MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

RÉPUBLIQUE DE CÔTE D'IVOIRE

Union – Discipline – Travail

MEMOIRE DE FIN D'ETUDE

Master professionnel Génie Informatique et Réseau

Conception et réalisation d'une application grand public de gestion ERP d'une PME

PRÉSENTÉ PAR

AMANGOUA YAPO YVES FREJUS

ENCADREUR PÉDAGOGIQUE

M. AKPATOU INNOCENT

ENSEIGNANT À PIGIER CÔTE D'IVOIRE

ANNÉE SCOLAIRE : 2021-2022

Sommaire

Dédicace

À mon père,
AMANGOUA FREDERIC, qui, depuis notre tendre enfance, s'évertue à tracer les sillons d'un avenir prometteur. Par son amour paternel et sa rigueur exemplaire, il continue encore aujourd'hui d'incarner à mes yeux un modèle de travail et de dévouement.

À ma mère,
Mme YAPO COCO épouse AMANGOUA, une femme dévouée et aimante, qui ne cesse de nous offrir son soutien indéfectible en toutes circonstances.

À ma sœur et son époux,
AMANGOUA ANITA Epse DONGOSSI et **DONGOSSI HERMANN**, qui ont toujours été présent à mes côtés. Dans les moments difficiles de ma vie, ils n'ont jamais failli à leur rôle de soutien et de conseillers exceptionnels.

À mon frère et ami, et à son épouse,
KONATE N'GOLO HAMED et **BAGO AUDE-BENEDICTE**, pour leur affection fraternelle et leur soutien constant qui me donnent toujours la force d'avancer.

À mon collègue devenu frère et ami,
NDIAYE ELHADJ, pour sa fraternité, sa présence infaillible et son soutien sincère qui m'ont accompagné tout au long de mon parcours.

À ma fiancée,
DOUMBIA FATIM, qui partage avec moi chaque instant, qu'il soit heureux ou éprouvant, et qui, par sa présence, m'apporte force et sérénité.

Remerciements

« Si j'ai pu voir aussi loin, c'est parce que j'étais perché sur les épaules de géants » NEWTON

Nous n'avons certainement pas vu aussi loin que NEWTON, mais nous ne sommes pas pour autant monté sur les épaules de moins géants que les siens.

Nous remercions tout le personnel de l'administration et les enseignants de PIGIER COTE D'IVOIRE ainsi que le personnel de l'entreprise GENIE INFORMATIQUE CÔTE D'IVOIRE et plus particulièrement :

- ✧ Mme ELLA ASSIE (Directrice de L'entreprise TAPA Conseil)
- ✧ M. AKPATOU INNOCENT (Ingénieur informaticien et professeur à PIGIER COTE D'IVOIRE)
- ✧ M. SANE ARNAUD (Ingénieur informaticien et professeur à PIGIER COTE D'IVOIRE)

Avant-propos

L'Institut Pigier Côte d'Ivoire figure parmi les institutions de référence en Côte d'Ivoire pour la formation académique et professionnelle. Depuis plusieurs décennies, il œuvre à former des cadres compétents et opérationnels dans divers domaines, notamment en Informatique, Marketing, Gestion, et Comptabilité.
Le MAster Génie Informatique et Réseaux (MAGIR) est une formation de niveau BAC+5 qui allie rigueur académique et compétences pratiques. Il prépare les étudiants à relever les défis du monde professionnel en leur inculquant des connaissances approfondies ainsi que des aptitudes techniques avancées.

Ce programme se décline en deux principales étapes. La première phase consiste en des cours théoriques, des travaux dirigés, et des ateliers pratiques visant à renforcer les bases techniques et méthodologiques des étudiants. La deuxième phase, axée sur la mise en pratique, s'effectue sous forme de stages en entreprise et de projets de fin d'études. Elle permet aux apprenants de s'immerger dans des environnements professionnels, de mettre en œuvre les compétences acquises et d'apporter des solutions innovantes aux problématiques rencontrées.

Dans ce cadre, nous avons réalisé un stage pratique au sein de l'entreprise [Nom de l'entreprise] pour finaliser notre cursus académique. Ce stage a donné lieu à des travaux sur le thème : Conception et réalisation d'une application grand publique de gestion de stock, client, trésorerie, achat et vente d'une PME, qui constitue l'objet du présent mémoire.

Résumé

Dans l'optique de faciliter et d'optimiser la gestion globale d'une PME, ce projet a pour objectif la conception et la réalisation d'une application grand public intégrant des modules pour la gestion des stocks, des clients, de la trésorerie, des achats et des ventes.
L'application vise à offrir une solution numérique accessible, permettant aux utilisateurs autorisés de gérer efficacement leurs opérations depuis leurs smartphone connecté à internet.

La phase initiale du projet a été consacrée à une étude conceptuelle détaillée, visant à organiser et structurer les processus de développement à travers des diagrammes et des modèles UML. La méthodologie adoptée repose sur les Processus Unifiés, garantissant une approche rigoureuse et progressive pour le développement de l'application.

Pour la phase de développement, MySQL a été choisi comme Système de Gestion de Bases de Données (SGBD), et l'ensemble du code a été rédigé avec Visual Studio Code en utilisant PHP et Dart, appuyés par les framework Laravel et Flutter pour garantir une architecture robuste et évolutive.

Abstract

In order to streamline and optimize the overall management of small and medium-sized enterprises (SMEs), this project aims to design and develop a user-friendly application incorporating modules for inventory management, client management, treasury, purchases, and sales.

The application is designed to provide an accessible digital solution, enabling authorized users to efficiently manage their operations from their smartphones connected to the internet.

The initial phase of the project focused on a detailed conceptual study to organize and structure the development processes using diagrams and UML models. The methodology adopted is based on the Unified Process, ensuring a rigorous and systematic approach to application development.

During the development phase, MySQL was chosen as the Database Management System (DBMS), and the codebase was implemented using Visual Studio Code with PHP and Dart, supported by the Laravel and Flutter frameworks to deliver a robust and scalable architecture.

Table des figures, photos et tableaux

Liste des accronymes

HTML : Hypertext Markup Language

HTTP : Hypertext Transport Protocol

HTTPS : Hypertext Transport Protocol Secure

MCD : Modèle Conceptuel des Données

MCT : Modèle Conceptuel des Traitements

MERISE : Méthode d'Étude et de Réalisation Informatique par Sous Ensemble

MLDr : Modèle Logique des Données relationnelles

MOpT : Modèle Opérationnel des Traitements

MOT : Modèle Organisationnel des Traitements

MPD : Modèle Physique des Données

MySQL : My Structured Query Language

PHP : Hypertext Preprocessor

PU : Processus Unifié

SGBD : Système de Gestion de Bases de Données

SGBDR : Système de Gestion de Bases de Données Relationnelles

SI : Système d'Information

SQL : Structured Query Language

UC : Use Case

UML : Unified Modeling Language

Introductions

L'informatique occupe une place centrale dans la gestion et le développement des petites et moyennes entreprises (PME). En tant que science du traitement automatique et rationnel de l'information, elle est devenue incontournable dans tous les secteurs d'activité, permettant aux entreprises d'optimiser leurs processus et de gagner en efficacité. Face à l'évolution constante des nouvelles technologies, de nombreuses applications de gestion ont vu le jour, offrant des solutions adaptées aux besoins spécifiques des structures, qu'elles soient petites ou grandes.

Dans ce contexte, la gestion des stocks, des clients, de la trésorerie, des achats et des ventes constitue un défi majeur pour les PME. Ces processus, lorsqu'ils sont mal gérés, peuvent engendrer des pertes financières, une mauvaise allocation des ressources et une baisse de la compétitivité.

Ce projet s'inscrit dans une volonté d'automatiser et de centraliser ces processus à travers la conception et la réalisation d'une application numérique dédiée. Il s'agit de développer un outil accessible, capable de répondre aux besoins d'une gestion simplifiée et efficace, tout en offrant une flexibilité d'utilisation via des terminaux connectés à internet.
Le sujet de notre projet est ainsi intitulé : « Conception et réalisation d'une application grand public de gestion ERP d'une PME ».

Notre étude se structure autour de quatre (4) grandes parties :

- ✧ La première partie, intitulée "Cadre de référence", présente le contexte général du projet, les problématiques rencontrées ainsi que les choix méthodologiques adoptés.

- ✧ La deuxième partie, "Étude préalable", analyse le système existant, identifie ses limites et propose des solutions adaptées, qui serviront de base au développement.

- ✧ La troisième partie, intitulée "Étude détaillée de la solution retenue", se concentre sur la description des cas d'utilisation essentiels. Elle inclut également l'identification des diagrammes UML et des classes associées, ainsi que la présentation des modèles d'analyse des données, garantissant une structuration efficace des fonctionnalités.

- ✧ Enfin, la quatrième partie, intitulée "Réalisation et déploiement", détaille l'étude technique menée, suivie de l'implémentation et du déploiement du système global. Cette partie s'achève avec des captures d'écran illustrant les principales interfaces et fonctionnalités de la solution finale.

Première partie : Cadre de reférence

I- Présentation de TAPA Conseil

1. Notre histoire

TAPA Conseil est né d'une histoire personnelle et d'une passion. Notre fondatrice, après avoir traversé les défis financiers et découvert la puissance d'une gestion éclairée, a décidé de transformer ses apprentissages en une mission de vie : aider les autres à éviter les erreurs coûteuses et à saisir les opportunités.

Tout a commencé avec des ateliers d'éducation financière dans des cercles restreints. Rapidement, les résultats positifs et les retours enthousiastes ont permis d'étendre notre vision. Aujourd'hui, TAPA Conseil accompagne particuliers et entreprises à chaque étape de leur parcours financier, avec une approche alliant pédagogie et stratégie.

2. Notre vision

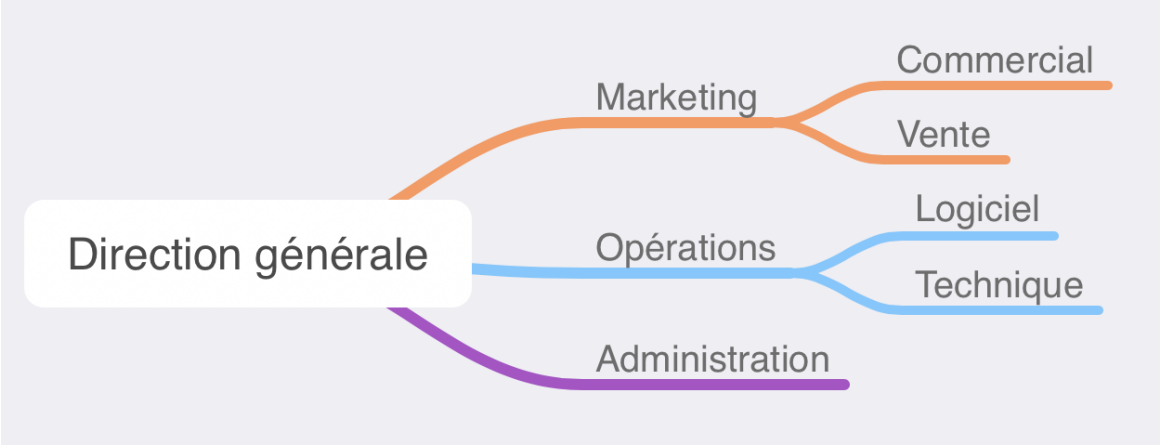
Chez TAPA Conseil, nous croyons en un avenir où chacun a les outils nécessaires pour prendre des décisions financières éclairées. Notre ambition est de devenir une référence dans l'éducation financière, l'entrepreneuriat et l'investissement, tout en inspirant des pratiques financières responsables et durables.

Nous sommes convaincus que chaque personne a le potentiel de bâtir une vie financière stable et prospère. Avec les bons conseils et un accompagnement adéquat, même les objectifs les plus ambitieux peuvent devenir des réalités tangibles.

3. Nos valeurs

- ✧ Accessibilité : Rendre la gestion financière compréhensible et accessible à tous.
- ✧ Innovation : Proposer des solutions modernes adaptées aux défis d'aujourd'hui.
- ✧ Empathie : Comprendre les besoins uniques de nos clients pour leur offrir un accompagnement personnalisé.
- ✧ Responsabilité : Encourager des pratiques financières éthiques et durables.
- ✧ Excellence : Garantir un service de qualité et des résultats mesurables.

4. Organisation



II- Présentation du projet

1. Problématique

Aujourd'hui, de nombreux petits commerçants et entrepreneurs en Côte d'Ivoire et dans la région continuent de gérer leurs affaires de manière traditionnelle, souvent en notant leurs ventes et leurs dépenses sur des carnets papier ou en les mémorisant. Ce mode de gestion manuel, bien que répandu, présente plusieurs inconvénients majeurs :

- ✧ Risque élevé de pertes de données : Les carnets peuvent être perdus, endommagés ou simplement oubliés. Cela conduit à une perte d'informations cruciales, telles que le suivi des ventes, des dettes clients ou des commandes fournisseurs.

- ✧ Absence de visibilité sur les performances : Les commerçants ont du mal à suivre leurs revenus, leurs dépenses et leurs stocks de manière précise et en temps réel. Cela peut entraîner des erreurs dans la gestion de stock (ruptures ou excédents), des pertes financières non identifiées, et des difficultés à anticiper les besoins futurs.

- ✧ Manque de temps et d'habitude : Beaucoup de commerçants n'ont ni le temps ni l'habitude de structurer leur gestion avec des outils complexes ou des processus formalisés. Cette absence de rigueur impacte directement la rentabilité de leur entreprise, car ils ne disposent pas d'une vue claire de leur santé financière.

- ✧ Difficultés à accéder aux services financiers : En raison de cette gestion informelle et souvent approximative, ces commerçants ont du mal à fournir les documents financiers nécessaires pour accéder à des crédits bancaires ou des partenariats stratégiques. Cela limite leurs opportunités de croissance.

2. Objectif général

Notre application se propose de résoudre cette problématique en offrant une solution numérique simple, intuitive et adaptée à ces commerçants. Elle permet de passer d'une gestion manuelle à une gestion digitalisée, sans complexité excessive. L'application permettra notamment à une PME de gérer efficacement ses stocks, sa clientèle, sa trésorerie, ses achats et ses ventes au sein d'une seule application mobile intégrée.

III- Choix de la méthode d'analyse

1. Méthode MERISE

MERISE (Méthode d'Étude et de Réalisation Informatique des Systèmes d'Entreprise), est une méthode d'analyse informatique née vers 1978 en France. Elle est très répandue de nos jours et est beaucoup utilisée dans la conduite et la conception de projets informatiques.

1.1 Les principes généraux

MERISE est une méthode qui a une double vocation. C'est avant tout une méthode de conception de systèmes informatiques (SI) à savoir :

- ✧ Une approche globale du SI menée parallèlement sur les données et les traitements.
- ✧ Une description du SI par niveaux :
  - ■ Le niveau conceptuel
  - ■ Le niveau logique ou organisationnel
  - ■ Le niveau physique ou opérationnel
- ✧ Une description du SI utilisant un formalisme de représentation précis, simple rigoureux pour la description des données

✧ Une représentation visuelle des modèles conceptuels.

Aussi, MERISE propose une démarche de développement de ce SI à travers :

✧ Un découpage du processus de développement en quatre étapes
  - Étude préalable
  - Étude détaillée
  - Réalisation
  - Mise en œuvre

✧ Une description de la structure de travail à mettre en place pour mener à bien le développement du SI.

1.2 Présentation des niveaux de conception

L'étude suivant MERISE distingue trois niveaux de conception de système d'informations :

1.2.1 Niveau conceptuel

À ce niveau, il est établi une description des finalités de l'entreprise en précisant le « QUOI » tout en faisant abstraction des contraintes organisationnelles et techniques. Il sera fait une description des données stables ou données invariantes du SI et de l'ensemble des règles de gestion qui y sont appliquées au niveau des concepts par le biais d'un formalisme qui peut se traduire en termes de :

✧ Modèle conceptuel des données (MCD) La description des données et des relations est réalisée à partir du formalisme individuel suivant : Objet, Relation, Propriété.

✧ Modèle conceptuel des traitements (MCT) Ses concepts : Processus, Opération, Évènement, Résultat, synchronisation

✧

1.2.2 Niveau organisationnel ou logique

Ce niveau définit l'organisation qu'il est souhaitable de mettre en place dans l'entreprise pour atteindre les objectifs souhaités. Il faut préciser les choix d'organisation qui seront pris en compte :

✧ La répartition des tâches entre l'homme et la machine Conception d'une application de gestion de stock et du processus achat

- ✧ Le mode de fonctionnement : temps réel (conversationnel), temps différé (batch)
- ✧ La répartition géographique des données et des traitements. En claire, ce niveau décrit le « qui fait quoi et où ». Les modèles associés à ce niveau de description sont :
- ✧ Modèle Logique des Données (MLD), qui peut être selon le cas Codasyl, Relationnel, Fichier classique.
- ✧ Modèle Organisationnel des Traitements (MOT), qui permet de représenter par phases les tâches exécutées et les postes de travail correspondants.

1.2.3 Niveau opérationnel ou physique

Il définit les organisations physiques des données au travers du Modèle Physique des Données (MPD) et la description des traitements effectués par unité de traitements au travers du Modèle Opérationnel des Traitements (MOPT). À ce niveau, le MOPT, décrit « LE COMMENT FAIRE ». La méthode de conception proposée par MERISE nous a présenté une vue globale des différents niveaux applicables par cette méthode pour mener à bien
un projet.

Cependant, ces concepts nécessitent une analyse détaillée du SI et une délimitation précise du domaine d'activités concerné.

NIVEAU D'ABSTRACTION	DONNEES	TRAITEMENTS
CONCEPTUEL	Modèle Conceptuel des Données (MCD)	Modèle Conceptuel des Traitements (MCT)
ORGANISATIONNEL Ou LOGIQUE	Modèle Logique des Données (MLD)	Modèle Organisationnel des Traitements (MOT)
Modèle Organisationnel des Traitements (MOT)	Modèle Physique des Données (MPD)	Modèle Opérationnel des Traitements (MOpT)

Figure 1: Synthèse d'une étude conceptuelle avec MERISE

1.3 Avantages

MERISE faisant partie des approches dites systémiques a comme avantages :

- ✧ Structure : la méthode MERISE est une méthode structurée pour la conception de systèmes d'information. Elle est basée sur des étapes claires et des règles précises, ce qui rend la conception de systèmes d'information plus cohérente et plus facile à gérer.
- ✧ Scalabilité : la méthode MERISE peut être utilisée pour des projets de toutes tailles, des petits projets aux projets d'envergure nationale. Elle peut être adaptée en fonction de la complexité du projet.
- ✧ Visualisation : la méthode MERISE permet de visualiser les données et les processus du système d'information de manière claire et structurée. Cela permet de mieux comprendre le système d'information et de faciliter la communication entre les différents acteurs impliqués.
- ✧ Gestion de projet : la méthode MERISE est également utile pour la gestion de projet. Elle permet de suivre l'avancement du projet et de vérifier si toutes les étapes sont respectées.

1.4 Inconvénients

En dépit des avantages sus cités, MERISE a comme inconvénients :

- ✧ Complexité : la méthode MERISE peut être assez complexe pour les personnes qui ne sont pas familières avec la méthode. Elle nécessite une certaine expertise pour être appliquée correctement.
- ✧ Rigidité : la méthode MERISE peut être assez rigide. Elle ne permet pas toujours de prendre en compte les changements dans les besoins du système d'information pendant la conception.
- ✧ Coût : la méthode MERISE peut également être coûteuse. Elle nécessite souvent l'utilisation de logiciels spécialisés et la formation de personnel qualifié pour la mettre en œuvre.
- ✧ Temps : la méthode MERISE peut également prendre beaucoup de temps pour être mise en œuvre. Elle nécessite souvent des étapes complexes et peut prendre plus de temps que les méthodes moins structurées.

2. Processus Unifié (PU)

Le Processus Unifié (PU) est une méthode de développement logiciel itérative et incrémentale qui vise à offrir une approche structurée et rigoureuse pour la conception, la mise en œuvre et la maintenance de systèmes logiciels. Le PU est basé sur le paradigme orienté objet et met l'accent sur la collaboration entre les développeurs, la gestion des risques et la satisfaction des besoins des utilisateurs finaux. Le processus se concentre sur la production de livrables qui reflètent les activités de développement. Le PU est également personnalisable et s'adapte aux besoins de chaque projet, en offrant une flexibilité dans la sélection des activités, des outils et des techniques de développement appropriés pour le projet en question. Il est organisé en quatre phases principales, chacune étant associée à des activités spécifiques :

2.1. Les phases du PU

Cette phase se concentre sur la compréhension du contexte commercial et des besoins de l'utilisateur, la définition des objectifs du projet et l'évaluation de la faisabilité du projet.

2.1.2. La phase d'élaboration

Cette phase se concentre sur la création d'une architecture logicielle, la planification du projet et l'identification des risques.

2.1.3. La phase de Construction

Cette phase se concentre sur la mise en œuvre du système, le développement des fonctionnalités et la réalisation des tests.

2.1.4. La phase de Transition

Cette phase se concentre sur le déploiement du système, la formation des utilisateurs finaux, la maintenance et le support. Le PU encourage également l'utilisation d'artefacts tels que les diagrammes UML, les modèles de processus, les documents de spécification de logiciels et les plans de projet pour faciliter la communication et la collaboration au sein de l'équipe de développement. UML fournit un moyen permettant de représenter diverses projections d'une même représentation grâce aux vues. Une vue est constituée d'un ou plusieurs diagrammes. Il y a trois (3) types de vue :

2.2. Les vues statiques, répresentation physique du système

2.2.1. Diagramme d'objet

Un diagramme d'objets est un diagramme de structure qui montre un ensemble d'objets et les relations entre eux, à un point particulier du temps pendant l'exécution du système modélisé. On peut considérer un diagramme d'objets comme étant une instance d'un diagramme de classe.

2.2.2. Diagramme de classe

Le diagramme de classe est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques. Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes sont utilisées dans la programmation orientée-objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

2.2.3. Diagramme de composants

Le diagramme de composants décrit l'organisation du système du point de vue des éléments logiciels comme les modules (paquetages, fichiers sources, bibliothèques, exécutables), des données (fichiers, bases de données) ou encore d'éléments de configuration (paramètres, scripts, fichiers de commandes). Ce diagramme permet de mettre en évidence les dépendances entre les composants (qui utilisent quoi).

2.2.4. Diagramme de déploiement

En UML, un diagramme de déploiement est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux. Les éléments utilisés par un diagramme de déploiement sont principalement les nœuds, les composants, les associations et les artefacts. Les caractéristiques des ressources matérielles physiques et des supports de communication peuvent être précisées par stéréotype.

2.2.5. Diagramme de paquetages

Les diagrammes de paquetages sont la représentation graphique des relations existant entre les paquetages (ou espaces de noms) composant un système, dans le langage Unified Modeling Language (UML).

2.2.6. Diagramme structure composite

Dans le langage UML, le diagramme de structure composite expose la structure interne d'une classe ainsi que les collaborations que cette dernière rend possible. Les éléments de ce diagramme sont les parties (en anglais parts), les ports par le biais desquels les parties interagissent entre elles, avec différentes instances de la classe ou encore avec le monde extérieur, et enfin les connecteurs reliant les parties et les ports. Une structure composite est un ensemble d'éléments interconnectés collaborant dans un but commun lors de l'exécution d'une tâche. Chaque élément se voit attribuer un rôle dans la collaboration.

2.3. Les vues dynamiques, répresentation fonctionnelle du système

2.3.1. Diagramme de séquence

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language (UML).

Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un scénario d'un Diagramme des cas d'utilisation. Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets. La dimension verticale du diagramme représente le temps, permettant de visualiser l'enchaînement des actions dans le temps, et de spécifier la naissance et la mort d'objets. Les périodes d'activité des objets sont symbolisées par des rectangles, et ces objets dialoguent à l'aide de messages.

2.3.2. Diagramme de collaboration

Un diagramme de collaboration est un diagramme d'interactions UML 1, simplification d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets, et où la

chronologie n'intervient que de façon annexe. Il consiste en un graphe dont les nœuds sont des objets et les arcs (numérotés selon la chronologie) les échanges entre ces objets. Les diagrammes de collaboration ont été remplacés en UML 2. Par les diagrammes de communication.

2.3.3. Diagramme de temps

Un diagramme de temps est un diagramme d'interaction où l'attention est portée sur les contraintes temporelles dans le langage UML 2.

2.3.4. Diagramme de communication

Un diagramme de communication est un diagramme d'interactions UML 2 (appelé diagramme de collaboration en UML 1), représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets. En fait, le diagramme de séquence et le diagramme de communication sont deux vues différentes mais logiquement équivalentes (on peut construire l'une à partir de l'autre) d'une même chronologie. Ils sont dits isomorphes. C'est une combinaison entre le diagramme de classes, celui de séquence et celui des cas d'utilisation. Il rend compte à la fois de l'organisation des acteurs aux interactions et de la dynamique du système. C'est un graphe dont les nœuds sont des objets et les arcs (numérotés selon la chronologie) les échanges entre objets.

2.4. Les vues comportementales

2.4.1. Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils
interagissent avec les cas d'utilisation (use cases).

2.4.2. Diagramme d'états-transition

Un diagramme états-transitions est un schéma utilisé en génie logiciel pour représenter des automates déterministes. Il fait partie du modèle UML et s'inspire principalement du formalisme des statecharts et rappelle les grafcets des automates. S'ils ne permettent pas de comprendre globalement le fonctionnement du système, ils sont directement transposables en algorithme. Tous

les automates d'un système s'exécutent parallèlement et peuvent donc changer d'état de façon indépendante.

2.4.3. Diagramme d'activité

Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements en parallèle (multithreads ou multiprocessus). Le diagramme d'activité est également utilisé pour décrire un flux de travail (workflow).

2.5. Avantages

Structuration du développement : Le PU offre une structure claire pour le développement de logiciels, en définissant des phases et des activités précises qui doivent être accomplies à chaque étape. Cela aide à garantir que le développement suit une approche systématique et cohérente.

- ✧ Adaptabilité : Le PU est personnalisable et peut être adapté aux besoins de chaque projet. Cela permet de sélectionner les activités, les outils et les techniques de développement appropriés pour chaque projet.
- ✧ Collaboration et communication : Le PU met l'accent sur la collaboration et la communication entre les membres de l'équipe de développement. Cela permet d'assurer que tous les membres de l'équipe travaillent ensemble pour atteindre les objectifs du projet. Conception d'une application de gestion de stock et du processus achat
- ✧ Gestion des risques : Le PU met également l'accent sur la gestion des risques, en identifiant les risques potentiels dès le début du projet et en développant des stratégies pour les atténuer.
- ✧ Qualité du produit : Le PU met l'accent sur la qualité du produit logiciel final en utilisant des artefacts tels que les diagrammes UML et les modèles de processus pour assurer que toutes les exigences ont été prises en compte et que le système est bien conçu.
- ✧ Flexibilité : Le PU permet une grande flexibilité dans la planification du projet et l'adaptation aux changements, en offrant la possibilité d'itérations et d'ajustements en cours de route.

2.6 Inconvénients

Complexité : Le PU est un processus complexe qui peut être difficile à mettre en œuvre pour les petites équipes de développement ou les projets à court terme.

- ✧ Coût : Le PU peut être coûteux en termes de temps, de ressources et de budget nécessaires pour mettre en œuvre toutes les activités et les phases du processus.
- ✧ Temps : Le PU peut également prendre plus de temps que d'autres méthodes de développement logiciel en raison de sa nature itérative et de son emphase sur la documentation.
- ✧ Rigidité : Bien que le PU soit personnalisable, il peut être considéré comme trop rigide pour certains projets qui nécessitent plus de flexibilité dans la planification et la mise en œuvre.
- ✧ Dépendance aux outils : Le PU nécessite souvent l'utilisation d'outils spécifiques pour la modélisation, la gestion de projets, le suivi des exigences, etc. Cela peut créer une dépendance aux outils et entraîner des coûts supplémentaires pour l'acquisition et la formation de l'équipe. Conception d'une application de gestion de stock et du processus achat

- ✧ Adaptabilité : Bien que le PU soit personnalisable, il peut être difficile de l'adapter à des projets très différents ou à des équipes de développement très différentes.

3. Analyse et justification du choix

MERISE	Processus Unifié
Méthode d'analyse et de conception de système d'information qui se concentre sur la modélisation des données, des traitements et des flux d'informations	Méthode de développement logiciel qui se concentre sur la gestion de projet et sur l'élaboration d'un processus itératif et incrémental
Comporte trois phases principales : - La phase de spécification - La phase de conception - La phase de réalisation	Comporte quatre phases principales : - L'élaboration - La construction - La transition - La production
Méthode linéaire qui se déroule en suivant des	Méthode itérative et incrémentale qui consiste

étapes claires et précises	en des cycles de développement successifs
Se concentre sur la structuration des données, des traitements et des flux d'informations	Se concentre sur la structuration du développement logiciel dans son ensemble
Utilise des modèles de données, de traitements et de flux d'informations	Utilise des modèles de processus, de cas d'utilisation, de classes, etc.

Figure 2: Comparatif MERISE – Processus

Une autre approche sera de comparer PU et MERISE par un système de pointage. En effet les points attribués seront faits selon les besoins du projet :

	MERISE	Processus Unifié
Principe de construction	7	7
Formalisme	5	8
Type d'approche	7	7
Cycle de vie	4	9
Niveau d'analyse	4	8
Découpage	6	6
Point de vue	4	8
Notion du temps	5	7
Type de système	4	8
Application	5	8
Apprentissage	8	5
Support logiciel	7	7
Niveau conception	7	4
Total	73	92

Figure 3: Comparatif MERISE – Processus Unifié (avec points)

Pondération :

- ✧ 1 à 3 correspond à une réponse faible de la solution
- ✧ 4 à 6 correspond à une réponse moyenne de la solution
- ✧ 7 à 10 correspond à une bonne réponse de la solution

En conclusion, avec 92 points, Processus Unifié est largement supérieur à MERISE pour l'exécution de notre projet. Nous retenons donc comme méthode d'analyse le Processus Unifié (PU).

Deuxième partie : Etude préalable

I- Etude de l'existant

1. Description du système actuel

1.1. Gestion des Ventes et des Clients

La gestion des ventes et des clients repose principalement sur des méthodes manuelles et informelles. La majorité des commerçants utilisent des cahiers, des feuilles volantes ou leur mémoire pour suivre les transactions et créances. Ces pratiques peuvent entraîner des pertes d'informations, ce qui complique le suivi des performances financières et la gestion des relations clients.

Lors d'une vente, le commerçant note généralement les détails du produit, la quantité et le prix, mais ces informations sont souvent dispersées et non centralisées, ce qui fausse l'analyse financière à la fin de la journée, de la semaine ou du mois. En cas de rupture de stock ou d'erreur de suivi, certaines transactions passent inaperçues, ce qui empêche une évaluation fiable de la rentabilité de l'activité.

Le suivi des clients est lui aussi informel. Les commerçants entretiennent des relations personnelles avec leurs acheteurs, mais ils ne disposent pas toujours d'un système structuré pour enregistrer les coordonnées ou analyser les habitudes d'achat. La fidélisation repose sur la proximité et la régularité des échanges, sans mécanismes précis de suivi.

Pour les ventes à crédit, les créances sont souvent notées dans un cahier, mais ces informations peuvent facilement être oubliées ou mal gérées. Cela entraîne des retards de paiement qui impactent la trésorerie. En l'absence d'un suivi précis, il devient difficile de récupérer les sommes dues, obligeant le commerçant à faire preuve de patience, au risque de perdre des clients ou d'accumuler des pertes.

1.2. Processus de gestion de stock

La gestion de stock se fait également de manière manuelle et souvent approximative. Les commerçants notent les entrées et sorties de produits sur des carnets ou des feuilles volantes. Quand

de nouveaux produits arrivent, les commerçants doivent consigner la date, la quantité, et la description des produits dans un registre, mais ce processus est souvent négligé ou mal suivi.

Lorsqu'un produit est vendu ou transféré vers un autre magasin ou entrepôt, la sortie de stock est notée manuellement, mais cela peut être omis si le commerçant est pressé ou oublie. Cela conduit à une mauvaise gestion de stock, avec des risques fréquents de rupture de stock ou d'accumulation de produits non vendus. Cette absence de visibilité sur les stocks complique également les réapprovisionnements et rend difficile l'anticipation des besoins futurs.

1.3. Processus de gestion de la trésorerie

La gestion de la trésorerie est également faite manuellement. Les commerçants se basent sur les chiffres de vente qu'ils ont notés dans leurs carnets pour calculer leurs bénéfices ou pertes à la fin de la journée. Toutefois, sans une centralisation automatique des ventes et des dépenses, les commerçants se trompent souvent dans leurs calculs, ce qui entraîne des incohérences dans leurs résultats financiers.

De plus, cette gestion approximative rend difficile la préparation de bilans financiers crédibles nécessaires à l'obtention de financements ou de partenariats avec des institutions bancaires ou des investisseurs.

2. Modélisation du système actuel

2.1. Processus de gestion de stock

2.1.1. Gestion des ventes et des clients

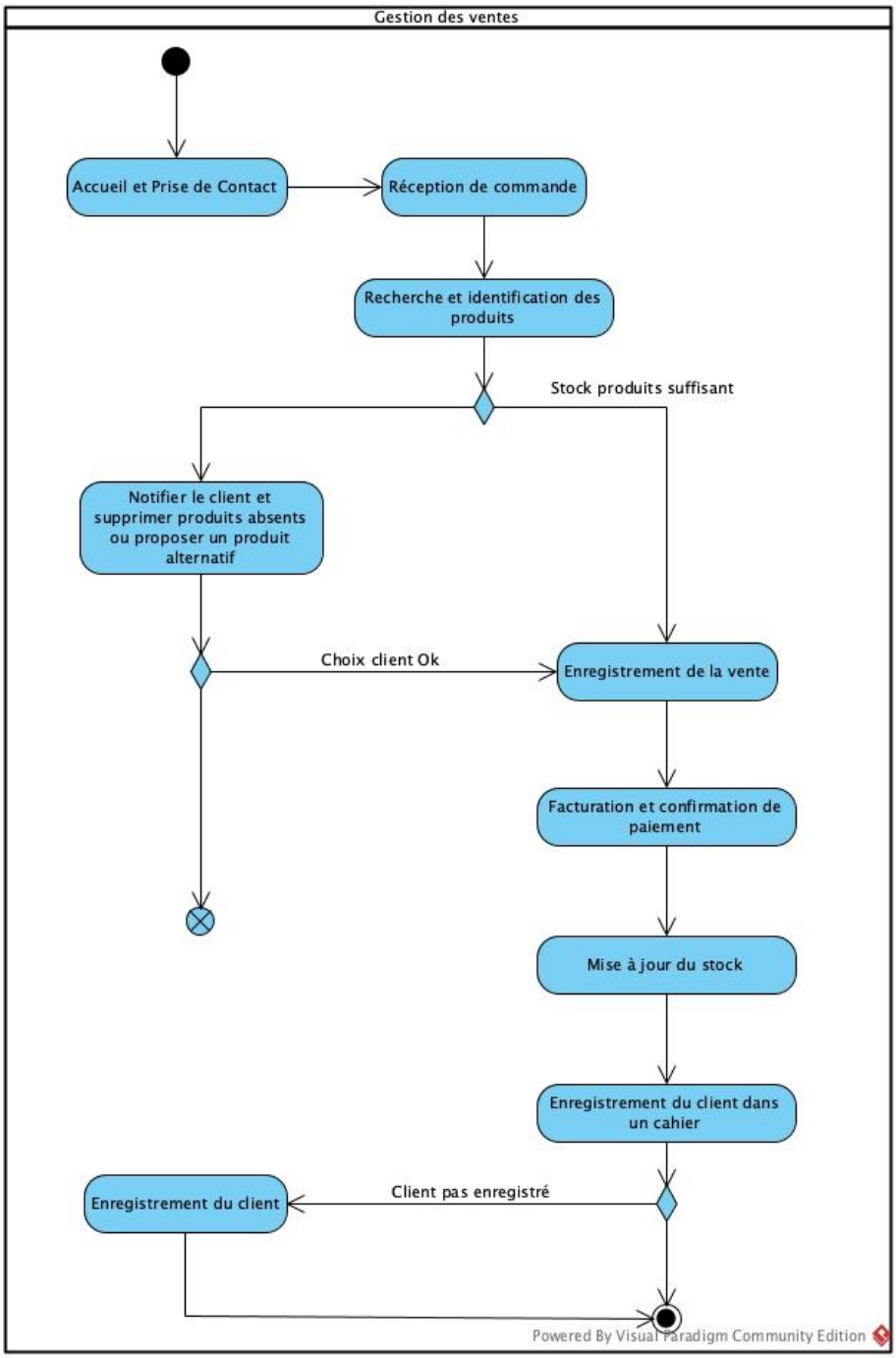


Figure 4: Modélisation du système actuel du processus de vente

2.1.2. Entrée de stock

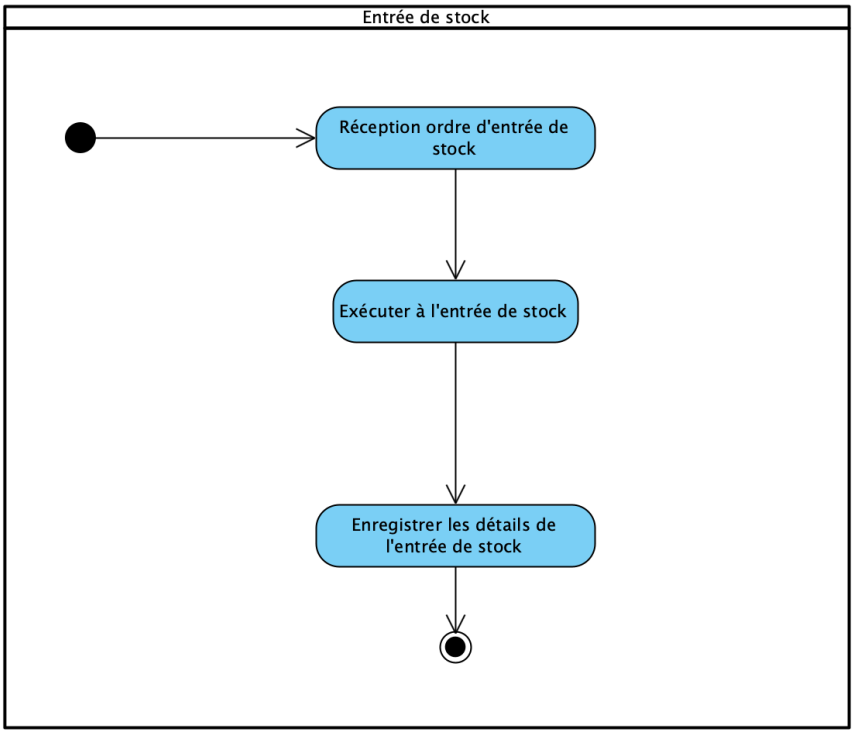


Figure 5: Modélisation du système actuel du processus d'entrée de stock

2.1.3. Sortie de stock

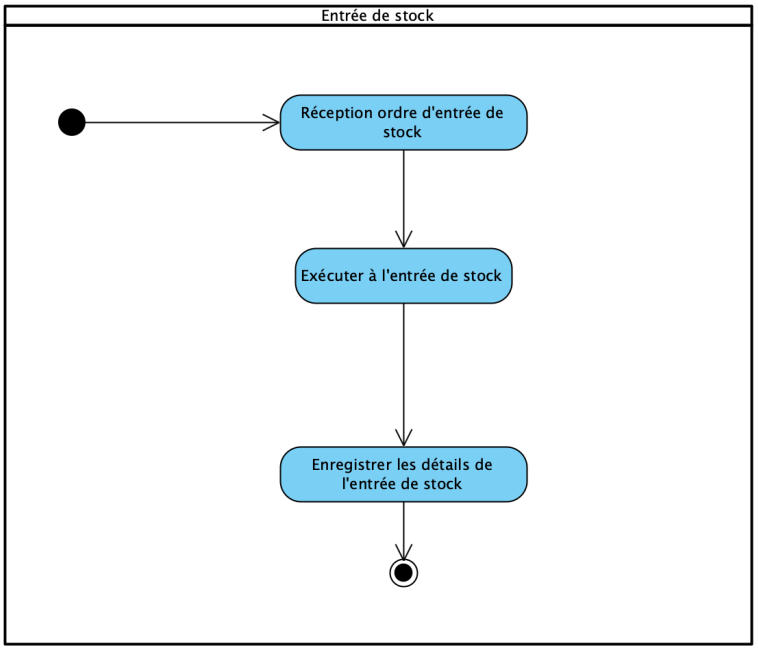


Figure 6: Modélisation du système actuel du processus de sortie de stock

2.1.4. Inventaire

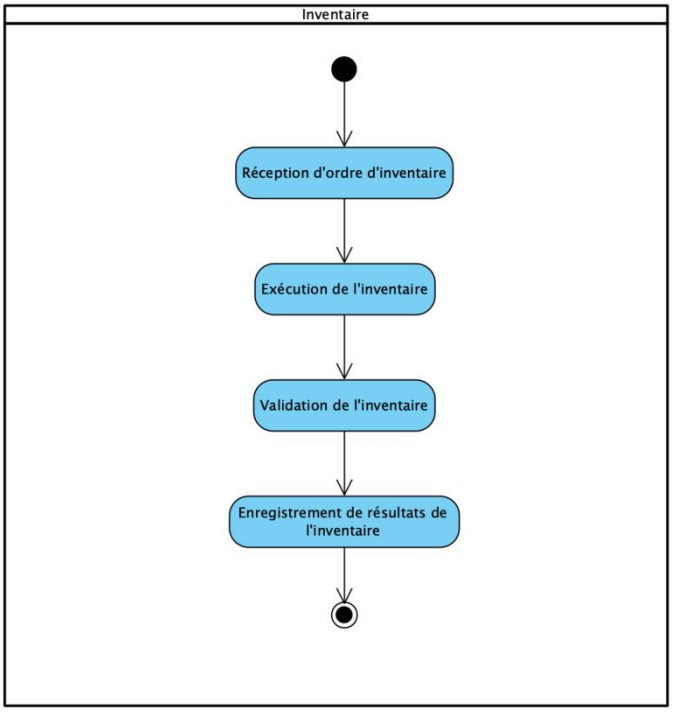


Figure 7: Modélisation du système actuel du processus des inventaires

2.1.5. Achat de marchandise ou de matière première

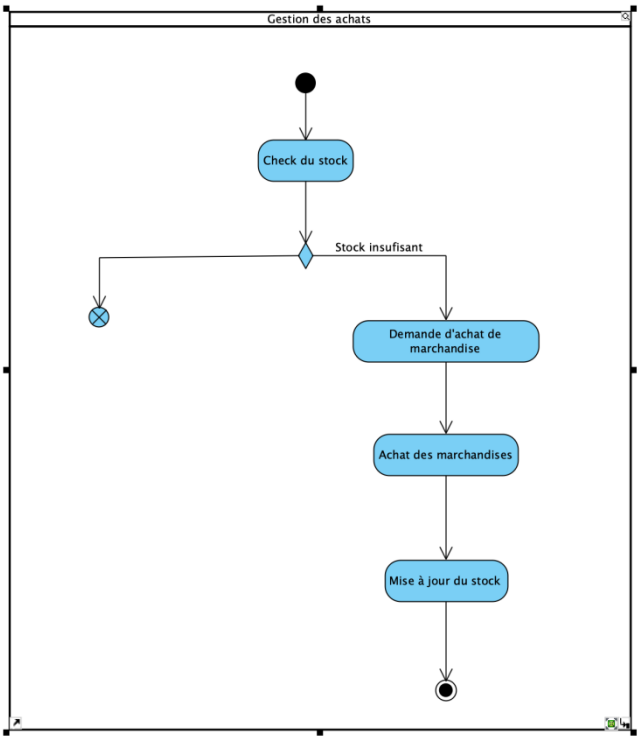


Figure 8: Modélisation du système actuel du processus des achats

2.2. Processus de gestion de la trésorerie

2.2.1. En cas de vente

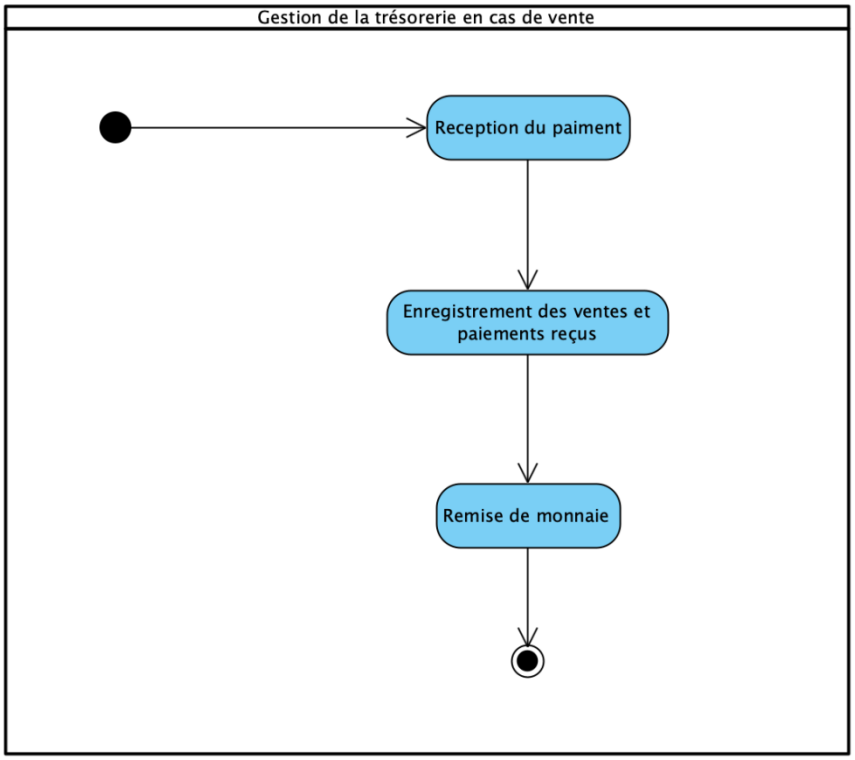


Figure 9: Modélisation du système actuel du processus de gestion trésorerie en cas de vente

2.2.2. En cas de dépense

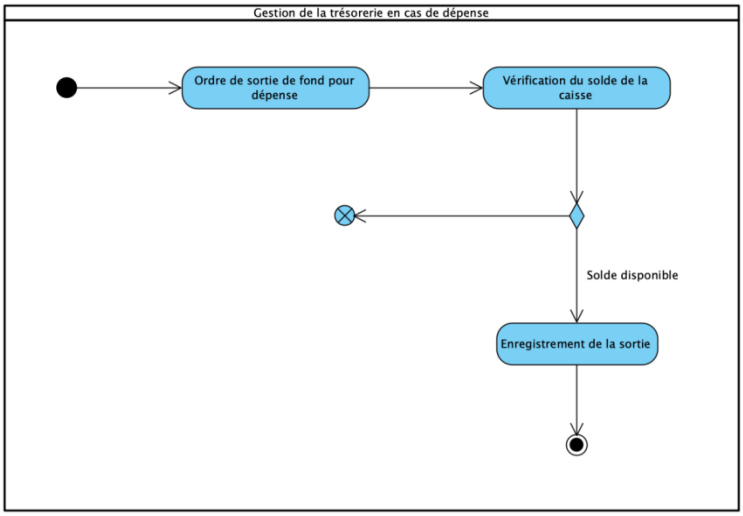


Figure 10: Modélisation du système actuel du processus de gestion de la trésorerie en cas de dépense

II- Analyse de l'existant

1. Objectifs et besoins des utilisateurs

1.1. Processus de gestion des Ventes et des clients

La gestion des ventes et des clients doit offrir aux utilisateurs la possibilité de suivre les commandes en cours, d'enregistrer les paiements et de consulter l'historique des achats. Il est également essentiel de faciliter la gestion des relations avec les clients à travers des suivis de créance client en cas de vente à crédit.

1.2. Processus de gestion de stock

La gestion de stock devrait permettre aux utilisateurs de connaître en temps réel le niveau de chaque article dans les stocks et de retracer les mouvements d'inventaire. L'objectif est également d'améliorer la validation des inventaires pour garantir une cohérence entre les données et la réalité sur le terrain.

1.3. Processus de gestion de la trésorerie

La gestion de la trésorerie doit permettre de suivre les encaissements et décaissements en temps réel, offrant une visibilité claire sur les flux financiers. Il est aussi crucial d'anticiper les besoins en liquidités et d'améliorer le processus de gestion des paiements et des dépenses pour assurer une stabilité financière.

2. Critiques de l'existant

Les problèmes rencontrés avec le système actuelle peuvent se résumer en ces points :

- ✧ Risque élevé de perte de données
- ✧ Travail fastidieux quant à la recherche de détails ou informations quelconques concernant une opération effectuée
- ✧ Risque d'incohérences de données

III- Propositions des solutions

1. Première solution

La première solution proposée est une application mobile intégrant l'ensemble des modules précédemment mentionnés.

1.1. Description

L'utilisateur pourra installer l'application, conçue avec une architecture client-serveur, sur son ordinateur. Elle lui offrira une gestion simplifiée des informations liées à l'activité de son entreprise et lui permettra de bénéficier des fonctionnalités d'automatisation qu'elle propose.

1.2. Avantages

- ✧ Réduction du temps d'accès aux données commerciales de l'entreprise.
- ✧ Accélération des recherches grâce à des critères de filtrage pertinents.
- ✧ Accès à des statistiques détaillées basées sur les activités commerciales quotidiennes.
- ✧ Amélioration de la collaboration entre les équipes via un accès partagé aux informations en temps réel.
- ✧ Sécurisation des données sensibles grâce à des niveaux d'accès et des sauvegardes régulières.
- ✧ Automatisation des rapports et alertes pour un suivi proactif des indicateurs clés.

1.3. Inconvénients

Parmi les inconvénients de cette solution, on peut noter :

- ✧ Une limitation de la mobilité de l'utilisateur.
- ✧ Des processus de mise à jour complexes et chronophages.
- ✧ Des coûts potentiels liés à la maintenance et au support technique. Une courbe d'apprentissage pouvant ralentir l'adoption par les utilisateurs.

2. Deuxième solution

La deuxième solution propose de développer une application mobile reposant également sur une architecture client-serveur et intégrant les fonctionnalités précédemment mentionnées.

2.1. Description

La deuxième solution repose sur le développement d'une application mobile utilisant une architecture client-serveur, permettant une synchronisation en temps réel entre les données locales et le serveur central. Cette approche garantit une meilleure accessibilité aux informations, offrant aux utilisateurs la possibilité de consulter et d'enregistrer des données à tout moment, même en déplacement. L'application proposera également des fonctionnalités essentielles telles que la gestion des ventes, des stocks et de la trésorerie, tout en intégrant des outils d'automatisation pour simplifier les tâches répétitives. De plus, grâce aux mises à jour centralisées, l'application pourra évoluer rapidement avec un minimum d'effort de la part des utilisateurs.

2.2. Avantages

Voici quelques avantages liés à la solution que tu proposes, qui consiste à développer une application mobile avec une architecture client-serveur :

Avantages de l'architecture client-serveur :

- ✧ Synchronisation en temps réel : Les utilisateurs ont accès à des données à jour, ce qui leur permet de prendre des décisions éclairées rapidement, même lorsqu'ils sont en déplacement.

- ✧ Automatisation des tâches répétitives : L'intégration d'outils d'automatisation réduit la charge de travail manuelle, ce qui augmente la productivité et permet aux utilisateurs de se concentrer sur des tâches à plus forte valeur ajoutée.

- ✧ Evolutivité et mises à jour centralisées : L'application peut facilement intégrer de nouvelles fonctionnalités et améliorations sans nécessiter d'efforts de la part des utilisateurs, garantissant ainsi une expérience toujours à jour et optimisée.

2.3. Inconvénients

- ✧ Dépendance à la connectivité Internet : L'accès aux fonctionnalités et aux données de l'application dépend d'une connexion Internet stable. En cas de connexion faible ou absente, les utilisateurs peuvent rencontrer des difficultés pour consulter ou enregistrer des informations.

- ✧ Coûts de développement et de maintenance : Le développement d'une application client-serveur peut être plus coûteux en raison de la complexité de l'architecture.

- ✧ Problèmes de sécurité : Bien que les données soient centralisées, cela peut également représenter une cible attrayante pour les cyberattaques. Il est essentiel de mettre en place des mesures de sécurité robustes pour protéger les informations sensibles, ce qui peut compliquer le développement et augmenter les coûts.

3. Choix d'une solution

Après examen de chacune des solutions proposées, nous avons choisi de privilégier la deuxième option : le développement d'une application mobile utilisant une architecture client-serveur.

Cette solution a été retenue en raison de sa capacité à offrir un accès en temps réel aux données, permettant aux utilisateurs de consulter et d'enregistrer des informations à tout moment.

De plus, les utilisateurs sont très familiers avec leurs smartphone et le système des applications mobiles, ce qui facilite l'adoption et l'utilisation de l'application pour la gestion des ventes, des stocks et de la trésorerie.

Troisième partie : Etude détaillée de la solution retenue

I- Modélisation objet

1. Diagramme de cas d'utilisation

1.1. Généralités

1.1.1. Définition

Les rôles du diagramme de cas d'utilisation sont de recueillir, d'analyser et d'organiser les besoins, ainsi que de recenser les grandes fonctionnalités d'un système. Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs. Ainsi, ces cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système. Ils sont donc une vision orientée utilisateur de ce besoin.

1.1.2. Composantes

Le diagramme de cas d'utilisation se compose de trois éléments principaux :

1.1.2.1. L'acteur

Il représente le rôle joué par une personne ou un élément en interaction avec le système. Un même acteur peut agir de différentes façons. Exemple : Le mécanicien auto entretient l'automobile du client, la conduit pour l'essayer, ou la répare.

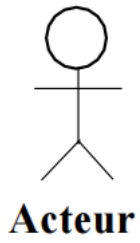


Figure 11: Représentation d'un acteur

1.1.2.2. Les cas d'utilisations ou "use cases"

Ils ont pour objet de faciliter la description sous forme graphique et textuelle de l'utilisation d'un système. Ils sont organisés en se servant de trois composants essentiels : les acteurs, les cas d'utilisation eux-mêmes, et le système.

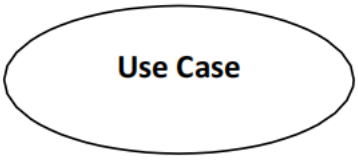


Figure 12: Représentation d'un cas d'utilisation

1.1.2.3. Le système

Il est construit progressivement à partir de l'ensemble des informations recueillies. La juxtaposition de l'ensemble des cas constitue le modèle des besoins appelé aussi spécification du système.

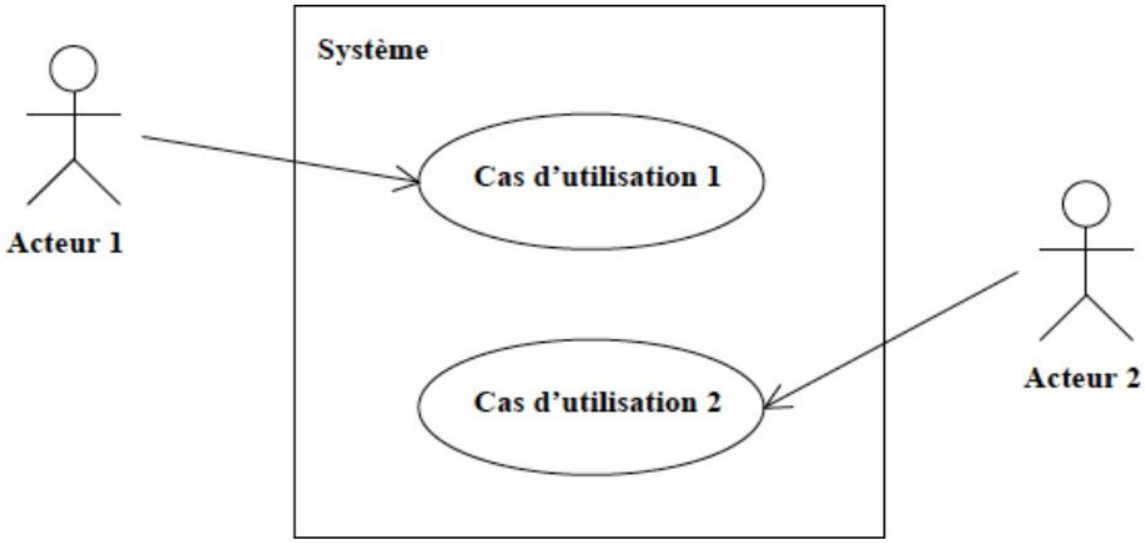


Figure 13: Représentation d'un diagramme de cas d'utilisation

1.1.3. Les acteurs de notre application

Les acteurs de notre application

- ✧ Le vendeur
- ✧ Le propriétaire (Owner)

1.2- Présentation du Diagramme de Cas d'Utilisation

1.2.1. Processus de gestion de stock

1.2.1.1. Capture du digramme

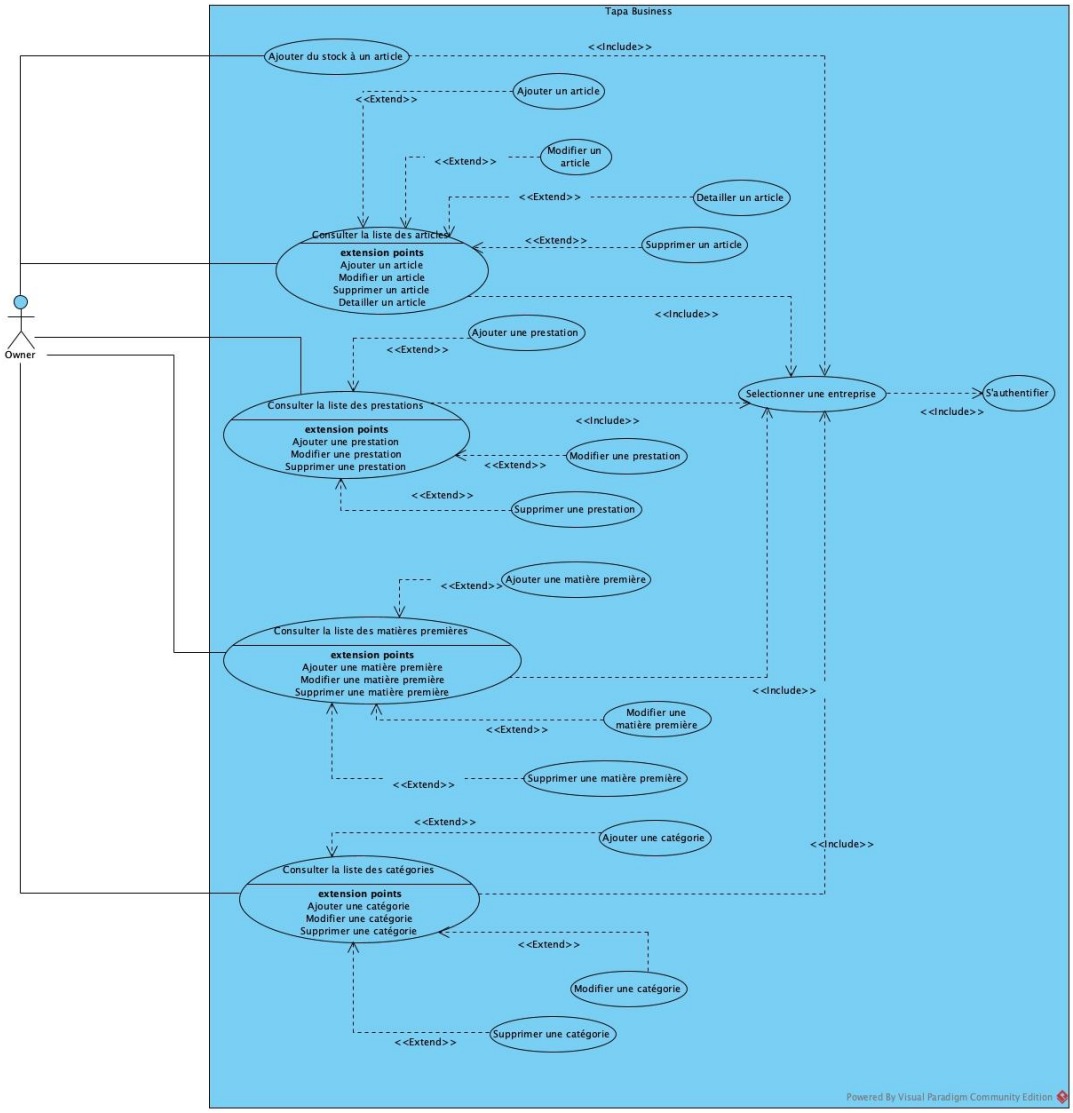


Figure 14: Diagramme des cas d'utilisation du processus de stock

1.2.1.2. Descriptions textuelles

✧ Cas d'utilisation : authentification

Titre	Authentification
Acteurs	Tous les utilisateurs de l'application
Description	S'authentifier

Scénario nominal	1. L'utilisateur entre son login et mot de passe dans les champs appropriés. 2. L'utilisateur valide ses entrées en appuyant sur se connecter. 3. L'application vérifie l'authenticité des informations de l'utilisateur. 4. L'application affiche la page d'accueil de l'application
énario alternatifs	Scénario alternatif 1 : « Ne pas remplir les champs » 1. L'utilisateur clique sur le bouton Se connecter sans rien saisir. 2. L'application indique le champ vide et qu'il faut le remplir. 3. Le cas d'utilisation reprend à l'action 1 du scénario nominal. Scénario alternatif 2 : « Fournir des accès erronés » 1. L'utilisateur clique sur le bouton Se connecter après avoir saisi des informations erronées. 2. L'application indique que les données d'identification sont erronées et le cas d'utilisation reprend à l'action 1 du scénario nominal
Préconditions	✧ Base de données créée

	✧ Base de données structurées ✧ Base de données contenant des utilisateurs
Postconditions	Accès à l'application après identification

✧ Cas d'utilisation : Sélectionner une entreprise

Titre	Sélectionner une entreprise
Acteurs	Tous les utilisateurs de l'application
Description	Sélectionner une entreprise dans l'application
Scénario nominal	1. L'utilisation se connecte à l'application 2. Une fois connecté, l'utilisateur sélectionne une entreprise dans l'appbar de la page d'accueil de l'application
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	L'utilisateur authentifié L'entreprise de l'utilisateur est déjà créée
Postconditions	L''entreprise sélectionnée est prise en compte dans tous les modules de l'application

✧ Cas d'utilisation : Consulter les articles

Titre	Consulter les articles
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Consulter la liste des articles
Scénario nominal	1. L'utilisateur clique sur « Articles et prestations » dans le groupe de menu « Gestion de stock » 2. L'application affiche une page qui affiche un bouton pour lister les articles ou les prestations

	3. L'utilisateur clique sur le bouton "Mes articles" 4. L'application affiche la liste des articles déjà créés.
Scénarios alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Consultation de la liste des articles existants

✧ Cas d'utilisation : Ajouter un article

Titre	Ajouter un article
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
scription	Ajouter un article
Scénario nominal	1. Depuis l'écran présentant la liste des articles, l'utilisateur clique sur le bouton « + » 2. L'application affiche l'écran permettant de renseigner les informations concernant le nouvel article. 3. L'utilisateur renseigne les informations concernant l'article et valide. 4. L'application enregistre les informations de l'article, retourne sur la page de précédente et actualise la liste des articles.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Ajout d'un nouvel article

✧ Cas d'utilisation : Consulter les détails un article

Titre	Détailler un article
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Afficher les détails d'un article
Scénario nominal	1. L'utilisateur clique sur l'élément désiré depuis la liste des articles 2. L'application affiche les détails de l'inventaire sélectionné.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Affichage des détails d'un article

✧ Cas d'utilisation : Ajouter du stock à un article

Titre	Détailler un article
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Afficher les détails d'un article
Scénario nominal	L'utilisateur clique sur l'élément désiré depuis la liste des articles 2. L'application affiche les détails de l'inventaire sélectionné.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié

	✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Affichage des détails d'un article

✧ Cas d'utilisation : Ajouter du stock à un article

Titre	Ajouter du stock à un article
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Ajouter du stock à un article
Scénario nominal	1. L'utilisateur clique le menu "achat de produits", puis sur le bouton "Articles" 2. L'application affiche la liste des achats d'articles 3. L'utilisateur clique sur le bouton "+" 4. L'application affiche la page d'ajout de stock. 5. L'utilisateur saisit les informations à renseigner puis valide l'entrée 6. L'application enregistre les informations saisies.
Scénario alternatifs	1. L'utilisateur clique sur l'article désiré depuis la liste des articles 2. L'application affiche le formulaire de modification des infos d'un article 3. L'utilisateur saisit la quantité voulue et valide

	4. L'application met à jour l'article et retourne sur la page précédente et actualise la liste des articles
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Affichage de la liste des articles

✧ Supprimer un article

Titre	Supprimer un article
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Supprimer un article
Scénario nominal	1. L'utilisateur clique sur l'article désiré depuis la liste des articles 2. L'application affiche le bouton de suppression en bas à droit de l'écran avec le symbole d'une poubelle 3. L'utilisateur clique sur le bouton "supprimer" et confirme la popup qui s'affiche 4. L'application supprime l'article et met à jour la liste des articles
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Suppression de l'article désiré

✧ Cas d'utilisation : Modifier un article

Titre	Modifier un article
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Modifier un article
Scénario nominal	✧ L'utilisateur clique sur l'article désiré depuis la liste des articles ✧ L'application affichera le formulaire de modification de l'article avec les infos de l'article déjà renseigné ✧ L'utilisateur modifie les infos à modifier et valide le formulaire ✧ L'application envoie les modifications au serveur, retourne sur la page de liste et met à jour la liste des articles.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	L'article édité est mis à jour

✧ Cas d'utilisation : Consulter les prestations

Titre	Consulter les prestations
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Consulter la liste des prestations

Scénario nominal	1. L'utilisateur clique sur « Articles et prestations » dans le groupe de menu « Gestion de stock » 2. L'application affiche une page qui affiche un bouton pour lister les articles ou les prestations 3. L'utilisateur clique sur le bouton "Mes prestations" 4. L'application affiche la liste des prestations déjà créées.
Scénarios alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Consultation de la liste des prestations existantes

✧ Cas d'utilisation : Ajouter une prestation

Titre	Ajouter une prestation
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Ajouter une prestation
Scénario nominal	1. Depuis l'écran présentant la liste des prestations, l'utilisateur clique sur le bouton « + » 2. L'application affiche l'écran permettant de renseigner les informations concernant la nouvelle prestation.

	3. L'utilisateur renseigne les informations concernant la prestation et valide. 4. L'application enregistre les informations de l'article, retourne sur la page de précédente et actualise la liste des prestations.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Ajout d'une nouvelle prestation

✧ Cas d'utilisation : Modifier une prestation

Titre	Modifier une prestation
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Modifier une prestation
Scénario nominal	1. L'utilisateur clique sur la prestation désirée depuis la liste des prestations 2. L'application affichera le formulaire de modification de la prestation avec les infos de la prestation déjà renseignés 3. L'utilisateur modifie les infos à modifier et valide le formulaire 4. L'application envoie les modifications au serveur, retourne sur la page de liste et met à jour la liste des prestations.

Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	La prestation éditée est mise à jour

✧ Supprimer une prestation

Titre	Supprimer une prestation
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Supprimer une prestation
Scénario nominal	1. L'utilisateur clique sur la prestation désirée depuis la liste des prestations 2. L'application affiche le bouton de suppression en bas à droit de l'écran avec le symbole d'une poubelle 3. L'utilisateur clique sur le bouton "supprimer" et confirme la popup qui s'affiche 4. L'application supprime la prestation et met à jour la liste de l'application
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Suppression de la prestation désirée

✧ Cas d'utilisation : Consulter les matières premières

Titre	Consulter les prestations
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Consulter la liste des matières premières
Scénario nominal	1. L'utilisateur clique sur « Matières premières » dans le groupe de menu « Gestion de stock » 2. L'application affiche la liste prestations déjà créées.
Scénarios alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Consultation de la liste des prestations existantes

✧ Cas d'utilisation : Ajouter une matière première

Titre	Ajouter une matière première
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Ajouter une matière première
Scénario nominal	1. Depuis l'écran présentant la liste des matières premières, l'utilisateur clique sur le bouton « + » 2. L'application affiche l'écran permettant de renseigner les

	informations concernant la nouvelle matière première. 3. L'utilisateur renseigne les informations concernant la matière première et valide. 4. L'application enregistre les informations de la matière première, retourne sur la page de précédente et actualise la liste des matières premières.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Ajout d'une nouvelle matière première

✧ Cas d'utilisation : Modifier une matière première

Titre	Modifier une matière première
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Modifier une prestation
Scénario nominal	1. L'utilisateur clique sur la matière première désirée depuis la liste des matières premières 2. L'application affichera le formulaire de modification des matières premières avec les infos de la matière première sélectionnée déjà renseignées

	3. L'utilisateur modifie les infos à modifier et valide le formulaire 4. L'application envoie les modifications au serveur, retourne sur la page de liste et met à jour la liste des matières premières.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	La matière première éditée est mise à jour

✧ Supprimer une matière première

Titre	Supprimer une matière première
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Supprimer une matière première
Scénario nominal	1. L'utilisateur clique sur la matière première désirée depuis la liste des matières premières 1. L'application affiche le bouton de suppression en bas à droit de l'écran avec le symbole d'une poubelle 2. L'utilisateur clique sur le bouton "supprimer" et confirme la popup qui s'affiche

	3. L'application supprime la matière première et met à jour la liste de l'application
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Suppression de la matière première désirée

✧ Cas d'utilisation : Consulter les catégories

Titre	Consulter les catégories
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Consulter la liste des catégories
Scénario nominal	1. L'utilisateur clique sur « Catégorie d'articles » dans le groupe de menu « Gestion de stock » 2. L'application affiche la liste catégories déjà créées.
Scénarios alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Consultation de la liste des catégories existantes

✧ Cas d'utilisation : Ajouter une catégorie

Titre	Ajouter une catégorie
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Ajouter une catégorie
Scénario nominal	1. Depuis l'écran présentant la liste des catégories, l'utilisateur clique sur le bouton « + » 2. L'application affiche l'écran permettant de renseigner les informations concernant la nouvelle catégorie. 3. L'utilisateur renseigne les informations concernant la catégorie et valide. 4. L'application enregistre les informations de la catégorie, retourne sur la page de précédente et actualise la liste des catégories.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Ajout d'une nouvelle catégorie

✧ Cas d'utilisation : Modifier une catégorie

Titre	Modifier une catégorie
Acteurs	Tous les utilisateurs ayant accès au module « Stock »

Description	Modifier une catégorie
Scénario nominal	1. L'utilisateur clique sur la catégorie désirée depuis la liste des catégories 2. L'application affichera le formulaire de modification de la prestation avec les infos de la catégorie déjà renseignés 3. L'utilisateur modifie les infos à modifier et valide le formulaire 4. L'application envoie les modifications au serveur, retourne sur la page de liste et met à jour la liste des catégories.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	La catégorie éditée est mise à jour

✧ Supprimer une catégorie

Titre	Supprimer une catégorie
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Supprimer une catégorie
Scénario nominal	1. L'utilisateur clique sur la catégorie désirée depuis la liste des catégories

	2. L'application affiche le bouton de suppression en bas à droit de l'écran avec le symbole d'une poubelle 3. L'utilisateur clique sur le bouton "supprimer" et confirme la popup qui s'affiche 4. L'application supprime catégorie et met à jour la liste de l'application
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Suppression de la catégorie désirée

1.2.2. Processus de gestion de la trésorerie

1.2.2.1. Capture du diagramme

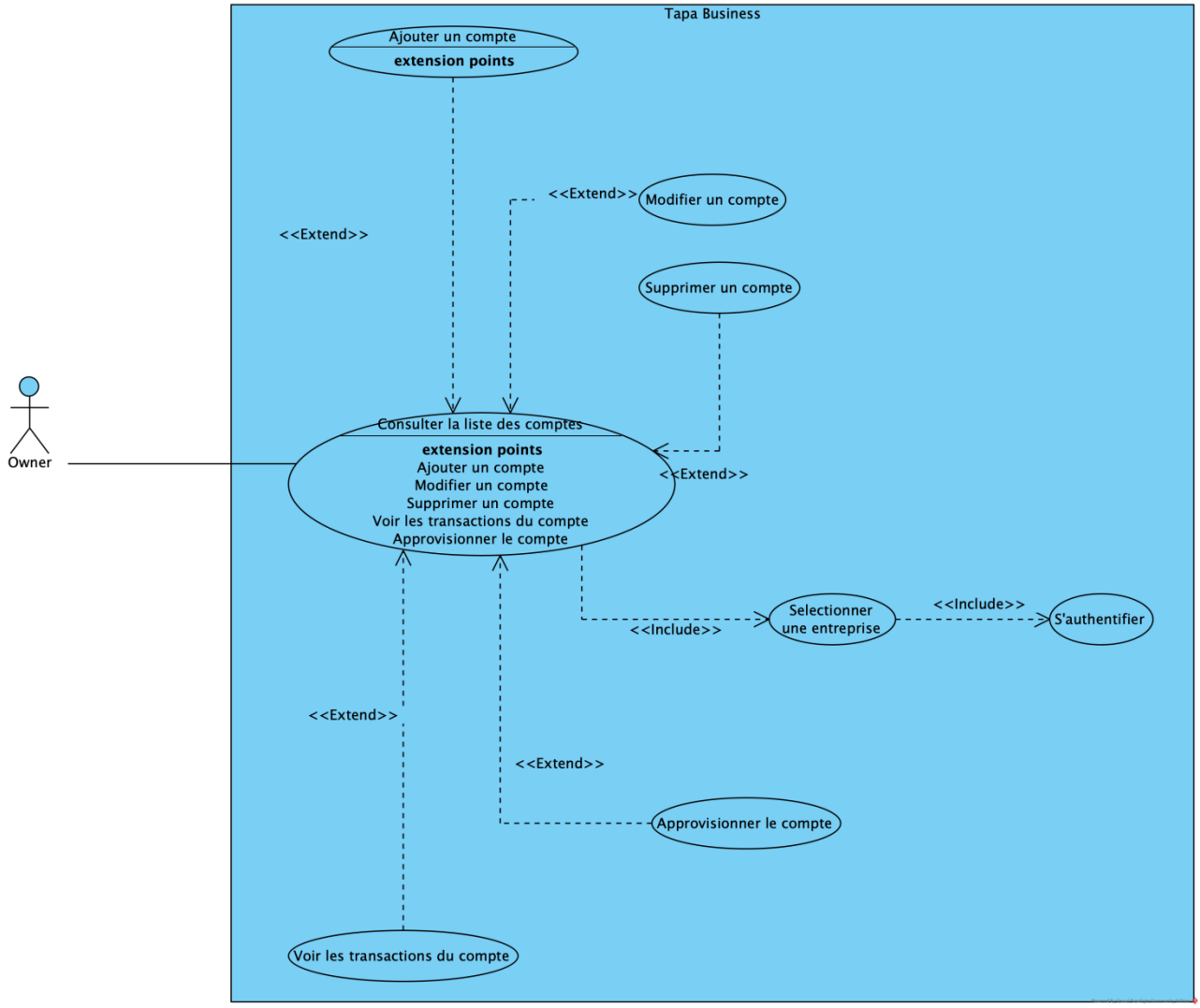


Figure 15: Diagramme des cas d'utilisation du processus de gestion de la trésorerie

1.2.2.2. Descriptions textuelles

✧ Cas d'utilisation : Consulter les comptes

Titre	Consulter les comptes
Acteurs	Tous les utilisateurs ayant accès au module « Gestion de trésorerie »
Description	Consulter la liste des comptes
Scénario nominal	1. L'utilisateur clique sur « Trésoreries » dans le menu

	2. L'application affiche la liste comptes déjà créés.
Scénarios alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Gestion de trésorerie »
Postconditions	Consultation de la liste des comptes existants

✧ Cas d'utilisation : Ajouter un compte

Titre	Ajouter un compte
Acteurs	Tous les utilisateurs ayant accès au module « Gestion de trésorerie »
Description	Ajouter un compte
Scénario nominal	1. Depuis l'écran présentant la liste des comptes, l'utilisateur clique sur le bouton « + » 2. L'application affiche l'écran permettant de renseigner les informations concernant le nouveau compte. 3. L'utilisateur renseigne les informations concernant la catégorie et valide. 4. L'application enregistre les informations de le compte, retourne sur la page de précédente et actualise la liste des comptes.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif

Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Gestion de trésorerie »
Postconditions	Ajout d'un nouveau compte

✧ Cas d'utilisation : Modifier un compte

Titre	Modifier un compte
Acteurs	Tous les utilisateurs ayant accès au module « Gestion de trésorerie »
Description	Modifier un compte
Scénario nominal	1. L'utilisateur clique sur le compte désiré depuis la liste des comptes 2. L'application affichera le formulaire de modification de la prestation avec les infos du compte déjà renseignés 3. L'utilisateur modifie les infos à modifier et valide le formulaire 4. L'application envoie les modifications au serveur, retourne sur la page de liste et met à jour la liste des comptes.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Gestion de trésorerie »
Postconditions	Le compte édité est mis à jour

✧ Supprimer un compte

Titre	Supprimer un compte
Acteurs	Tous les utilisateurs ayant accès au module « Gestion de trésorerie »
Description	Supprimer un compte
Scénario nominal	1. L'utilisateur clique sur le compte désiré depuis la liste des comptes 2. L'application affiche le bouton de suppression en bas à droit de l'écran avec le symbole d'une poubelle 3. L'utilisateur clique sur le bouton "supprimer" et confirme la popup qui s'affiche 4. L'application supprime le compte et met à jour la liste de l'application
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Gestion de trésorerie »
Postconditions	Suppression de la catégorie désirée

✧ Approvisionner un compte

Titre	Approvisionner un compte
Acteurs	Tous les utilisateurs ayant accès au module « Gestion de trésorerie »
Description	Approvisionner un compte

Scénario nominal	1. L'utilisateur clique sur le bouton "+" de la ligne du compte désiré depuis la liste des comptes 2. L'application affichera le formulaire d'approvisonnement 3. L'utilisateur saisit le montant à ajouter et un commentaire (facultatif) et valide 4. L'application crédite le compte sélectionné, retourne sur la liste des comptes et l'actualise
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Gestion de trésorerie »
Postconditions	Le solde du compte sélectionné est crédité

✧ Voir les transactions d'un compte

Titre	Approvisionner un compte
Acteurs	Tous les utilisateurs ayant accès au module « Gestion de trésorerie »
Description	Approvisionner un compte
Scénario nominal	1. L'utilisateur clique sur le bouton avec l'icone de liste de la ligne du compte désiré depuis la liste des comptes

	2. L'application affichera les opérations qui ont été effectués sur ce compte depuis sa création
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Gestion de trésorerie »
Postconditions	Affichage des opérations effectuées sur le compte sélectionné

1.2.3. Processus de gestion des Ventes et des clients

1.2.3.1. Capture du diagramme

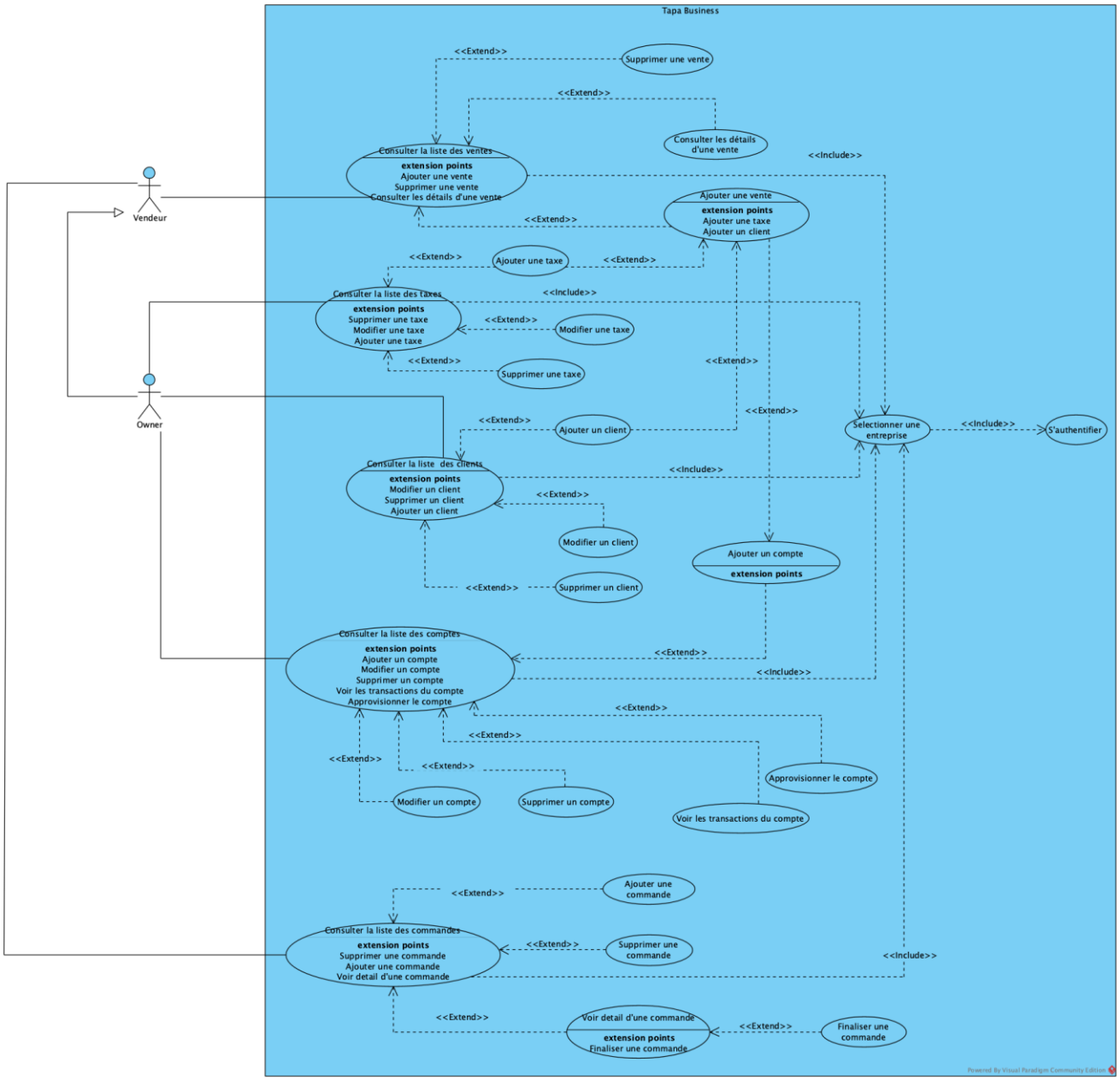


Figure 16: Diagramme des cas d'utilisation du processus de gestion des ventes et des clients

1.2.3.2. Descriptions textuelles

✧ Cas d'utilisation : Ajouter une prestation

Titre	Ajouter une prestation
Acteurs	Tous les utilisateurs ayant accès au module « Stock »

Description	Ajouter une prestation
Scénario nominal	1. Depuis l'écran la liste des prestations, l'utilisateur clique sur le bouton « + » 2. L'application affiche l'écran permettant de renseigner les informations concernant la nouvelle prestation. 3. L'utilisateur renseigne les informations concernant la prestation et valide. 4. L'application enregistre les informations de la prestation, retourne sur la page de précédente et actualise la liste des prestations.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif.
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Ajout d'une nouvelle prestation

✧ Cas d'utilisation : Consulter les catégories

Titre	Consulter les catégories
Acteurs	Tous les utilisateurs ayant accès au module « Stock »
Description	Consulter la liste des catégories de produit
Scénario nominal	✧ L'utilisateur clique sur « Catégories » dans le groupe d'onglets « Produits »

	✧ L'application affiche la liste des catégories déjà créées.
Scénario alternatifs	Ce cas d'utilisation n'a pas de scénario alternatif
Préconditions	✧ Utilisateur authentifié ✧ Utilisateur possédant la permission d'accéder au module « Stock »
Postconditions	Affichage de la liste des catégories

2. Diagramme de classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet : il est le seul obligatoire lors d'une telle modélisation. Pendant que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il est important de noter qu'un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation. Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation.

2.1. Composantes

Le diagramme de classe se compose des éléments suivants :

2.1.1. La classe

C'est la représentation d'un objet ou un ensemble d'objets possédant une structure et un comportement communs. Une classe se représente avec UML sous forme d'un rectangle divisé en trois sections.

- ✧ La première section contient le nom donné à la classe (non souligné).
- ✧ La deuxième section contient les attributs de la classe (nom, type, valeur par défaut éventuellement)
- ✧ La troisième section contient les opérations de la classe.

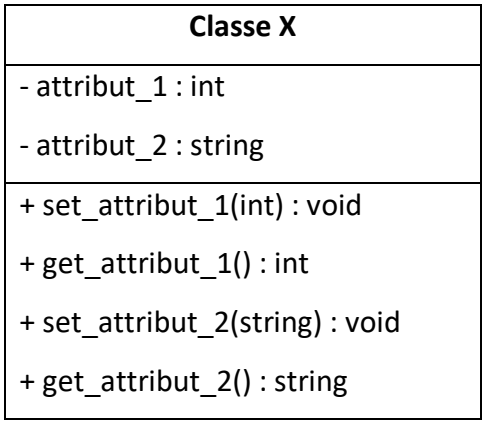


Figure 17: Composition d'une classe

2.1.2. L'attribut

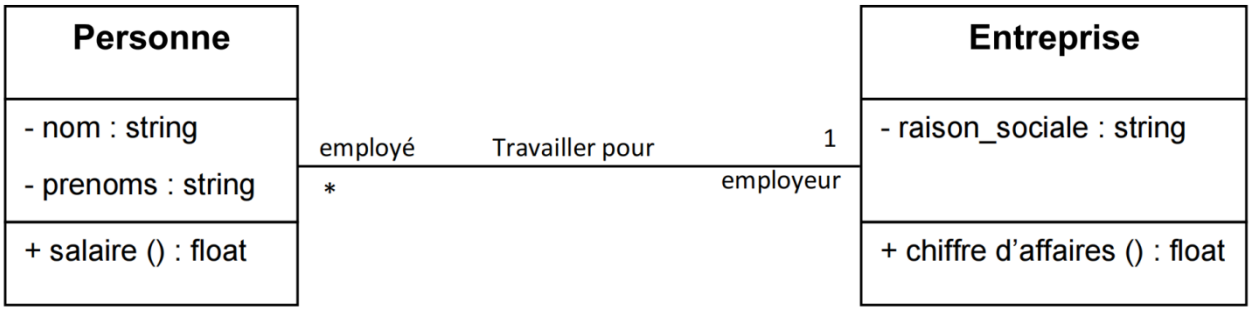
C'est une propriété élémentaire qui décrit l'état de l'objet de la classe. Il est représenté par un nom d'attribut et un type de données.

2.1.3. La méthode

C'est une fonctionnalité de la classe qui peut être appelée pour effectuer une action. Elle est représentée par le nom, les paramètres et le type de retour.

2.1.4. L'association

Il s'agit d'une relation entre deux classes (association binaire) ou plus (association n-aire) qui décrit les connexions structurelles entre leurs instances.



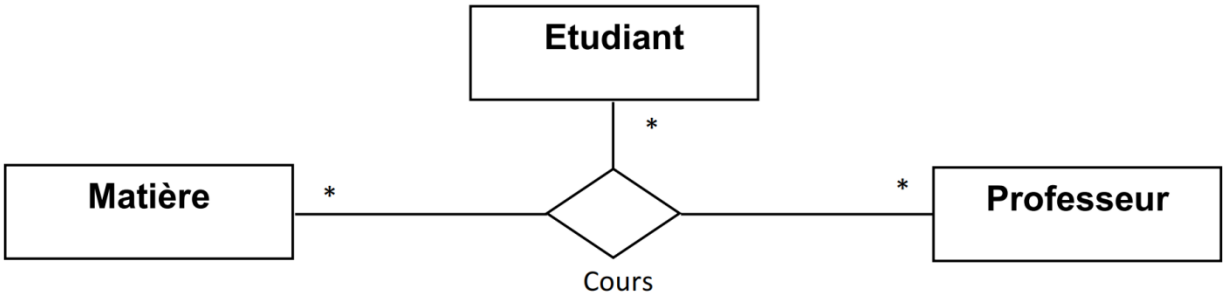


Figure 18: Composition d'une association

2.1.5. Le rôle

C'est le nom donné à chaque extrémité d'une association ; par extension, c'est la manière dont les instances d'une classe voient les instances d'une autre classe au travers d'une association.

2.1.6. La multiplicité

C'est le nombre d'objet (min… max) qui peuvent participer à une relation avec un autre objet dans le cadre d'une association.

2.1.7. L'agrégation

C'est une relation entre deux classes où une classe contient une ou plusieurs instances d'une autre classe. Elle est représentée par un diamant creux à l'extrémité de l'association qui pointe vers la classe contenante.

2.1.8. La composition

C'est une relation entre deux classes où une classe contient une ou plusieurs instances d'une autre classe, mais où ces instances sont exclusives à cette classe. Elle est représentée par un diamant plein à l'extrémité de l'association qui pointe vers la classe contenante.

2.1.9. La généralisation - spécialisation

La généralisation est une relation entre une classe parent et une ou plusieurs classes enfants. La classe parent contient les caractéristiques communes à toutes les classes enfants, tandis que les classes enfants contiennent les caractéristiques spécifiques à chaque classe. La classe parent est appelée classe générale, tandis que les classes enfants sont appelées classes spécialisées. La spécialisation est le processus de création de classes spécialisées à partir d'une classe générale.

2.2. Présentation du diagramme de classe

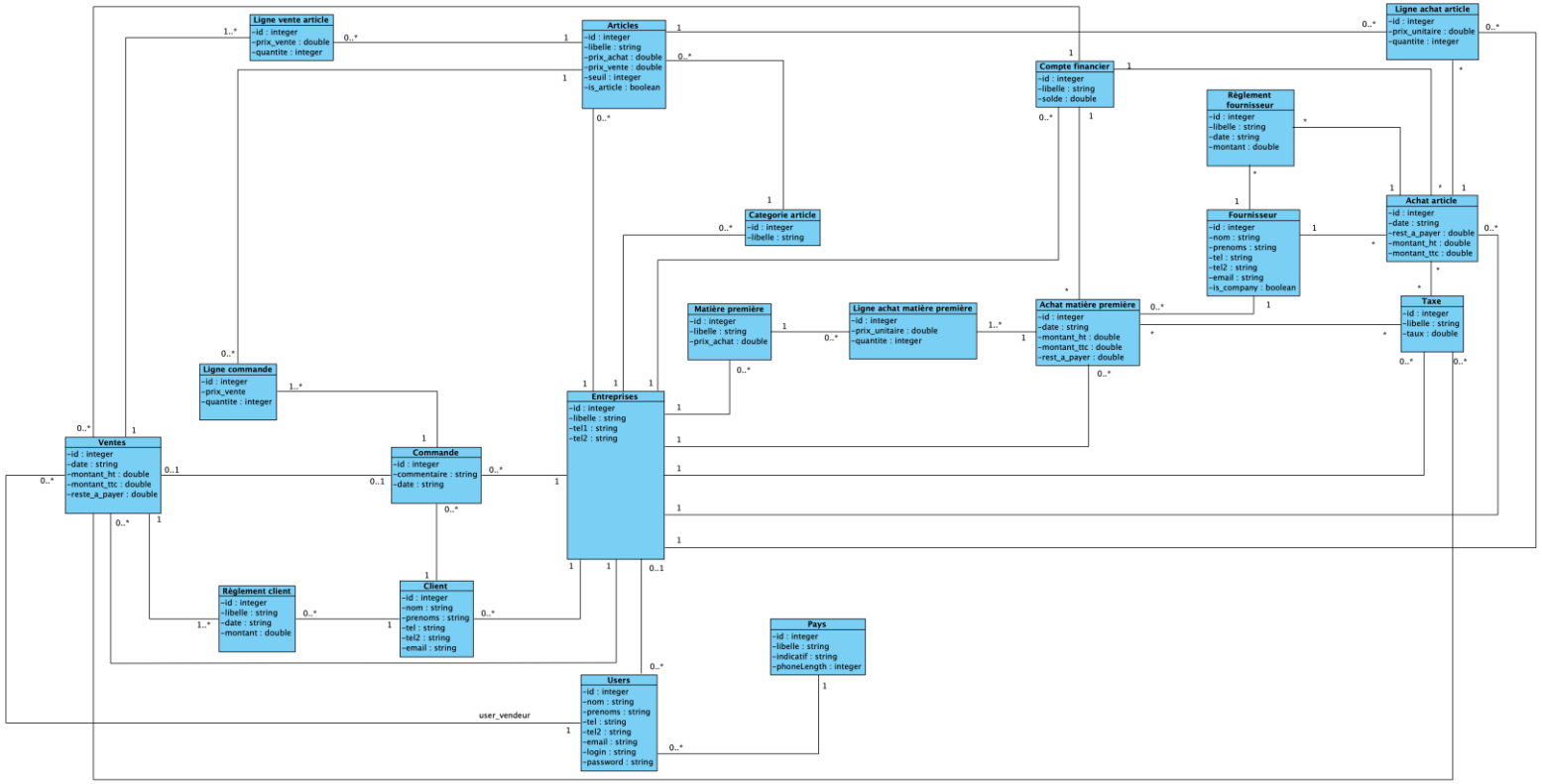


Figure 19: Diagramme de classe

II- Modélisation au niveau base de données

1. Présentation

Durant la phase d'analyse, nous avons suivi une démarche selon le modèle orienté-objet avec le processus unifié. Puisque nous allons implémenter notre base de données avec MYSQL, nous avons besoin de traduire certains objets pour qu'ils soient compatibles à un environnement relationnel.

2. Liste des tables de la base de données

2.1. Gestion des Ventes et des Clients

2.1.1. Table "Ventes"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
date	Date	20	
entreprise_id	Entier	20	Clé étrangère
compte_financier_id	Entier	20	Clé étrangère
montant_h	Double	20	
reste_a_payer	Double	20	
client_id	Entier	20	Clé étrangère
montant_ttc	Double	20	Clé étrangère
user_vendeur_id	Entier	20	Clé étrangère

2.1.2. Table "Clients"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
nom	Caractères	255	
prenoms	Caractères	255	
tel	Caractères	100	
tel2	Caractères	100	
email	Caractères	255	
entreprise_id	Entier	20	Clé étrangère

2.1.3. Table "Ligne vente article"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
prix_vente	Double	20	
quantite	Entier	11	
article_id	Entier	20	Clé étrangère
vente_id	Entier	20	Clé étrangère

2.1.4. Table "Règlement client"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
libelle	Caractères	255	
date	Date	20	
montant	Double	20	
client_id	Entier	20	Clé étrangère
vente_id	Entier	20	Clé étrangère

2.1.5. Table "Ligne commande"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
prix_vente	Double	20	
quantite	Entier	11	
article_id	Entier	20	Clé étrangère
commande_id	Entier	20	Clé étrangère

2.1.6. Table "Commande"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
commentaire	Text		
date	Date	20	
entreprise_id	Entier	20	Clé étrangère
vente_id	Entier	20	Clé étrangère
client_id	Entier	20	Clé étrangère

2.1.7. Table "Taxe vente"

Attributs	Type	Longueur	Remarques

id	Entier	20	Clé primaire
taxe_id	Entier	20	Clé étrangère
vente_id	Entier	20	Clé étrangère

2.1.8. Table "Taxe"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
libelle	Caractères		
taux	Double	10	
entreprise_id	Entier	20	

2.2. Processus de gestion de stock

2.2.1. Table "Articles"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
libelle	Caractères	255	
prix_achat	Double	20	
prix_vente	Double	20	
categorie_article_id	Entier	20	Clé étrangère
entreprise_id	Entier	20	Clé étrangère
quantite	Double	11	
seuil	Entier	11	
is_article	Entier	1	

2.2.2. Table "Achat articles"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
date	Date	20	
entreprise_id	Entier	20	Clé étrangère
compte_financier_id	Entier	20	Clé étrangère
montant_ht	Double	20	
montant_ttc	Double	20	
fournisseur_id	Entier	20	Clé étrangère
rest_a_payer	Double	20	

2.2.3. Table "Matière première"

Attributs	Type	Longueur	Remarques

id	Entier	20	Clé primaire
libelle	Caractères	255	
prix_achat	Double	20	
entreprise_id	Entier	20	Clé étrangère

2.2.4. Table "Achat matière première"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
date	Date	20	
entreprise_id	Entier	20	Clé étrangère
compte_financier_id	Entier	20	Clé étrangère
montant_ht	Double	20	
montant_ttc	Double	20	
rest_a_payer	Double	20	
fournisseur_id	Entier	20	Clé étrangère

2.2.5. Table "Ligne achat article"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
prix_unitaire	Double	20	
quantite	Entier	20	
entreprise_id	Entier	20	Clé étrangère
fournisseur_id	Entier	20	Clé étrangère
achat_id	Entier	20	Clé étrangère
article_id	Entier	20	Clé étrangère

2.2.6. Table "Ligne achat matière première"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
prix_unitaire	Double	20	
quantite	Entier	20	
achat_matière_première_id	Entier	20	Clé étrangère
matière_première_id	Entier	20	Clé étrangère

2.2.7. Table "Catégorie article"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
libelle	Caractères	255	
entreprise_id	Entier	20	Clé étrangère

2.2.8. Table "Taxe achat article"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
taxe_id	Entier	20	Clé étrangère
article_id	Entier	20	Clé étrangère

2.2.9. Table "Taxe achat matière première"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
taxe_id	Entier	20	Clé étrangère
achat_matiere_premiere_id	Entier	20	Clé étrangère

2.3. Processus de gestion de la trésorerie

2.3.1. Table "Compte financier"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
libelle	Caractères	255	
solde	Double	20	
entreprise_id	Entier	20	Clé étrangère

2.3.2. Table "Règlement client"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
libelle	Caractères	255	
date	Date	20	
montant	Double	20	
client_id	Entier	20	Clé étrangère
vente_id	Entier	20	Clé étrangère

2.3.3. Table "Règlement fournisseur"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
libelle	Caractères	255	
date	Date	20	

montant	Double	20	
fournisseur_id	Entier	20	Clé étrangère
achat_id	Entier	20	Clé étrangère

2.3.4. Table "Fournisseur"

Attributs	Type	Longueur	Remarques
id	Entier	20	Clé primaire
nom	Caractères	255	
prenoms	Caractères	255	
tel	Caractères	255	
tel2	Caractères	255	
email	Caractères	255	
entreprise_id	Entier	20	Clé étrangère
is_company	Entier	1	

Quatrième partie : Réalisation et déploiement

I- Réalisation

1. Script de création de la base de données

La création de la base de données n'a pas été réalisée directement par l'exécution d'un script. En effet, le framework utilisé propose une approche légèrement différente, bien que, en arrière-plan, un script soit effectivement exécuté. Le processus se déroule comme suit :

1.1. Création de fichiers de migration

Chaque table de la base de données est définie dans un fichier de migration, où sont spécifiés ses différents attributs ou propriétés, leurs types respectifs, ainsi que les relations avec les tables associées.

1.2. Exécution des fichiers de migration

L'exécution des fichiers de migration, qui génèrent automatiquement le script de données, se fait à l'aide d'une commande exécutée depuis le terminal après avoir configuré les fichiers de migration selon les besoins.

Cette méthode offre non seulement un contrôle préalable avant l'exécution du script sur la base de données, mais également une grande flexibilité, permettant de restructurer la base de données à tout moment.

1.3. Exemple de fichier de migration

```php
<?php

use ...

no usages  amangoua01
class CreateEntrepriseTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    amangoua01
    public function up()
    {
        Schema::create( table: 'entreprise', function (Blueprint $table) {
            $table->id();
            $table->integer( column: "active")->default( value: 0)->nullable();
            $table->integer( column: "version")->default( value: 0)->nullable();
            $table->string( column: "libelle",  length: 255)->nullable();
            $table->bigInteger( column: "logo_id")->nullable();
            $table->bigInteger( column: "owner_id")->nullable();
            $table->softDeletes();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    amangoua01
    public function down()
    {
        Schema::dropIfExists( table: 'entreprise');
    }
}
```

Figure 20: Capture d'exemple de fichier de migration

2. Présentation des outils utilisés

Avant de détailler les outils utilisés pour le développement de l'application, il est important de présenter l'environnement matériel requis pour son utilisation.

2.1. Environnement matériel

- ✧ Un appareil connecté par agent (Ordinateur, Tablette, Smartphone)
- ✧ Une connexion internet

2.2. Les outils utilisés

2.2.1. Editeur de code : Visual Studio Code & PhpStom

Il convient de mentionner qu'il existe de nombreux éditeurs de code disponibles, tels que Sublime Text et Atom, mais notre choix s'est porté sur Visual Studio Code et PhpStorm. Ces deux éditeurs ont été utilisés pour leurs fonctionnalités complémentaires : Visual Studio Code pour sa gratuité, sa légèreté, et ses extensions adaptées, et PhpStorm pour sa puissance et ses outils avancés, parfaitement adaptés aux exigences de notre projet.

2.2.2. Outils coté serveur

2.2.2.1. Framework Serveur : Laravel

De nombreux frameworks côté serveur, tels que Symfony, Spring Boot, ou Django, sont disponibles. Nous avons choisi d'utiliser Laravel en raison de sa simplicité, de ses nombreuses fonctionnalités, de sa large communauté de support, et de son adéquation avec les besoins spécifiques de notre projet. De plus, la familiarité préalable de l'équipe avec Laravel a également influencé ce choix.

Laravel est un framework web open-source, écrit en PHP, qui suit le principe Modèle-Vue-Contrôleur (MVC) et repose entièrement sur une approche orientée objet.

2.2.2.2. Langage de programmation : PHP

En accord avec le choix du framework Laravel, qui est un framework basé sur PHP, nous avons utilisé PHP comme langage de programmation côté serveur.

PHP est un langage de programmation open-source, largement employé pour générer des pages web dynamiques via un serveur HTTP, tout en pouvant également être utilisé comme un langage interprété pour des applications locales.

2.2.3. Outil coté client : Framework Flutter

Parmi les nombreuses technologies disponibles pour le développement d'applications mobiles, telles que React Native, Xamarin, ou encore Ionic, nous avons choisi d'utiliser Flutter comme framework côté client.

Flutter est un framework open-source développé par Google, permettant de créer des applications mobiles, web, et desktop à partir d'une seule base de code. Il utilise le langage de programmation Dart et repose sur une architecture réactive.

Le choix de Flutter a été motivé par plusieurs facteurs :

- ✧ Son interface utilisateur riche et personnalisable grâce à une vaste collection de widgets intégrés, facilitant la création d'interfaces modernes et réactives.
- ✧ Sa performance élevée, car Flutter compile directement en code natif, éliminant les limitations de ponts entre le framework et les composants natifs.
- ✧ Son support multiplateforme, permettant de développer pour Android, iOS, et d'autres plateformes avec un seul code source.
- ✧ Sa documentation exhaustive et sa large communauté qui offrent un support constant et des ressources abondantes pour accélérer le développement.

En résumé, Flutter s'est imposé comme l'outil idéal pour le développement de notre application en raison de sa flexibilité, de sa rapidité, et de sa capacité à répondre aux exigences de notre projet.

3. Quelques captures d'écran

Figure 21: Page d'accueil

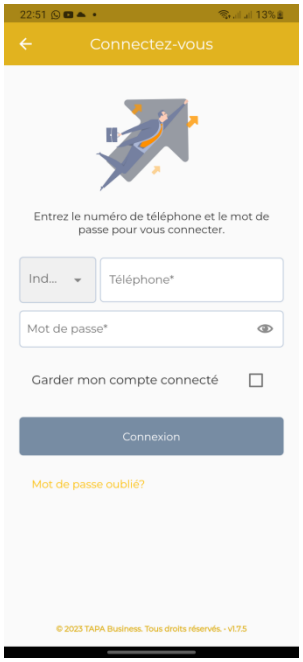


Figure 22: Page de login

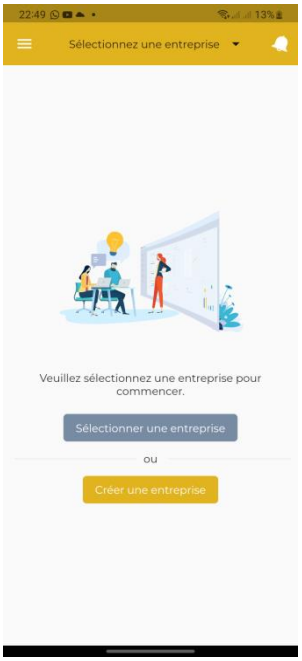


Figure 23: Page home de l'application

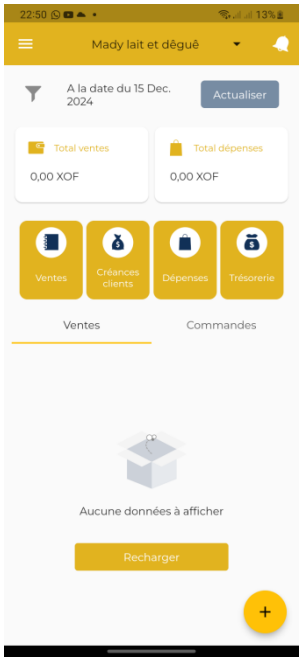


Figure 24: Dashboard

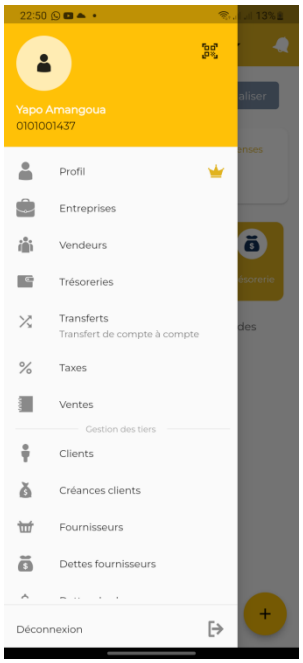


Figure 25: Menu de l'application

Figure 26: Liste des articles

Figure 27: Liste des ventes

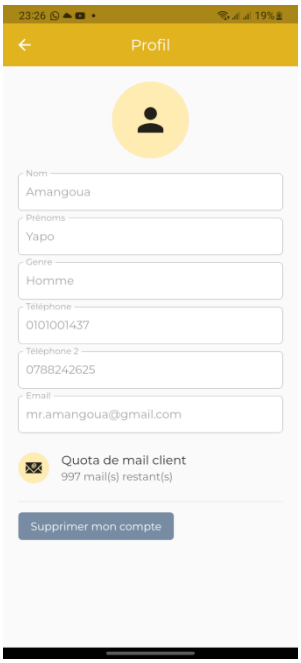


Figure 28: Profil de l'utilisateur

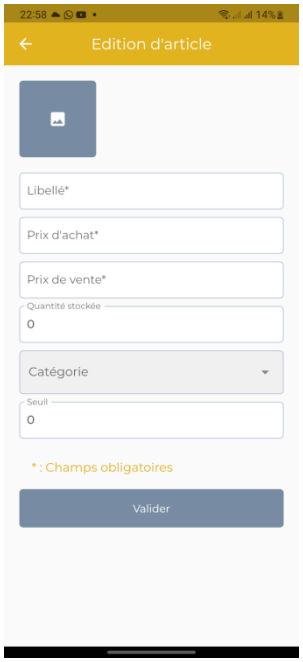


Figure 29: Page d'édition d'article

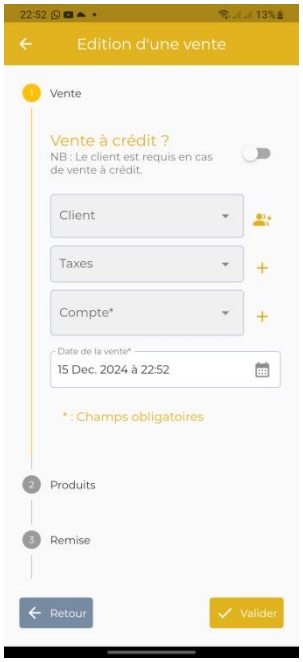


Figure 30: Page d'édition de vente

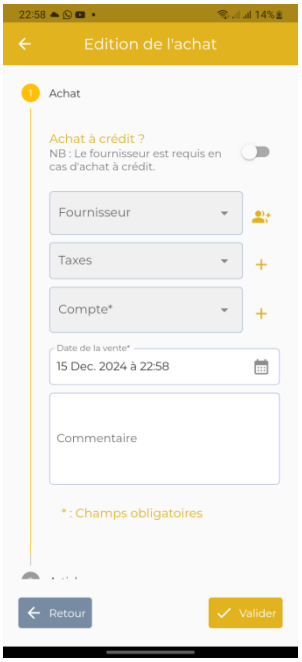


Figure 31: Page d'édition d'achat

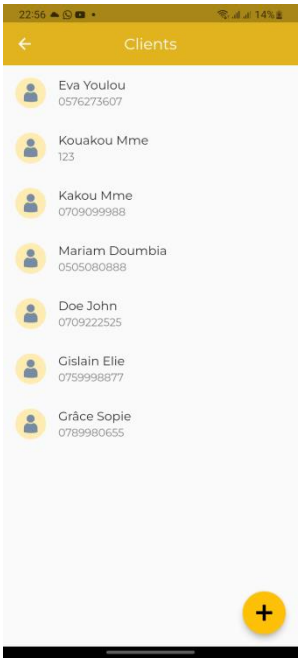


Figure 32: Liste des clients

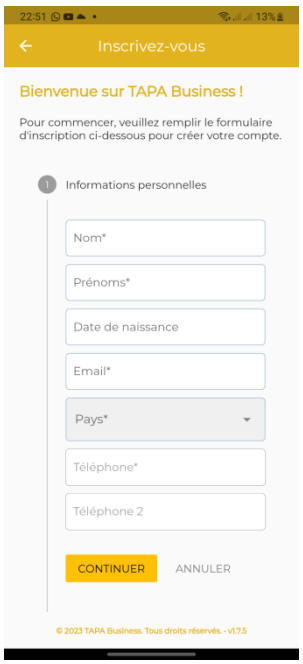


Figure 33: Page d'inscription

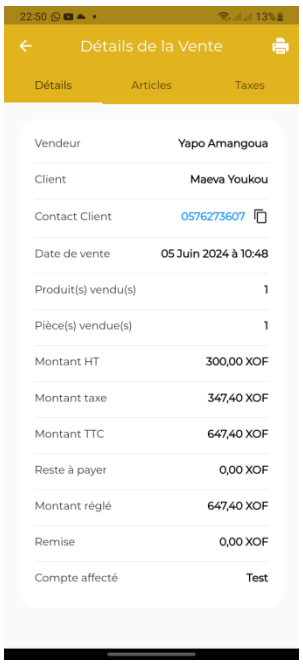


Figure 34: Détails d'une vente

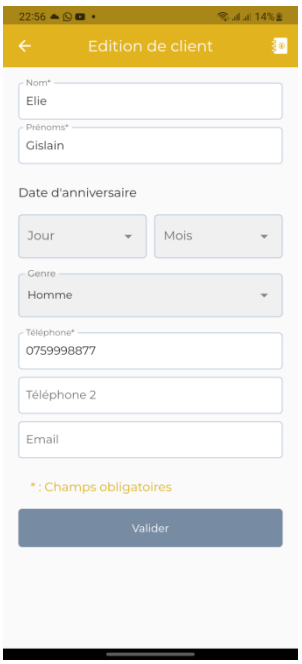


Figure 35: Page d'édition des clients

Figure 36: Page des transactions d'un compte

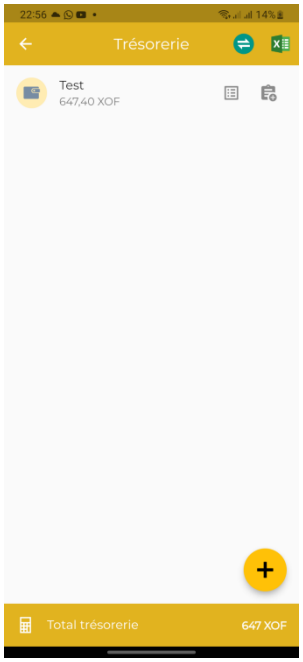


Figure 37: Liste des comptes

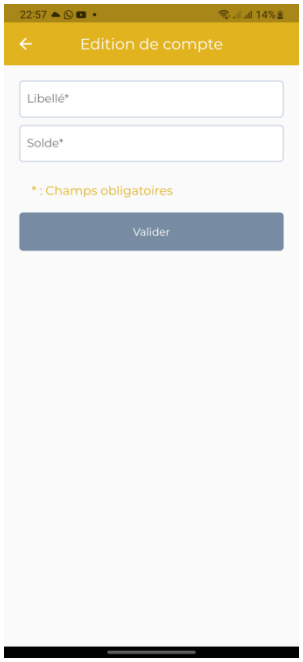


Figure 38: Page d'édtion de compte

II- Déploiement de la solution

1. Description de l'environnement de déploiement

1.1. Système d'exploitation Linux

La majorité des environnements de déploiement pour Laravel reposent sur des systèmes d'exploitation Linux, tels qu'Ubuntu, CentOS, Debian ou Red Hat. Cela s'explique par le fait que Laravel, développé en PHP, s'intègre naturellement avec les serveurs Linux, où PHP est largement utilisé comme langage de script.

1.2. Serveur Web : Apache version 2.4

Le serveur web le plus fréquemment utilisé pour les applications Laravel est Apache, bien que Nginx soit également une alternative largement adoptée. Ces deux serveurs web sont capables de traiter les requêtes HTTP et de diriger efficacement le trafic vers l'application Laravel.

1.3. Base de données : MySQL version 8.0

Laravel supporte divers systèmes de gestion de bases de données, notamment MySQL, PostgreSQL, SQLite et SQL Server. Il est donc essentiel de choisir un SGBD compatible avec Laravel pour garantir le bon fonctionnement de l'application.

1.4. PHP : version 7.3 ou 8.0

Laravel nécessite une version supérieure à celle minimum pour le projet qui est dans le fichier « composer.json » pour fonctionner correctement. Il est donc important de s'assurer que PHP est installé et configuré correctement sur le serveur.

1.5. Composer : version 2.4.4

Laravel utilise Composer, un gestionnaire de dépendances pour PHP, pour gérer les bibliothèques tierces et les packages nécessaires à l'application. Il est donc important de disposer d'une version de Composer compatible avec Laravel.

1.6. Outils de développement : Git & SSH

Afin de simplifier le développement et le déploiement de l'application, il est conseillé d'utiliser des outils comme Git, pour la gestion des versions, et SSH, pour un accès sécurisé et distant au serveur.

1.7. Google play store

Nous avons opté pour le déploiement de notre application mobile uniquement sur le Google Play Store afin de permettre à nos utilisateurs de télécharger facilement et de maintenir leur application à jour. Ce choix nous permet non seulement de toucher nos utilisateurs cibles habitués à ce magasin d'applications, mais aussi d'atteindre le maximum d'utilisateurs disposant d'un smartphone Android. Après avoir étudié notre public cible, il est apparu que la majorité utilise des appareils Android, c'est pourquoi nous n'avons pas émis de version iOS.

2. Procédure de déploiement

2.1. Application côté serveur

2.1.1. Préparation de l'environnement

Avant de déployer l'application, il est essentiel de vérifier que l'environnement de déploiement répond aux exigences spécifiques de notre application Laravel, notamment le système d'exploitation, le serveur web, la base de données, PHP, Composer, et Git (comme mentionné dans la section précédente).

2.1.2. Clonage du code source

La première étape pour déployer une application Laravel consiste à cloner le code source depuis le référentiel Git distant vers le serveur de production. Pour ce faire, on utilise la commande git clone :

```
git clone https://github.com/nom_utilisateur/mon_projet.git /var/www/html/mon_projet
```

2.1.3. Installation des dépendances

Après avoir cloné le code source, il est nécessaire d'installer les dépendances du projet à l'aide de Composer. Pour ce faire, se placer dans le répertoire du projet et lancer la commande suivante : composer install.

```
cd /var/www/html/mon_projet
composer install --no-dev --optimize-autoloader
```

2.1.4. Configuration de l'environnement de développement

Il est nécessaire de configurer les variables d'environnement de l'application, notamment les informations de connexion à la base de données et les clés de sécurité. Copier le fichier « .env.example » vers un fichier « .env », puis configurer les variables selon l'environnement de déploiement.

```
cp .env.example .env
```

2.1.5. Générer la clé de l'application

Laravel utilise une clé pour crypter les sessions de l'utilisateur et les cookies. Pour générer une nouvelle clé, exécuter la commande suivante :

```
php artisan key:generate
```

2.1.6. Configurer les autorisations

Il est important de s'assurer que les autorisations des fichiers et des répertoires sont correctement configurées. Pour cela, on utilise les commandes suivantes :

```
chmod -R 775 storage bootstrap/cache
chown -R www-data:www-data /var/www/html/mon_projet
```

2.1.7. Exécuter les migrations

Avant de déployer l'application, il est nécessaire de migrer la base de données pour s'assurer que les tables et les relations sont correctement créées. Pour cela, exécuter la commande suivante :

```
php artisan migrate
```

2.1.8. Optimiser l'application

Pour améliorer les performances de l'application, il est recommandé d'optimiser les fichiers de l'application. Pour cela, utiliser les commandes
suivantes :

```
php artisan cache:clear
php artisan config:clear
php artisan view:clear
php artisan route:clear
php artisan optimize
```

2.1.9. Redémarrer le serveur

Après avoir terminé toutes les étapes de déploiement, il est recommandé de redémarrer le serveur pour s'assurer que l'application est correctement déployée :

```
sudo systemctl restart apache2
```

2.2. Application mobile cliente

Pour rendre notre application disponible aux utilisateurs finaux, nous avons choisi de la publier sur le Google Play Store. Voici les étapes détaillées pour ajouter une application sur cette plateforme :

2.2.1. Créer un compte Google Play Developer

- ✧ Accéder au Google Play Console.
- ✧ Se connecter avec un compte Google existant ou en créer un.
- ✧ Payer les 25 USD de frais d'inscription uniques pour créer un compte développeur.
- ✧ Compléter les informations nécessaires (nom développeur, adresse email, numéro de téléphone).

2.2.2. Préparer l'application pour le déploiement

- ✧ Générer un fichier APK ou AAB :

Dans notre cas, nous avons opté pour le format AAB (Android App Bundle), recommandé pour une meilleure optimisation sur Google Play. Commande pour générer un AAB avec Flutter :

```
yvesamangoua@MacBook-Pro-2023-de-Yves ~ % flutter build appbundle
```

- ✧ Configurer l'icône et les visuels :
  - Ajouter l'icône de l'application (résolution recommandée : 512x512 px).
  - Préparer des captures d'écran de l'application pour différentes tailles d'appareils Android (smartphones et tablettes).

- ✧ Rédiger les métadonnées de l'application :
  - Titre : Nom de l'application (maximum 50 caractères).
  - Description brève : Une courte présentation (maximum 80 caractères).
  - Description complète : Détails des fonctionnalités et avantages de l'application (maximum 4 000 caractères).
  - Catégorie : Choisir une catégorie pertinente (ex. : Productivité, Jeux, etc.).
  - Adresse email du support.
  - Politique de confidentialité : Fournir un lien vers une page décrivant la gestion des données utilisateurs.

2.2.3. Configurer l'application dans Google Play Console

- ✧ Aller dans la Google Play Console et cliquer sur Créer une application.
- ✧ Remplir les informations suivantes :
  - Nom de l'application.
  - Langue par défaut.
  - Type d'application : Application ou Jeu.
  - Catégorie : Sélectionner la catégorie correspondante.
  - Déclarer si l'application contient des publicités.

- ✧ Téléverser le fichier AAB :
  - Se rendre dans la section Production > Créer une nouvelle version.
  - Téléverser le fichier .aab généré précédemment.
- ✧ Remplir la fiche de contenu :
  - Répondre aux questions sur le contenu de l'application (contenu sensible, cible d'âge, etc.).
- ✧ Configurer les tests (facultatif) :
  - Avant la mise en production, il est possible de déployer l'application en mode test interne ou test fermé pour vérifier son fonctionnement auprès d'un groupe restreint d'utilisateurs.

2.2.4. Déclarer les règles de confidentialité et de sécurité

- ✧ Fournir un lien vers la politique de confidentialité.
- ✧ Répondre au questionnaire sur la sécurité des données utilisateurs.

2.2.5. Publier l'application

- ✧ Une fois toutes les informations renseignées et le fichier téléversé, cliquer sur Revue et publication.
- ✧ Google effectuera une vérification de l'application, qui peut prendre quelques heures à plusieurs jours.
- ✧ Une fois l'examen terminé et validé, l'application sera publiée sur le Play Store et accessible aux utilisateurs.

Conclusion

À l'issue de notre étude, il est évident que ce projet nous a permis d'intégrer des concepts théoriques dans un cadre pratique. En particulier, l'utilisation de la méthode du Processus Unifié combinée au Langage de Modélisation Unifié (UML) a été cruciale pour structurer le développement de notre application. L'objectif initial, à savoir concevoir une solution numérique permettant la gestion des processus de stock et d'achats, a été concrétisé avec succès par la création d'une application web. Cette application a été développée en suivant un processus rigoureux, allant de l'étude préliminaire à l'analyse détaillée, avant d'aboutir au déploiement final.

La mise à disposition de cette application aux PME désirant optimiser la gestion quotidienne de ses activités permettra d'optimiser la gestion interne de ses opérations de stock et d'achats. Bien que ce projet représente une avancée significative, il n'en demeure pas moins perfectible. Nous soumettons volontiers notre travail aux critiques et suggestions constructives, car elles constituent une opportunité d'apporter des améliorations et d'assurer l'évolution continue de notre solution. Ce processus de rétroaction est essentiel pour enrichir l'application, garantir sa pertinence et répondre aux besoins évolutifs des utilisateurs.

Bibliographie

- UML 2 analyse et conception, 2008 – Joseph GABAY, David GABAY
- Comprendre Merise : Outils conceptuels et organisationnels – Jean-Patrick
- MATHERON
- MERISE : Guide pratique (Nouvelle édition) – Jean-Luc BAPTISTE

Webographie

- https://www.apprendre-laravel.fr
- https://youtube.com
- https://laracasts.com
- https://laravel.com
- https:// github.com
- https://openclassroms.com
- https://w3schools.in
- https://chat.openai.com
- https://www.lucidchart.com
- https://flutter.dev
- https://lecipefformationcanva.my.canva.site/tapa-conseil

Annexes

Facture

N° de facture	3228
Emis par	Mady lait et dêguê
Emis pour (client)	Eva Youlou
Contact client	0576273607
Date	05 Juin 2024 à 10:48

Produit	Quantité	Prix	Total
Dêguê de mil 300f	1	300,00 XOF	300,00 XOF
Test	-	115.8%	347,40 XOF

Montant HT	300,00 XOF
Reste à payer	0,00 XOF
Taxes	347,40 XOF
Remise	0,00 XOF
Montant TTC	647,40 XOF

Emis le 17 Dec. 2024 à 04:03 Page 1

1/1

Table des matières