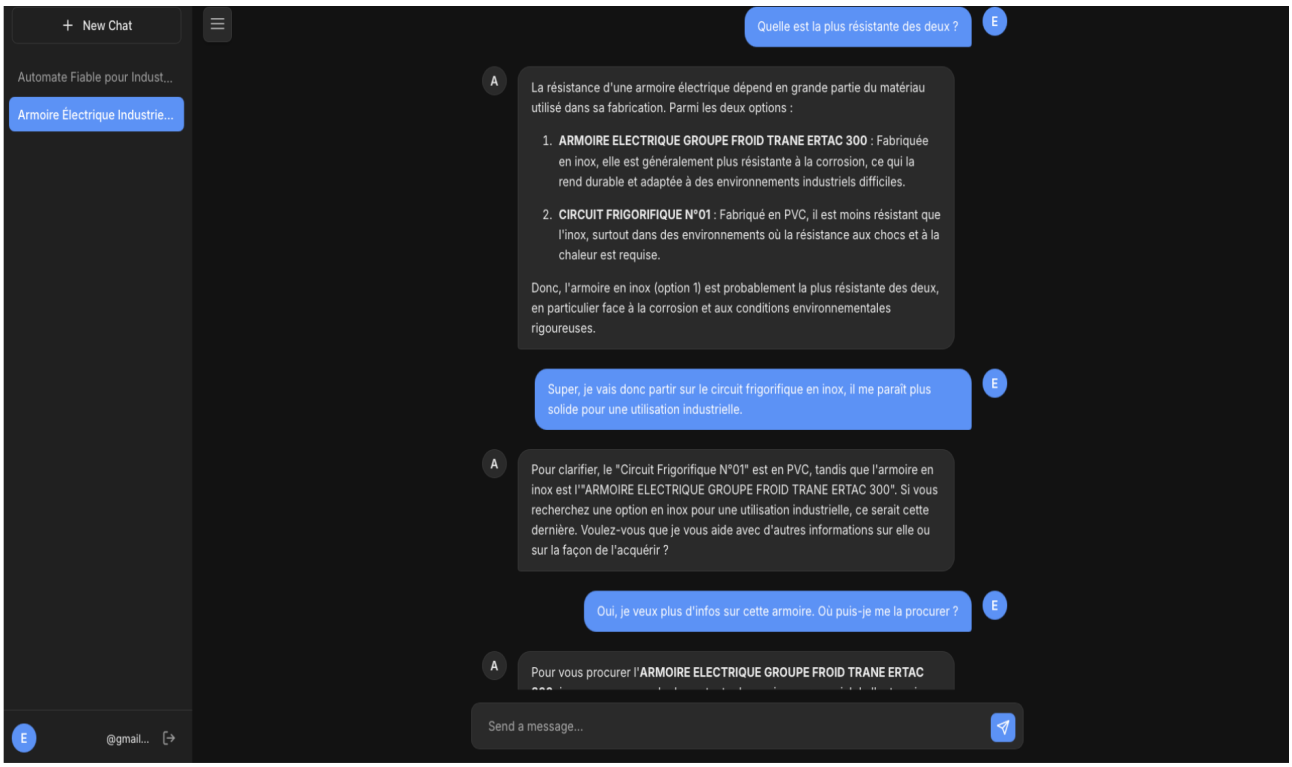

FIGURE 4.13 – Comparaison des options et recommandations techniques

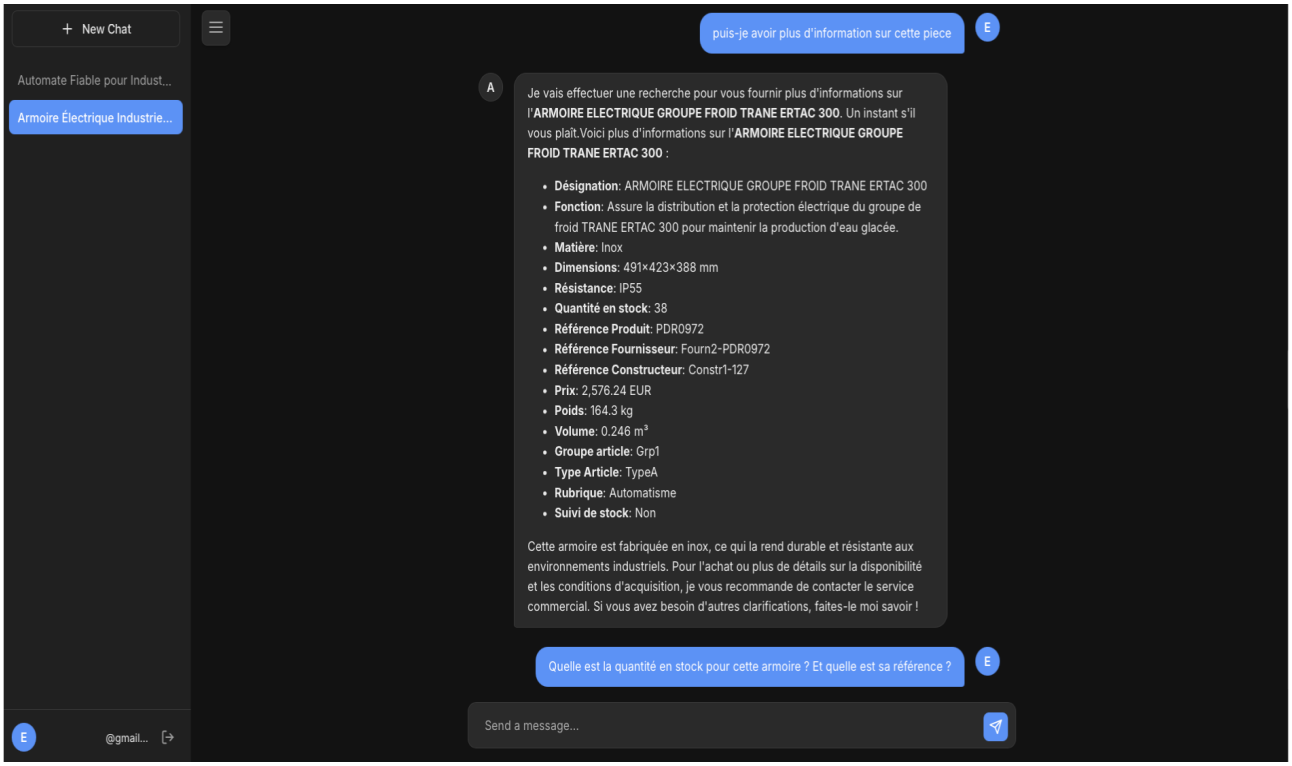

FIGURE 4.14 – Informations détaillées sur la pièce sélectionnée

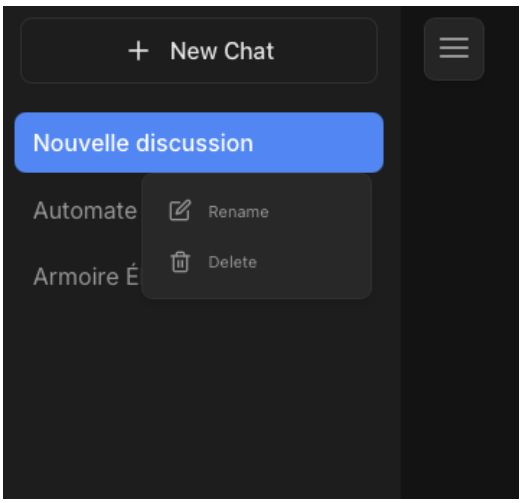

FIGURE 4.15 – Fonctionnalités de gestion d'une discussion : creation de chat, renommage et suppression

2. Sauvegarde de l'historique des conversations

Afin d'assurer la persistance des échanges entre l'utilisateur et l'agent conversationnel, le système enregistre chaque message dans une base de données hébergée sur Supabase.

Deux tables principales sont utilisées :

— `chat` : stocke les métadonnées de chaque conversation, notamment le titre, l'identifiant de l'utilisateur (`user_id`), ainsi que les horodatages de création et de mise à jour.

— `message` : enregistre chaque message individuel échangé, avec une distinction entre les rôles (`user` ou `assistant`), le contenu du message, et une référence au `chat_id` associé.

Chaque fois qu'un utilisateur interagit avec l'agent, un nouveau message est inséré dans la table `message`, associé à un identifiant de conversation existant ou nouvellement créé dans la table `chat`. Cela permet de reconstituer l'historique complet d'une session utilisateur pour un raisonnement contextuel efficace.

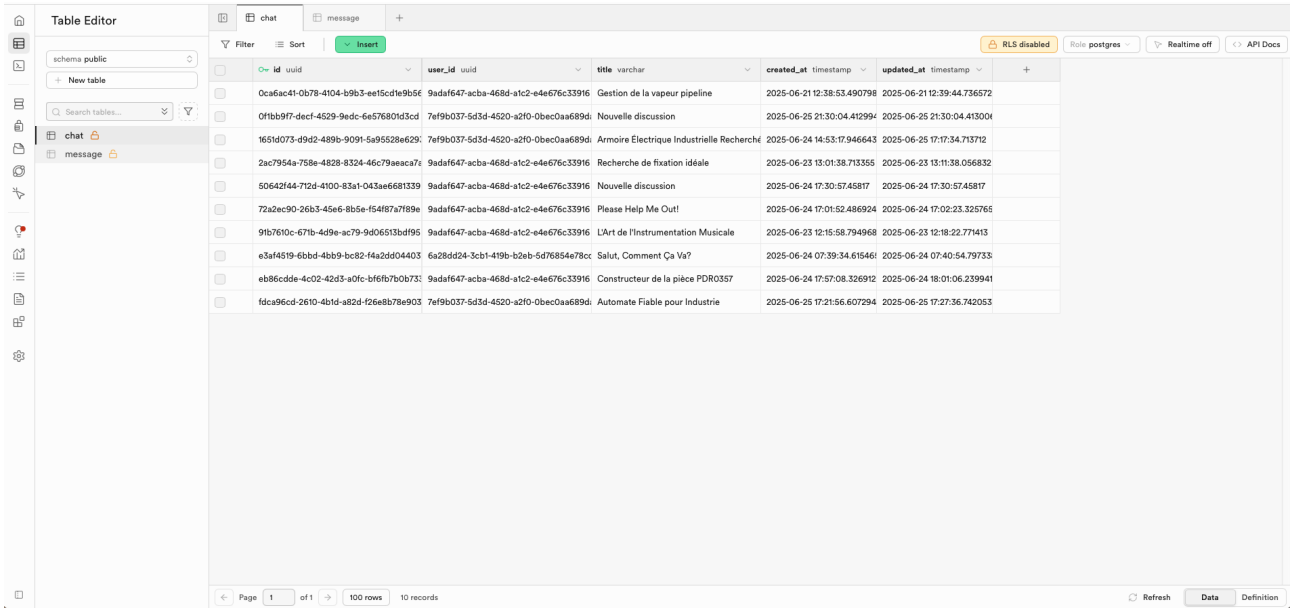

FIGURE 4.16 – Contenu de la table `chat` dans Supabase

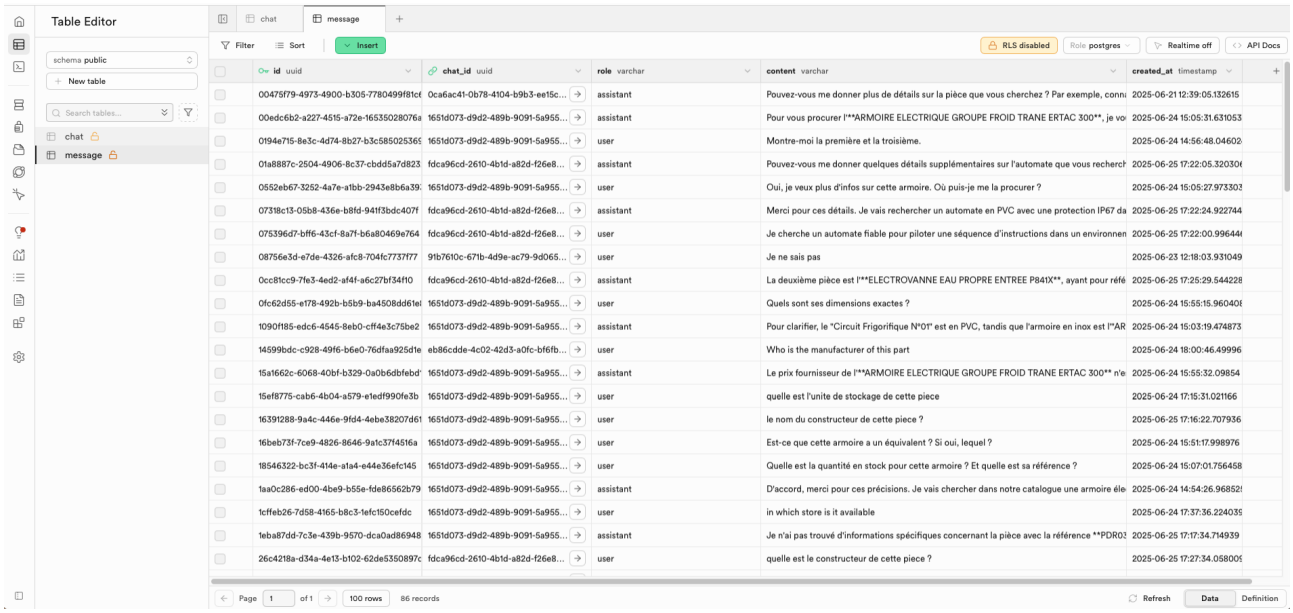

FIGURE 4.17 – Contenu de la table `message` dans Supabase

3.2 Sprint 2 : Gestion des stocks et des utilisateurs

3.2.1 Objectif :

Ce sprint a pour objectif de mettre en place les éléments fondamentaux de la gestion des stocks, qui comprend : la gestion des pièces de rechange, la gestion des magasins, ainsi que la gestion des types, des familles et des groupes de pièces. Il inclut également la gestion des utilisateurs du système : création des comptes, affectation à des groupes et rôles, attribution des permissions et des droits d'accès.Son but est d'établir une structure solide qui servira de base pour les fonctionnalités développées dans les sprints suivants.

Le tableau 4.7 ci-dessous représente le Sprint Backlog de ce sprint.

Fonctionnalités	**Description**	**User Stories associées**
S2-F01	Gestion des PDR (ajout, modification, suppression)	US02
S2-F02	Gestion des Magasins (ajout, modification, suppression)	US02
S2-F03	Gestion des types, familles et groupes des PDR (ajout, modification, suppression)	US02
S2-F04	Gestion des comptes utilisateurs (ajout, modification, suppression)	US01
S2-F05	Affecter les utilisateurs à des groupes	US01
S2-F06	Affecter des droits d'accées aux utilisateurs selon les groupes	US01

TABLE 4.7: Sprint Backlog du Sprint 2

3.2.2 Prototype

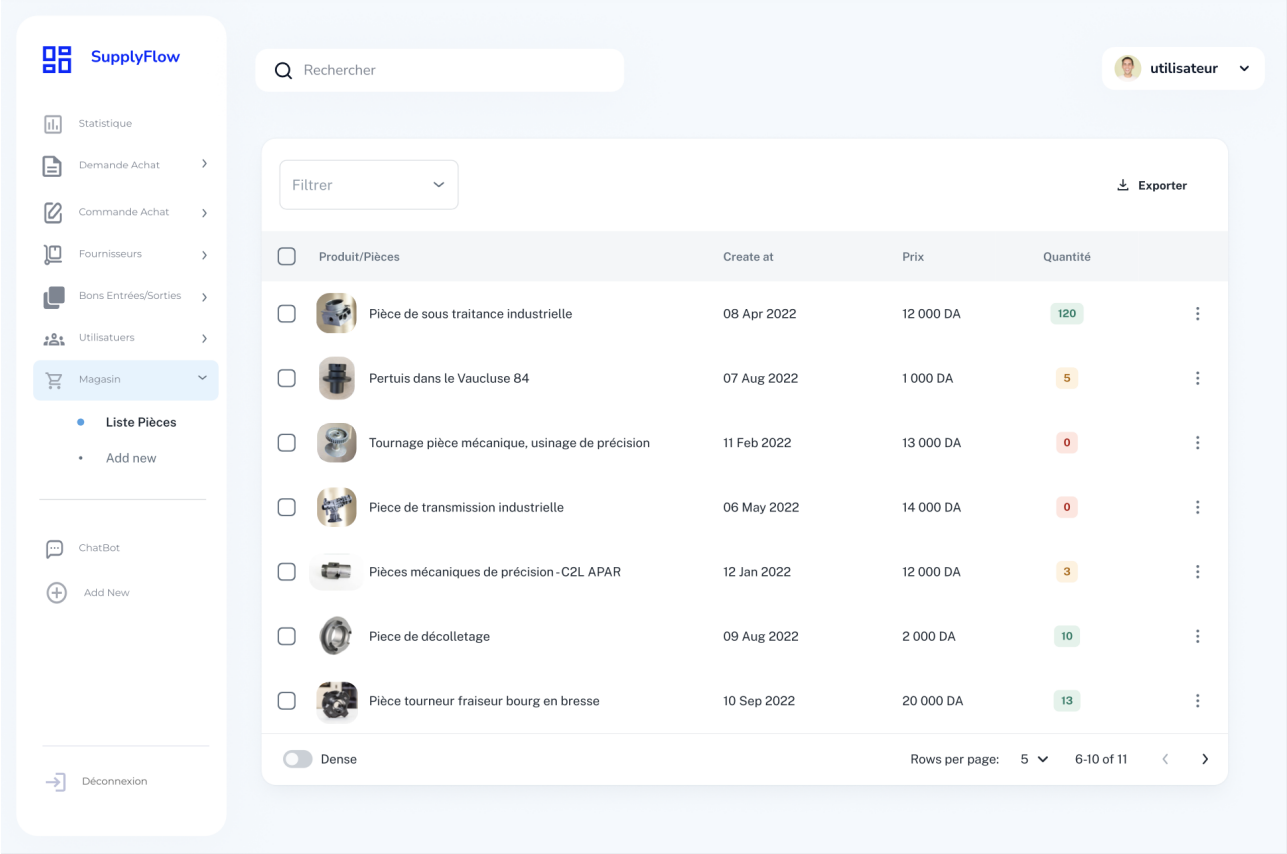

FIGURE 4.18 – Design de l'interface de la liste des pieces

3.2.3 Conception

Cette section présente les diagrammes UML conçus durant le Sprint 2, illustrant les fonctionnalités, les acteurs impliqués et l'architecture du système.

1. **Diagramme de cas d'utilisation** Le diagramme de cas d'utilisation du second Sprint est présenté dans la figure 4.19 suivante.

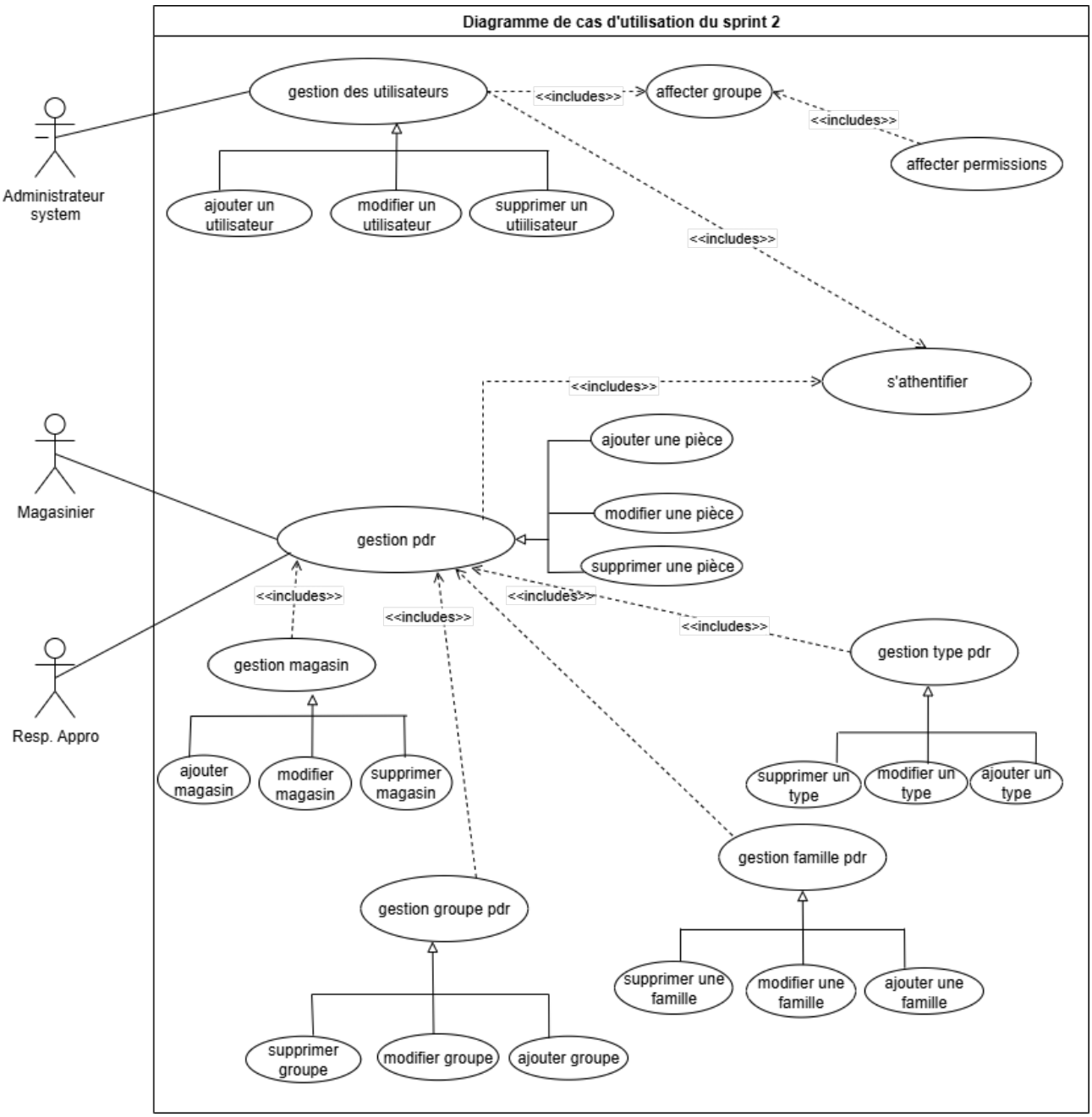

FIGURE 4.19 – Diagramme de cas d'utilisation du Sprint 2

Le tableau ci-dessous 4.8 présente la description textuelle du cas d'utilisation "Gestion des Utilisateurs"

Nom du cas : Gérer les utilisateurs
But : Permettre à un administrateur de gérer les utilisateurs du système, en leur attribuant une identité, un groupe, et des droits d'accès appropriés.
Acteur principal : Administrateur
Acteur secondaire : Système d'authentification
Date de création : 23/04/2025
Date de mise à jour : 01/07/2025
Responsable : Èquipe technique de développement
Version : 1.0
Séquencement : Le cas d'utilisation commence lorsqu'un administrateur accède à l'interface de gestion des utilisateurs.
Préconditions : — L'administrateur est authentifié et il a les droits d'accès à la gestion des utilisateurs.
Enchaînement nominal : (a) L'administrateur accède à l'interface de gestion des utilisateurs. (b) Il clique sur le bouton « Ajouter un utilisateur ». (c) Il remplit le formulaire de création avec : nom, prénom, mot de passe, email, rôle. (d) Il sélectionne le groupe d'appartenance (maintenancier, magasinier, acheteur, etc.). (e) Il valide la création. (f) Le système enregistre l'utilisateur et lui affecte les droits correspondant à son groupe. (g) Une confirmation de création s'affiche à l'écran.
Enchaînements alternatifs : **A1 : Données invalides ou incomplètes** — Le système affiche un message d'erreur précisant les champs incorrects ou manquants. — L'administrateur corrige les erreurs et soumet à nouveau. — La séquence nominale reprend au point 3.
Enchaînements d'exception : **E1 : Échec de l'enregistrement dans la base** — Le système affiche une erreur technique. — Aucune donnée n'est enregistrée. — L'administrateur est invité à réessayer ultérieurement.
Postconditions : — Un nouvel utilisateur est ajouté au système. — Ses permissions sont configurées en fonction de son groupe. — Il pourra accéder aux modules selon les droits qui lui sont attribués.

TABLE 4.8 – Description textuelle du cas d'utilisation "Gérer les utilisateurs"

Le tableau ci-dessous 4.9 présente la description textuelle du cas d'utilisation "Gestion des pièces de rechanges"

Nom du cas : Gérer les pièces de rechange
But : Permettre aux utilisateurs habilités de gérer les pièces de rechange, incluant la création, modification et suppression d'articles, et la classification par type, famille et groupe.
Acteur principal : Magasinier / Responsable approvisionnement
Acteur secondaire : Néant
Date de création : 23/04/2025
Date de mise à jour : 01/07/2025
Responsable : Èquipe technique de développement
Version : 1.0
Séquencement : Le cas d'utilisation commence lorsqu'un utilisateur accède à l'interface de gestion des pièces de rechange.
Préconditions : — L'utilisateur est authentifié et appartient au groupe des magasiniers ou responsables approvisionnement.
Enchaînement nominal : (a) L'utilisateur accède à l'interface de gestion des pièces. (b) Il choisit de créer une nouvelle pièce. (c) Il renseigne les informations nécessaires : type d'article, famille, groupe, désignation, etc. (d) Il valide la création. (e) Le système confirme la création avec une notification de succès. (f) L'utilisateur peut aussi modifier ou supprimer des pièces existantes. (g) Le système enregistre toutes les modifications.
Enchaînements alternatifs : **A1 : Informations incomplètes ou erronées** — Le système affiche un message d'erreur précisant les champs incorrects. — L'utilisateur corrige les informations. — La séquence nominale reprend au point 3.
Enchaînements d'exception : **E1 : Erreur d'enregistrement dans la base** — Le système affiche une erreur technique. — Aucune modification n'est sauvegardée.
Postconditions : — Les données des pièces sont enregistrées ou mises à jour. — Les pièces sont disponibles pour les processus de réservation, de commande ou de consultation.

TABLE 4.9 – Description textuelle du cas d'utilisation "Gérer les pièces de rechange"

2. **Diagramme de séquence** La figure 4.20 ci dessous illustre le diagramme de séquence du cas d'utilisation "Gestion des PDR".

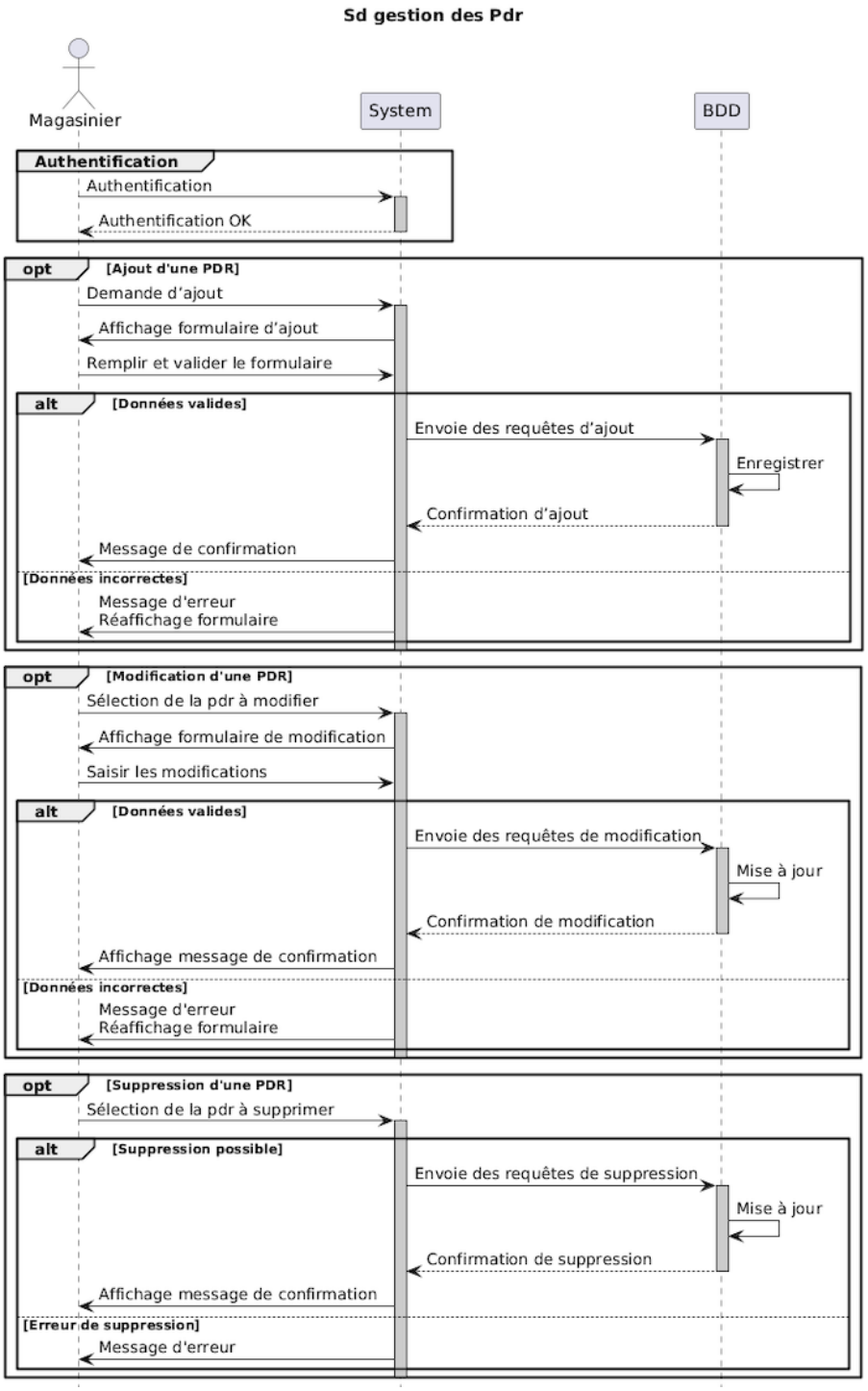

FIGURE 4.20 – Diagramme de séquence du cas d'utilisation "Gestion des PDR"

La figure 4.21 suivante illustre le diagramme de séquence du cas d'utilisation " S'authentifier"

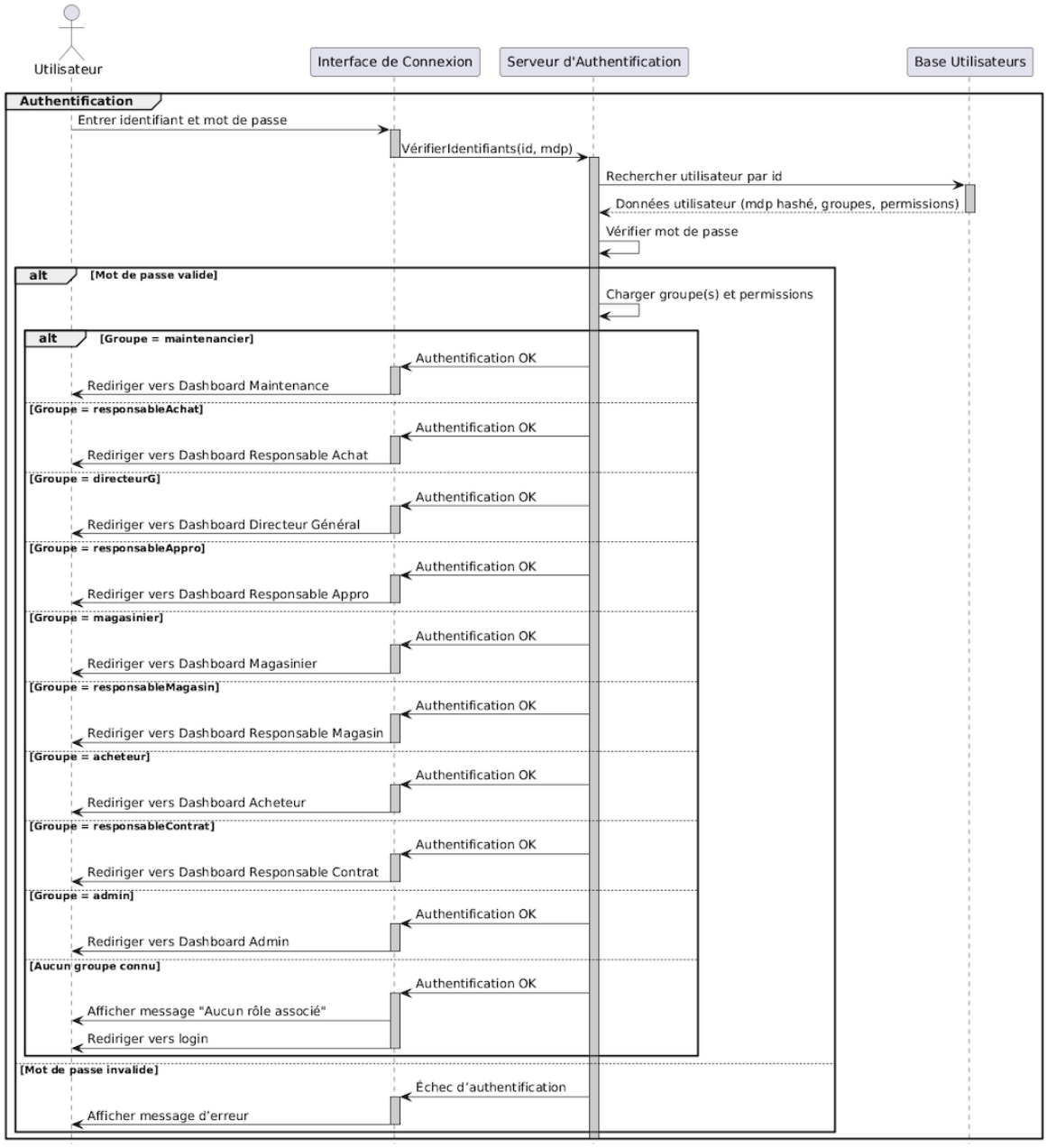

FIGURE 4.21 – Diagramme de séquence du cas d'utilisation "S'Authentifier"

3. **Diagramme de classe** La figure 4.22 ci-dessous, présente le diagramme de classe elaboré pour le second Sprint.

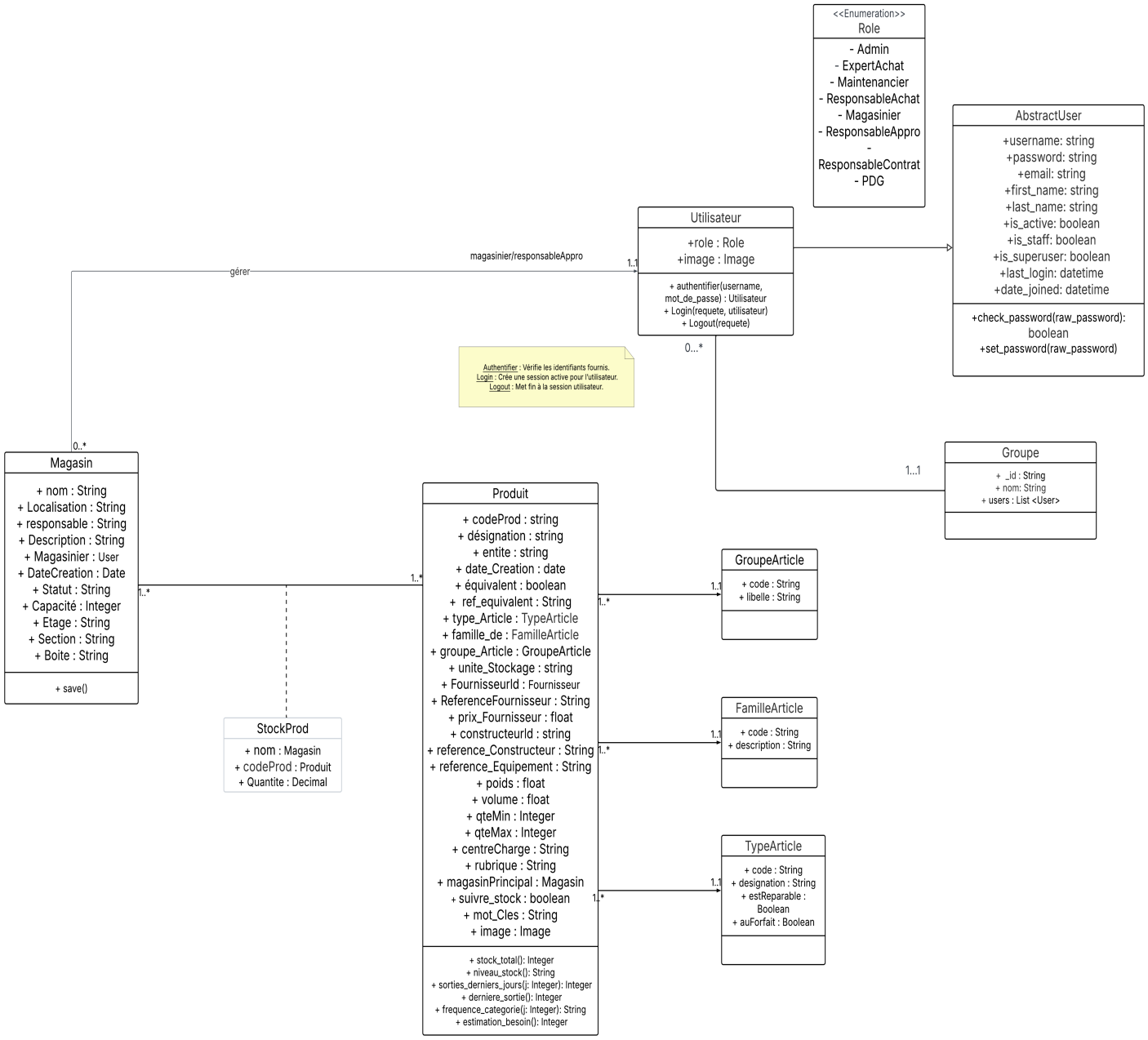

FIGURE 4.22 – Diagramme de classe du Sprint 2

3.2.4 Développement

Pour ce second Sprint, nous avons mis en place un ensemble de modèles permettant de gérer les piéces de rechange, les stocks et les utilisateurs ainsi que leurs droit d'accées. Nous avons également intégré deux pratiques agiles essentielles, à notre méthode de travail :

— **Le refactoring** : une pratique qui nous a permis d'améliorer la qualité du code, de réduire les duplications et d'optimiser l'architecture sans modifier les fonctionnalités existantes.

— **L'intégration continue** : en intégrant régulièrement nos modifications dans une branche partagée via GitHub, nous avons pu tester automatiquement les échanges et détecter rapidement d'éventuelles erreurs. Cela a renforcé la stabilité du projet et facilité la collaboration en équipe.

Ces pratiques nous ont guidés vers une réalisation progressive, simple et fonctionnelle, des modules développés lors de ce sprint. Ces pratiques nous ont permis de structurer efficacement le développement de notre application. Nous présentons ci-dessous les principaux modèles mis en place durant ce second sprint.

1. Le Modèle `Product`

Le modèle `Product` regroupe les informations essentielles relatives aux pièces de rechange. Il inclut notamment les champs suivants : codeProd, designation, referenceFournisseur, qauntiteMin et quantiteMax, ainsi que des références à trois modèles permettant de faciliter le classement et la recherche :

- **`TypeArticle`** : contient les champs `code`, `libelle`, ainsi que deux attributs booléens `auForfait` et `estReparable`, servant à indiquer si la pièce est incluse dans un forfait ou réparable.
- **`FamilleArticle`** : comprend les champs `code` et `libelle`.
- **`GroupeArticle`** : contient également `code` et `libelle`.

Afin de gérer le stockage multi-emplacement, le modèle `Magasin`a été mis en place. Il regroupe des informatins relatives à chaque magasin tels que le nom, le responsable, la localisation et la capacite max. Le modèle `StockProd`, une classe d'association entre `Product` et `Magasin`, permet quant à lui, de représenter avec précision la quantité disponible d'une pièce donnée dans chaque magasin. Il contient les champs quantite, codeProduit et nomMagasin.

2. Modèle `User`

Le modèle `User` hérite du modèle standard `AbstractUser` fourni par Django. Ce dernier comprend déjà les champs usuels nécessaires tels que : password, username, first-name, last-name et email. Nous avons enrichi ce modèle en ajoutant des champs personnalisés :

- `role` : qui indique la fonction de l'utilisateur dans le système.
- `image` : une photo de profil pour chaque utilisateur.

Les utilisateurs sont ensuite associés à des groupes, représentant les divers acteurs du processus d'approvisionnement :

- Administrateur.
- Magasinier.
- Maintenancier.
- Responsable approvisionnement.
- Responsable achat.
- Directeur général.
- Acheteur.
- Responsable contrat.

3. Mécanismes d'authentification et contrôle d'accès

Afin de sécuriser l'accès à l'application, nous avons utilisé les mécanismes d'authentification natifs de Django, combinés à une gestion des sessions et des autorisations.

Connexion / Déconnexion : Les vues suivantes assurent la gestion des authentifications :

Listing 4.1 – Vue de Connexion

```
    def login_page(request):
    forms = LoginForm()
    if request.method == 'POST':
        forms = LoginForm(request.POST)
        if forms.is_valid():
            username = forms.cleaned_data['username']
            password = forms.cleaned_data['password']
            user = authenticate(username=username, password=password)
            if user:
                login(request, user)
                return redirect('page_redirect')

    context = {'form': forms}
    return render(request, 'users/login.html', context)
```

Listing 4.2 – Vue de Déconnexion

```
   def logout_page(request):
    logout(request)
    return redirect('login')
```

Sessions : chaque utilisateur connecté dispose d'une session active qui permet de suivre son activité de manière sécurisée.

Décorateurs de restriction d'accès Afin de filtrer les droit d'accès des utilisateurs nous avons eu recours a des décorateurs de contrôle d'accès tels que :

— `@login_required` : empêche l'accès aux utilisateurs non connectés.

— `@user_passes_test` : permet de restreindre l'accès selon le rôle ou le groupe d'appartenance.

Redirection dynamique selon le rôle : Une fois connecté, l'utilisateur est redirigé automatiquement vers le tableau de bord correspondant à son rôle grâce à la vue suivante :

Listing 4.3 – Fonction de redirection des utilisateurs

```
  @login_required
def page_redirect(request):
    print("Redirection depuis page_redirect() - Utilisateur:", request
       .user)
    print("Groupes de l'utilisateur:", list(request.user.groups.all())
       )

    if request.user.groups.filter(name='maintenancier').exists():
        return redirect('dashboard_maintenance')
    elif request.user.groups.filter(name='responsableAchat').exists():
        return redirect('dashboard_responsableAchat')
    elif request.user.groups.filter(name='directeurG').exists():
        return redirect('dashboard_directeurG')
    elif request.user.groups.filter(name='responsableAppro').exists():
        return redirect('dashboard_responsableAppro')
```

```
    elif request.user.groups.filter(name='magasinier').exists():
        return redirect('dashboard_magasinier')
    elif request.user.groups.filter(name='responsableMagasin').exists
       ():
        return redirect('dashboard_responsableMagasin')
    elif request.user.groups.filter(name='acheteur').exists():
        return redirect('dashboard_acheteur')
    elif request.user.groups.filter(name='responsableContrat').exists
       ():
        return redirect('dashboard_responsableContrat')
    elif request.user.groups.filter(name='admin').exists():
        return redirect('dashboard')
    else:
        messages.warning(request,"Votre compte n'est associe a aucun
           role.")
        return redirect('login')
```

Ces mecanisme permettent de garantir que chaque utilisateur accéde uniquement aux fonctionnalités qui lui sont autorisées selon son profil.

3.2.5 Réalisation

Dans cette section, nous présentons quelques interfaces utilisateur développées au cours de ce second sprint. Leur conception a été guidée par les retours recueillis auprès des utilisateurs, afin de mieux répondre à leurs préférences et besoins fonctionnels.

La figure 4.23 montre l'interface utilisateur de la connexion :

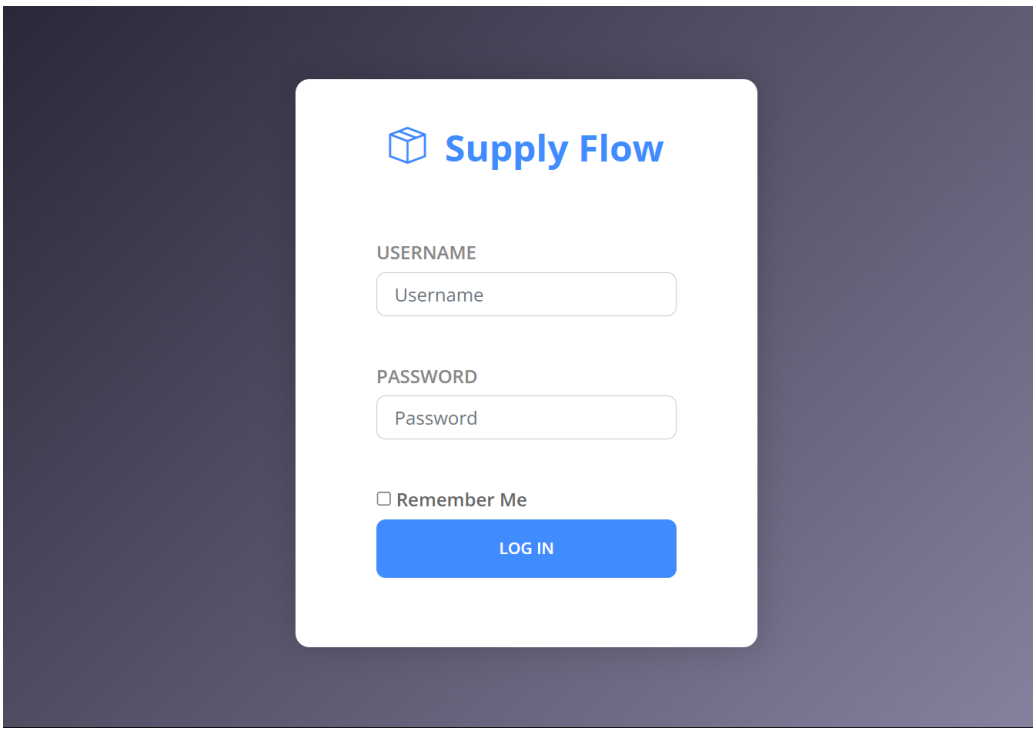

FIGURE 4.23 – Le page d'authentification

La figure 4.24 montre un exemple du profil d'un utilisateur :

FIGURE 4.24 – Profil utilisateurs

3.3 Sprint 3 : Gestion des demandes d'achat, des réservations et des commandes d'achat

3.3.1 Objectif :

L'objectif de ce sprint est de permettre à un utilisateur de soumettre une demande de réservation ou une demande d'achat, et de mettre en place la logique de création des commandes d'achat. Il inclut également l'intégration d'un workflow de validation, impliquant des valideurs successifs, pour garantir le suivi et l'approbation des demandes selon le circuit défini.

Le tableau 4.10 représente le Sprint Backlog de ce sprint.

Fonctionnalités	**Description**	**User Stories associées**
S3-F01	Création et soumission de demandes de réservation	US06
S3-F02	Consultation des demandes de réservation	US07
S3-F03	Suivi de l'état des demandes de réservation : en attente, validée, refusée	US08
S3-F04	Validation des demandes de réservation selon un circuit	US13, US15
S3-F05	Création et soumission de demandes d'achat	US10
S3-F06	Consultation des demandes d'achat	US11
S3-F07	Suivi de l'état des demandes d'achat (en attente, validée, refusée)	US11
S3-F08	Validation des demandes d'achat selon un circuit	US13, US14
S3-F09	Passage des commandes d'achat	US17
S3-F10	Génération des bons de commande	US18

S3-F11	Consultation de l'historique des commandes	US16
S3-F12	Validation des commandes d'achats	US13, US14
S3-F13	Notifications en temps réel pour le suivi des validations et actions	US26
S3-F14	Consulter son tableau de bord	US09

TABLE 4.10: Sprint Backlog du Sprint 3

3.3.2 Prototype

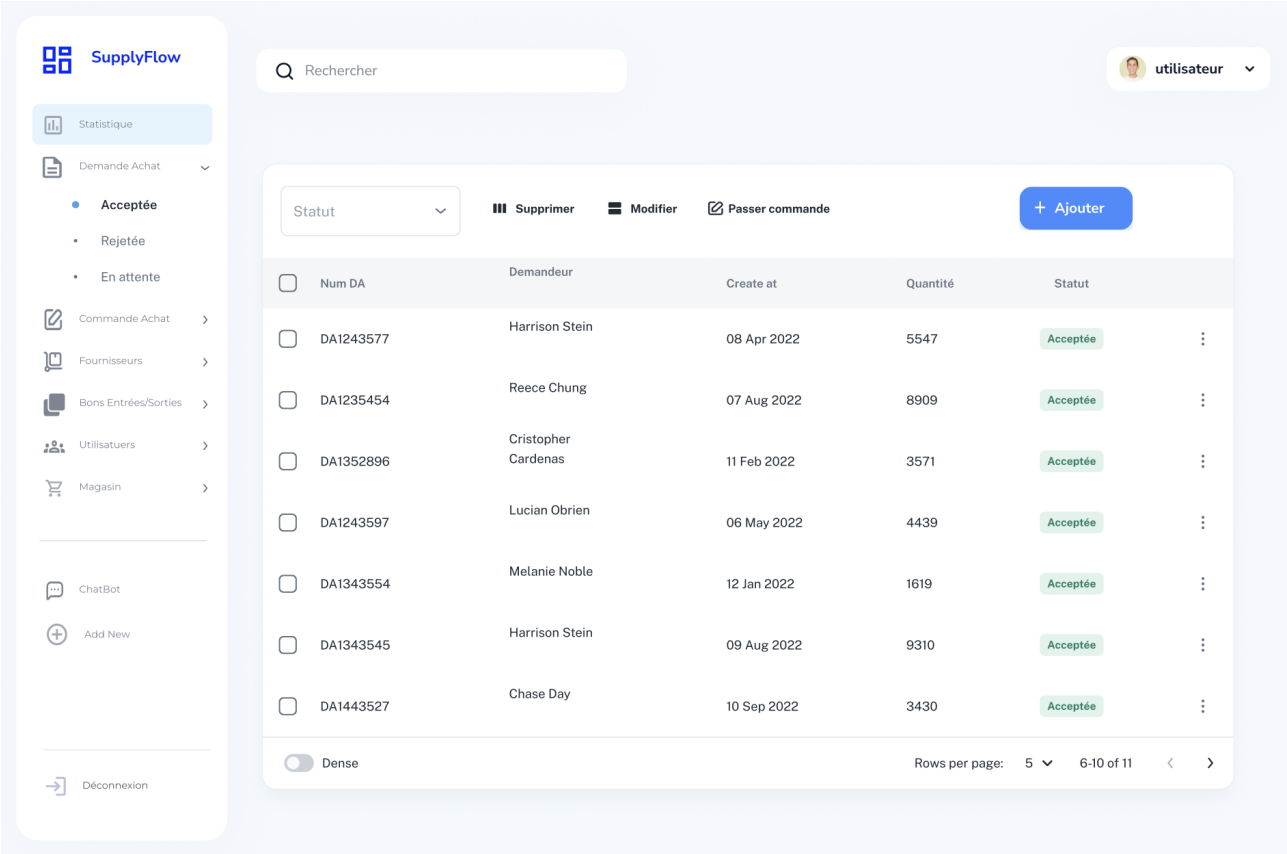

FIGURE 4.25 – Design de l'interface de la liste des Demandes D'achat

3.3.3 Conception

Cette section est consacrée à la présentation des diagrammes UML élaborés durant ce 3eme sprint. Ils servent à représenter les cas d'utilisation, les interactions entre les composants, ainsi que la structure générale du système.

1. **Diagramme de cas d'utilisation** La figure 4.26 ci-dessous, illustre le diagramme de cas d'utilisation elaboré pour le Sprint 3 :

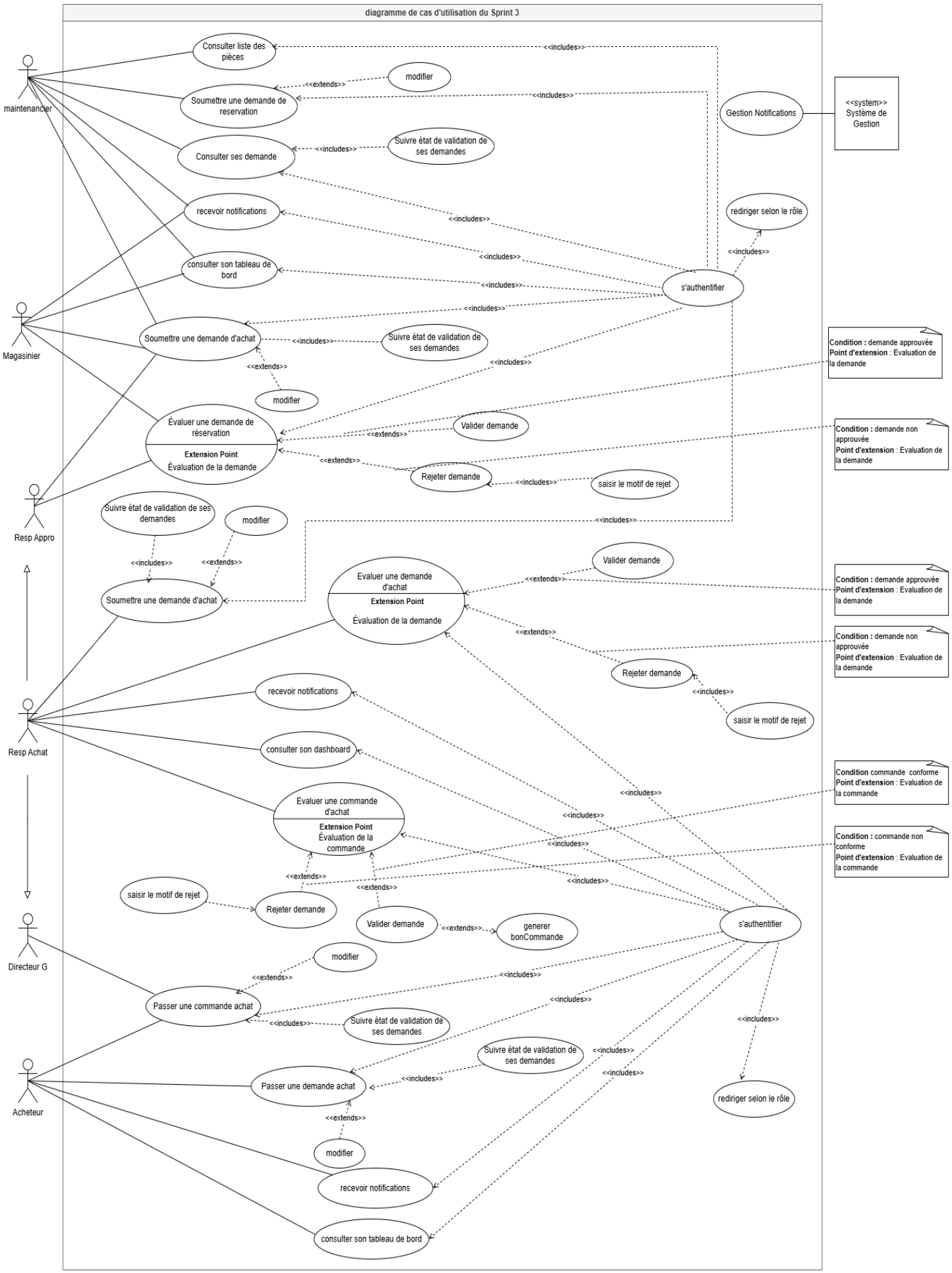

FIGURE 4.26 – Diagramme de cas d'utilisation du sprint 3

Le tableau 4.11 Présente la description du cas "Soumettre une demande de réservation"

Nom du cas : Soumettre une demande de réservation
But : Permettre à un utilisateur du groupe maintenancier de formuler une demande de réservation pour des pièces nécessaires à une intervention.
Acteur principal : Maintenancier
Acteur secondaire : Système de notification des valideurs
Date de création : 01/05/2025
Date de mise à jour : 01/07/2025.
Responsable : Équipe technique de développement.
Version : 1.0
Séquencement : Le cas d'utilisation commence lorsqu'un maintenancier accède au formulaire de création d'une demande de réservation.
Préconditions : — L'utilisateur est authentifié et le formulaire de demande est accessible.
Enchaînement nominal : (a) Le maintenancier accède à la page de création d'une demande. (b) Il renseigne les informations requises : — Liste des pièces à réserver, Date souhaitée de réservation, Motif de la demande. (c) Il valide le formulaire. (d) Le système vérifie la complétude et la validité des informations. (e) Le système enregistre la demande dans la base de données. (f) Le système notifie les valideurs responsables du traitement de la demande. (g) Un accusé de soumission est affiché à l'utilisateur.
Enchaînements alternatifs : **A1 : Informations invalides ou incomplètes** — Le système affiche un message d'erreur détaillant les champs incorrects ou manquants. — Le maintenancier peut corriger les données et soumettre à nouveau. — La séquence nominale reprend au point 3.
Enchaînements d'exception : **E1 : Erreur d'enregistrement en base de données** — Le système affiche un message d'erreur technique. — Aucune demande n'est enregistrée. — L'utilisateur est invité à réessayer ultérieurement.
Postconditions : — La demande de réservation est enregistrée et mise en attente de validation. — Les valideurs ont été notifiés. — L'utilisateur a reçu un accusé de réception.

TABLE 4.11 – Description textuelle du cas d'utilisation "Soumettre une demande de réservation"

Le tableau 4.12 Présente la description du cas "Evaluer une demande de réservation"

Identification
Nom du cas : Évaluer une demande de réservation. **But** : Permettre aux utilisateurs habilités (responsable approvisionnement et magasinier) d'évaluer et de valider une demande de réservation soumise par un maintenancier. **Acteurs principaux** : Responsable approvisionnement, Magasinier. **Acteur secondaire** : Maintenancier. **Date de création** : 03/05/2025. **Date de mise à jour** : 01/07/2025. **Responsable** : Équipe technique de développement. **Version** : 1.0.
Séquencement
Le cas d'utilisation commence lorsqu'une demande de réservation est soumise par un maintenancier et qu'elle entre dans le circuit de validation.
Préconditions
— La demande de réservation a été correctement saisie et soumise par un maintenancier. — Les utilisateurs (valideurs) sont authentifiés.
Enchaînement nominal (scénario principal)
(a) Le responsable approvisionnement accède à la liste des demandes en attente. (b) Il consulte les détails de la demande. (c) Il valide ou rejette la demande. (d) Si la demande est validée, le magasinier est notifié. (e) Le magasinier accède à la demande transmise. (f) Il effectue à son tour la validation. (g) Le système met à jour le statut de la demande et notifie le maintenancier.
Enchaînements alternatifs
A1 : Si la demande est correcte et ne nécessite pas de rejet : - Le système transmet la demande au magasinier pour traitement. **A2 :** Si le responsable approvisionnement rejette la demande : — Le système transmet tout de même la demande au magasinier. — Le magasinier est notifié, mais ne peut qu'effectuer un rejet (la validation lui est bloquée). — La demande est alors définitivement rejetée et le maintenancier en est informé. **A3 :** Le magasinier tente de valider une demande qui n'a pas encore été traitée par le responsable approvisionnement : - Le système bloque l'action.
Enchaînements d'exception
E1 : Le valideur n'est pas authentifié : — Le système redirige vers la page de connexion. **E2 :** La demande de réservation n'existe pas ou a été supprimée : — Le système affiche un message d'erreur. — L'utilisateur est redirigé vers la liste des demandes.
Postconditions
Le statut de la demande devient « validée » ou « rejetée ». Le maintenancier est informé de la décision finale.

TABLE 4.12 – Description textuelle du cas d'utilisation "Évaluer une demande de réservation"

2. **Diagramme de séquence** Le diagramme illustré dans la figure 4.27 ci-dessous, décrit l'aspect chronologique du cas d'utilisation "Valider une demande d'achat" du troisiéme sprint.

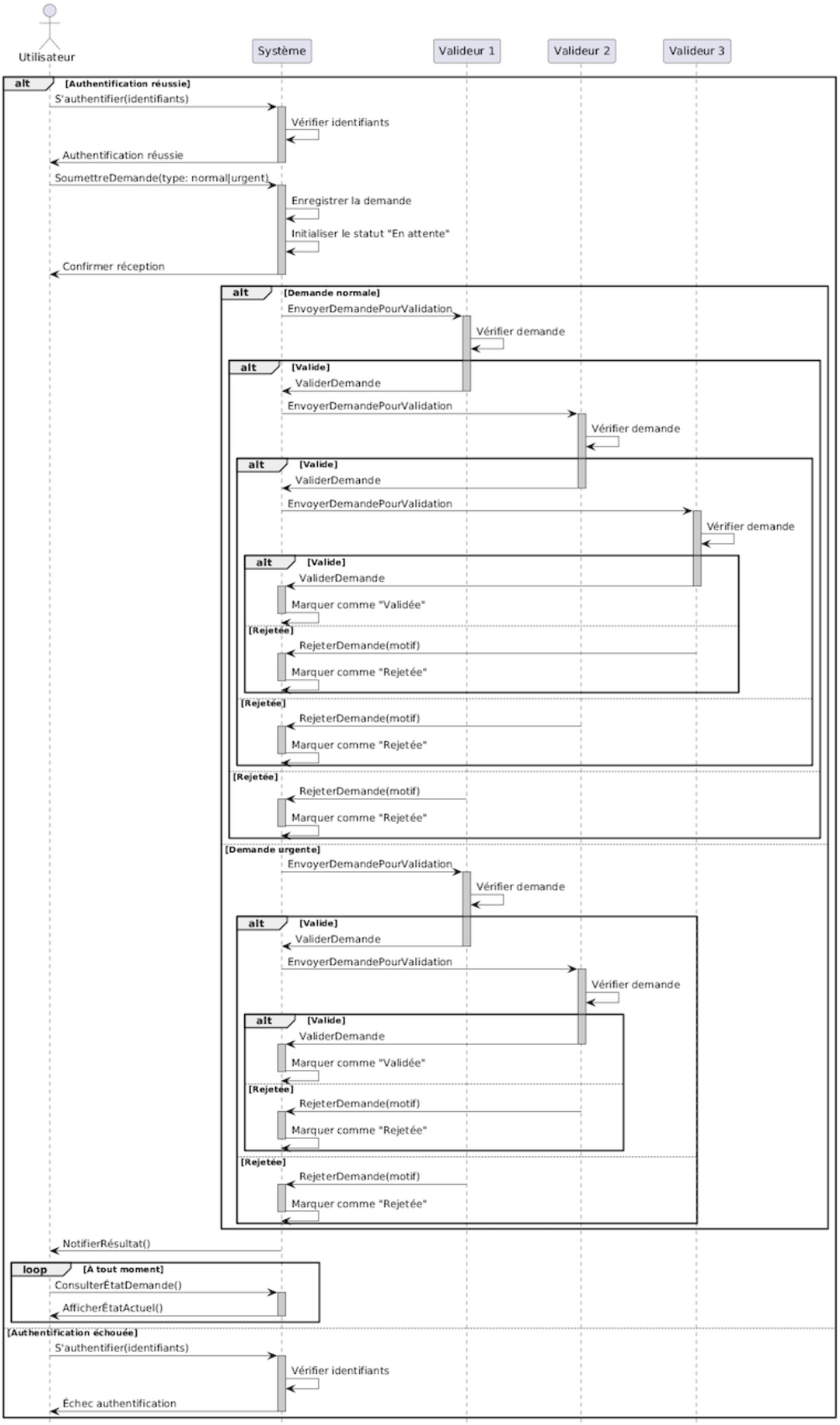

FIGURE 4.27 – Diagramme de Sequence du cas "Valider une demande d'achat"

3. **Diagramme d'état transition** La figure suivante présente le diagramme d'état transition du cas "Valider une demande d'achat", il retrace les differents états par lesquelles une demande passe.

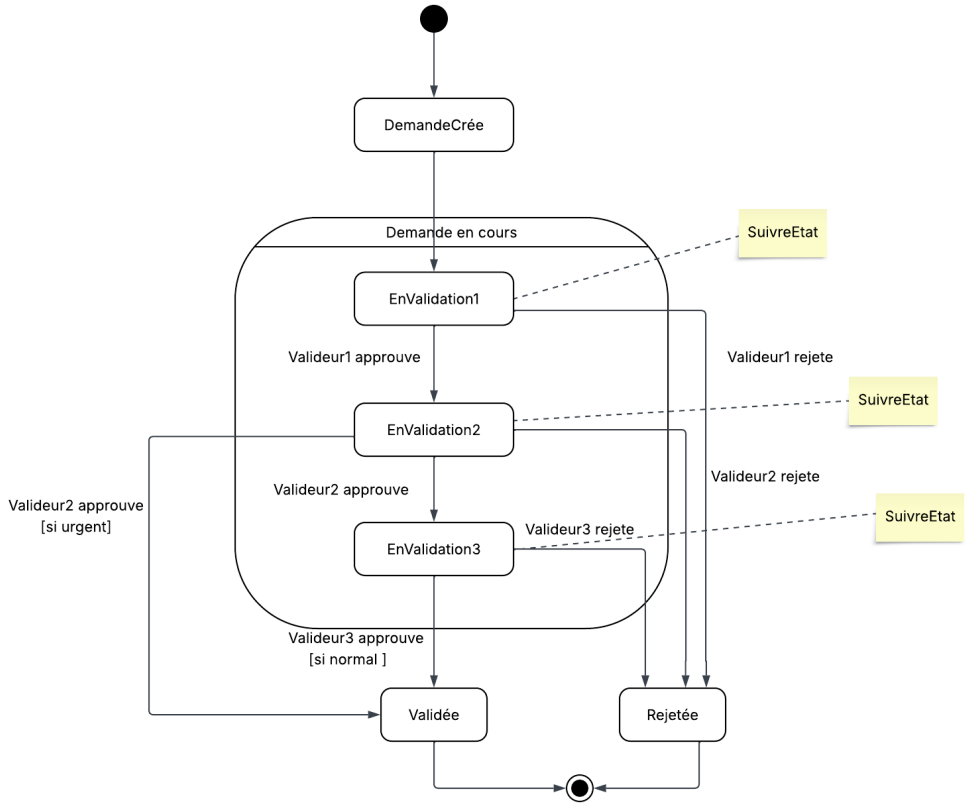

FIGURE 4.28 – Diagramme d'état transition du cas "Validation d'une demande d'achat"

4. **Diagramme de classe** La figure 4.29 ci-dessous décrit le diagramme de classe elaboré pour le Sprint 3 :

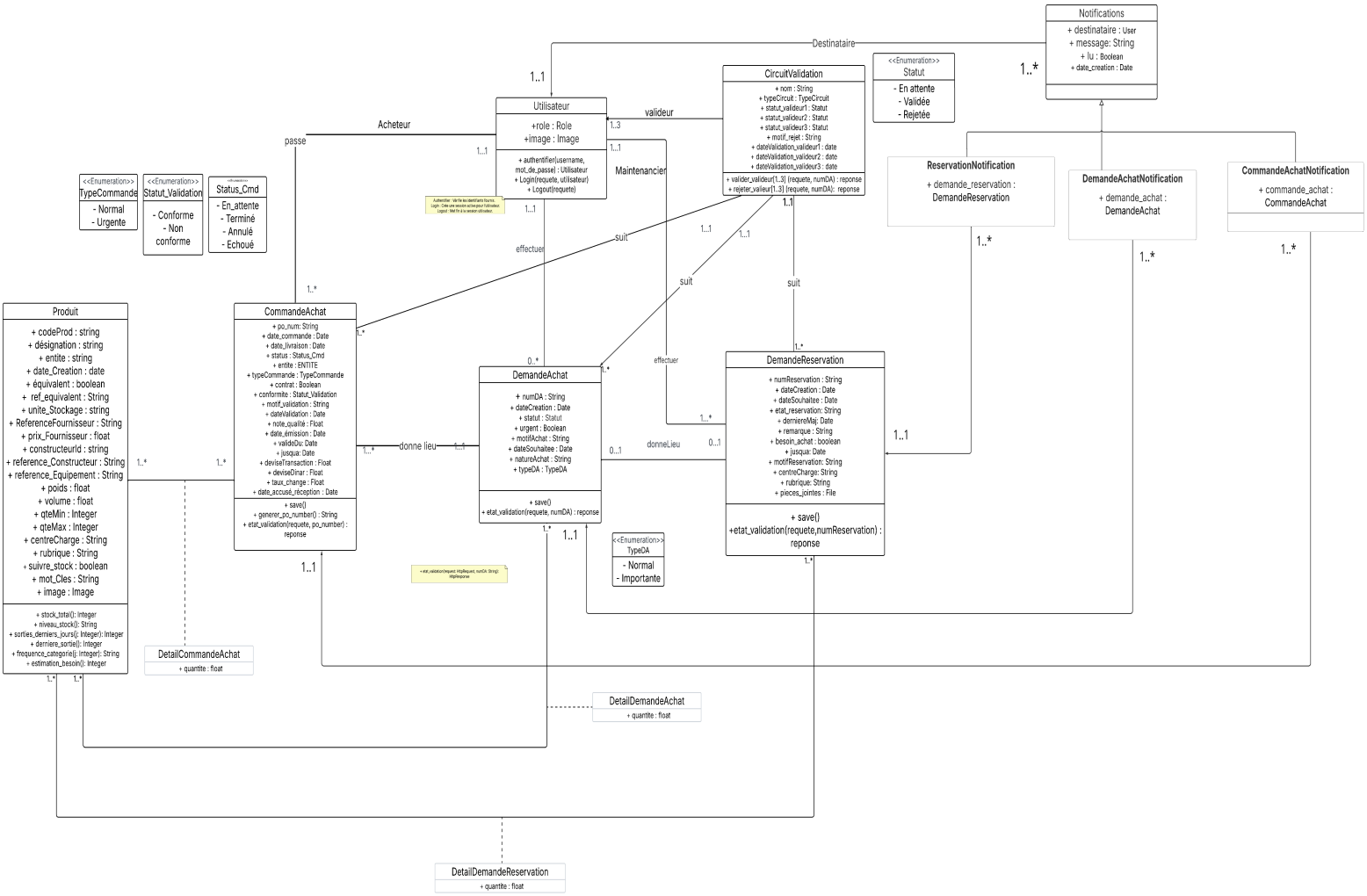

FIGURE 4.29 – Diagramme de classe du Sprint 3

3.3.4 Développement

Dans le cadre du Sprint 3, nous avons mis en place une architecture modulaire pour structurer efficacement les différentes étapes du processus d'achat en développant trois applications Django distinctes chacune dédiée à un type spécifique de demande :

1. **Module `DemandeReservation`**
   Ce module permet de gérer les demande de réservation et permet aux utilisateurs du groupe *maintenancier* de soumettre une demande pour une ou plusieurs pièces de rechange, suite à l'expression d'un besoin qui peut être :
   - **correctif** telle une panne constatée.
   - **préventif** dans le cas d'un entretien planifié sur une machine.
   - **amélioratif** pour améliorer la performance d'une machine.

   Le créateur spécifie aussi une liste de un ou plusieurs pièces, ainsi la date pour laquelle ces dernières sont requises.

2. **Module `DemandeAchat`**
   Ce module permet aux utilisateurs de soumettre des demande d'achat, soit de manière directe en remplissant un formulaire de création, en spécifiant le type : normale ou urgente,

une liste d'un ou plusieurs pièces la date pour laquelle elles sont requises. Soit de façon indirecte à partir d'une demande de réservation validée.

3. **Module `CommandeAchat`**
   Ce module permet la gestion des commandes d'achat. Elles sont créer par les utilisateurs du groupe *acheteur* ou *directeurGénéral* à partir d'une demande d'achat validée en spécifiant le fournisseur ainsi que le contrat.

Le modèle `CircuitValidation`

L'un des aspects principaux de ce sprint, est la mise en place d'un circuit de validation pour tout type de demande : achat, réservation et commande d'achat créee. La table `CircuitValidation` formalise les étapes de validation propre à chaque type de demande :

Listing 4.4 – Table Circuit Validation

```
class CircuitValidation(models.Model):
    nom = models.CharField(max_length=255, default='normal')
    valideur1 = models.ForeignKey(User, related_name='valideur1',
        on_delete=models.CASCADE,)
    statut_valideur1 = models.CharField(max_length=10, choices=[(
       'en attente', 'En Attente'), ('validee', 'Validee'), ('
       rejetee', 'Rejetee')], default='en attente')
    date_validation_valideur1 = models.DateTimeField(null=True,
       blank=True)


    valideur2 = models.ForeignKey(User, related_name='valideur2',
        on_delete=models.CASCADE, null=True, blank=True)
    statut_valideur2 = models.CharField(max_length=10, choices=[(
       'en attente', 'En Attente'), ('validee', 'Validee'), ('
       rejetee', 'Rejetee')], default='en attente')
    date_validation_valideur2 = models.DateTimeField(null=True,
       blank=True)

    valideur3 = models.ForeignKey(User, related_name='valideur3',
        on_delete=models.CASCADE, null=True, blank=True)
    date_validation_valideur3 = models.DateTimeField(null=True,
       blank=True)
    statut_valideur3 = models.CharField(max_length=10, choices=[(
       'en attente', 'En Attente'), ('validee', 'Validee'), ('
       rejetee', 'Rejetee')], default='en attente')
    type_circuit = models.CharField(max_length=10, choices=[('
       normal', 'Normal'), ('urgent', 'Urgent')])
    motif_rejet = models.TextField(null=True, blank=True)
```

Dans le cas d'une commande ou demande d'achat le type de circuit et le nombre de valideurs varient selon le créateur et le type : urgente ou normale, de la demande :

— **Circuit normal** : dans le cas d'une demande d'achat ou d'une commande de type normale, un circuit standard de type "Normal" est assigné à cette dérniere et il implique trois niveaux de validations :

1. Responsable approvisionnement
2. Responsable achat
3. Directeur général

— **Circuit urgent** : pour répondre à un besoin nécessitant une réactivité accrue et une prise de décision accélérée, une demande d'achat ou une commande de type urgente est créée. Le circuit urgent prévoit deux valideurs seulement, la validation du Directeur général n'est pas requise afin de gagner plus de temps. Ce circuit implique donc :

1. Responsable approvisionnement
2. Responsable achat

— **Cas Particulier** : les demandes d'achats créé par le responsable achat ne requiert que deux valideurs qui sont : le responsable lui même et le directeur général. Les demandes d'achat crée par le directeur général ne requiert que sa propre validation.

Dans le cas d'une demande de réservation, un seul type de circuit est associé à chaque demande,avec deux valideurs qui sont :

1. Responsable approvisionnement
2. Magasinier

4. **Justification des choix des valideurs**
Le choix des valideurs dans chaque circuit a été guidé par des considérations de risque et de rapidité de traitement.

— **Demande d'Achat :**

1. Le responsable approvisionnement est le premier valideur, car il a une connaissance directe des seuils de stock ainsi que des planifications logistique. Son rôle est d'évaluer la pertinence de la demande.
2. Une fois la demande validée par le responsable approvisionnement, le second valideur : le responsable d'achat vérifie la faisabilité sur le plan financiers et vérifie les conditions commerciales.
3. Le directeur général est le dernier valideur dans le cas d'un circuit de type "Normal", il intervient pour donner un accord final, notamment pour les montants élevés.

— **Commande d'Achat**

1. Le responsable approvisionnement vérifie la conformité de la commande par rapport aux demandes d'achat validées, il s'assure également que la commande respecte les quantités prévues.
2. Le responsable achat intervient en deuxiéme après le premier valideur, afin de s'assurer que la commande est bien conforme à la politique d'achat de l'organisation.
3. Le directeur Général est le dernier valideur car son intervention est nécessaire pour autoriser des dépenses importantes, et éviter tout engagement non necessaire.

— **Demande de réservation** : puisqu'il s'agit d'un mouvement interne en stock, deux valideurs en lien avec l'approvisonnement et le stock suffisent :

1. Le responsable approvisionnement se charge de vérifier que la demande de réservation est justifiée par un besoin réel et légitime. Son rôle est crucial pour éviter les réservations excessives.
2. Le magasinier après la validation du responsable approvisionnement, le magasinier se charge de vérifier la disponibilité des quantités spécifiées dans la demande.

5. **Le module `notifications` :**
   Afin de rendre le processus plus interactif et d'assurer une meilleure traçabilité, un système de notifications a été mis en place. Celui-ci permet d'informer en temps réel les utilisateurs concernés à chaque étape de validation d'une demande, tels que :
   — Notifier les valideurs qu'une demande (achat ou reservation) ou une commande est en attente de leur validation. Voici un exemple de code qui permet de notifier automatiquement le premier valideur du circuit (le responsable approvisionnement) lorsqu'une demande ou une commande est créée et nécessite son évaluation :

Listing 4.5 – Vue de notifiaction du premier valideur

```
def notifier_responsable_appro(obj):
    """
    Notifie le responsable achat a la creation d'une DA, DR ou
        CA.
    """

    class_name = obj.__class__.__name__

    if class_name == 'DemandeAchat':
        circuit = getattr(obj, 'circuit_validation', None)
        if circuit and circuit.nom not in ['resp_achat', '
           dg_seul']:
            responsable = getattr(circuit, 'valideur1', None)
            if responsable:
                DemandeAchatNotification.objects.create(
                    destinataire=responsable,
                    demande_achat=obj,
                    message=f"Nouvelle commande d'achat #{obj.
                       id} en attente de votre validation."

                )


    elif class_name == 'PurchaseOrder':
        responsable = getattr(obj.workflow, 'valideur1', None)
        if responsable:
            CommandeAchatNotification.objects.create(
                destinataire=responsable,
                commande_achat=obj,
                 message=f"Nouvelle commande d'achat #{obj.id}
                     en attente de votre validation."

            )

    elif class_name == 'DemandeReservation':
        responsable = getattr(obj, 'valideur_responsableAppro'
           , None)

        if not responsable:
            try:
                groupe = Group.objects.get(name='
                   responsableAppro')
```

```
                responsable = groupe.user_set.first()
            except Group.DoesNotExist:
                responsable = None

        if responsable:
            ReservationNotification.objects.create(
                destinataire=responsable,
                demande_reservation=obj,
                message=f"Nouvelle commande d'achat #{obj.id}
                    en attente de votre validation."

            )
```

Une fois la demande ou la commadne est créée, l'envoi de la notification est déclenchée. Voici un extrait de code, situé dans la vue de création d'une demande d'achat, qui illustre ce mécanisme :

Listing 4.6 – Déclenchement de la notification après la création d'une demande

```
    demande.circuit_validation = circuit  # Creation du
        circuit de validation
    demande.save()  # Enregistrement de la demande d'achat
    notifier_responsable_appro(demande)  # Notification du
        premier valideur
```

— Notifier le demandeur (le createur de la demande ou commande) que sa demande ou commande est évaluée (validée ou rejetée), afin de l'informer du statut final et de l'inviter à consulter les détails. Le code suivant illustre la fonction utilisée pour générer ces notifications en fonction du type de l'objet concerné :

Listing 4.7 – Vue de notification du createur d'une demande/commande

```
    def notifier_createur(obj):

    demandeur = getattr(obj, 'demandeur', None)
    if not demandeur:
        return

    class_name = obj.__class__.__name__

    # Pour DA
    if class_name == 'DemandeAchat':
        if obj.statut in ['validee', 'rejetee']:
            message = f"Votre demande d'achat #{obj.numDa} a
                ete {obj.statut}. Consultez vos demandes."
            DemandeAchatNotification.objects.create(
                destinataire=demandeur,
                demande_achat=obj,
                message=message
            )
    # Pour DR
    elif class_name == 'DemandeReservation':
        if obj.etat_reservation in ['validee', 'rejetee']:
            message = f"Votre demande de reservationtion #{obj
```

```
                .id} a ete {obj.etat_reservation}. Consultez
                vos demandes."
            ReservationNotification.objects.create(
                destinataire=demandeur,
                demande_reservation=obj,
                message=message
            )
    # Pour PO
    elif class_name == 'PurchaseOrder':
        if obj.status in ['conforme', 'non conforme']:
            message = f"Votre commande d'achat #{obj.id} a ete
                marquee comme {obj.status}."
            CommandeAchatNotification.objects.create(
                destinataire=demandeur,
                commande_achat=obj,
                message=message
            )
```

L'appel à cette fonction se fait à la fin du processus de validation ou de rejet de la demande. Voici deux exemples dans le cas d'une demande d'achat validée et rejetée :

Listing 4.8 – Notification du créateur lors de l'évaluation

```
      # Cas de validation
      demande.statut = 'validee'
      demande.save()
      notifier_createur(demande)

     #Cas de rejet
        demande.statut='rejetee'
        demande.save()
        notifier_createur(demande)
```

Affichage des notifications

Afin de rendre les notifications accessibles en temps réel depuis tous les templates(via une icône en cloche dans la barre de navigation), nous avons utilisé un *context processor*. Ce mécanisme propre à Django, permet d'injecter automatiquement, dans chaque page, les notifications non lues de l'utilisateur connecté, sans avoir à les recalculer manuellement dans chaque vue. Ainsi tout utilisateur connecté peut consulter ses notifications en permanence.

Voici l'extrait de code utilisé dans le fichier `context_processors.py` :

Listing 4.9 – Injection des notifications non lues

```
def navbar_notifications(request):
if not request.user.is_authenticated:
    return {}

da_notifs = DemandeAchatNotification.objects.filter(
   destinataire=request.user, lu=False)
dr_notifs = ReservationNotification.objects.filter(
   destinataire= request.user, lu=False)
po_notifs = CommandeAchatNotification.objects.filter(
   destinataire= request.user, lu=False)
```

```
    all_notifs = list(da_notifs)  + list(dr_notifs) + list(
        po_notifs)
    all_notifs.sort(key=lambda x: x.date_creation, reverse=True)

    return {
        'notifications': all_notifs[:5],  # les 5 dernieres
        'notification_count': len(all_notifs)
    }
```

6. **Choix des KPI en fonction des utilisateurs :**

L'implémentation du tableau de bord a été conçue selon une logique **personnalisée et orientée métier**, afin de proposer à chaque utilisateur des **indicateurs clés de performance (KPI)** spécifiques à ses responsabilités. L'objectif est d'assurer une prise de décision efficace, un meilleur suivi opérationnel, et une vision synthétique des tâches quotidiennes. Ci-dessous, une présentation détaillée des KPI retenus pour chaque rôle :

- **Responsable approvisionnement** : Le responsable approvisionnement intervient en amont du processus, en validant les demandes (achat et réservation). Il accède à :
  - **Demandes d'Achat (DA)** : total, traitées, en attente, taux de traitement et délai moyen.
  - **Demandes de Réservation (DR)** : total, traitées, en attente, taux de traitement et délai moyen.
- **Directeur général** Le directeur général a une vision macro du processus. Ses KPI sont axés sur la performance financière :
  - **Demandes d'Achat (DA)** : total, traitées, en attente, taux de traitement et délai moyen.
  - **Commandes d'Achat** : total, traitées, en attente, taux de traitement et délai moyen.
  - **Valeur du stock** : par pièce et valeur totale.
  - **Montant des achats validés** : valeur cumulée des commandes traitées.
- **Maintenancier** Acteur de terrain, le maintenancier initie des réservations de pièces. Les indicateurs sont :
  - **Demandes d'Achat (DA)** : total, validées, en attente, rejetées.
  - **Demandes de Réservation (DR)** : total, validées, en attente, rejetées.

Ces KPI ont été essentiels pour assurer un suivi rigoureux des demandes (achat et réservation) et optimiser les validations à chaque étape du processus.

3.3.5 Réalisation

Les figures suivantes illustrent les interfaces développées dans le cadre du Sprint 3. Elles ont été conçues et ajustées à la suite des retours des utilisateurs, collectés lors d'une première version de la solution.

La figure 4.30 montre l'interface utilisateur du suivi des état de validation d'une demande d'achat de l'utilisateur connecté :

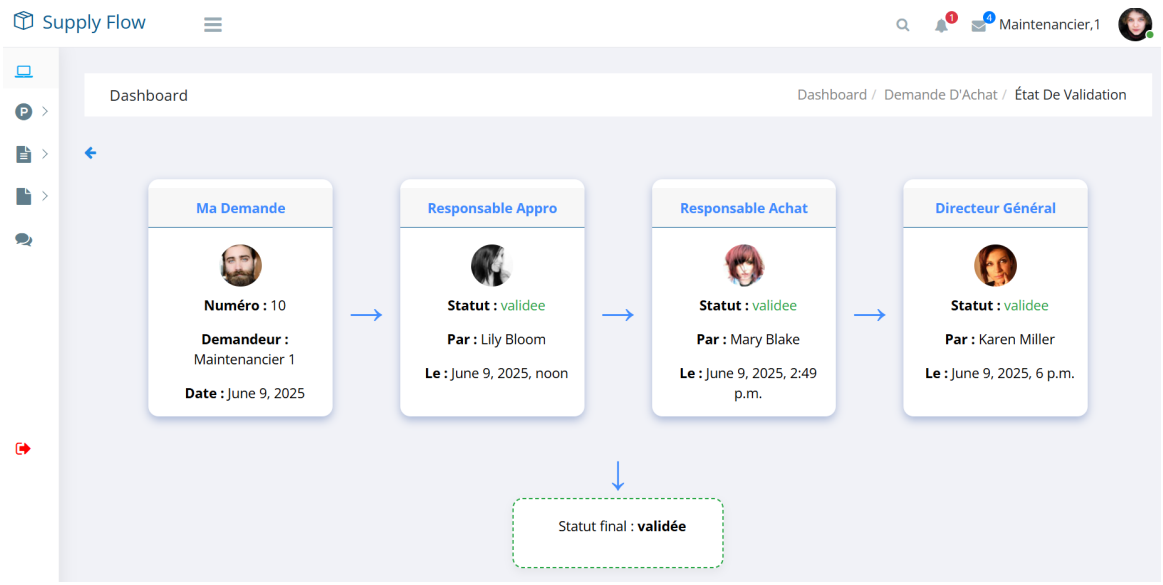

FIGURE 4.30 – Le Suivi de l'état de validation d'une demande d'achat

La figure 4.31 montre l'interface utilisateur d'une demande d'achat en attente de validation par un valideur :

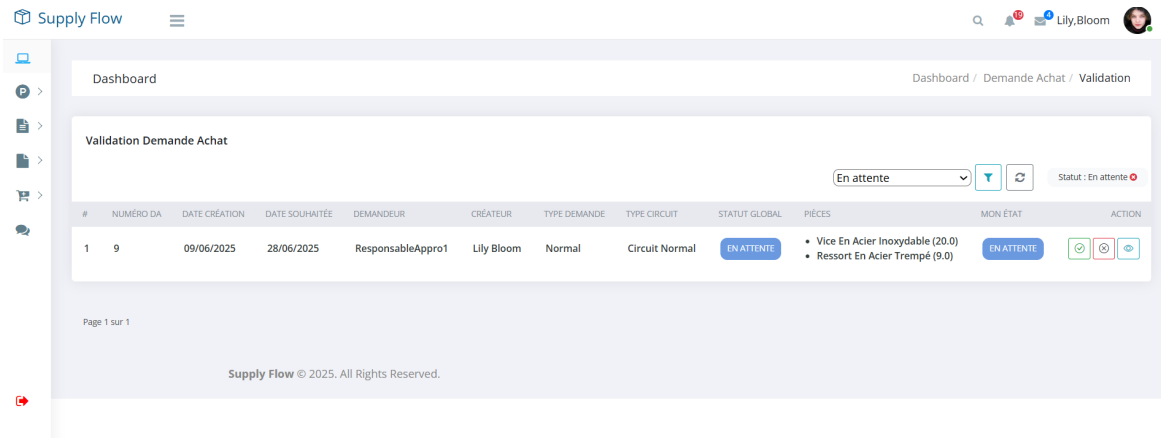

FIGURE 4.31 – Liste des demandes d'achat en attente de validation

3.4 Sprint 4 : Gestion des fournisseurs et des contrats

3.4.1 Objectif :

Ce dernier sprint se concentre sur la gestion des fournisseurs, le suivi de leurs performances, ainsi que la gestion des contrats, des bons de réception et des sorties de stock. Les utilisateurs peuvent créer et valider des contrats, visualiser des indicateurs de performance (tels que le respect des délais), et suivre les mouvements de stock associés aux livraisons ou aux sorties.

Le tableau 4.13 ci-dessous représente le Sprint Backlog de ce dernier sprint.

Fonctionnalités	**Description**	**User Stories associées**
S4-F01	Création et gestion des fournisseurs	US19

S4-F02	Évaluation des performances des fournisseurs (qualité, délais, exécution)	US19
S4-F02	Consulter l'historique des performances des fournisseurs	US19
S4-F03	gestion des contrats fournisseurs	US21
S4-F04	Gestion des RFA, prises en charge et cycle de vie	US21
S4-F05	Evaluation de la conformité des contrats	US22
S4-F06	Suivi des contrats et des engagements contractuels	US21, US22
S4-F07	Suivi des sorties de stock (bons de sortie)	US25
S4-F08	Génération des bons de sorties	US25
S4-F09	Suivi des arrivages de stock	US24
S4-F10	Génération des bons d'entrée	US24
S4-F11	Suivi global des actions et supervision par le Directeur Général	US23
S4-F12	Finaliser la Suivi des stock	US04
S4-F13	Finaliser les Notifications	US26
S4-F13	Consulter le tableau de bord	US09

TABLE 4.13: Sprint Backlog du Sprint 4

3.4.2 Prototype

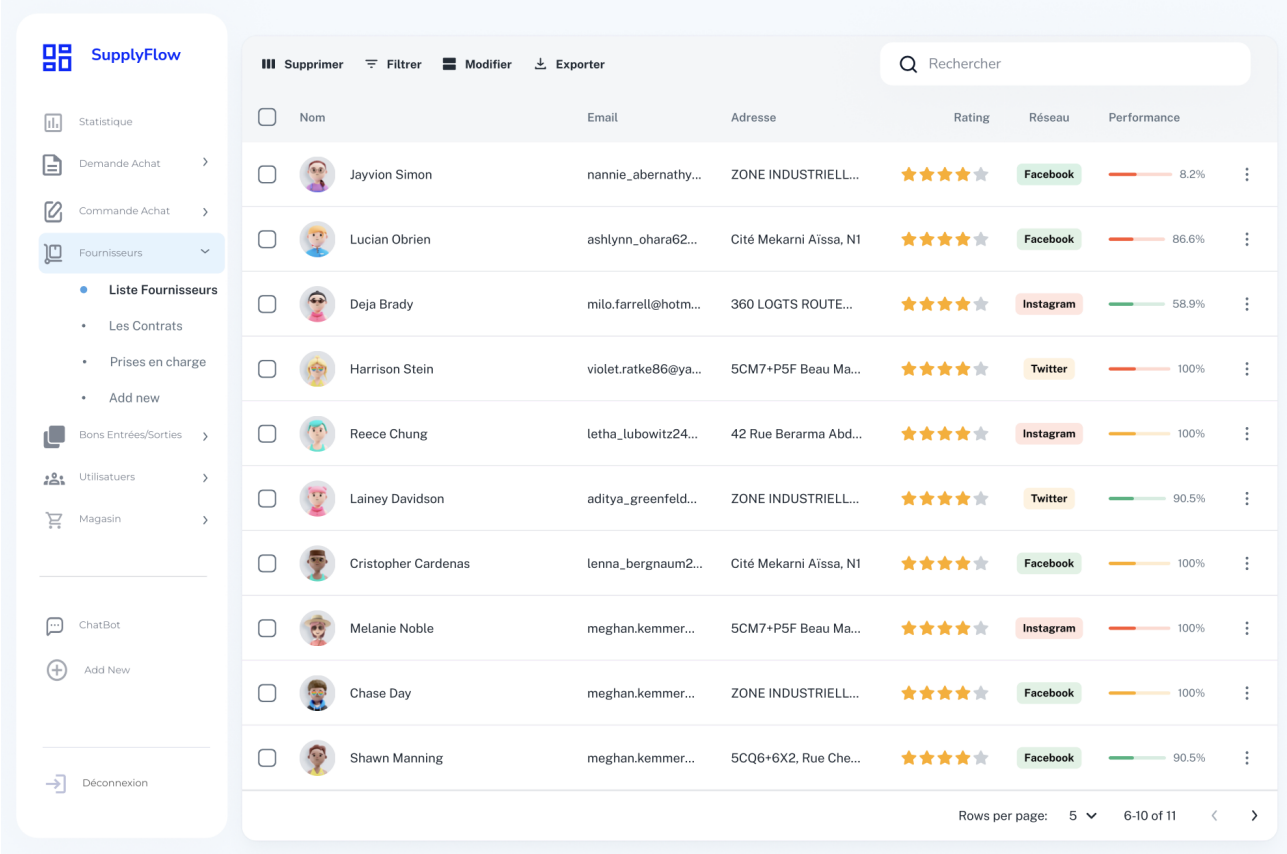

FIGURE 4.32 – Design de l'interface de la liste des fournisseurs

3.4.3 Conception

Dans cette partie, nous présentons les principaux diagrammes UML réalisés lors du Sprint 4. Ils ont pour but de modéliser les fonctionnalités à développer, les interactions entre les acteurs, ainsi que la structure du système.

1. **Diagramme de cas d'utilisation** L'illustration ci dessous 4.33 présente le diagramme de cas d'utilisation du dérnier Sprint.

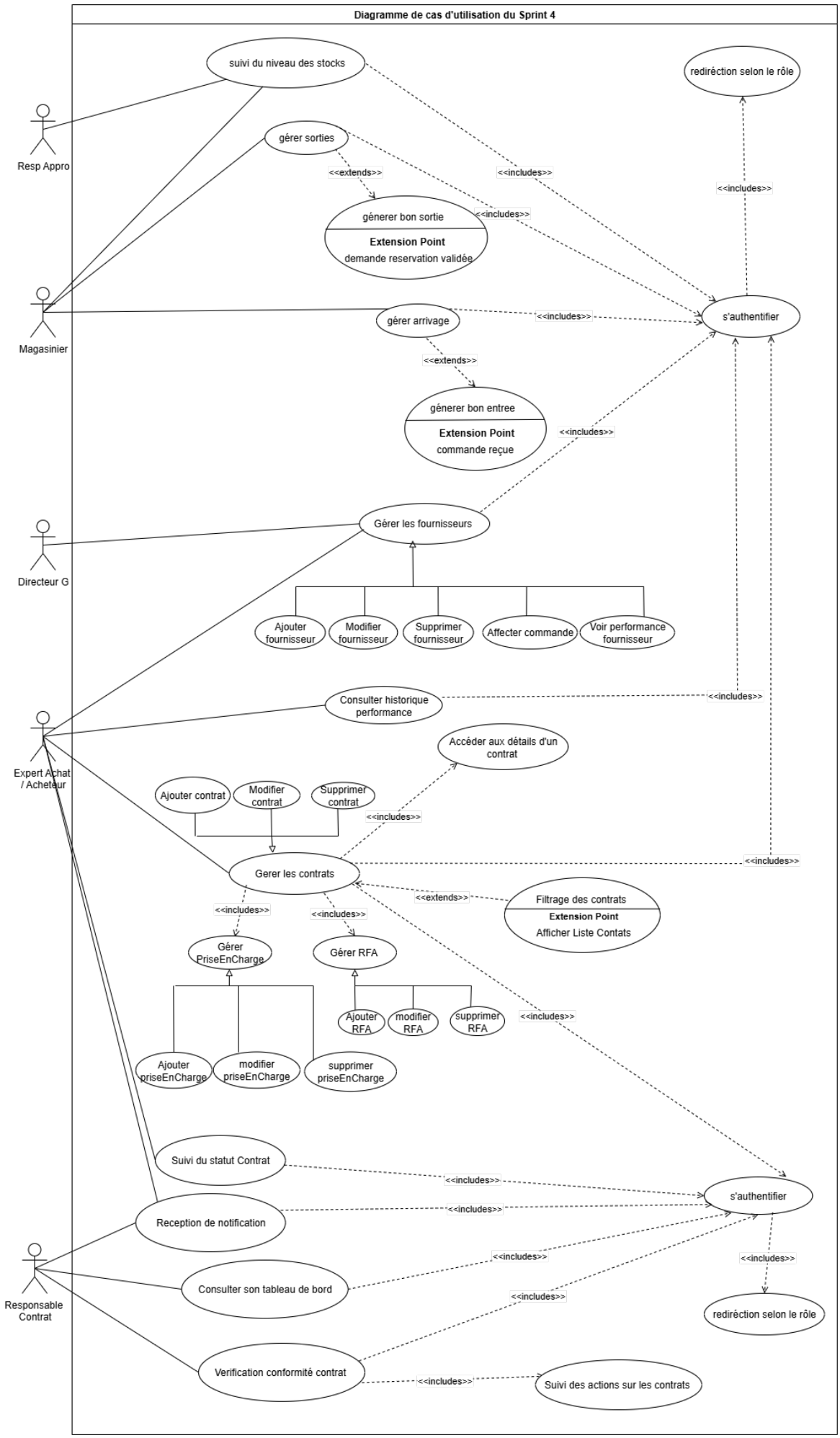

FIGURE 4.33 – Diagramme de cas d'utilisation du Sprint 4

Ce tableau ci-dessous 4.14 montre la description textuelle du cas d'utilisation "Évaluation des performances des fournisseurs" :

Nom du cas : Évaluation des performances des fournisseurs
But : Permettre au magasinier d'évaluer objectivement les fournisseurs en fonction de la qualité des livraisons et du respect des délais, afin d'éclairer les décisions d'achat futures.
Acteur principal : Magasinier
Acteur secondaire : Système de gestion des fournisseurs
Date de création : 23/05/2025
Date de mise à jour : 01/07/2025
Responsable : Équipe technique de développement.
Version : 1.0
Séquencement : Le cas d'utilisation commence lorsqu'un magasinier souhaite effectuer une évaluation suite à la réception d'une commande clôturée.
Préconditions : — La commande associée au fournisseur a été livrée et clôturée. — Les données de réception, contrôle qualité et délai sont disponibles.
Enchaînement nominal : (a) Le magasinier accède au module d'évaluation des fournisseurs. (b) Il sélectionne un fournisseur à partir de l'historique des commandes. (c) Il renseigne les critères d'évaluation : — Qualité des produits livrés, respect des délais de livraison, taux de conformité ou d'exécution, commentaires éventuels ... (d) Il valide l'évaluation. (e) Le système enregistre les notes et les commentaires. (f) Le score global du fournisseur est mis à jour dans l'historique.
Enchaînements alternatifs : **A1 : Données de livraison incomplètes** — Le système bloque la soumission de l'évaluation. — Un message informe l'utilisateur de l'impossibilité d'évaluer sans données complètes. — Le magasinier peut vérifier et attendre la mise à jour des données.
Enchaînements d'exception : **E1 : Échec de l'enregistrement de l'évaluation** — Une erreur technique empêche l'enregistrement. — Le système affiche un message d'échec et l'évaluation est annulée.
Postconditions : — L'évaluation du fournisseur est enregistrée. — Le score de performance est actualisé. — L'information est disponible pour les prochaines décisions d'achat.

TABLE 4.14 – Description textuelle du cas d'utilisation "Évaluation des performances des fournisseurs"

2. **Diagramme de séquence** Le diagramme illustré dans la figure 4.34 ci-dessous, decrit l'aspect chronologique du cas d'utilisation "Gerer les Arrivages et les Sorties" du quatrième sprint.

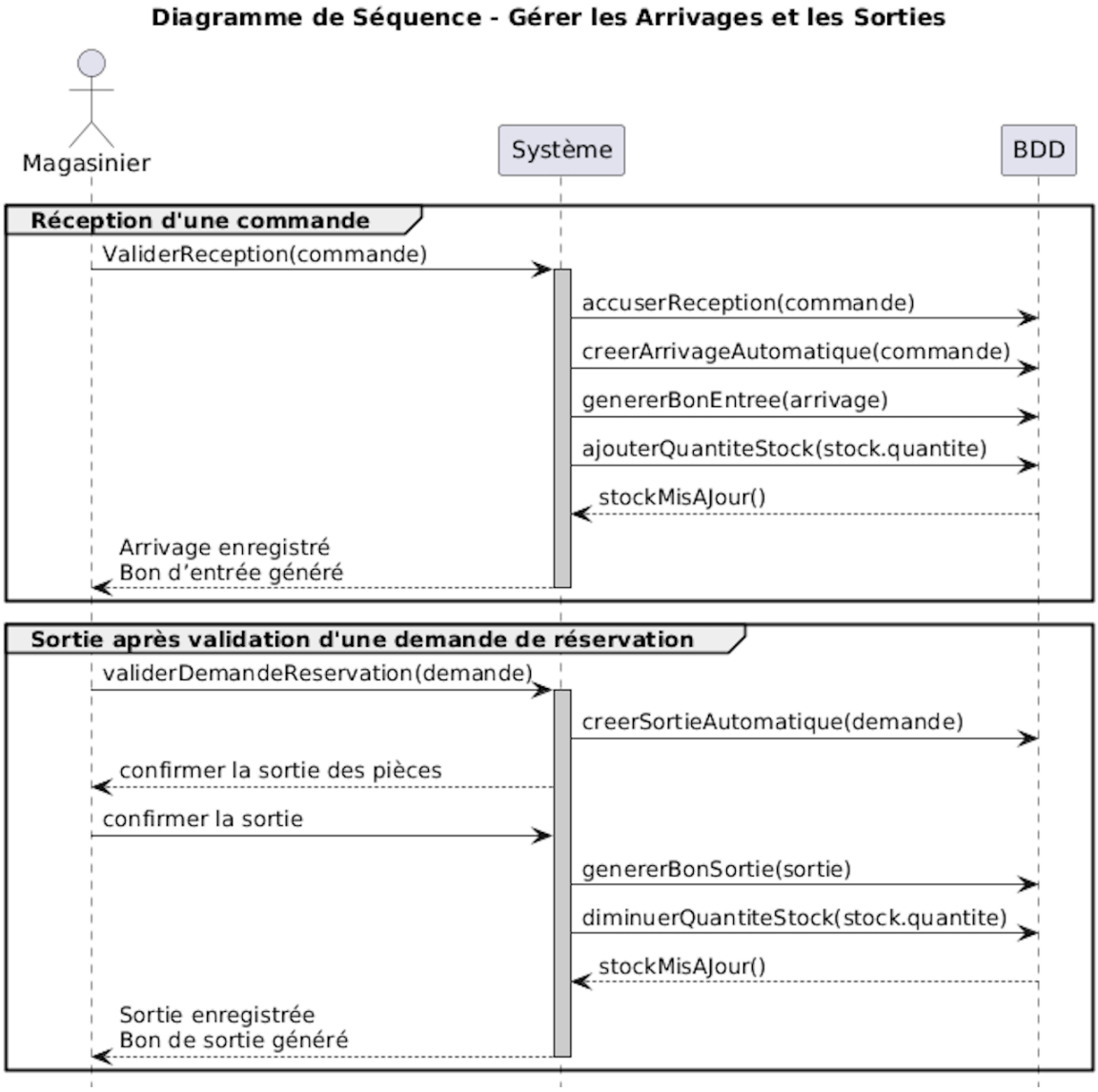

FIGURE 4.34 – Diagramme de Sequence du cas "Gerer les Arrivages et les Sorties"

Le diagramme illustré dans la figure 4.35 ci-dessous, decrit l'aspect chronologique du cas d'utilisation "Ajout contrat et Validation contrat" du quatrième sprint.

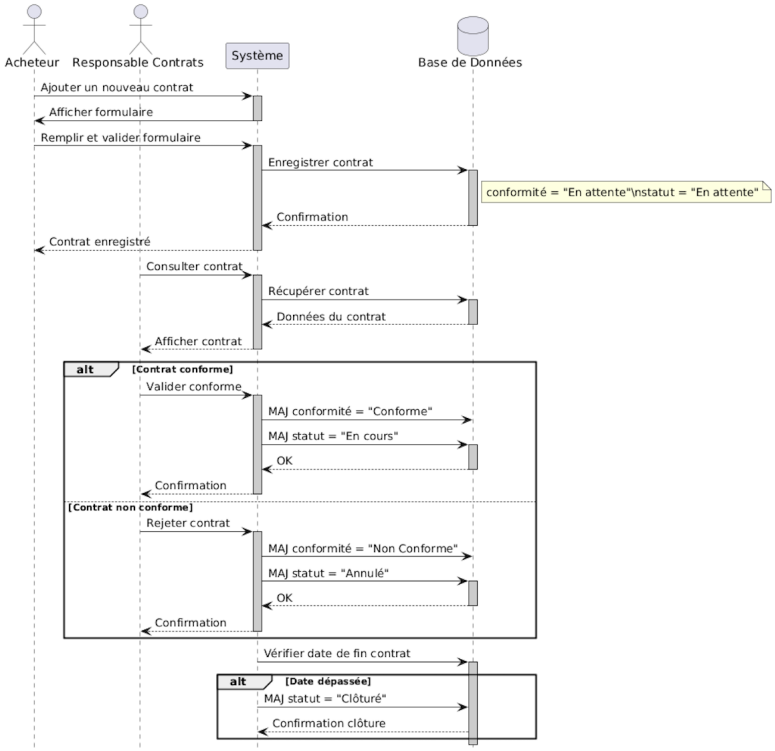

FIGURE 4.35 – Diagramme de Sequence du cas "Ajout contrat et Validation contrat"

3. **Diagramme d'état transition** La figure suivante présente le diagramme d'état transition du cas "Gerer les Arrivages et les Sorties", il retrace les differents états pour la creation des arrivage, des sorties ainsi que des bons associés.

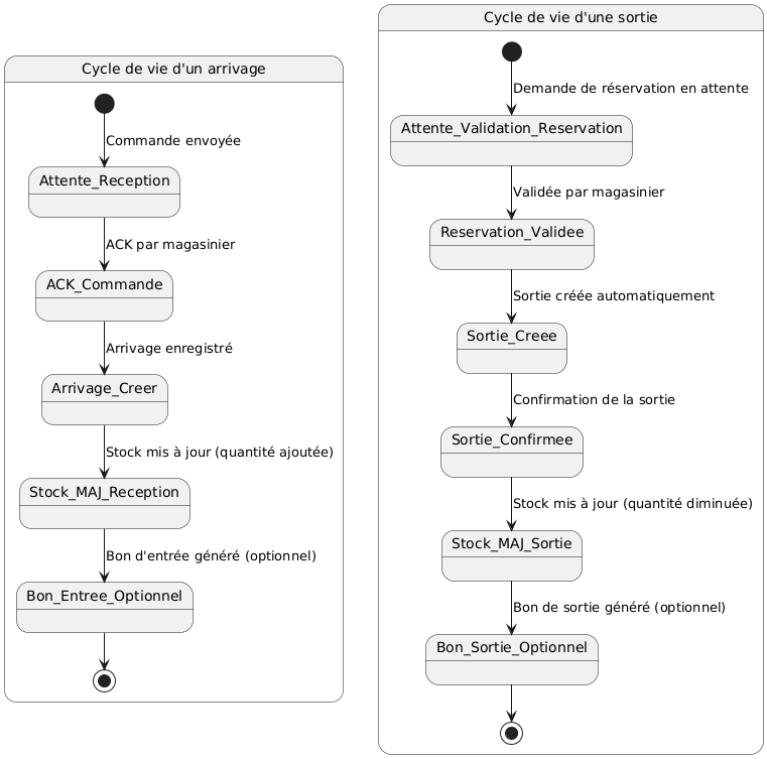

FIGURE 4.36 – Diagramme d'état transition du cas "Gerer les Arrivages et les Sorties"

4. **Diagramme de classe** La figure ci-dessous 4.37, illustre le diagramme de cas d'utilisation du sprint 4

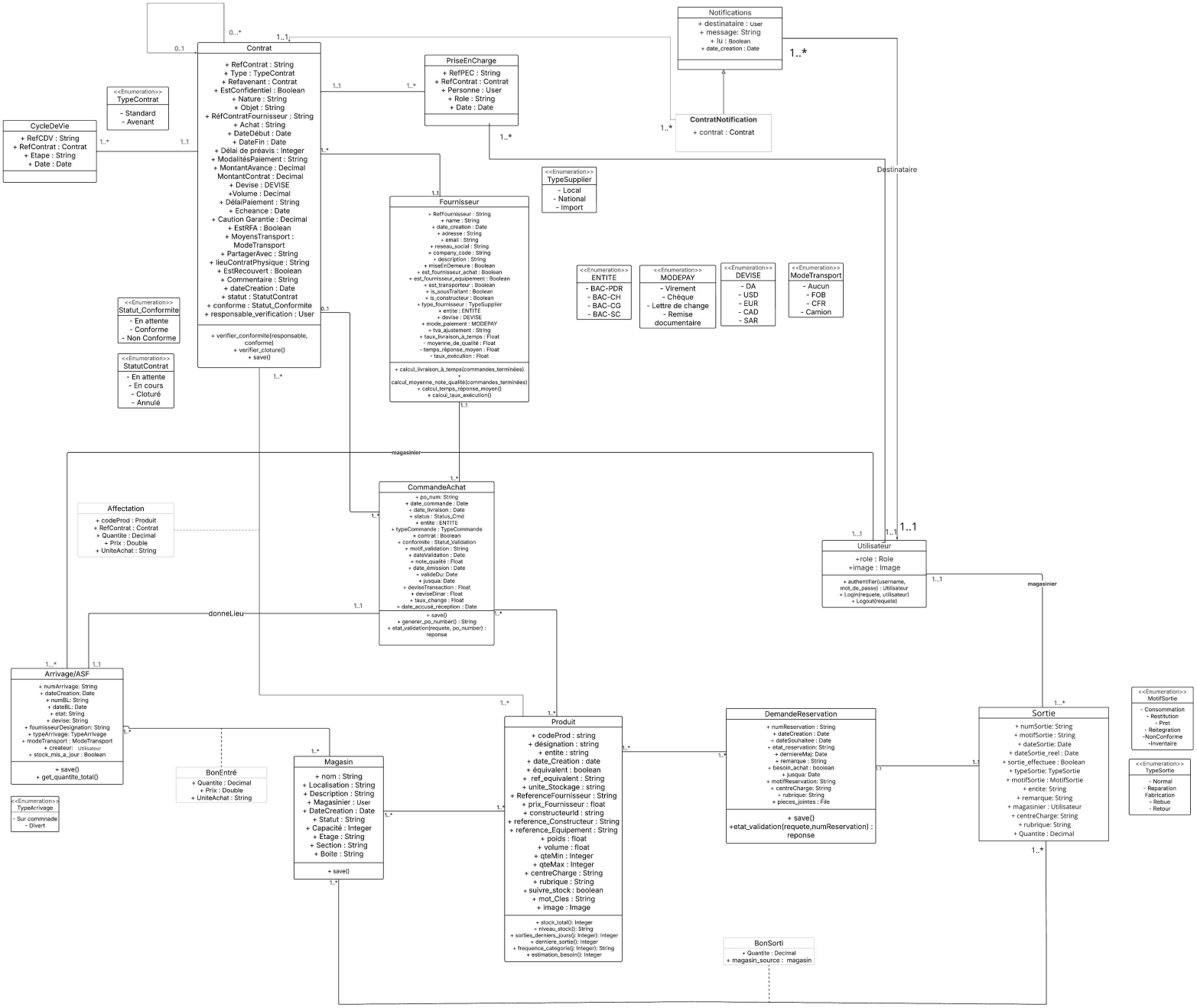

FIGURE 4.37 – Diagramme de classe du Sprint 4

3.4.4 Développement

Nous avons mis en place un ensemble de modèles permettant de gérer efficacement les fournisseurs, leurs contrats, ainsi que les différentes étapes liées au cycle de vie des relations contractuelles. Voici les principales réalisations :

1. Modèle `Supplier` – Gestion des fournisseurs

Le modèle `Supplier` centralise toutes les informations relatives aux fournisseurs : identifiants, coordonnées, statut, type (local, national, importé), entité associée (ex : BAC-PDR), devise et mode de paiement. Des champs booléens permettent également d'identifier leur rôle (fournisseur achat, constructeur, etc.).

Évaluation de la performance des fournisseurs

Dans le cadre du Sprint 4, nous avons intégré plusieurs indicateurs de performance dans le module de gestion des fournisseurs, afin de permettre un pilotage objectif et basé sur des données mesurables.

a. Taux de livraison à temps (`on_time_delivery_rate`)

But : Évaluer la ponctualité du fournisseur.

— Les commandes complétées (`completed_orders`) sont analysées.

— Si un `arrivage` a été enregistré avant ou à la date de livraison prévue, la commande est considérée comme livrée à temps.

— Le taux est calculé comme suit :

$$\text{Taux de livraison à temps} = \left(\frac{\text{Commandes livrées à temps}}{\text{Total commandes complétées}} \right) \times 100$$

Listing 4.10 – Classe Supplier et ses méthodes

```
    def on_time_delivery_calculator(self, completed_orders):
    from arrivage.models import Arrivage
    total_completed_orders = completed_orders.count()

    if total_completed_orders == 0:
        self.on_time_delivery_rate = None
        self.save()
        return

    on_time_delivered_count = 0
    for order in completed_orders:
        expected_date = order.delivery_date
        if expected_date:
            arrivages = Arrivage.objects.filter(numCommande=order)
                .order_by('dateCreation')
            for arrivage in arrivages:
                if arrivage.dateCreation and arrivage.dateCreation <=
                   expected_date:
                    on_time_delivered_count += 1
                    break

    self.on_time_delivery_rate = (on_time_delivered_count /
       total_completed_orders) * 100
    self.save()
```

b. Note moyenne de qualité (`quality_rating_avg`)

But : Mesurer la qualité des livraisons.

— Seules les commandes avec une note qualité renseignée (`quality_rating`) sont prises en compte.

— La moyenne est calculée selon la formule :

$$\text{Note qualité moyenne} = \frac{\text{Somme des notes qualité}}{\text{Nombre de commandes notées}}$$

Listing 4.11 – Calcul de la moyenne des notes de qualité

```
def quality_rating_avg_calculator(self,completed_orders):
        quality_rated_order = completed_orders.filter(
           quality_rating__isnull = False)
        total_quality_rated_order = quality_rated_order.count()
        if total_quality_rated_order > 0:
            quality_rating_sum = quality_rated_order.aggregate(Sum('
               quality_rating'))
                 ['quality_rating__sum']
            self.quality_rating_avg = quality_rating_sum /
               total_quality_rated_order
        else:
            self.quality_rating_avg = 0
        self.save()
```

c. Temps moyen de réponse (`average_response_time`)

But : Évaluer la réactivité du fournisseur.

— On mesure l'écart entre la date d'émission de la commande (`issue_date`) et la date d'accusé de réception (`acknowledgment_date`).

— La moyenne est exprimée en heures :

$$\text{Temps moyen de réponse (h)} = \frac{\text{Total des durées (en heures)}}{\text{Nombre de commandes reconnues}}$$

Listing 4.12 – Calcul du temps moyen de réponse

```
def average_response_time_calculator(self):
    acknowledged_orders = self.purchaseorder_set.filter(
       acknowledgment_date__isnull=False)

    total_response_time = 0
    total_orders = 0
    for order in acknowledged_orders:
        if order.issue_date and order.acknowledgment_date:
            time_diff = order.acknowledgment_date - order.issue_date
            time_diff_seconds = abs(time_diff.total_seconds())
            total_response_time += time_diff_seconds / 3600
            total_orders += 1
    if total_orders > 0:
        self.average_response_time = total_response_time /
           total_orders
    else:
        self.average_response_time = None

    self.save()
```

d. Taux de réalisation des commandes (`fulfillment_rate`)

But : Évaluer la capacité du fournisseur à exécuter les commandes.

— Il s'agit du pourcentage de commandes finalisées (`completed`) parmi les commandes non annulées.

— La formule de calcul est :

$$\text{Taux de réalisation} = \left(\frac{\text{Commandes réussies}}{\text{Commandes totales hors annulées}} \right) \times 100$$

Listing 4.13 – Calcul du taux d'accomplissement des commandes

```
def fulfillment_rate_calculator(self):
    successful_orders = self.purchaseorder_set.filter(status='
        completed', issue_date__isnull=False)
    total_orders = self.purchaseorder_set.exclude(status='canceled').
        count()

    fulfillment_rate = (successful_orders.count() / total_orders) *
        100 if total_orders > 0 else 0

    self.fulfillment_rate = fulfillment_rate
    self.save()
```

2. Mise à jour automatique des indicateurs via la méthode `save` du modèle `PurchaseOrder`

Pour garantir la cohérence des indicateurs de performance du fournisseur, la méthode `save` du modèle `PurchaseOrder` a été surchargée afin d'actualiser automatiquement plusieurs champs clés lors de la sauvegarde d'une commande. Le comportement implémenté est le suivant :

— Lorsque le statut de la commande devient `'completed'` (complétée) et que les dates de livraison (`delivery_date`) et d'émission (`issue_date`) ne sont pas encore définies, elles sont automatiquement renseignées avec la date et l'heure actuelles.

— Si la commande existante est modifiée, et que son statut ou ses dates importantes changent, les méthodes de recalcul des indicateurs de performance du fournisseur associé sont appelées :

  — `fulfillment_rate_calculator()` : mise à jour du taux de réalisation des commandes.

  — `on_time_delivery_calculator(completed_orders)` : recalcul du taux de livraison à temps, basé sur l'ensemble des commandes complétées.

  — `average_response_time_calculator()` : recalcul du temps moyen de réponse, lorsque la date d'accusé de réception change.

  — `quality_rating_avg_calculator(completed_orders)` : mise à jour de la note moyenne de qualité, si une nouvelle note est renseignée lors de la complétion.

— Cette automatisation garantit que toutes les statistiques liées à la performance du fournisseur restent à jour et reflètent précisément l'état courant des commandes.

Listing 4.14 – Méthode `save` du modèle `PurchaseOrder`

```
def save(self, *args, **kwargs):
    if self.status == 'completed' and not self.delivery_date:
```

```
            self.delivery_date = timezone.now()
        if self.status == 'completed' and not self.issue_date:
            self.issue_date = timezone.now()

        completed_orders = self.supplier.purchaseorder_set.filter(status='
            completed')

        if self.pk is not None:  # Si l'objet existe d j
            old_order_obj = PurchaseOrder.objects.get(pk=self.pk)
            if self.status != old_order_obj.status:
                self.supplier.fulfillment_rate_calculator()
            if self.status == 'completed' and old_order_obj.status != '
                completed':
                self.supplier.on_time_delivery_calculator(completed_orders
                    )
            if self.acknowledgment_date != old_order_obj.
                acknowledgment_date:
                self.supplier.average_response_time_calculator()
            if self.status == 'completed' and old_order_obj.status != '
                completed' and self.quality_rating is not None:
                self.supplier.quality_rating_avg_calculator(
                    completed_orders)

        super().save(*args, **kwargs)
```

3. Modèle `Contract` – Gestion des contrats fournisseurs

Le cœur de cette fonctionnalité repose sur le modèle `Contract`, qui gère :

- La référence et le type de contrat (`Standard` ou `Avenant`),
- La liaison avec le fournisseur concerné,
- Les informations juridiques et financières (montants, échéances, délais, modalité de paiement, devise, lieu de conservation, etc.),
- La conformité du contrat et son **cycle de vie** via les statuts :
  - `en_attente`, `en_cours`, `cloture`, `annule`.
- Une méthode `verifier_conformite` permet de mettre à jour la conformité et le statut automatiquement selon l'évaluation effectuée par un responsable.
- Le statut est aussi mis à jour automatiquement dans la méthode `save`, en fonction de la conformité et de la date de fin du contrat (clôture automatique).

4. Modèles complémentaires

Pour enrichir la gestion contractuelle, plusieurs entités ont été ajoutées :

- `PriseEnCharge` : trace les prises en charge d'un contrat par des utilisateurs.
- `SupplierLifecycle` : permet de suivre les étapes du cycle de vie d'un contrat fournisseur (par exemple validation, exécution, résiliation).
- `RFA` : gère les remises de fin d'année liées à un contrat, avec montant et état de recouvrement.
- `SupplierSupport` : identifie les personnes impliquées dans le support ou le suivi du fournisseur, leur rôle et date d'intervention.

Le modéle `Notification`

Comme expliqué dans le Sprint 3, nnous avons déjà développé la majeure partie du module de notifications. Cependant, nous souhaitons désormais l'enrichir afin d'inclure la notification des contrats avec comme destinataire : les utilisateurs appartenant au groupe « responsable contrat ». Pour cela, nous avons ajouté la fonction suivante dans le fichier `views.py` :

Listing 4.15 – Notification à la création d'un contrat

```
    def send_contract_notification(responsable_contrat, contract):
    # Creer une notification dans la base de donn es
    message = f"Un nouveau contrat a ete cree : {contract.ref_contrat
       }."
    ContratNotification.objects.create(
        destinataire =responsable_contrat,
        message=message,
        contrat=contract
    )
```

Pour permettre l'affichage de ces notifications dans l'interface utilisateur nous avons mis à jour le fichier `context_processors.py` comme suit :

Listing 4.16 – Ajout des notifications de contrat dans toutes les templates

```
def navbar_notifications(request):
    if not request.user.is_authenticated:
        return {}

    contrat_notifs = ContratNotification.objects.filter(destinataire=
       request.user, lu=False)
    all_notifs =  list(contrat_notifs)
    all_notifs.sort(key=lambda x: x.date_creation, reverse=True)

    return {
        'notifications': all_notifs[:5],  # les 5 dernieres
        'notification_count': len(all_notifs)
    }
```

5. Modèles `Arrivage / Sortie`

Le modèle `Arrivage` représente la réception des pièces suite à la réception d'une commande d'achat. Un arrivage est automatiquement créé lorsqu'une commande d'achat est réceptionnée. Il contient des informations relatives à la commande d'achat, telles que le fournisseur, la devise, le mode de transport, ainsi qu'un champ booléen `stock_mis_a_jour` indiquant si les quantités reçues ont été enregistrées dans le stock ou non. Un bon de livraison peut être généré à partir de cet arrivage.

Le modèle `Sortie` correspond à la sortie de pièces du magasin, créée lorsqu'une demande de réservation est validée. Il inclut des informations liées à la demande de réservation, telles que la date souhaitée et les pièces concernées, ainsi que la date de sortie réelle, qui est remplie une fois que le champ booléen `sortie_effectuee` est mis à vrai. Ce champ permet de contrôler la génération de bon de sortie et la mise à jour des stocks.

Le suivi des stocks

Pour assurer un suivi précis des stocks, plusieurs fonctions clés ont été implémentées au niveau du modèle `Produit`. Ces fonctions permettent de mesurer la quantité disponible, d'évaluer les seuils critiques, d'analyser la fréquence des sorties, et d'estimer les besoins futurs.

1. **Calcul des stocks total**
   Le but de cette fonction est de connaître la quantité totale de pièces disponibles en stock. Le calcul se fait en agrégeant toutes les quantités présentes dans les différents magasins à l'aide de la formule suivante :

$$\text{Stock total} = \sum_{i=1}^{n} q_i$$

   Tels que :
   - i représente un magasin.
   - qi est la quantité de la pièce dans le magasin i.
   - n est le nombre total de magasin.

Listing 4.17 – Fonction de calcul du stock total

```
    def stock_total(self):
        return self.details_pieces.aggregate(Sum('quantite'))['
           quantite__sum'] or 0
```

2. **Niveau des stocks**
   Cette fonction permet de déterminer et de classer le niveau des stocks en comparant la quantité totale (récupérée via la fonction `stock_total`) à deux seuils définis :
   - `quantité minimale` *qte_min*, en dessous de laquelle le stock est considéré insuffisant (risque de rupture).
   - `quantité maximale` *qte_max*, au-delà de laquelle le stock est considéré trop élevé (risque de surstock).

   Cela permet de visualiser rapidement l'état du stock à l'aide d'un code couleur :
   - **Rouge** : la quantité dépasse la quantité maximale (surstock).
   - **Orange** : la quantité est inférieure à la quantité minimale (stock insuffisant).
   - **Jaune** : la quantité est exactement égale à la quantité minimale (alerte).
   - **Vert** : la quantité est comprise entre les deux seuils (stock normal).

Listing 4.18 – Fonction de calcul du niveau du stock

```
def niveau_stock(self):
    qte = self.stock_total
    if qte > self.qte_max:
        return "rouge"
    elif qte == self.qte_min:
        return "jaune"
    elif self.qte_min < qte <= self.qte_max:
        return "vert"
    elif qte < self.qte_min:
        return "orange"
```

3. **Nombre de sorties sur les derniers jours**
Cette fonction sert à compter le nombre de fois qu'une pièce a été sortie du stock au cours des n derniers jours (par défaut 30 jours). Cela permet d'évaluer la fréquence d'utilisation d'une pièce.

$$\text{Nombre de sorties} = \text{Total des sorties où Date sortie} \geq \text{Aujourd'hui} - J$$

Tels que :
— J représente le nombre de jours à considérer en arrière à partir d'aujourd'hui (par défaut, $J = 30$).

Listing 4.19 – Fonction sorties_derniers_jours

```
def sorties_derniers_jours(self, jours=30):
    from bonSortie.models import DetailBonSortie
    date_limite = timezone.now() - timedelta(days=jours)
    return DetailBonSortie.objects.filter(
        piece=self,
        bon_sortie__date_sortie__gte=date_limite
    ).count()
```

4. **Délai depuis la dernière sortie**
Cette fonction calcule le nombre de jours écoulés depuis la dernière sortie d'une pièce du stock, Elle sert à mesurer depuis combien de jours elle n'a pas été utilisée. Voici la formule de calcul :

$$\text{Dernière sortie (en jours)} = \text{date actuelle} - \text{date dernière sortie}$$

Listing 4.20 – Fonction derniere_sortie

```
def derniere_sortie(self):
    from bonSortie.models import DetailBonSortie

    derniere = DetailBonSortie.objects.filter(
        piece=self
    ).order_by('-bon_sortie__date_sortie').first()

    if derniere:
        return (timezone.now().date() - derniere.bon_sortie.
           date_sortie.date()).days
    return None
```

5. **Catégorie de fréquence des sorties**
Cette fonction classe la fréquence d'utilisation d'une pièce en catégories en se basant sur la moyenne quotidienne des sorties sur une période donnée :
— **Aucune** si aucune sortie n'a été enregistré pour cette pièce.
— **Faible** si il y a une sortie au maximum tous les 5 jours.
— **Moyenne** si une sortie est enregistré environ tous les 2 à 5 jours.
— **Forte** dans le cas d'une sortie tous les 2 jours
Le calcul de la moyenne se fait avec cette formule :

$$\text{Moyenne de sortie par jour} = \frac{\text{Nombre total de sorties}}{\text{Nombre de Jour}}$$

Listing 4.21 – Fonction frequence_categorie

```
  def frequence_categorie(self, jours=30):
        sorties = self.sorties_derniers_jours(jours)
        moyenne_jour = sorties / jours if jours else 0

        if sorties == 0:
            return "Aucune"
        elif moyenne_jour <= 0.2:
            return "Faible"
        elif moyenne_jour <= 0.5:
            return "Moyenne"
        else:
            return "Forte"
```

6. **Estimation du besoin**
   Cette fonction estime la quantité approximative de pièces à prévoir pour le mois à venir (30 jours), en se basant sur la consommation moyenne quotidienne pour planifier les commandes et éviter les ruptures :

$$\text{estimation} = \text{round}\left(\frac{\text{Nombre total de sorties}}{\text{Nombre de jours}}\right) \times 30$$

Listing 4.22 – Fonction estimation_besoin

```
def estimation_besoin(self):
        jours = 30
        sorties = self.sorties_derniers_jours(jours)
        if sorties == 0:
            return 0
        moyenne_jour = sorties / jours
        return round(moyenne_jour * 30)
```

7. **Choix des KPI en fonction des utilisateurs** : Une première série de KPI a été développée durant le Sprint 3, principalement autour du traitement des demandes. À l'issue de la mise en place du suivi des stocks dans ce Sprint 4, nous avons enrichi ces indicateurs en y intégrant les profils liés à l'exécution des achats et au pilotage global du stock, voici les indicateurs ajoutés :
   - **Acheteur** L'acheteur est en première ligne du processus d'acquisition. Il dispose des indicateurs suivants :
     - **Volumes globaux** : nombre de DA, commandes, fournisseurs, contrats.
     - **Commandes** : en cours, validées, liées à un contrat, réalisées.
   - **Responsable achat** : Le responsable des achats supervise l'ensemble des acheteurs. Il dispose d'une vision synthétique à travers :
     - **Demandes d'Achat (DA)** : total, traitées, en attente, taux de traitement, délai moyen.
     - **Commandes d'Achat** : total, traitées, en attente, taux de traitement, délai moyen.

     Cela facilite le pilotage stratégique et le rééquilibrage des charges si nécessaire.

- **Responsable Contrat**
  - **Volumes globaux** : nombre de DA, commandes, fournisseurs, contrats.
  - **Contrats par statut** : en cours, clôturé, annulé, etc.
  - **Contrats par conformité** : conforme, non conforme, en attente.
- **Responsable approvisionnement** Dans le cadre du suivi des stocks, ces deux profils disposent d'indicateurs communs pour assurer une gestion optimale des flux :
  - **Vue sur les stocks disponibles :** visibilité en temps réel sur les niveaux de stock par produit et par magasin, facilitant la prise de décision lors des validations.
- **Magasinier** Le magasinier intervient dans l'évaluation des **demandes de réservation**. Ses outils de suivi incluent :
  - **Indicateurs globaux** : nombre total de DA, DR, arrivages, et sorties.
  - **Suivi des produits suivis** : stock total, niveau, besoin estimé, fréquence de sortie.
  - **Graphiques hebdomadaires** : volumes d'arrivages par semaine.
  - **Répartition des types de sorties**
  - **État des arrivages**
  - **Taux de remplissage des magasins**

  Grâce à ces éléments, le magasinier peut gérer ses priorités et éviter les ruptures de stocks.

Ces indicateurs ont permis d'enrichir le tableau de bord existant en y intégrant le suivi des stocks, offrant ainsi une vision plus complète et du processus d'achat de des approvisionnements.

3.4.5 Réalisation

Cette phase consiste à traduire visuellement les fonctionnalités développées en interfaces utilisateur complètes et fonctionnelles. Les figures suivantes illustrent les interfaces développées dans le cadre du Sprint 4, qui portent sur la gestion des fournisseurs et leurs performances, des contrats, les sorties et arrivages ainsi que le suivi du stock :

La figure 4.38 montre l'interface utilisateur de suivi des performances fournisseurs :

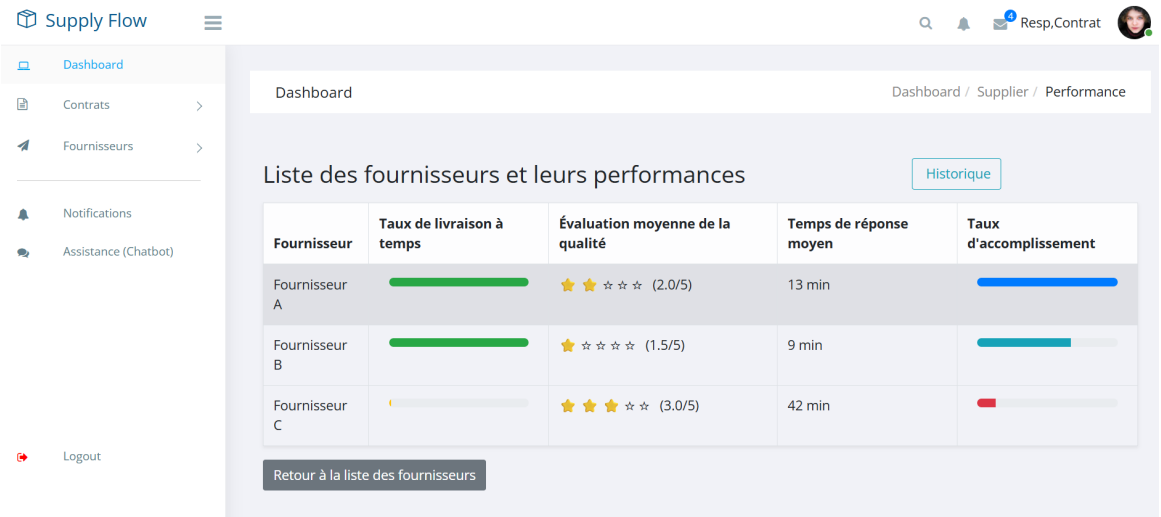

FIGURE 4.38 – Visualisation des performances fournisseur

La figure 4.39 montre l'interface utilisateur de la gestion des contrats :

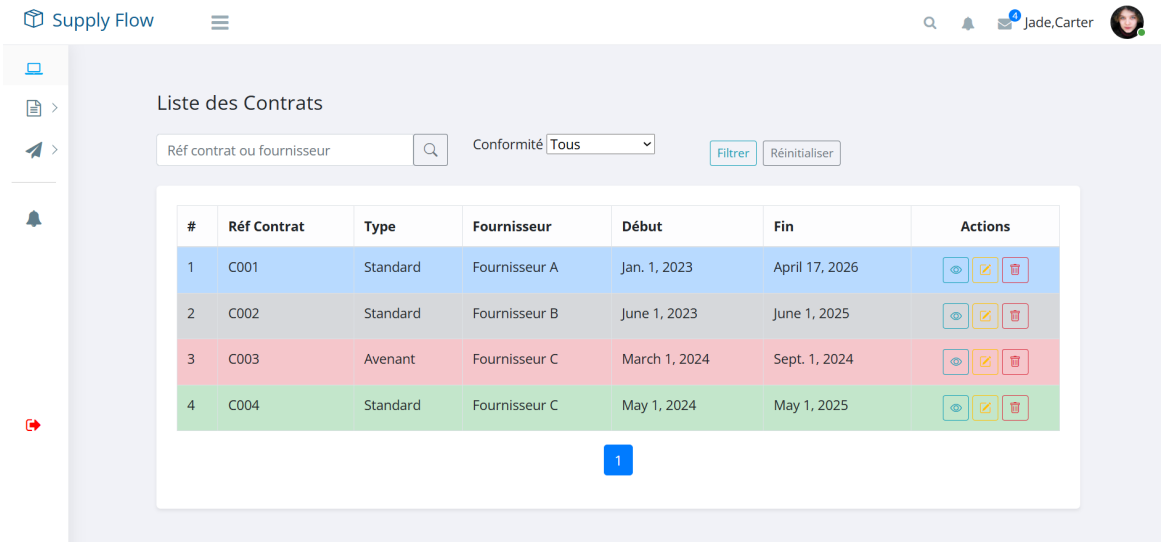

FIGURE 4.39 – La liste des contrats

La figure 4.40 montre l'interface utilisateur d'un exemple d'une sortie :

FIGURE 4.40 – Detail d'un bon de sortie

La figure 4.41 montre l'interface utilisateur le tableau de bord du suivi des niveaux des stocks :

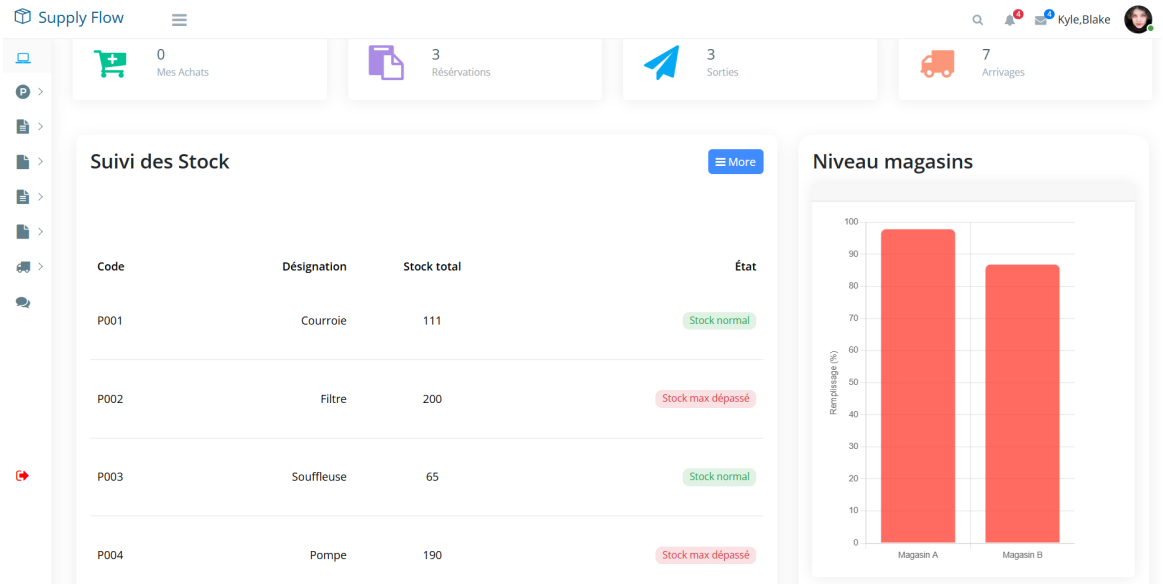

FIGURE 4.41 – Le Suivi des stocks

4 Diagramme de classes global

Cette section présente le diagramme de classes UML global du système, regroupant l'ensemble des entités manipulées dans les different modules développés au cours des sprints.

Il illustre la structure finale de l'application et offre une vue d'ensemble sur les liens entre ses différentes composantes. Cela nous a permis de mieux comprendre la logique du système et les interactions entre les différentes parties de l'architecture.

La figure ci-dessous 4.42, présente ce diagramme de classes global :

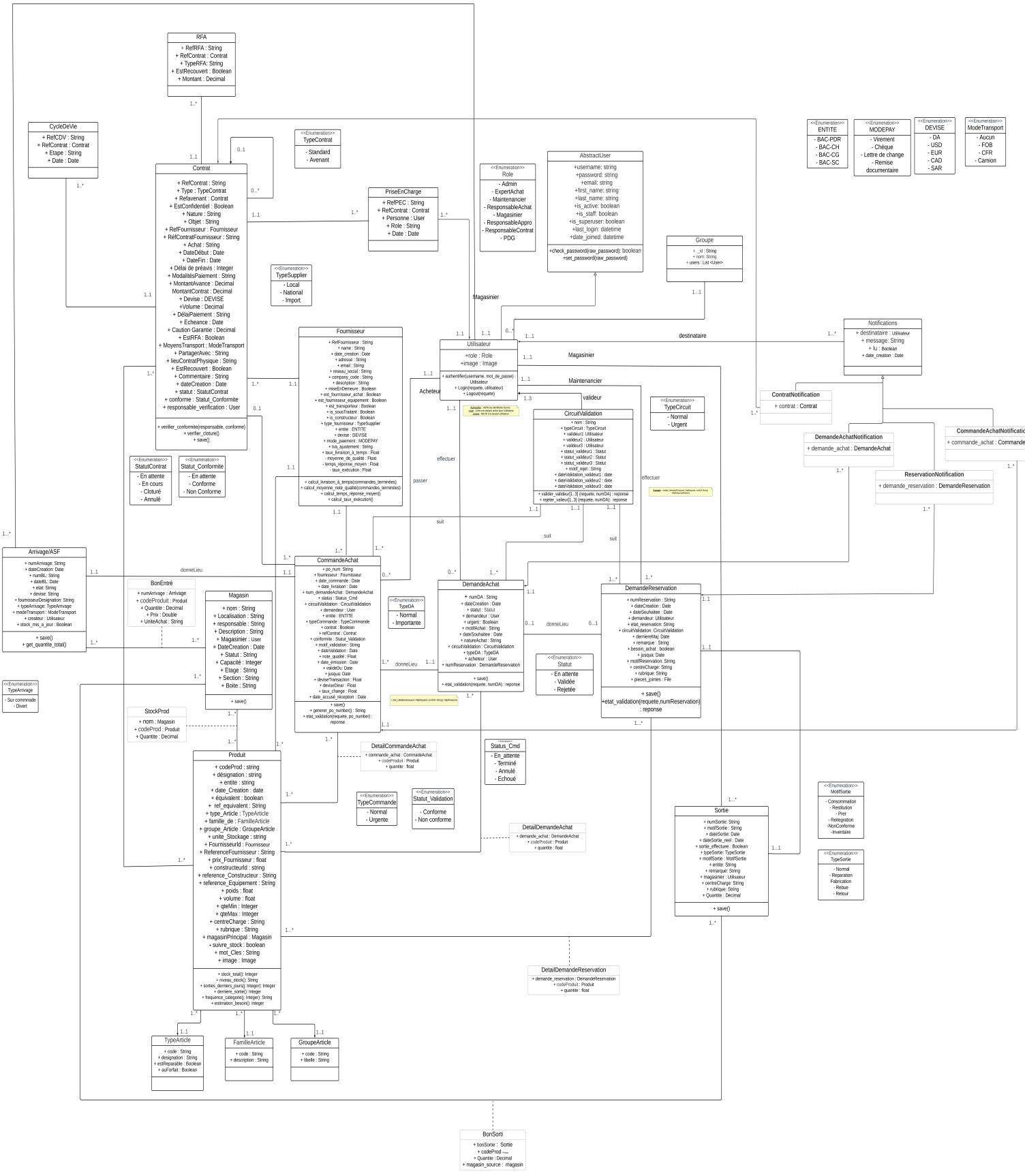

FIGURE 4.42 – Diagramme de classe global

5 Conclusion

La phase de conception, développement et mise en œuvre, découpée en quatre sprints selon la méthode agile, a permis d'aboutir à une solution fonctionnelle, cohérente avec les besoins identifiés lors de l'étude préalable. Chaque sprint a donné lieu à une itération complète intégrant à la fois la conception (avec diagrammes de cas d'utilisation, séquence, classe, et état-transition lorsque pertinent), la prototypage rapide des interfaces, ainsi qu'un développement incrémental accompagné d'explications de code détaillées.

Les interfaces réalisées ont été documentées par des captures d'écran, permettant d'illustrer concrètement les fonctionnalités livrées à chaque étape. Cette approche agile a favorisé la flexibilité, la validation continue et l'implication progressive des utilisateurs dans l'évolution du système.

Cette phase se conclut par une version stable de l'application, prête à être soumise à des tests. Ces derniers, abordés dans le chapitre suivant, seront menés de manière séquentielle selon une approche en cascade, afin d'évaluer la robustesse et la performance du système.

Chapitre 5

Tests et Déploiement

1 Introduction

Ce chapitre présente les différentes phases de test et les stratégies de déploiement adoptées pour garantir la fiabilité, la robustesse et l'efficacité de l'application développée dans le cadre de ce projet. Étant donné la nature critique des processus métiers pris en charge — notamment la gestion des fournisseurs, des circuits de validation, et l'assistance intelligente via un chatbot — une approche rigoureuse de vérification a été mise en œuvre.

Les tests ont initialement suivi une structuration de type séquentiel (inspirée du modèle en cascade), comme dans la phase d'analyse et l'étude préalable. Cependant, la phase de tests a dû être ajustée et a évolué vers une démarche plus itérative et interactive.

Cette évolution méthodologique s'explique par la nécessité de tester certaines fonctionnalités complexes (comme le chatbot ou le circuit de validation) au contact des utilisateurs réels, et de réajuster le système selon leurs retours. Ainsi, bien que l'ossature des tests reste structurée (tests unitaires, fonctionnels, intégration), des boucles de feedback rapides ont été mises en place, permettant d'introduire un degré d'agilité pragmatique dans cette phase critique.

Il ne s'agit donc pas d'une adoption formelle du cadre Agile, mais plutôt d'une hybridation raisonnée entre les deux logiques :

— Une logique **séquentielle** pour assurer la couverture complète et structurée des tests ;

— Une logique **itérative** pour intégrer les ajustements nécessaires liés à l'usage réel (notamment pour les interfaces ou les systèmes d'assistance intelligents).

Ce compromis méthodologique visait également à éviter de tomber dans la *crise du logiciel*, c'est-à-dire les difficultés engendrées par un regroupement tardif des tests dans un modèle strictement séquentiel, rendant difficile la prise en compte des erreurs et des retours utilisateurs, et compromettant ainsi la qualité finale du produit.

Enfin, le déploiement n'a été effectué qu'à la fin du projet, une fois l'ensemble des tests validés. Ce choix garantit une stabilité optimale de l'application en production et respecte la logique séquentielle adoptée tout au long du développement.

Les sections suivantes détaillent les différentes campagnes de tests menées pour chaque fonctionnalité développée pendant les sprints, en mettant en évidence leur rôle dans la validation progressive du système ainsi que les outils, méthodes et résultats associés :

— Tests fonctionnels et d'usage du **chatbot intelligent** ;

— Tests unitaires des **opérations CRUD** ;

— Vérification du **circuit de validation multi-acteurs** ;

— Évaluation des **performances fournisseurs via Postman**.

2 Tests

2.1 Tests du chatbot

Objectifs des tests : Les tests visaient à évaluer les aspects suivants :

- La capacité du chatbot à interpréter correctement des descriptions textuelles techniques ou approximatives.
- La pertinence des résultats retournés, c'est-à-dire la capacité à proposer la bonne pièce ou les pièces les plus proches.
- La robustesse du système face aux formulations variées (abréviations, fautes, formulations incomplètes) ;
- La facilité d'utilisation et l'ergonomie de l'interface pour les utilisateurs non techniques.

Méthodologie adoptée : Un panel de 5-6 utilisateurs issus du service maintenance et magasin a été sélectionné. Chaque utilisateur a interagi avec le chatbot via l'interface web afin de simuler des recherches réelles de pièces détachées. Les scénarios testés incluaient :

- La recherche d'une armoire électrique à partir d'une description libre (ex : *"Je cherche une armoire électrique pour l'industrie"*) ;
- L'interaction incrémentale avec l'agent pour affiner une recherche à partir d'informations vagues ou incomplètes (ex : *"Elle est utilisée pour gérer un groupe froid"*) ;
- La prise en compte des critères de comparaison tels que la résistance des matériaux ou les dimensions pour guider le choix (ex : *"Quelle est la plus résistante des deux ?"*) ;
- La récupération d'informations détaillées sur une pièce sélectionnée (ex : référence, fournisseur, poids, prix...).

Indicateurs de performance : Pour mesurer objectivement les résultats des tests, les indicateurs suivants ont été utilisés :

- **Taux de correspondance exacte** : pourcentage de cas où la bonne pièce a été proposée en premier ;
- **Taux de correspondance acceptable** : cas où la bonne pièce figure parmi les 3 premières propositions ;
- **Temps moyen de réponse** : temps écoulé entre l'envoi de la description et la réception de la réponse ;
- **Taux de satisfaction utilisateur** : mesuré via un formulaire de retour post-interaction.

Résultats obtenus :

- **Taux de correspondance exacte** : **76 %** ;
- **Taux de correspondance acceptable** : **91 %** ;
- **Temps moyen de réponse** : environ **5-20 secondes**, grâce à l'utilisation de FAISS pour l'indexation vectorielle et du modèle RAG ;
- **Satisfaction générale** : **89 %** des utilisateurs se sont déclarés satisfaits ou très satisfaits.

Commentaires des utilisateurs : Les retours qualitatifs ont mis en lumière les points suivants :

— Le chatbot permet un gain de temps important par rapport aux recherches manuelles ;

— Il est particulièrement utile en cas d'oubli du nom exact d'une pièce ;

— Certaines améliorations pourraient être envisagées, notamment l'intégration de visuels des pièces c'est a dire ajouter une recherche par images.

Cette phase de test a confirmé la pertinence du chatbot dans un contexte industriel réel. Il constitue un outil d'aide efficace pour les équipes techniques, en facilitant l'identification rapide de pièces de rechange à partir de descriptions souvent imprécises. Des perspectives d'évolution sont envisagées.

2.2 Tests des opérations CRUD

Afin de tester les fonctionnalités de notre application, nous avons utilisé le module `TestCase` fourni par Django, qui hérite de la bibliothèque standard de Python : `unittest`, une bibliothèque permet d'écrire, d'organiser et d'exécuter des tests unitaires. Par convention, Django recherche automatiquement ces tests dans tout fichier nommé `tests.py` présent dans l'arborescence du projet. [55]

Pour tester les modèles de notre base de données, nous avons opté pour la classe `django.test.TestCase`, qui est une extension de `unittest.TestCase`, adaptée aux applications Django. Cette classe facilite grandement la phase de test en intégrant plusieurs fonctionnalités :

— La création automatique d'une base de données temporaire dédiée aux tests, qui est supprimer dès que l'ensemble des tests ont été complètement exécutés.

— L'isolation des données de test des vraies données, donc la base réelle n'est jamais affectée.

— Une intégration directe avec les modèles Django, permettant de manipuler les données via l'ORM.

Nous avons ainsi mis en place des tests unitaires pour les opérations CRUD (Create, Read, Update, Delete) [56], qui représentent les actions de base pour créer et gérer les données :

— **Create** : ajouter un nouvel enregistrement dans la base de données ;

— **Read** : lire ou récupérer des données existantes ;

— **Update** : modifier un enregistrement existant ;

— **Delete** : supprimer une donnée.

Voici un exemple de tests que nous avons réalisés pour valider les opérations de création et de modification sur le modèle `Product` :

Listing 5.1 – Test unitaire de la création de Piéce

```
from django.test import TestCase
from users.models import User
from .models import Product, TypeArticle, FamilleArticle,
   GroupeArticle, Supplier, Magasin

class ProductModelTest(TestCase):
    # s'execute avant chaque test pour eviter de repeter du code
    def setUp(self):
        # Cr ation des objets (foreign key)
        self.user = User.objects.create_user(username='testuser',
           password='123')
```

```
        self.type_article = TypeArticle.objects.create(code="TA004",
            designation="Type D")
        self.famille = FamilleArticle.objects.create(code="FA004",
            description="Famille D")
        self.groupe = GroupeArticle.objects.create(code="GA004",
            libelle="Groupe D")
        self.fournisseur = Supplier.objects.create(ref_fournisseur="
            supplier 4")
        self.magasin = Magasin.objects.create(nom="Magasin D",
            capacite=100)

    def test_create_product(self):
        % # Cr ation du produit avec tout ses attributs
        product = Product.objects.create(
            codeProd="P009",
            createur=self.user,
            designation="Pompe Hydraulique",
            entite="BAC-CH",
            equivalent=False,
            type_article=self.type_article,
            famille_de=self.famille,
            groupe_article=self.groupe,
            unite_stockage="pi ce",
            fournisseur_id=self.fournisseur,
            reference_fournisseur="F004",
            prix_fournisseur=150.5,
            constructeur_id="Constructeur 4",
            reference_constructeur="C004",
            poids=20.5,
            volume=12.25,
            qte_min=2,
            qte_max=10,
            suivre_stock=True,
            centre_charge="Centre 2",
            rubrique="Pi ces de rechange",
            mot_cles="courroie, machine, entra nement",
            magasin_principal=self.magasin
        )

        % # V rification des valeurs de l'objet cree
        self.assertEqual(product.codeProd, "P009")
        self.assertEqual(product.designation, "Pompe Hydraulique")
        self.assertEqual(product.entite, "BAC-CH")
        self.assertTrue(product.suivre_stock)
        self.assertEqual(product.magasin_principal.nom, "Magasin D")
```

Afin d'exécuter les tests, la commande suivante est utilisée dans le terminal. Elle permet à Django de rechercher tous les fichiers `tests.py` dans les applications installées :

```
python manage.py test
```

Une fois l'exécution terminée avec succès, le terminal affiche ceci :

Listing 5.2 – Resultat du test unitaire

```
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------
Ran 1 tests in 1.857s

OK
Destroying test database for alias 'default'...
```

— La ligne `Creating test database...` indique que Django a créé une base temporaire dédiée aux tests.

— Les points (`.`) correspondent à des tests réussis (un seul test dans notre cas).

— Le message `OK` confirme que tous les tests se sont bien déroulés.

— Enfin, la base de données de test est automatiquement détruite.

Ainsi, le succès du test confirme que l'ajout d'un produit dans la base de données fonctionne correctement. Les données saisies respectent bien les contraintes définies dans le modèle, et l'enregistrement est effectué sans erreur.

2.3 Tests du circuit de validation

Afin de garantir la fiabilité du processus de validation d'une demande d'achat, nous avons mis en place une série de tests unitaires à l'aide du module `TestCase` de Django. Ces tests visent à vérifier le bon fonctionnement du circuit de validation composé de trois valideurs, ainsi que l'impact de leurs actions sur le statut global de la demande.

Objectifs des tests. Les tests ont été conçus pour valider les comportements suivants :

— La capacité pour chaque valideur de changer son statut de validation.

— L'enchaînement correct des validations (ex. : le valideur 2 ne peut valider qu'après le valideur 1).

— La propagation correcte du rejet (si un valideur rejette, le processus s'arrête et la demande passe en statut « rejetée »).

— L'évolution automatique du statut de la demande d'achat vers « validée » lorsque tous les valideurs ont validé.

Mise en œuvre. Dans le fichier de tests, quatre cas principaux ont été simulés :

1. Validation par le premier valideur uniquement.
2. Validation par le second valideur après la validation du premier.
3. Rejet par le troisième valideur et vérification du motif de rejet ainsi que du changement de statut de la demande.
4. Validation complète du circuit (trois valideurs) et passage automatique de la demande en statut « validée ».

Listing 5.3 – Tests du circuit de validation

```
from django.test import TestCase
from django.utils import timezone
from users.models import User
```

```
from demandeAchat.models import DemandeAchat, CircuitValidation
from store.models import Product

class CircuitValidationTestCase(TestCase):
    def setUp(self):
        self.valideur1 = User.objects.create_user(username="valideur1"
            , password="test123")
        self.valideur2 = User.objects.create_user(username="valideur2"
            , password="test123")
        self.valideur3 = User.objects.create_user(username="valideur3"
            , password="test123")
        self.demandeur = User.objects.create_user(username="demandeur"
            , password="test123")
        self.createur = User.objects.create_user(username="createur",
           password="test123")

        self.circuit = CircuitValidation.objects.create(
            nom="Circuit Test",
            valideur1=self.valideur1,
            valideur2=self.valideur2,
            valideur3=self.valideur3,
            type_circuit='normal'
        )

        self.da = DemandeAchat.objects.create(
            motifAchat="Achat test",
            natureAchat="pdr",
            typeDa="normal",
            demandeur=self.demandeur,
            createur=self.createur,
            circuit_validation=self.circuit
        )

    def test_valideur1_valide(self):
        self.circuit.statut_valideur1 = 'validee'
        self.circuit.date_validation_valideur1 = timezone.now()
        self.circuit.save()
        self.assertEqual(self.circuit.statut_valideur1, 'validee')
        self.assertIsNotNone(self.circuit.date_validation_valideur1)

    def test_valideur2_valide_apres_valideur1(self):
        self.circuit.statut_valideur1 = 'validee'
        self.circuit.date_validation_valideur1 = timezone.now()
        self.circuit.save()
        self.circuit.statut_valideur2 = 'validee'
        self.circuit.date_validation_valideur2 = timezone.now()
        self.circuit.save()
        self.assertEqual(self.circuit.statut_valideur2, 'validee')
        self.assertIsNotNone(self.circuit.date_validation_valideur2)

    def test_valideur3_rejete(self):
        self.circuit.statut_valideur1 = 'validee'
        self.circuit.statut_valideur2 = 'validee'
```

```
        self.circuit.statut_valideur3 = 'rejetee'
        self.circuit.motif_rejet = "Budget insuffisant"
        self.circuit.save()
        self.da.statut = 'rejet e'
        self.da.save()
        self.assertEqual(self.circuit.statut_valideur3, 'rejetee')
        self.assertEqual(self.circuit.motif_rejet, "Budget insuffisant
           ")
        self.assertEqual(self.da.statut, 'rejet e')

    def test_circuit_complet_valide(self):
        self.circuit.statut_valideur1 = 'validee'
        self.circuit.statut_valideur2 = 'validee'
        self.circuit.statut_valideur3 = 'validee'
        self.circuit.date_validation_valideur1 = timezone.now()
        self.circuit.date_validation_valideur2 = timezone.now()
        self.circuit.date_validation_valideur3 = timezone.now()
        self.circuit.save()
        self.da.statut = 'valid e'
        self.da.save()
        self.assertEqual(self.da.statut, 'valid e')
```

Résultats. Tous les tests ont été exécutés avec succès, confirmant que :

— Les transitions de statut sont correctement enregistrées.

— Le motif de rejet est bien sauvegardé dans le modèle `CircuitValidation`.

— Le statut global de la demande d'achat évolue conformément aux décisions des valideurs.

Listing 5.4 – Sortie console lors de l'exécution des tests

```
(venv) macbookair@MacBook-Air-de-Macbook myProject-main-2 % python
   manage.py test demandeAchat
Found 4 test(s).
Creating test database for alias 'default'...
System check identified some issues:
System check identified 24 issues (0 silenced).
....
----------------------------------------------------------------------
Ran 4 tests in 4.340s

OK
Destroying test database for alias 'default'...
```

Les tests du circuit de validation permettent ainsi d'assurer la robustesse du processus décisionnel autour des demandes d'achat. Ils garantissent que chaque acteur impliqué dans la validation peut interagir avec le système de manière fiable et conforme aux règles de l'entreprise.

2.4 Tests des performances des fournisseurs avec Postman

Dans le cadre de l'évaluation des fournisseurs, nous avons implémenté une API REST permettant de tester plusieurs indicateurs de performance clés. Cette API a été testée à l'aide de l'outil Postman. Elle reçoit en paramètre le code de référence d'un fournisseur et retourne divers métriques calculées à partir des données de commandes et d'arrivages.

Exemple URL de test : `GET http://127.0.0.1:8000/store/suppliers/F001/test-performance/`

Exemples de métriques calculées :

- **Taux de livraison à temps (On-time Delivery)** : pourcentage des commandes reçues avant ou à la date de livraison prévue.
- **Délai moyen de livraison** : nombre moyen de jours de retard pour les livraisons tardives.
- **Note de qualité moyenne** : moyenne des notes de qualité attribuées aux commandes réceptionnées.
- **Temps moyen de réponse** : délai moyen (en heures) entre l'émission de la commande et son accusé de réception.
- **Taux de conformité (Compliance)** : pourcentage des commandes dont la quantité livrée correspond exactement à la quantité commandée.
- **Taux de satisfaction** : basé sur les livraisons complètes, la qualité et le respect des délais.
- **Nombre moyen de livraisons par commande** : nombre d'arrivages en moyenne par commande.
- **Score global du fournisseur** : moyenne de plusieurs indicateurs pondérés.

Résultat attendu : la réponse JSON retournée permet d'avoir une vue synthétique de la performance du fournisseur et peut servir de base pour un classement ou une décision d'achat. Voici un exemple de réponse retournée par Postman :

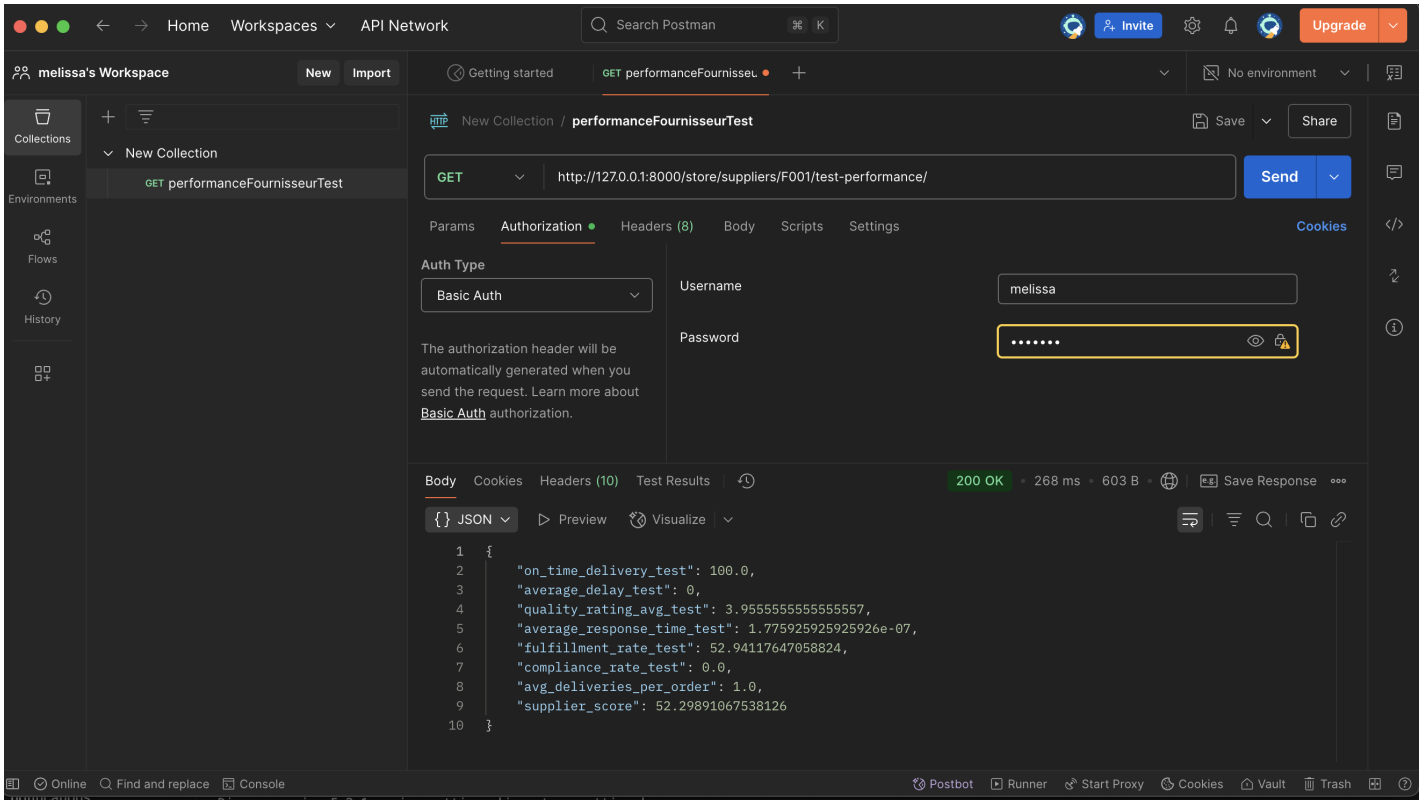

FIGURE 5.1 – Postman HTTP Request

Ce test permet ainsi de juger la fiabilité et la réactivité du fournisseur dans une optique d'optimisation de la chaîne d'approvisionnement.

3 Déploiement

3.1 Choix de la plateforme de déploiement

Pour le déploiement de notre application, nous avons opté pour la plateforme **Render**, un service cloud moderne qui permet d'héberger des applications web sans gérer manuellement des serveurs [57]. Render propose une intégration directe avec GitHub, un système de déploiement automatique à chaque mise à jour du code, ainsi qu'une interface intuitive pour la configuration des variables d'environnement. De plus, l'offre gratuite est largement suffisante pour des projets académiques ou de démonstration.

3.2 Étapes de déploiement sur Render

Le déploiement est effectué automatiquement à partir du dépôt GitHub. À chaque nouveau *push* sur la branche principale, Render déclenche un processus de build et de redéploiement.

Les étapes de configuration sont les suivantes :

— Création d'un dépôt GitHub contenant le projet.

— Ajout d'un fichier `requirements.txt` pour la gestion des dépendances.

— Configuration du fichier `settings.py` avec prise en charge de la base de données PostgreSQL.

— Création d'un nouveau Web Service sur Render.

— Connexion du service au dépôt GitHub.

— Définition des variables d'environnement nécessaires, telles que `SECRET_KEY`, `DEBUG` ou encore les informations de la base de données.

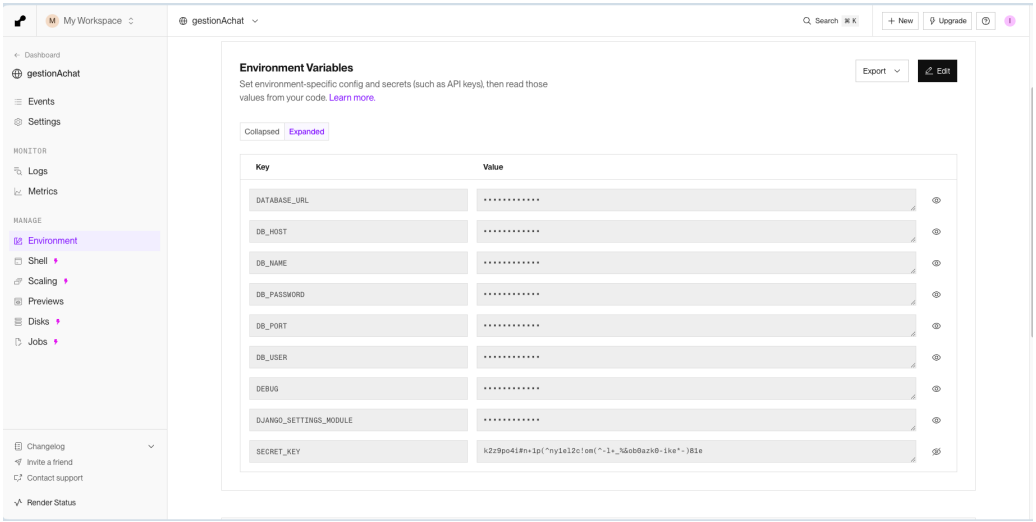

FIGURE 5.2 – Configuration des variables d'environnement sur Render

Une fois déployée, l'application est rendue accessible publiquement à travers une URL sécurisée (HTTPS) fournie par Render.

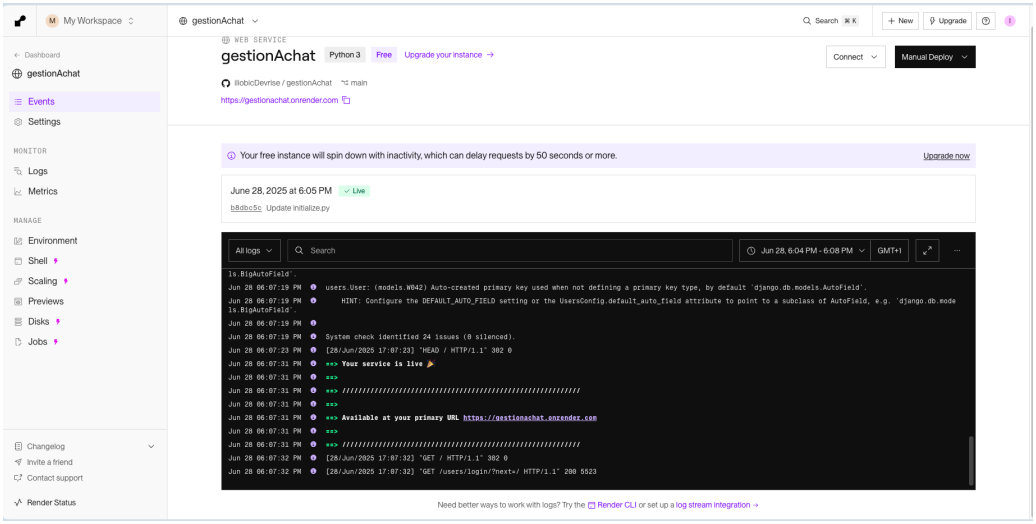

FIGURE 5.3 – Application en ligne et opérationnelle sur Render

3.3 Avantages

— **Simplicité de configuration** : aucune connaissance approfondie en administration système n'est requise.

— **Déploiement automatisé** : basé sur GitHub, ce qui facilite l'intégration continue.

— **Support natif de PostgreSQL** : avec gestion simplifiée des connexions et variables.

— **Scalabilité** : Render permet d'augmenter les ressources (RAM, CPU) en fonction des besoins.

4 Conclusion

Ce chapitre a permis de renforcer la qualité globale de l'application grâce à une série de tests rigoureusement menés, allant des tests unitaires aux tests utilisateurs. Ces validations successives ont permis de détecter et corriger les anomalies, d'assurer la conformité fonctionnelle des différents modules, et de garantir la robustesse de l'ensemble du système. Enfin, le déploiement de l'application sur la plateforme **Render** a permis de mettre en ligne une version stable et accessible, consolidant ainsi le cycle de développement jusqu'à la mise en production.

Conclusion Générale

Ce travail a été réalisé dans le cadre de notre projet de fin de cycle Master en Génie logiciel. Il avait pour principal objectif, le développement d'une application de gestion du processus d'achat des pièces de rechange avec l'intégration de l'intelligence artificielle destinée à faciliter la reconnaissance des pièces. Cette intégration vise à optimiser le temps de traitement et à réduire les erreurs,particulièrement dans un contexte industriel comme celui de l'entreprise Cevital, dont la base de données contient un grand nombre de références de pièce de rechange.

Afin d'atteindre cet objectif, nous avons commencé par présenter l'entreprise d'accueil et le contexte de notre stage. Nous avons ensuite introduit la méthodologie de conception hybride : combinant la robustesse du modèle en cascade et la flexibilité de la méthode Scrum, que nous avons adoptée durant tout le processus de développement. Une approche qui nous a permis de structurer efficacement notre travail tout en restant agile face aux imprévus.

Dans le cadre de la phase agile, nous avons mis en place un sprint 0 consacré à l'organisation du projet : l'identification des user stories, la rédaction du Product Backlog, la planification des sprints, et choix des outils et technologies utiliser pour l'implementation du projet (Framework Django, Python, Jira, etc.). Le travail a ensuite été divisé en quatre sprints, chacun documenté avec les diagrammes nécessaires et les livrables et documents produits.

Notre application, permet la gestion des différentes composante du processus d'achat : gestion des pièces de rechange, des utilisateurs et leurs droits d'accès, soumission et validation des différentes demandes (achat et réservation), passation et réception des commandes d'achat, ainsi que la gestion des fournisseurs, l'évaluation de leur performance,la gestion des contrats et le suivi des seuils de stock. Elle intègre également un chatbot conversationnel exploitant l'IA pour reconnaître les pièces de rechange directement à partir des descriptions textuelles fournies par des utilisateurs.

Malgré les difficultés rencontrées, notamment liées à la collecte d'informations au sein de l'entreprise d'accueil pour des raisons de sécurité et de restrictions internes et le manque de temps, nous avons pu réaliser et livrer une application fonctionnelle répondant aux besoins identifiés.

En guise de perspectives futures, nous envisageons d'enrichir notre application avec de nouvelles fonctionnalités, telles que la création de facture après validation d'une commande d'achat, l'intégration avancée d'une reconnaissance d'image pour les pièces, ainsi qu'un module de planification prenant en compte les consommations, les contraintes de durée de travail hebdomadaire et mensuelle, et la planification budgétaire liée aux opérations d'achat.

Annexe

Modèle Logique de Données (MLD)

1. Les règles de passage

- **Règle 1** : Chaque classe du diagramme de classes devient une relation. Son identifiant devient la clé primaire, et les autres champs deviennent des attributs. [58]
- **Règle 2** : Une association de type **un à plusieurs** (1 :N) se traduit par l'ajout d'une clé étrangère dans la table du côté "plusieurs" (N), qui référence la clé primaire de la table du côté "un" (1). [58]
- **Règle 3** : Une association de type **plusieurs à plusieurs** (N :N) se traduit par la création d'une nouvelle table intermédiaire qui contiendra une clé primaire composée des clés étrangères référençant les relations concernées.[58]
- **Règle 4 : Cas particulier** : Une association de type **un à un** (1 :1) se traduit par l'ajout d'une clé étrangère dans l'une des deux tables au choix. [58]
- **Règle 5 : Héritage** : L'héritage est modélisé par la création d'une relation pour chaque classe, de sorte que la super-classe (la classe mére) devient une relation contenant les attributs communs, puis chaque sous-classe (classe fille) devient une relation qui reprend la clé primaire de la super-classe (comme clé primaire et étrangère) et ajoute ses propres attributs.
- **Règle 6 : Classe auto-référencée** : Si une classe possède une association vers elle-même (récursivité), la relation correspondante contiendra une clé étrangère qui référence sa propre clé primaire. Par exemple : un employé peut avoir un manager, qui est lui-même un employé.

Le MLD

- **AbstractUser**(id, username,password, email, first-name, last-name, is-active, is-staff, is-superuser, last-login, date-joined)
- **user**(#id, role, image)
- **groupe**(id, nom, users)
- **fournisseur**(ref-fournisseur, name, adress, created-at, email, socialmetwork, company-code, description, mise-en-demeure, is-purchase-supplier, is-equipment-supplier, is-transporter, is-subcontractor, is-manufacturer, supplier-type, entity, currency, payment-mode, vat-adjustment, on-time-delivery-rate, quality-rating-avg, average-response-time, fulfillment-rate)
- **typeArticle**(code, designation, estReparable, au-forfait)
- **familleArticle**(code, description)
- **groupeArticle**(code, libelle)

- **magasin**(id, nom, localisation, description, #responsable, date-creation, statut, capacite, etage, section, boite)
- **produit**(codeProd, designation, entite, date-creation, equivalent, ref-equivalent, #type-article, #famille-de, #groupe-article, uniteStockage, #fournisseur, ref-fournisseur, constructeur, ref-constructeur, ref-equipement, poids, volume, qtte-min, qtte-max, centre-charge, rubrique, mots-cles, image)
- **stockProd**(#nom-magasin, #code-produit, quantite)
- **demandeReservation**(numReservation, dateCreation, dateSouhaitee, #demandeur, etat-reservation, #circuitValidation, derniereMaj, remarque, besoin-achat, jusqua, motifReservation, centreCharge, rubrique, pieces-jointes)
- **detailDemandeReservation**(#demande, #piece, quantite)
- **circuitValidation**(id, nom,#valideur1, #valideur2, #valideur3, statut-valideur1, date-validation-valideur1, statut-valideur2, date-validation-valideur2, date-validation-valideur3, statut-valideur3, type-circuit, motif-rejet)
- **demandeAchat**(numDA, dateCreation, statut, #demandeur, urgent, motifAchat, date-Souhaitee, natureAchat, #circuitValidation, typeDA, #acheteur, #numReservation)
- **detailDemandeAchat**(#demande-achat, #piece, quantite)
- **commandeAchat**(po-num, #fournisseur, date-commande, date-livraison, #num-demandeAchat, status, #circuitValidation, #demandeur, entite, typeCommande, contrat, #refContrat, conformite, motif-validation, dateValidation, note-qualité, date-emission, valideDu, jusqua, deviseTransaction, deviseDinar, taux-change, date-accuse-reception)
- **detailCommandeAchat**(#commande, #product, quantite)
- **Sortie**(numSortie, motifSortie, #demandeReservation, dateSortie, dateSortie-reel, sortie-effectuee, typeSortie, entite, remarque, #magasinier, centreCharge, rubrique, Quantite)
- **BonSortie**(#bon-sortie, #piece, quantite-sortie, #magasin-source)
- **arrivage**(numArrivage, dateCreation, numBL, dateBL, etat, #produit, #numCommande, devise, #foumisseurCode, foumisseurDesignation, typeArrivage, modeTransport, createur, #numReservation, stock-mis-a-jour)
- **bonEntree**(#arrivage, #product, quantite, #magasin-source)
- **contrat**(RefContrat, Type, #Refavenant, EstConfidentiel, Nature, Objet, #RefFournisseur, RefContratFournisseur, Achat, DateDebut, DateFin, Delai-de-preavis, Modalites-Paiement, MontantAvance, MontantContrat, Devise, Volume, DelaiPaiement, Echeance, Caution-Garantie, EstRFA, MoyensTransport, PartagerAvec, lieuContratPhysique, EstRecouvert, Commentaire, dateCreation, statut, conforme, #responsable-verification)
- **cycledeVie**(RefCDV, #RefContrat, Etape, Date)
- **rfa**(RefRFA, #RefContrat, TypeRFA, EstRecouvert, Montant)
- **priseencharge**(RefPEC, #RefContrat, #Personne, Role, Date)
- **notifications**(id,destinataire, message, lu, datecreation)
- **notificationDemandeAchat**(#id,#demande-achat)
- **notificationDemandeReservation**(#id,#demande-reservation)
- **notificationCommandeAchat**(#id,#commande-achat)
- **notificationContrat**(#id,#contrat)

Bibliographie

[1] Manutan. Chaîne d'approvisionnement : définition, rôle et enjeux. Disponible sur https://www.manutan.com/blog/fr/lexique/chaine-dapprovisionnement-definition-role-et-enjeux, publié le 21 mars 2024, (consulté en avril 2025).

[2] Flowie. Le processus d'achat en entreprise le guide complet. Disponible sur https://www.get-flowie.com/articles/processus-dachat-en-entreprise-le-guide-complet, (consulté en avril 2025).

[3] Agicap. *Défnition entreprise*. disponible sur https://agicap.com/fr/article/definition-entreprise/ (consulté en mai 2025).

[4] IBM. *Qu'est-ce que l'intelligence artificielle ?*. Disponible sur : https://www.ibm.com/fr-fr/think/topics/artificial-intelligence (consulté en mai 2025).

[5] IBM. *Qu'est-ce qu'un chatbot ?*. Disponible sur : https://www.ibm.com/fr-fr/think/topics/chatbots (consulté en mai 2025).

[6] Cevital. *l'histoire du groupe*. disponible sur https://www.cevital.com/lhistoire-du-groupe/ (consulté en mars 2025)

[7] IONOS. WaterFall Model , Disponible sur : https://www.ionos.fr/digitalguide/sites-internet/developpement-web/modele-en-cascade/. (Consulté en février 2025)

[8] Atlassian. (s.d). Consulté le 15/02/2025 : https://www.atlassian.com/fr/agile/scrum

[9] Tuleap. (s.d). Consulté le 15/02/2025 : https://www.tuleap.org/fr/agile/comprendre-methode-agile-scrum-10-minutes#Scrum-cest-quoi

[10] Ken Schwaber & Jeff Sutherland (2020) *The Scrum Guide*

[11] PLanZone. (s.d). Consulté le 16/02/2025 : https://www.planzone.fr/blog/quest-ce-que-la-methodologie-extreme-programming

[12] Boehm, B. & Turner, R. (2003). *Balancing Agility and Discipline : A Guide for the Perplexed*. Addison-Wesley.

[13] Ambler, S. & Lines, M. (2012). *Disciplined Agile Delivery : A Practitioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press

[14] Project Management Institute (2017). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Sixth Edition* et *Agile Practice Guide*.

[15] McKinsey & Company. (2024). *How to create an agile organization*. Disponible sur : https://www.mckinsey.com

[16] BCG. (2021). *How Companies Can Manage a Successful Transformation*. Disponible sur : https://www.bcg.com

[17] BCG. (2023). *Agile Development's Biggest Failure Point—and How to Fix It*. Disponible sur : https://www.bcg.com

[18] McKinsey & Company. (2023). *The Future of DevOps in Enterprise IT*. Disponible sur : https://www.mckinsey.com

[19] La galaxie de Futura. (s.d). Consulté le 14/02/2025, à partir de : https://www.futura-sciences.com/tech/definitions/informatique-uml-3979/

[20] Lucidchart. *Qu'est-ce qu'un diagramme de cas d'utilisation UML ?*. disponible sur https://www.lucidchart.com/pages/fr/diagramme-de-cas-dutilisation-uml

[21] Lucidchart. *Qu'est-ce qu'un diagramme de séquence UML ?*. disponible sur https://www.lucidchart.com/pages/fr/diagramme-de-sequence-uml

[22] Lucidchart. *Qu'est-ce qu'un diagramme de classe UML ?*. disponible sur https://www.lucidchart.com/pages/fr/diagramme-de-classes-uml

[23] Lucidchart. *Qu'est-ce qu'un diagramme d'états-transitions ?*. disponible sur https://www.lucidchart.com/pages/fr/diagramme-etats-transitions-uml

[24] Blog Gestion de projet, 2024https://blog-gestion-de-projet.com/analyse-pestel/

[25] Le coin des Entrepreneurs, Consulté le 17-02-2024https://www.lecoindesentrepreneurs.fr/analyse-pestel

[26] Ad Valoris. (s.d). Consulté le 14/02/2025, https://www.advaloris.ch/gestion-de-projet/definir-besoins-gestion-projet.

[27] Ad Valoris. (s.d). Consulté le 14/02/2025, https://www.advaloris.ch/gestion-de-projet/definir-besoins-gestion-projet.

[28] IBM. (s.d.). Consulté le 14/02/2025, https://www.ibm.com/docs/fr/dmrt/9.5?topic=diagrams-actors

[29] PM coaching. Consulté le 15/02/2025, https://www.pm-coaching.org/blog/blog074

[30] Python Documentations. *Le tutoriel Python* disponible sur https://docs.python.org/fr/3.13/tutorial/

[31] Django Documentations. *Coup d'œil sur Django* disponible sur https://docs.djangoproject.com/fr/5.2/intro/overview/.

[32] Oracle. disponible sur https://www.oracle.com/fr/database/definition-postgresql/ (consulté en Juin 2025)

[33] ESGI. *Quelle est la différence entre HTML, CSS et JavaScript ?* https://www.esgi.fr/actualites/17022022-quelle-est-la-difference-entre-html-css-javascript publié en février 2022.

[34] Mdn Web Docs. *Qu'est-ce que le JavaScript ?*. disponible sur https://developer.mozilla.org/fr/docs/Learn_web_development/Core/Scripting/What_is_JavaScript

[35] Chart.js. *Chart.js Docs*. disponible sur https://www.chartjs.org/docs/latest/

[36] Atlassian. *Jira*. Disponible sur https://www.atlassian.com/fr/software/jira, consulté en juin 2025.

[37] Miro. *Centre d'aide Miro*. Disponible sur https://help.miro.com/hc/fr, consulté en juin 2025.

[38] GitHub. *Documentation GitHub*. Disponible sur https://docs.github.com/fr, consulté en juin 2025.

[39] PlantUML. Disponible sur https://plantuml.com/fr/, consulté en juin 2025.

[40] Lucidchart. *Lucid Documents*. Disponible sur https://www.lucidchart.com/pages/fr, consulté en juin 2025.

[41] DataScientest. *PyCharm : tout savoir*. Disponible sur https://datascientest.com/pycharm-tout-savoir, consulté en juin 2025.

[42] Microsoft. *Visual Studio Code*. Disponible sur https://visualstudio.microsoft.com/fr/, consulté en juin 2025.

[43] DataScientest. *Postman*. Disponible sur https://datascientest.com/postman-tout-savoir consulté en juin 2025.

[44] CloudFlare. *Qu'est-ce qu'un grand modèle de langage (LLM) ?*. disponible sur https://www.cloudflare.com/fr-fr/learning/ai/what-is-large-language-model/ (consulté en juin 2025)

[45] Oracle. *Artificial Intelligence*. disponible sur https://www.oracle.com/africa-fr/artificial-intelligence/ (consulté en juin 2025)

[46] DataScientest. *Le word Embedding*. disponible sur https://datascientest.com/le-word-embedding (consulté en juin 2025)

[47] DataScientist. *Agents ReAct* . disponible sur https://datascientist.fr/blog/agents-react-la-nouvelle-frontiere-de-l-ia-generative (consulté en juin 2025)

[48] DataScientest. *APi, tout savoir*. disponible sur https://datascientest.com/api-tout-savoir (consulté en juin 2025)

[49] IBM. *Qu'est-ce que LangChain ?*. Disponible sur https://www.ibm.com/fr-fr/think/topics/langchain (consulté en juin 2025)

[50] IBM. *Qu'est-ce que LangGraph ?*. Disponible sur https://www.ibm.com/think/topics/langgraph (consulté en Juin 2025)

[51] FastApi. disponible sur https://fastapi.tiangolo.com/ (consulté en juin 2025)

[52] Supabase. disponible sur https://supabase.com/ (consulté en juin 2025)

[53] SQLModel. disponible surhttps://sqlmodel.tiangolo.com/ (consulté en juin 2025)

[54] Meta. disponible sur https://ai.meta.com/tools/faiss/ (consulté en juin 2025)

[55] Django Documentations. *VÉcriture et lancement de tests*. Disponible sur : https://docs.djangoproject.com/fr/5.2/topics/testing/overview/ (consulté en juin 2025).

[56] IONOS, *CRUD : les opérations de base de données les plus importantes*, 2023 Disponible sur : https://www.ionos.fr/digitalguide/sites-internet/developpement-web/crud-les-operations-de-base-de-donnees-les-plus-importantes/ (consulté en juin 2025).

[57] Render. Disponible sur : https://render.com/docs (consulté en juin 2025)

[58] Chochois, Ph. « Base de données : Modèle Logique de Données », pp. 1–4.

Résumé

Dans le cadre de la transformation digitale des processus industriels, la gestion efficace des achats joue un rôle stratégique dans l'amélioration des performances des entreprises. Ce mémoire, intitulé « Conception, développement et réalisation d'une application web de gestion du processus d'achat de Cevital avec intégration d'intelligence artificielle », présente la mise en œuvre d'une solution visant à digitaliser, automatiser et enrichir le cycle d'achat au sein de l'entreprise Cevital.

Le projet repose sur une approche méthodologique hybride, combinant les modèles en cascade (Waterfall) et Agile, permettant une analyse structurée tout en conservant une flexibilité d'adaptation aux retours fréquents. L'application développée permet non seulement la gestion complète du processus d'achat (de la demande à la réception), mais intègre également un chatbot intelligent basé sur des techniques d'intelligence artificielle, facilitant l'assistance utilisateur et la recherche d'informations.

Ce mémoire détaille chaque phase du développement, de l'analyse des besoins à la mise en production, en soulignant les choix techniques, les difficultés rencontrées et les solutions adoptées.

Mots-clés : Digitalisation, Achat, Cevital, Application Web, Intelligence Artificielle, Chatbot.

Abstract

As part of the digital transformation of industrial processes, effective purchasing management plays a strategic role in improving company performance. This dissertation, entitled *"Design, Development, and Implementation of a Web Application for Purchase Process Management at Cevital with Artificial Intelligence Integration"*, presents the implementation of a solution aimed at digitizing, automating, and enriching the purchase cycle within Cevital.

The project is based on a hybrid methodological approach combining the Waterfall and Agile models, enabling structured analysis while maintaining the flexibility to incorporate frequent feedback. The developed application not only enables full management of the purchasing process (from request to reception), but also integrates an intelligent chatbot based on artificial intelligence techniques, facilitating user assistance and information retrieval.

This dissertation details each phase of the development process, from requirements analysis to deployment, highlighting technical choices, challenges encountered, and adopted solutions.

Keywords: Digitalization, Purchasing, Cevital, Web Application, Artificial Intelligence, Chatbot.