

REPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie

MINISTRE DE L'ENSEIGNEMENT
SUPERIEUR

UNIVERSITE DE DOUALA

INSTITUT UNIVERSITAIRE DE
TECHNOLOGIE



REPUBLIC OF CAMEROON
Peace-Work-Fatherland

MINISTER OF HIGHER
EDUCATION

UNIVERSITY OF DOUALA

UNIVERSITY INSTITUTE OF
TECHNOLOGY



RAPPORT DE PROJET ACADEMIQUE

**THÈME : CONCEPTION ET DEVELOPPEMENT D'UNE
PLATEFORME DE MISE EN RELATION ENTRE VENDEURS ET
CONSOMMATEURS, AVEC UN SYSTÈME INTEGRE DE
PROMOTION DES PRODUITS**

*Projet effectué en vue de l'obtention de la Licence de Technologie en **Génie
Logiciel***

Par :

YIMGA TCHOUTA ORNELLA YVANA

Sous l'encadrement de :

Encadreur académique

Mr MBOMPIEZE Joel

Année Académique :2024-2025

DEDICACE

“A ma famille”

REMERCIEMENTS

Le présent travail n'aurait été possible sans la contribution de nombreuses personnes. À cet effet, je remercie particulièrement :

- **Dieu** pour la santé, la force renouvelée chaque jour et pour tout ce qui ne cesse de faire.

- **Pr Jacques ETAME**, Directeur de l'IUT de Douala, pour le cadre agréable d'étude offert pour ma formation.

- **Pr ESSIBEN**, (chef département informatique à IUT) pour sa disponibilité et surtout sa détermination dans la formation des étudiants ;

- **M. Joel MBOMPIEZE** mon encadreur académique pour sa disponibilité, son dynamisme et son soutien intellectuel.

- Ainsi, j'aimerais exprimer ma profonde gratitude aux enseignants de l'Institut Universitaire de Technologie (IUT) de Douala pour leurs conseils judicieux, précieux et leurs directives pertinentes pour l'intérêt qu'ils ont portés à mon sujet.

- Je tiens à exprimer ma profonde gratitude et mes sincères remerciements à tous ceux qui m'ont aidé dans l'élaboration de ce projet.

- Je suis très reconnaissantes envers toutes les personnes qui ont contribué de près ou de loin à l'élaboration de ce travail.

- mes parents pour le soutien et les sacrifices abattus pour la réalisation de mes études.

- **Mes frères et sœurs**, pour leurs amours, leurs soutiens et les conseils au quotidien ;

SOMMAIRE

RESUME

Ce travail porte sur la conception et le développement d'une plateforme digitale innovante de mise en relation entre vendeurs et consommateurs, intégrant un système performant de promotion des produits, visant à créer un espace numérique sécurisé, ergonomique et performant qui répond aux besoins spécifiques des utilisateurs tout en assurant la fiabilité et la transparence des transactions. Cette plateforme offre aux vendeurs un outil efficace pour commercialiser leurs produits et aux consommateurs un accès simplifié à une large gamme d'articles, en automatisant la gestion des ventes, des paiements, du suivi des commandes et en intégrant des outils analytiques pour optimiser les performances commerciales et la stratégie marketing. La méthodologie adoptée comprend une analyse approfondie des besoins du marché et des utilisateurs, une étude concurrentielle, une phase de conception basée sur la modélisation UML, et un développement technique reposant sur une architecture évolutive et scalable, avec une attention particulière portée à l'ergonomie, la sécurité et à l'intégration du système de promotion. Après des tests utilisateurs et plusieurs itérations, la plateforme fonctionnelle intègre des fonctionnalités clés telles que la gestion multi-vendeurs, le catalogue produit, le système de paiement sécurisé, l'espace personnel, le tableau de bord analytique, la gestion des permissions et rôles, une interface claire et ergonomique, ainsi que l'automatisation des processus de vente et de suivi, validant ainsi sa viabilité technique et commerciale. En somme, la réussite de ce projet repose sur une compréhension fine des besoins utilisateurs, une planification rigoureuse, un équilibre entre technologie robuste, expérience utilisateur soignée, conformité réglementaire, et l'intégration d'une stratégie marketing ciblée accompagnée d'un soutien aux vendeurs, garantissant ainsi l'adoption et la croissance durable de la plateforme.

ABSTRACT

This work focuses on the design and development of an innovative digital platform that connects sellers and consumers, integrating an efficient product promotion system, aiming to create a secure, ergonomic, and high-performance digital space that meets the specific needs of users while ensuring transaction reliability and transparency. This platform provides sellers with an effective tool to market their products and consumers with simplified access to a wide range of items, automating the management of sales, payments, order tracking, and incorporating analytical tools to optimize commercial performance and marketing strategy. The adopted methodology includes an in-depth analysis of market and user needs, a competitive study, a design phase based on UML modeling, and a technical development relying on an evolving and scalable architecture, with particular attention to ergonomics, security, and integration of the promotion system. After user testing and several iterations, the functional platform integrates key features such as multi-seller management, product catalog, secure payment system, personal space, analytical dashboard, role and permission management, a clear and ergonomic interface, as well as automation of sales and tracking processes, thus validating its technical and commercial viability. In summary, the success of this project relies on a thorough understanding of user needs, rigorous planning, a balance between robust technology, refined user experience, regulatory compliance, and the integration of a targeted marketing strategy combined with seller support, thereby ensuring the adoption and sustainable growth of the platform.

LISTES DES FIGURES

LISTES DES TABLEAUX

LISTES DES ABREVIATIONS

INTRODUCTION

De nos jours, de nombreuses entreprises font face à des défis croissants en matière de commercialisation et de visibilité de leurs produits d'où la nécessité d'une plateforme digitale de mise en relation devient un atout majeur. C'est dans cette perspective que notre étude vise à étudier et réaliser une plateforme digitale innovante de mise en relation entre vendeurs et consommateurs répondant aux exigences complexes du commerce électronique moderne. En effet, ce domaine est en plein essor grâce aux avancées des technologies web et du commerce numérique. Les systèmes traditionnels de vente basés sur les canaux physiques sont progressivement complétés par des plateformes numériques offrant des avantages considérables tels qu'une meilleure accessibilité, une plus grande flexibilité dans les promotions, une gestion centralisée améliorée et une intégration facilitée avec d'autres systèmes de marketing digital. Afin de concilier la théorie à la pratique, nous avons mené une analyse approfondie des besoins du marché et des solutions existantes. Par ailleurs, comment optimiser l'efficacité et l'efficience d'une plateforme de mise en relation vendeurs-consommateurs pour assurer une navigation fluide, une promotion ciblée des produits et une expérience utilisateur optimale, tout en garantissant la sécurité des transactions et la protection des données personnelles des utilisateurs ? Dans ce document, nous allons exposer trois chapitres notamment : la présentation du contexte du projet, en mettant en avant les enjeux actuels du commerce électronique, l'état de l'art des solutions existantes et l'analyse des besoins spécifiques. Ensuite, nous aborderons la conception, modélisation et spécifications de la solution, de par l'architecture proposée, les modèles de données et les spécifications fonctionnelles détaillées. Enfin, nous détaillerons la réalisation, validation et déploiement de la plateforme, en couvrant tous les aspects, de l'implémentation du système à la validation des fonctionnalités et au déploiement de la solution complète.

CHAPITRE 1 : PRESENTATION DU CONTEXTE DU PROJET

Dans ce chapitre, nous présenterons le contexte dans lequel entre notre projet, la problématique qu'elle vient résoudre. Par la suite, nous parlerons de la méthode d'analyse utilisée et les diagrammes adéquats pour le bon fonctionnement de notre application.

I. Le commerce électronique et les plateformes de mise en relation

I.1. Définition et évolution du commerce électronique

Le commerce électronique, désigne l'ensemble des échanges commerciaux réalisés via Internet, transformant profondément les modes de vente traditionnels en permettant aux entreprises de vendre des produits ou services à distance, 24h/24 et sans limite géographique. Depuis ses débuts dans les années 90 avec des catalogues en ligne simples et des paiements manuels, le e-commerce a rapidement évolué vers des plateformes sophistiquées qui automatisent la gestion des paiements, des stocks, des promotions et du service après-vente, tout en offrant des interfaces sécurisées et faciles à utiliser. L'essor des smartphones, les progrès des technologies web et la popularité des réseaux sociaux ont renforcé son rôle clé dans l'économie numérique, offrant aux grandes entreprises comme aux vendeurs indépendants la possibilité d'atteindre un large public sans lourds investissements en infrastructure.

I.2. Typologie des plateformes de commerce électronique

Il existe aujourd'hui plusieurs types de plateformes e-commerce, chacune répondant à des logiques et des besoins différents :

- Les boutiques en ligne individuelles (B2C) : Ce sont des sites créés par une seule entreprise ou un commerçant pour vendre directement ses produits à des clients.
- Les plateformes B2B (Business to Business) : Elles facilitent les échanges entre professionnels (grossistes, distributeurs, fournisseurs) et permettent la gestion de

commandes en gros, avec des fonctions adaptées aux besoins spécifiques du commerce inter-entreprises.

- Les marketplaces multi-vendeurs (C2C ou B2C élargi) : Ces plateformes permettent à plusieurs vendeurs d'ouvrir leur propre boutique au sein d'un même site, et de proposer leurs produits aux consommateurs. Des exemples connus incluent Amazon, Etsy ou Jumia. Les marketplaces se démarquent par leur capacité à mutualiser l'audience, centraliser les paiements et offrir des outils communs (promotion, livraison, évaluation...).

Rôle	Description
Centralisation de l'offre	Agrège des milliers de produits et de vendeurs, facilitant la navigation pour l'utilisateur.
Mise en concurrence	Permet aux consommateurs de comparer les prix et les avis pour un même produit.
Accessibilité pour les vendeurs	Réduit les barrières à l'entrée : pas besoin de créer un site ou d'investir dans une logistique coûteuse.
Services mutualisés	Intègre paiement sécurisé, service client, gestion des retours, etc.
Internationalisation	Ouvre l'accès à des marchés étrangers sans infrastructure locale.

Tableaux1 : Rôles des marketplaces multi-vendeurs

Dans le cadre de notre projet, nous nous sommes orientés vers le modèle marketplace, permettant à des vendeurs de s'enregistrer, créer leur boutique, publier des produits, recevoir des commandes de la part des consommateurs etc...

II. Problématique et objectifs de la plateforme développée

II.1. Problématique rencontrée par les utilisateurs

L'essor du commerce électronique a bouleversé les habitudes d'achat, mais cette évolution ne bénéficie pas équitablement à tous les acteurs. En particulier, de nombreux **petits vendeurs ou commerçants locaux**, souvent issus du secteur informel ou non numérique, se heurtent à des difficultés majeures lorsqu'il s'agit de **digitaliser leur activité** notamment :

- **Absence de présence en ligne pour les vendeurs non digitalisés** : Beaucoup de commerçants n'ont aucune visibilité sur internet, ce qui limite leur clientèle, les empêche de vendre en ligne, de suivre les nouvelles habitudes d'achat et de booster leurs ventes.
- **Manque de moyens pour créer une boutique en ligne** : La création d'un site e-commerce demande des compétences techniques, un budget et du temps, ce qui empêche de nombreux petits vendeurs de se lancer dans le commerce en ligne.
- **Faible maîtrise des techniques de promotion et marketing digital sur les réseaux sociaux** : Même en ligne, les vendeurs ont du mal à se rendre visibles à cause d'un manque de compétences en marketing, en référencement, et d'outils d'analyse ou de publicité adaptés.

II.2. Objectifs poursuivis par la plateforme développée

Face aux multiples freins rencontrés par les petits vendeurs pour se lancer dans le commerce en ligne, cette plateforme vise à répondre de manière concrète aux besoins identifiés, en apportant des **solutions simples, accessibles et efficaces** notamment :

- **Démocratiser l'accès à la vente en ligne pour les petits vendeurs** : L'objectif principal est de permettre aux vendeurs ayant des moyens financiers limités ni compétences techniques de créer facilement leur boutique en ligne, de gérer leurs produits et commandes, et d'élargir leur clientèle sans obstacles majeurs.
- **Offrir un espace centralisé pour améliorer l'expérience des consommateurs** : La plateforme vise à regrouper un ensemble varié de vendeurs sur une interface unique,

afin de simplifier la recherche, la comparaison et l'achat pour les utilisateurs, tout en leur proposant un parcours fluide et intuitif.

- **Instaurer un environnement sécurisé et contrôlé :** Pour garantir la confiance des utilisateurs, la plateforme intègre un système d'approbation des vendeurs, des paiements sécurisés, une gestion centralisée des contenus, et une protection des données personnelles des acheteurs.
- **Un système de promotion assuré par l'équipe marketing :** Un objectif clé est de permettre aux vendeurs faire la promotion de leurs produits sur les réseaux sociaux avec finalisations du paiement sur notre plateforme pour améliorer la visibilité de leurs produits et stimuler les ventes.

Cette plateforme se veut être un outil inclusif et accessible, qui relie les vendeurs en quête de visibilité et les consommateurs à la recherche de simplicité et de variété, dans un cadre sécurisé, léger et facile à utiliser, sans prétendre concurrencer les géants du secteur.

CHAPITRE 2 : ANALYSE ET CONCEPTION DE LA PLATEFORME DIGITALE

Dans ce chapitre, nous présenterons la méthode d'analyse utilisée et les diagrammes adéquats pour le bon fonctionnement de notre application. Par la suite, nous parlerons de la conception et de la réalisation de la plate-forme.

I. PRESENTATION ET JUSTIFICATION DU CHOIX DE LA METHODE D'ANALYSE

Lors de la réalisation d'un projet de développement logiciel, il est essentiel d'utiliser une méthode d'analyse appropriée pour comprendre les besoins, spécifier les fonctionnalités et concevoir l'architecture du système. L'Unified Modeling Language (UML) est l'une des méthodes les plus répandues et populaires pour la modélisation et l'analyse des systèmes logiciels. Dans cette section, nous présenterons UML comme méthode d'analyse pour notre rapport de stage en génie logiciel et justifierons ce choix

I.1. Présentation : les diagrammes UML

L'Unified Modeling Language (UML) est un langage graphique standard utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système logiciel. Il fournit un ensemble de diagrammes qui permettent de représenter les différentes perspectives d'un système, allant de l'analyse des besoins à la conception détaillée. UML offre une notation graphique intuitive, ce qui facilite la communication entre les différentes parties prenantes du projet, telles que les développeurs, les concepteurs, les testeurs et les clients.

En plus d'UML, il existe d'autres méthodes de modélisation et d'analyse, parmi lesquelles :

- MERISE
- OOSE
- OMT
- BOOCH ou OOD

Un diagramme UML est une représentation graphique qui s'intéresse à un aspect précis du modèle. Chaque diagramme possède une structure propre, et les types d'éléments de modélisation qui le composent sont prédéfinis. Un type de diagramme UML véhicule une sémantique précise, c'est-à-dire qu'un type de diagramme offre toujours la même vue du système. En combinant les différents diagrammes, UML offre une vue complète des aspects statiques et dynamiques d'un système.

UML comporte 13 types de diagrammes, organisés en deux vues principales : la vue statique et la vue dynamique.

❖ *Vue Statique du Système*

La vue statique du système est composée de 6 diagrammes :

- **Diagramme de classes**
- **Diagramme d'objets**
- **Diagramme de composants**
- **Diagramme de déploiement**
- **Diagramme de paquetages**
- **Diagramme de structures composites**

Parmi ces diagrammes, nous utiliserons principalement le diagramme de classes dans notre analyse.

❖ *Vue Dynamique du Système*

La vue dynamique du système comprend 7 diagrammes :

- **Diagramme de cas d'utilisation**
- **Diagramme d'activités**
- **Diagramme d'états-transitions**
- **Diagrammes d'interaction**
- **Diagramme de séquence**
- **Diagramme global d'interaction**
- **Diagramme de temps**

Pour la suite, nous utiliserons le diagramme de cas d'utilisation et le diagramme de séquence.

I. Justification du choix de méthode

Le choix de l'Unified Modeling Language (UML) comme méthode d'analyse pour notre projet de développement logiciel est justifié par plusieurs points essentiels :

1. Standardisation

UML est un langage standard reconnu par l'ISO, garantissant la compatibilité entre divers outils et facilitant l'échange d'informations entre équipes. Cette norme assure que les diagrammes sont compréhensibles et utilisables globalement.

2. Communication

La notation graphique intuitive d'UML facilite la communication entre les développeurs, concepteurs, testeurs et clients. Les diagrammes sont accessibles même pour les non-experts, améliorant la clarté des exigences et des spécifications.

3. Complétude

UML propose une gamme complète de diagrammes couvrant les aspects statiques et dynamiques d'un système. Cette variété permet une modélisation exhaustive des fonctionnalités et de la structure, minimisant les risques d'omissions.

4. Flexibilité

UML s'adapte à toutes les phases du cycle de vie du développement, de l'analyse des besoins à la conception détaillée. Cette flexibilité permet des ajustements continus en fonction des évolutions du projet et des nouvelles exigences.

En résumé, UML est un outil puissant et polyvalent, indispensable pour la modélisation, l'analyse et la communication efficace dans notre projet de développement logiciel.

II. CONCEPTION, MODELISATION ET REALISATION DU LOGICIEL

II.1. RAPPEL SUR LES API ET LES APPLICATIONS WEB

II.1.1. Généralités sur les API

Les API (Application Programming Interfaces) sont des outils essentiels dans le développement de logiciels et d'applications. Elles permettent aux différentes applications et services de communiquer entre eux de manière structurée et standardisée. Voici une présentation des API :

- **Qu'est-ce qu'une API ?** Une API est un ensemble de règles et de protocoles qui permettent à deux logiciels différents de communiquer entre eux. Elle définit les méthodes et les formats de données qui peuvent être utilisés pour l'échange d'informations. Les API facilitent l'intégration de différentes fonctionnalités et services dans une application existante.

Pour illustrer ce qu'est une API, imaginez-la comme un serveur dans un restaurant. Le serveur écoute votre commande, se rend en cuisine, prend les produits alimentaires commandés et revient vers vous avec la commande. Cet exemple simple décrit bien le fonctionnement d'une API.

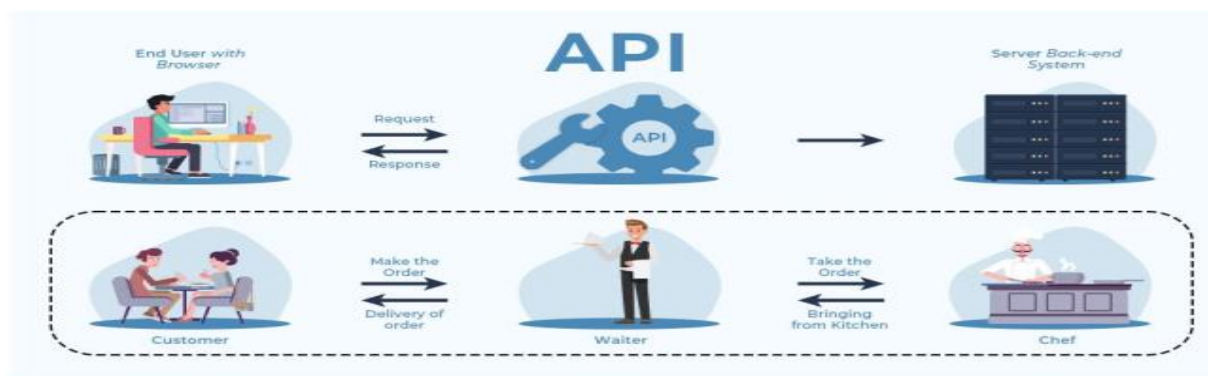


Figure 1 : Schéma de fonctionnement d'une API

➤ Principaux types d'API

- ◆ **API Web** : Utilisées pour permettre aux applications de communiquer via Internet en utilisant des protocoles comme HTTP et des formats de données comme JSON et XML.
- ◆ **API REST** : Un style d'API Web utilisant les verbes HTTP (GET, POST, PUT, DELETE) pour interagir avec les ressources, souvent utilisé pour créer des services web évolutifs et flexibles.

➤ Utilisation des API :

- **Intégration de services tiers** : Pour intégrer des services comme les paiements, les réseaux sociaux et les cartes.
 - **Développement d'applications web** : Pour accéder aux fonctionnalités du système d'exploitation des appareils mobiles (caméra, GPS, notifications push).
 - **Échange de données entre applications** : Facilite l'échange de données entre applications internes et externes.
- **Documentation et gestion des API** : Les fournisseurs d'API fournissent généralement une documentation détaillée pour aider les développeurs à comprendre comment utiliser l'API, les méthodes disponibles, les paramètres, les formats de données, etc. Les plateformes de gestion d'API sont également utilisées pour gérer les accès, surveiller les performances, analyser l'utilisation et appliquer des politiques de sécurité.

En résumé, les API sont des outils puissants qui facilitent la communication et l'intégration entre les différentes applications et services. Elles permettent aux développeurs de tirer parti de fonctionnalités existantes et de construire des applications plus robustes, flexibles et interconnectées.

II.1.2. Généralités sur les Applications Web

Les applications web occupent une place centrale dans l'écosystème numérique actuel. Elles permettent aux utilisateurs d'interagir avec des services via un navigateur, sans nécessiter d'installation préalable, et sont accessibles depuis divers types d'appareils (ordinateurs, tablettes, smartphones).

➤ Qu'est-ce qu'une application web ?

Une application web est une application logicielle accessible via un navigateur Internet (comme Chrome, Firefox ou Safari), et qui s'exécute sur un serveur distant. Contrairement aux logiciels traditionnels installés localement, l'application web ne nécessite pas de téléchargement ni d'installation, et peut être utilisée sur n'importe quel appareil connecté à Internet.

Elle repose généralement sur une architecture **client-serveur**, où l'interface utilisateur est affichée dans le navigateur (frontend) et les traitements logiques s'effectuent sur un serveur distant (backend).

➤ Typologie des applications web :

Les applications web peuvent être classées selon leur complexité et leur niveau d'interactivité :

- **Applications web statiques** : Affichent du contenu fixe. Elles sont rapides à charger mais limitées en interaction.
- **Applications web dynamiques** : Génèrent du contenu interactif en fonction des actions de l'utilisateur. Elles s'appuient sur des technologies serveur comme PHP, Node.js, ou Laravel.
- **Single Page Applications (SPA)** : L'interface est chargée une seule fois, puis mise à jour dynamiquement (ex. : React, Angular). Elles offrent une expérience fluide semblable à celle d'une application native.
- **Progressive Web Apps (PWA)** : Applications web améliorées qui peuvent fonctionner hors ligne, envoyer des notifications et être installées comme une application mobile.

➤ Technologies clés pour le développement des applications web :

- **Frontend (interface utilisateur)** : HTML, CSS, JavaScript, avec des bibliothèques ou frameworks comme **React**, **Vue.js**, ou **Angular**.

- **Backend (logique métier et base de données) :** Frameworks comme **Laravel**, Django, Node.js, connectés à des bases de données telles que **MySQL**, PostgreSQL ou MongoDB.
- **API REST :** Interface de communication entre le frontend et le backend, facilitant l'échange de données.
- **Sécurité :** Protocoles HTTPS, gestion des authentifications (par exemple **Laravel Sanctum**, OAuth), et protection des données utilisateurs.
- **Outils de déploiement :** Services comme **Vercel**, **Railway**, ou **Heroku** pour héberger et mettre en ligne rapidement une application.

➤ **Importance des applications web :**

- **Accessibilité multiplateforme :** Elles fonctionnent sur tous les systèmes d'exploitation disposant d'un navigateur web.
- **Déploiement rapide :** Une seule mise à jour suffit pour que tous les utilisateurs accèdent à la nouvelle version.
- **Évolutivité :** Elles sont facilement extensibles pour intégrer de nouvelles fonctionnalités ou répondre à une forte demande.
- **Centralisation des données :** Toutes les informations sont stockées sur un serveur, ce qui facilite la gestion, la sauvegarde et la sécurisation.
- **Rentabilité :** Développement généralement moins coûteux qu'une application native, tout en touchant un public plus large.

II.1.3. Fonctionnement des api dans le projet

Dans cette section, nous présentons le mécanisme de communication entre les différentes parties de l'application via l'API. Notre plateforme de mise en relation entre vendeurs et consommateurs repose sur une architecture **API RESTful**, qui permet une interaction fluide entre le **backend Laravel** (serveur/API) et le **frontend React** (interface utilisateur).

L'authentification est gérée via **Laravel Sanctum**, garantissant un accès sécurisé aux ressources selon les rôles (vendeur, client, administrateur).

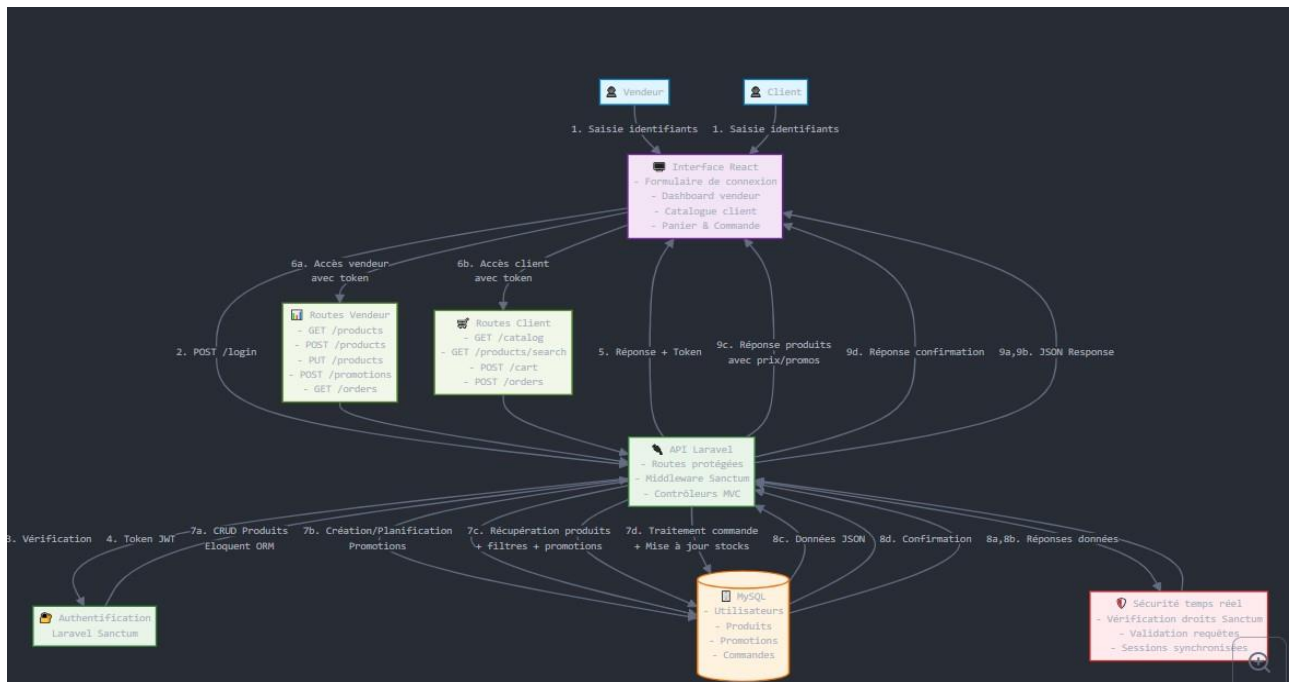


Figure2 : Fonctionnement général de l'API dans la plateforme

Description du fonctionnement :

1. Connexion de l'utilisateur (vendeur ou client)

L'utilisateur saisit ses identifiants dans l'interface React. Une requête HTTP POST est envoyée à l'API Laravel.

→ Laravel traite la requête via le middleware Sanctum pour vérifier l'identité et émettre un **jeton d'authentification**.

2. Accès aux fonctionnalités selon le rôle

Une fois authentifié, l'utilisateur peut accéder à différentes routes API selon son rôle :

- Les **vendeurs** accèdent aux modules de gestion de produits, création de promotions, consultation des commandes.
- Les **clients** peuvent consulter les produits, appliquer des filtres, ajouter au panier et passer commande.

3. Ajout / modification de produits

Lorsqu'un vendeur ajoute ou modifie un produit via le tableau de bord React, une requête API POST ou PUT est envoyée.

→ L'API Laravel enregistre ces données dans la base MySQL via le **modèle MVC** (modèle Eloquent).

4. **Création de promotions dynamiques**

Le vendeur utilise une interface intuitive pour créer des réductions. Une requête API POST est transmise avec les paramètres de promotion.

→ L'API valide, stocke et planifie la promotion pour affichage sur les bons produits.

5. **Consultation du catalogue client**

Le client interagit avec l'interface React : il consulte les produits, trie, recherche...

→ Ces actions déclenchent des requêtes GET vers l'API Laravel, qui renvoie les données JSON mises à jour (produits, prix, promos).

6. **Passation de commande**

Le client finalise son achat. Une requête API POST contenant les détails de la commande est envoyée.

→ L'API traite la commande, l'associe à l'utilisateur, met à jour les stocks et envoie une réponse de confirmation.

7. **Mise à jour temps réel (états et sécurité)**

À chaque étape, l'API vérifie les droits d'accès via **Sanctum**, renvoie les bonnes réponses au frontend et conserve les échanges synchronisés entre base de données, utilisateurs et sessions.

II.2.1. Architecture logicielle de la plateforme

L'architecture logicielle de la plateforme a été conçue selon une approche moderne, modulaire et évolutive, afin de garantir une séparation claire des responsabilités, une maintenabilité facilitée et une bonne répartition des charges côté client et serveur.

1. **Modèle MVC côté back-end (Laravel)**

Le framework **Laravel** a été utilisé pour le développement du back-end. Il repose sur le modèle MVC (Model – View – Controller), qui permet de séparer :

- **Les modèles** : responsables de l'accès et de la manipulation des données (produits, utilisateurs, commandes, etc.).

- **Les vues** : dans notre cas, pas utilisées directement puisque l'interface est gérée côté front-end.
- **Les contrôleurs** : qui contiennent la logique applicative et les règles métier.

Cette structure permet une meilleure lisibilité du code, une évolutivité accrue, et une organisation claire des différentes responsabilités logicielles.

2. API RESTful pour la communication client-serveur

Le front-end, développé avec React.js, ne dépend pas directement des vues Laravel. La communication entre les deux couches se fait via une API RESTful :

- Les requêtes sont envoyées par le client (React) au serveur Laravel via des appels HTTP (GET, POST, PUT, DELETE).
- Le serveur retourne des réponses structurées en JSON, facilitant leur traitement côté front-end.
- Cela permet aussi une indépendance technologique entre l'interface et le cœur métier, ce qui rend la plateforme facilement adaptable à d'autres interfaces à l'avenir (application mobile, etc.).

3. Authentification sécurisée avec Laravel Sanctum

Pour gérer l'identification et l'authentification des utilisateurs, la solution Laravel Sanctum a été intégrée. Elle offre un mécanisme de gestion de sessions et de tokens léger, adapté aux SPA (Single Page Applications) comme celles construites avec React. Ce système permet :

- Une authentification sécurisée sans exposition des identifiants sensibles.
- Une gestion des autorisations par rôle (vendeur, acheteur, administrateur).
- La persistance des sessions utilisateur avec une politique de sécurité rigoureuse.

4. Déploiement cloud avec Railway et Vercel

Pour assurer la mise en production de la plateforme :

- Le front-end React a été déployé sur Vercel, une plateforme spécialisée dans l'hébergement d'applications modernes, assurant un chargement rapide et un CDN intégré.
- Le back-end Laravel ainsi que la base de données MySQL ont été déployés sur Railway, qui offre un environnement cloud flexible, avec intégration continue et gestion simplifiée des services backend.

Cette infrastructure permet :

- Une séparation claire des environnements front et back, facilitant les mises à jour indépendantes.
- Une scalabilité rapide en cas d'augmentation du trafic ou du nombre d'utilisateurs.
- Un déploiement automatisé à partir des branches du dépôt Git, favorisant une livraison continue.

II.3. MODELISATION PROPREMENT DITE DE L'APPLICATION

Puisque tout système nécessite une modélisation, ceci dans le but de faciliter la compréhension et matérialiser les interactions entre les différents objets du système. Nous allons ici faire la présentation des diagrammes UML nécessaire à la compréhension de notre système.

1. Le diagramme de cas d'utilisation

Le diagramme de cas d'utilisation (ou *use case diagram* en UML) est un outil de modélisation qui permet de représenter **les interactions entre les utilisateurs (appelés "acteurs") et le système**. Il permet également de recueillir, d'analyser, d'organiser les besoins et de recenser les grandes fonctionnalités d'un système. Il représente ainsi un bon moyen de communication entre le maître d'ouvrage et le maître d'œuvre. Il s'agit donc de la première étape que nous avons suivie dans le cadre de la modélisation.

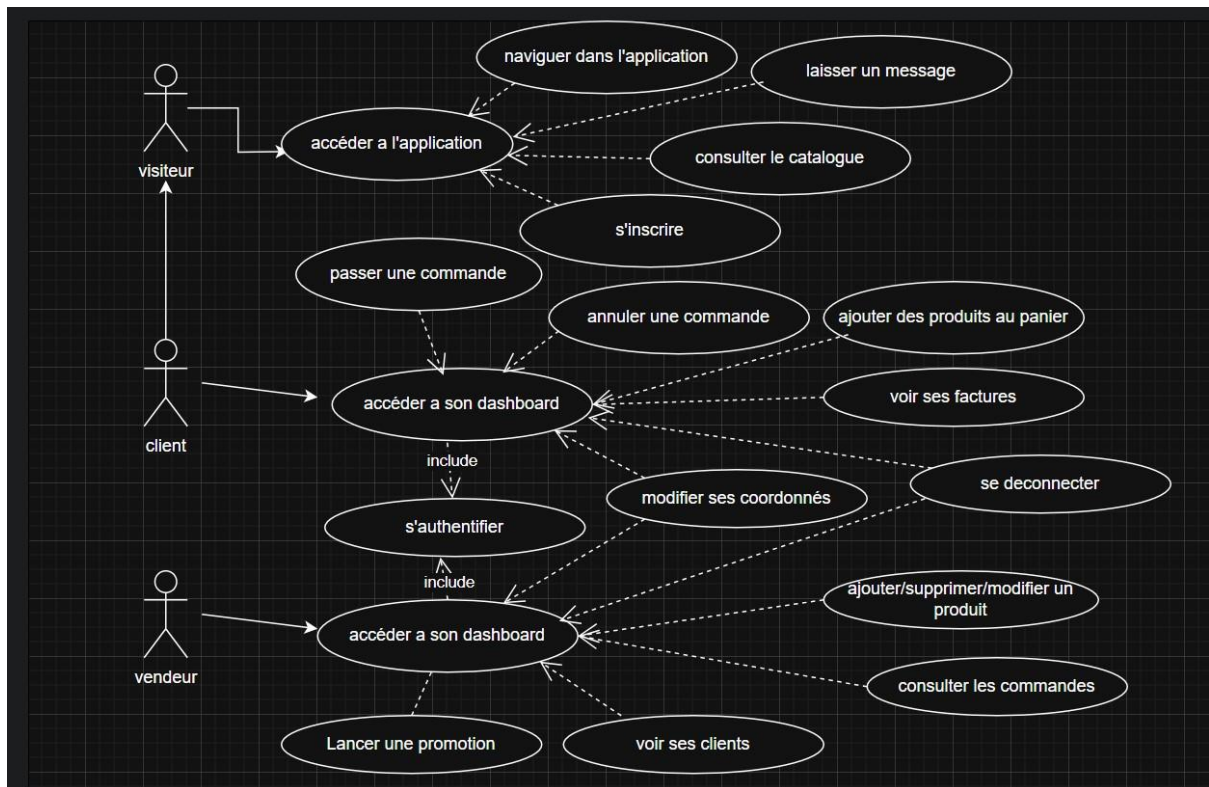


Figure 3 : diagramme de cas d'utilisation (début)

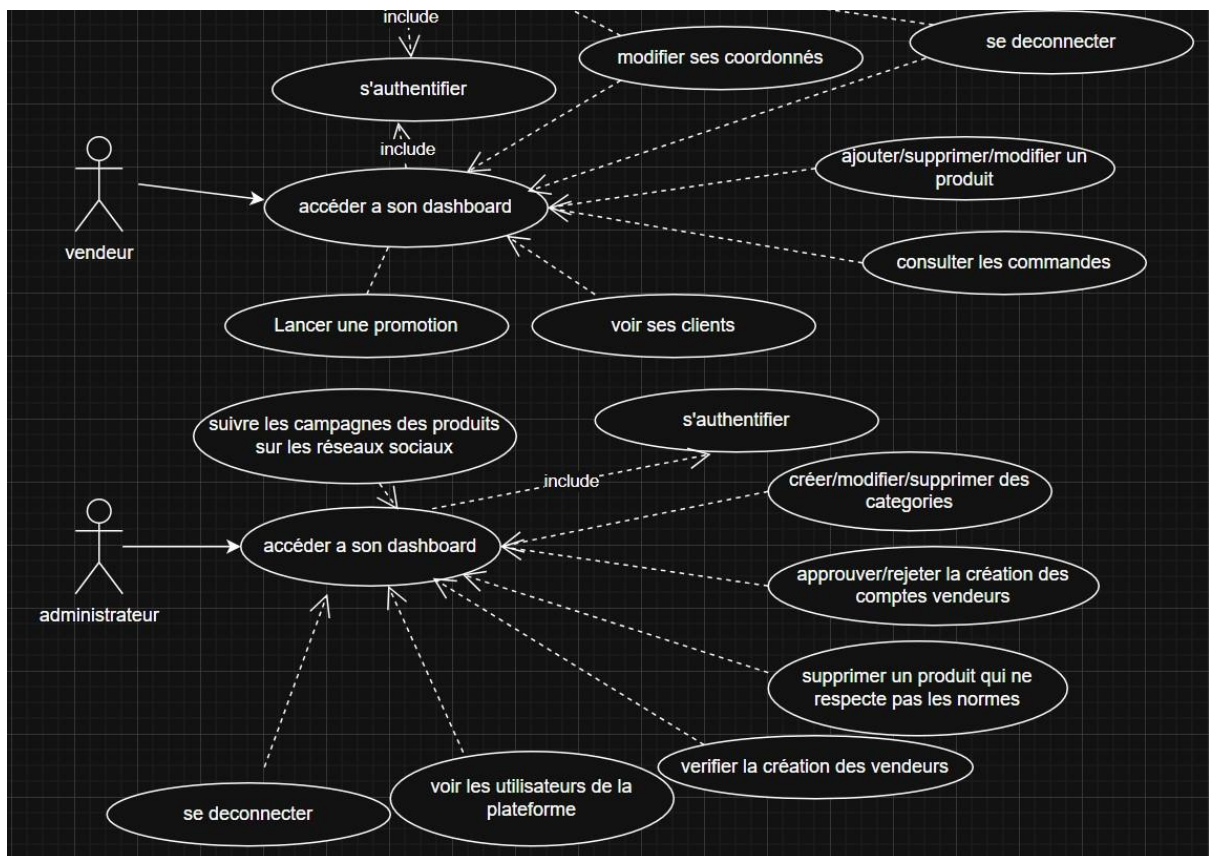


Figure 4 : diagramme de cas d'utilisation (fin)

2. Le diagramme de classe

Le diagramme de classes est considéré comme le plus important dans la Modélisation Orientée Objets (M.O.O) et est indispensable lors de ce type de modélisation. Il représente les concepts du domaine d'application ainsi que les concepts internes créés spécifiquement pour l'implémentation d'une application, indépendamment du langage de programmation orienté objets utilisé. Il s'agit d'une vue statique, car elle ne prend pas en compte le facteur temporel dans le comportement du système.

Les principaux éléments de cette vue statique sont les classes et leurs relations, notamment l'association, la généralisation, ainsi que divers types de dépendances telles que la réalisation et l'utilisation.

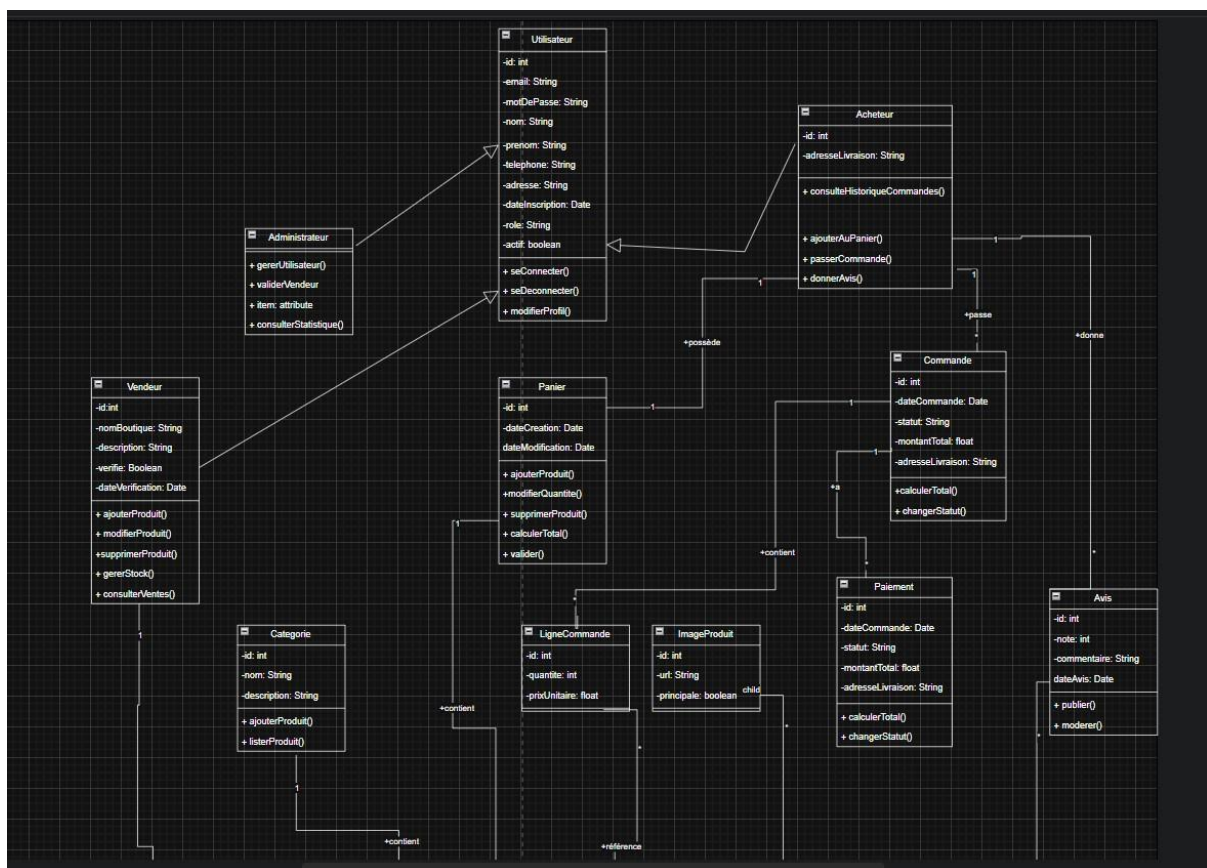


Figure 5 : Diagramme de classe (début)

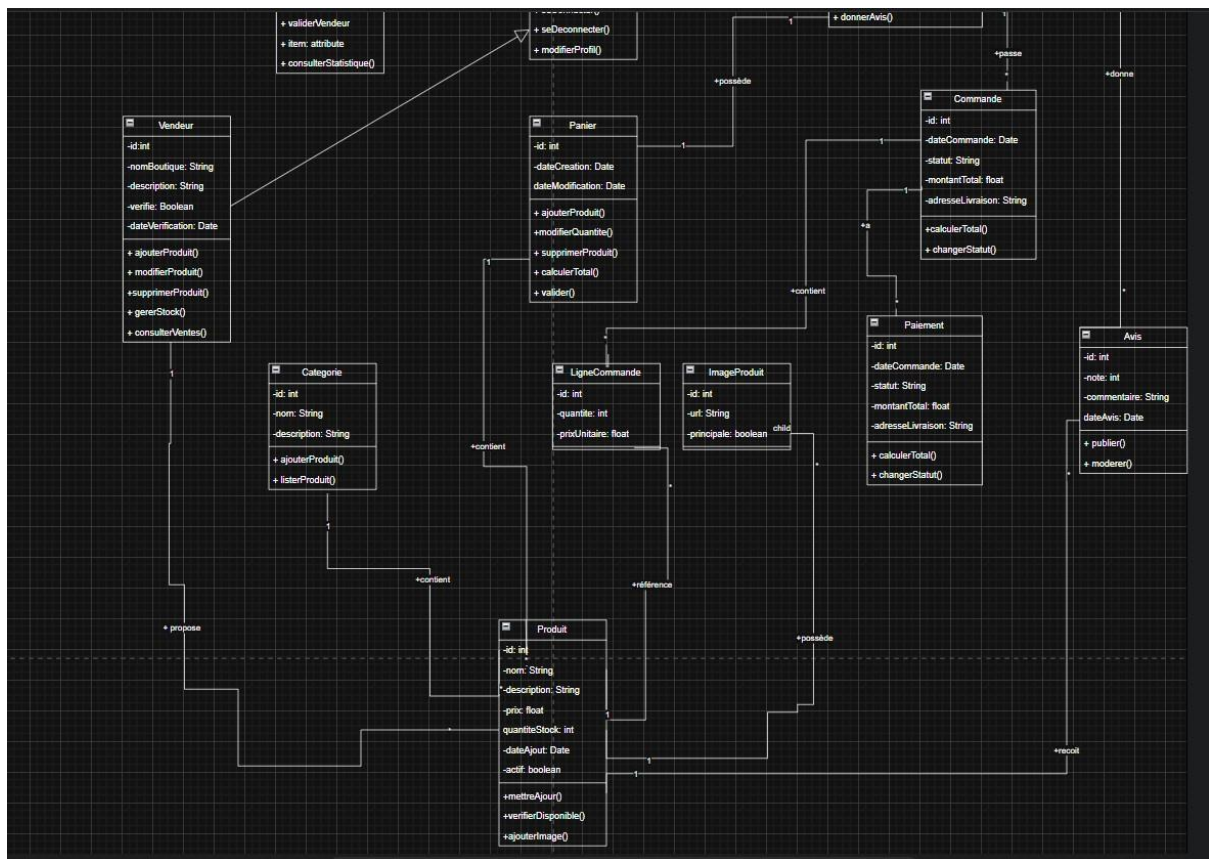


Figure 6 : Diagramme de classe (fin)

3. Diagramme de séquence

Un **diagramme de séquence** est un type de diagramme UML qui sert à **modéliser les interactions entre différents objets ou acteurs d'un système au fil du temps**. Il décrit comment les éléments du système (utilisateurs, composants, classes, etc.) **échangent des messages dans un ordre chronologique** pour accomplir un scénario ou une fonctionnalité spécifique.

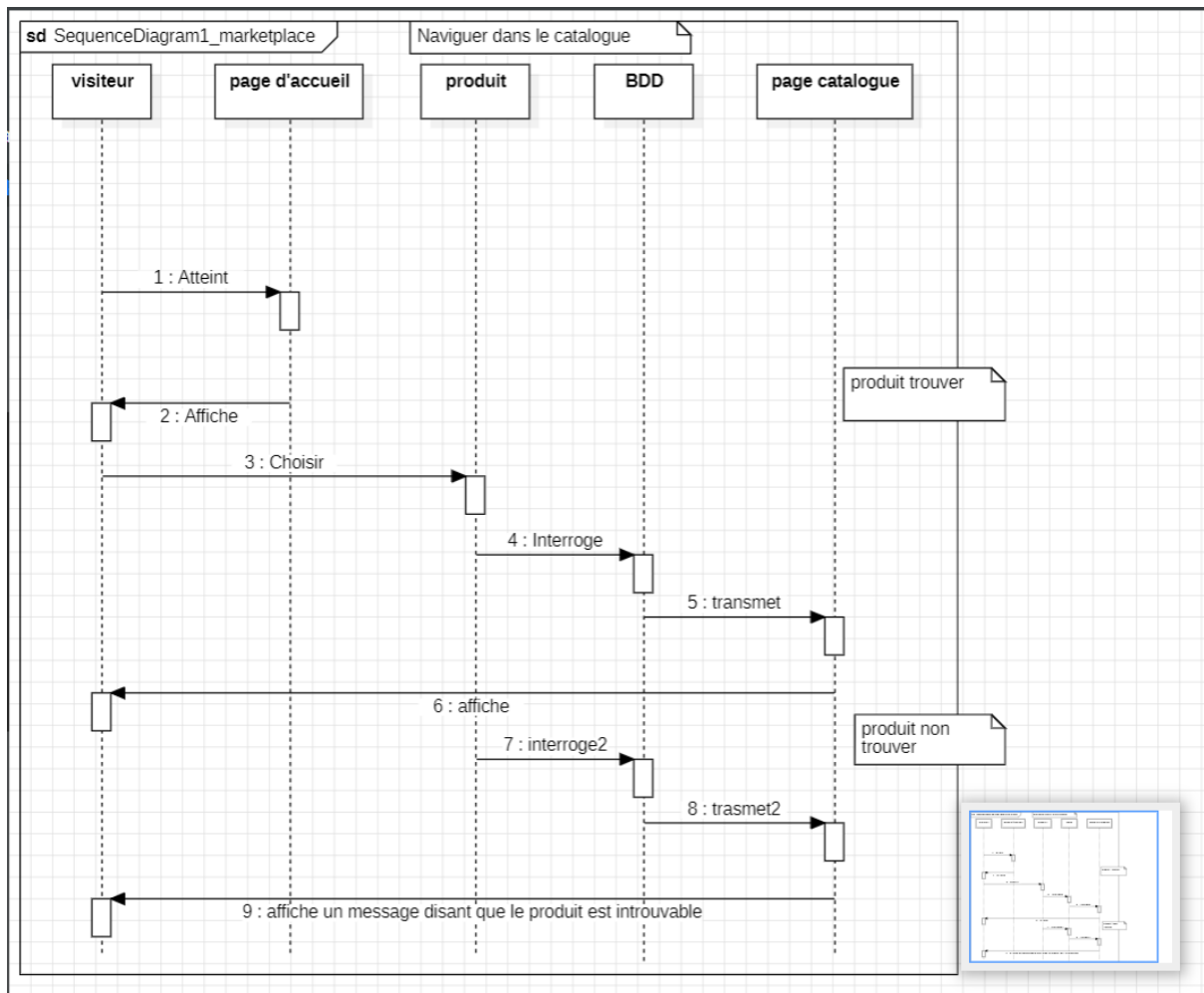


Figure7 : Diagramme de séquence de la navigation dans le catalogue

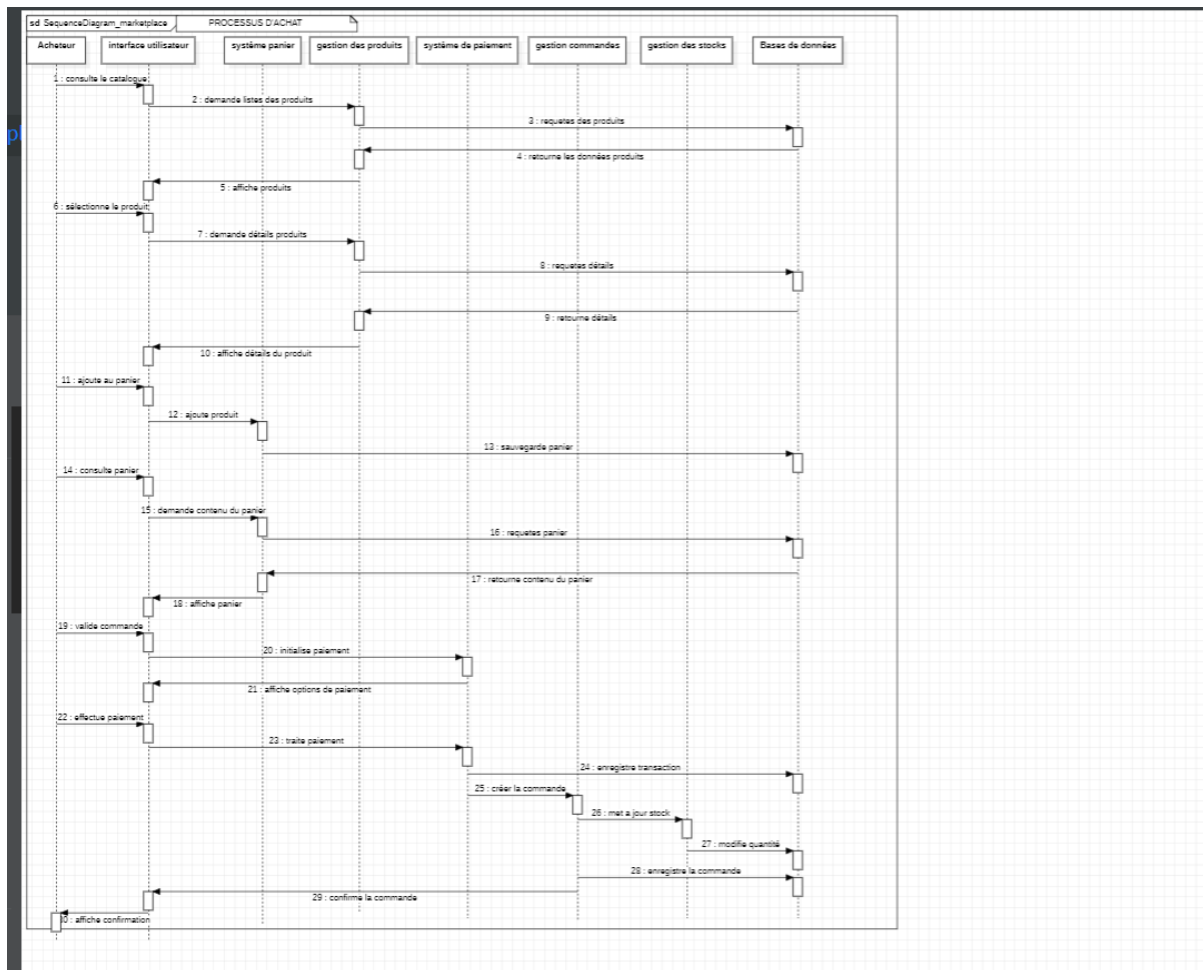


Figure 8 : Diagramme de séquence du processus d'achat

II.4. PHASE DE RÉALISATION DU LOGICIEL

Il existe plusieurs modèles de génie logiciel, également appelés modèles de développement logiciel, qui décrivent les étapes et les processus impliqués dans la création d'un logiciel. Pour notre projet, nous avons suivi le modèle en cascade (ou Waterfall model). Ce modèle suit une approche séquentielle linéaire, où les étapes de développement sont organisées de manière séquentielle, du début à la fin. Chaque phase doit être complétée avant de passer à la suivante, ce qui rend le processus rigide et peu flexible.

Voici les différentes phases du modèle en cascade :

- **Analyse des besoins** : Dans cette phase, les exigences du logiciel sont recueillies et analysées en collaboration avec les parties prenantes. Les fonctionnalités, les

contraintes et les objectifs du système sont identifiés et documentés dans un cahier des charges.

- **Conception** : Une fois les exigences définies, la phase de conception commence. Elle comprend la conception architecturale, la conception détaillée et la conception de l'interface utilisateur. Les décisions relatives à l'architecture du système, à la structure des modules et aux interactions entre eux sont prises pendant cette phase.
- **Implémentation** : Dans cette phase, le code source du logiciel est écrit en utilisant les langages de programmation appropriés. Les développeurs traduisent les spécifications de conception en un code fonctionnel et réalisent les fonctionnalités du logiciel.
- **Test** : Une fois l'implémentation terminée, la phase de test commence. Les tests unitaires, les tests d'intégration et les tests système sont effectués pour s'assurer que le logiciel fonctionne conformément aux spécifications. Les erreurs et les bugs sont identifiés et corrigés.
- **Déploiement** : Après avoir passé avec succès les tests, le logiciel est déployé dans l'environnement de production. Cela peut inclure l'installation du logiciel sur les serveurs, la configuration du système et la formation des utilisateurs.
- **Maintenance** : Une fois que le logiciel est déployé, il entre dans la phase de maintenance. Les mises à jour, les correctifs de bogues et les améliorations fonctionnelles sont apportés en réponse aux commentaires des utilisateurs et aux besoins changeants du système.

Le modèle en cascade est simple et facile à comprendre, car chaque phase est bien définie et les résultats d'une phase sont utilisés comme entrée pour la phase suivante.

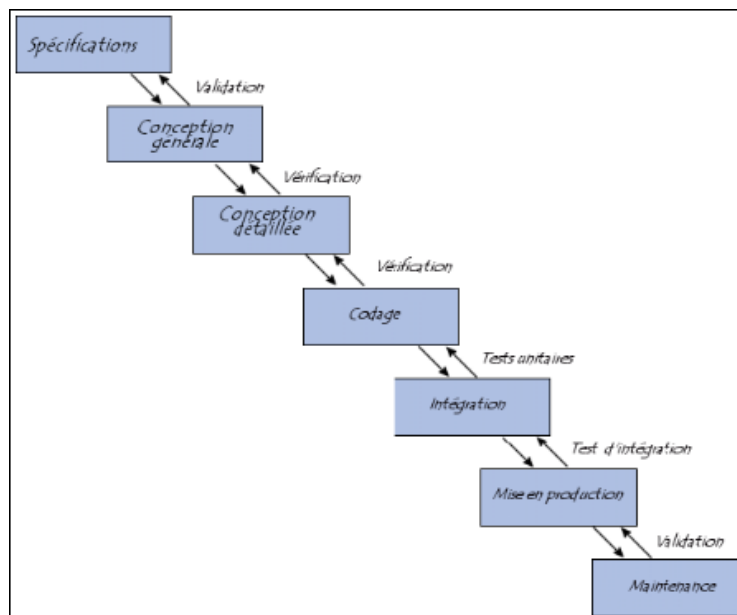


Figure 9 : Modèle en cascade ou Waterfall model

Ce chapitre nous aura permis d'adopter une approche analytique afin de concevoir la plate-forme de mise en relation des vendeurs et consommateurs avec systèmes de promotions intégrés en utilisant des diagrammes appropriés et certaines bonnes pratiques de développement.

CHAPITRE 3 : PRESENTATION DE LA SOLUTION

Dans ce chapitre 3, nous allons présenter la solution que nous avons proposée pour résoudre le problème posé. Nous détaillerons les différentes interfaces de la solution. Avant cela, nous introduirons les outils utilisés.

I. PRESENTATION DES LOGICIELS, LANGAGE DE PROGRAMMATION

I.1. OUTILS UTILISÉ

Afin de mener à bien notre travail, nous avons utilisé plusieurs outils qui nous ont assistés dans la modélisation, la programmation, les tests. Parmi eux, nous pouvons citer :

➤ Visual Studio Code (VS Code) :



VS Code est un éditeur de code source polyvalent utilisé pour écrire, modifier et déboguer du code. Il offre une multitude de fonctionnalités et d'extensions pour faciliter le développement logiciel dans différents langages de programmation. VS Code est largement utilisé par les développeurs en raison de sa légèreté, de sa grande flexibilité et de sa riche sélection d'extensions. Il facilite la productivité des développeurs en offrant des fonctionnalités telles que la complétion automatique du code, le débogage intégré, le contrôle de version Git et l'intégration avec d'autres outils de développement.

➤ Draw.io :



Draw.io est un outil de modélisation graphique utilisé pour créer des diagrammes UML, des organigrammes, des schémas de réseau et d'autres types de diagrammes. Il permet de concevoir visuellement l'architecture et les composants du système. Draw.io est particulièrement utile pour représenter les structures de données, les flux de travail et les interactions entre les différents éléments du système. Il aide à clarifier et à communiquer les concepts de conception avec l'équipe de développement.

➤ XAMPP :



XAMPP est un serveur web open source qui regroupe Apache, MySQL, PHP et Perl. Il facilite l'installation et la configuration d'un environnement de serveur web local pour le développement et les tests. XAMPP permet de créer et de tester des applications web dynamiques sur sa propre machine, sans nécessiter de connexion Internet. XAMPP est important dans le développement web car il fournit tous les composants essentiels pour exécuter des applications web basées sur PHP et MySQL. Il permet aux développeurs de créer et de tester des sites web localement avant de les déployer sur un serveur en production. Il est également utilisé pour l'apprentissage et la formation au développement web.

➤ **Postman :**



Postman est un outil de développement et de test d'API. Il permet aux développeurs de concevoir, de construire et de tester des API en envoyant des requêtes HTTP et en inspectant les réponses. Postman facilite l'exploration et la validation des API en fournissant une interface conviviale pour envoyer différentes requêtes et analyser les résultats. Postman est largement utilisé dans le développement d'API car il permet de tester facilement les routes et de vérifier la conformité des réponses. Il facilite également la documentation des API, la création de collections de routes, la réalisation de tests automatisés et la collaboration entre les membres d'une équipe de développement. Postman est essentiel pour assurer la qualité et la fiabilité des API dans le processus de développement logiciel.

➤ **Git :**



Git est un système de contrôle de version distribué qui permet de suivre et de gérer les modifications du code source au fil du temps. Il facilite le travail en équipe en permettant à plusieurs développeurs de collaborer sur un même projet, de fusionner leurs contributions et de revenir à des versions antérieures du code en cas d'erreur, garantissant ainsi une meilleure traçabilité et sécurité du développement.

➤ **Figma :**



Figma est un outil de conception d'interface utilisateur (UI) et de prototypage collaboratif. Il permet aux designers, développeurs et autres parties prenantes de créer des maquettes interactives directement depuis un navigateur, sans nécessiter d'installation locale.

Figma facilite la création, la modification et le partage de maquettes graphiques, tout en intégrant des fonctionnalités de collaboration en temps réel. Grâce à une interface intuitive et des outils puissants de design vectoriel, il permet de concevoir des interfaces modernes et réactives pour les applications web et mobiles.

I.2. LANGAGES DE PROGRAMMATION

Un langage de programmation est un ensemble de règles et de symboles utilisés pour écrire des instructions compréhensibles par un ordinateur. Il permet aux programmeurs de communiquer avec l'ordinateur en lui donnant des instructions précises pour exécuter des tâches spécifiques. Tout au long de notre travail, nous avons utilisé les langages de programmation suivants :

➤ **Langages natifs :**

- **JavaScript :**



Souvent abrégé JS, JavaScript est un langage de programmation de haut niveau, interprété et orienté objet. Il est principalement utilisé pour le développement web et offre des fonctionnalités pour rendre les sites web interactifs et dynamiques. Il est utilisé pour manipuler le contenu HTML, gérer les événements utilisateur, effectuer des requêtes AJAX pour récupérer des données du serveur, créer des animations, valider les formulaires, et bien plus encore. JavaScript peut également être utilisé côté serveur avec Node.js pour développer des applications. Il est largement utilisé dans les frameworks et bibliothèques populaires tels que React, Angular, Vue.js pour le front-end et avec diverses autres technologies côté serveur.

- **PHP :**



PHP (Hypertext Preprocessor) est un langage de script côté serveur conçu pour le développement web. Il est largement utilisé pour créer des pages web dynamiques et interagir

avec des bases de données. PHP est intégré dans le HTML et peut être utilisé pour gérer des formulaires, des sessions, des cookies, et effectuer des opérations complexes côté serveur. Il est compatible avec de nombreux systèmes de gestion de bases de données, ce qui en fait un choix populaire pour le développement web.



- **SQL (Structured Query Language) :**

SQL est un langage de requêtes utilisé pour communiquer avec les bases de données relationnelles. Il permet de créer, modifier et interroger des bases de données, de manipuler des données et d'effectuer des opérations telles que l'insertion, la mise à jour et la suppression de données. SQL est essentiel pour la manipulation et la gestion des données dans les applications qui utilisent des bases de données relationnelles.

➤ **Frameworks :**

Nous avons choisi d'utiliser deux frameworks, notamment React pour le frontend et Laravel pour le backend.



- **Laravel :**

Laravel est un framework PHP qui suit l'architecture MVC (Modèle-Vue-Contrôleur). Il offre des fonctionnalités telles que la gestion des requêtes et des réponses, un ORM intégré (Eloquent) pour interagir avec la base de données, des middlewares et des hooks pour personnaliser le flux de l'application. Avec une documentation complète et une communauté active, Laravel est un choix solide pour développer des applications web robustes et maintenables.

➤ **REACT**



React est un framework JavaScript dont le rôle est de faciliter la création d'interfaces utilisateur interactives et réactives pour les applications web. Développé par Facebook, il permet de construire des composants réutilisables qui se mettent à jour automatiquement en fonction des changements de données, sans recharger toute la page. React améliore ainsi les

performances, la modularité et la maintenabilité des applications, tout en offrant une expérience utilisateur fluide et dynamique.

➤ BOOTSTRAP



Bootstrap est un framework CSS dont le rôle est de faciliter et d'accélérer le développement d'interfaces web modernes et responsives grâce à une bibliothèque de composants HTML, CSS et JavaScript prêts à l'emploi. Créé par Twitter, ce framework permet de concevoir des sites adaptatifs qui s'affichent correctement sur tous les types d'écrans (ordinateurs, tablettes, smartphones) sans avoir à tout coder depuis zéro. Il offre une base cohérente pour structurer et styliser rapidement une page web avec un design professionnel, tout en assurant une compatibilité optimale entre les navigateurs.

➤ Base de données :



Nous utiliserons MySQL comme système de gestion de base de données en raison de sa compatibilité avec Laravel et de sa fiabilité dans la gestion des données.

II. PRESENTATION DES INTERFACES

1. Interface de connexion

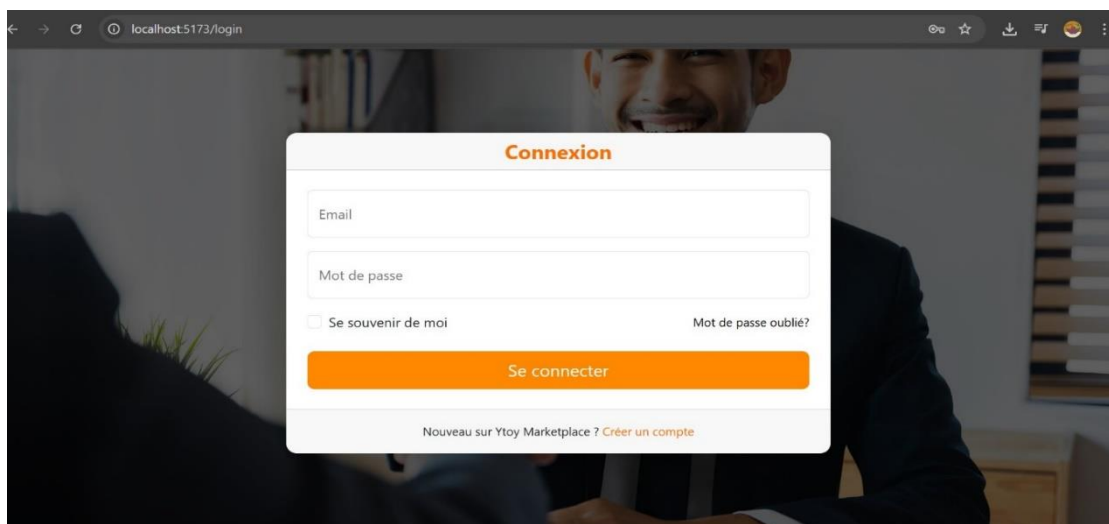


Figure 10 : page de connexion

Sur cette interface web, les utilisateurs (vendeurs, acheteurs, administrateurs) possédant déjà un compte peuvent se connecter directement en utilisant leur identifiant utilisateur et leur mot de passe. Les utilisateurs n'ayant pas encore de compte peuvent également accéder à cette interface pour naviguer vers l'écran de création de compte.

2. Interface principale ou page d'accueil

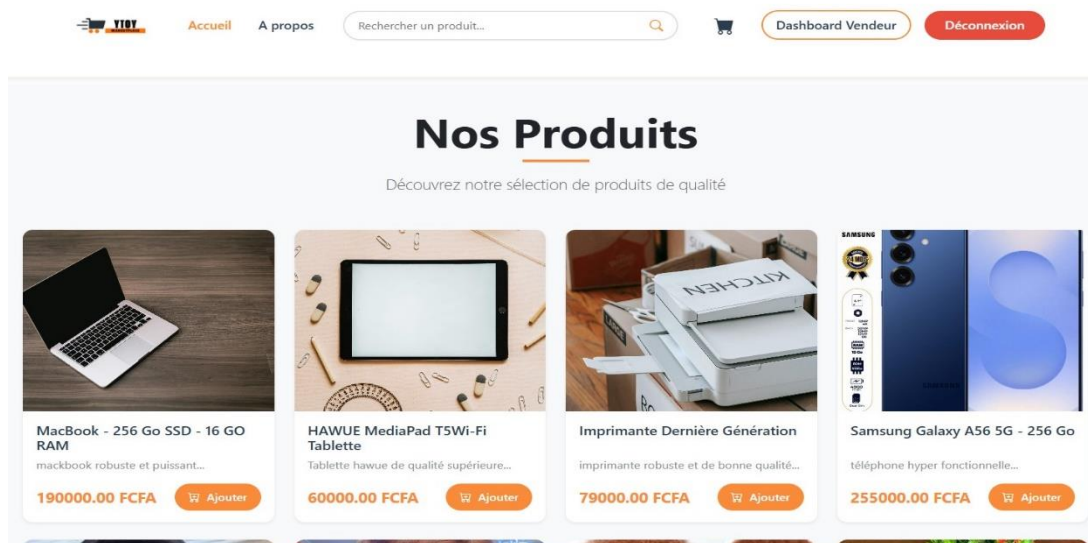


Figure 11 : interface principale ou page d'accueil

Sur cette interface web, les clients peuvent passer la commande d'un article et voir les détails de chaque article. Ils peuvent également consulter leurs paniers en temps réel et passer la commande.

3. Interface du panier

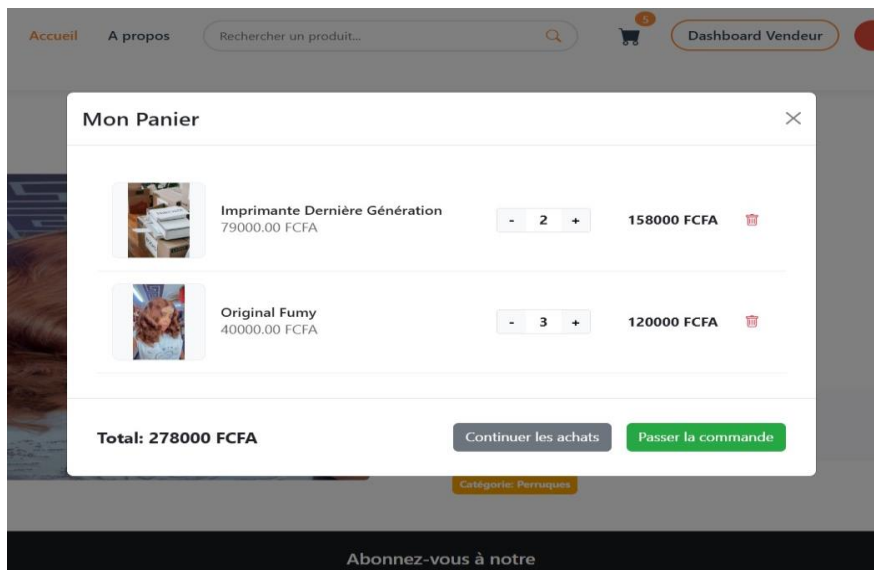


Figure 12 : interface du panier

Sur cette interface web, les clients peuvent visualiser leurs paniers, voir le total de leurs achats. Ils peuvent également passer la commande des articles qu'ils ont ajouté au panier.

4. Interface du Dashboard client

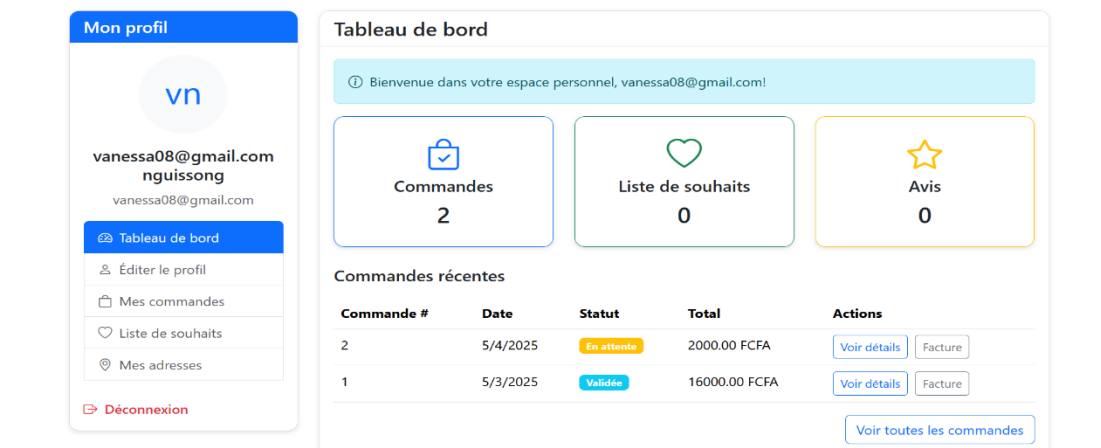


Figure 13 : Tableau de bord des clients

Sur cette interface web, les clients peuvent visualiser l'historique de leurs commandes, télécharger leurs factures au format PDF. Ils peuvent également éditer leurs profils.

5. Interface du Dashboard Vendeur

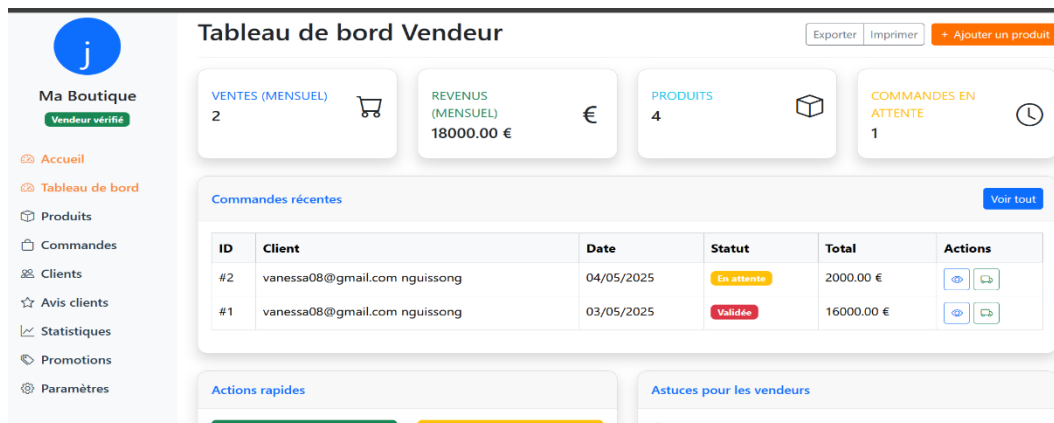


Figure 14 : Tableau de bord des vendeurs

Sur cette interface web, les vendeurs peuvent visualiser l'historique des commandes de leurs clients, ajouter des produits, voir ses ventes, ses revenus, lancer une campagne de promotion, voir ses clients, lancer une promotion.

6. Interface du Dashboard Administrateur

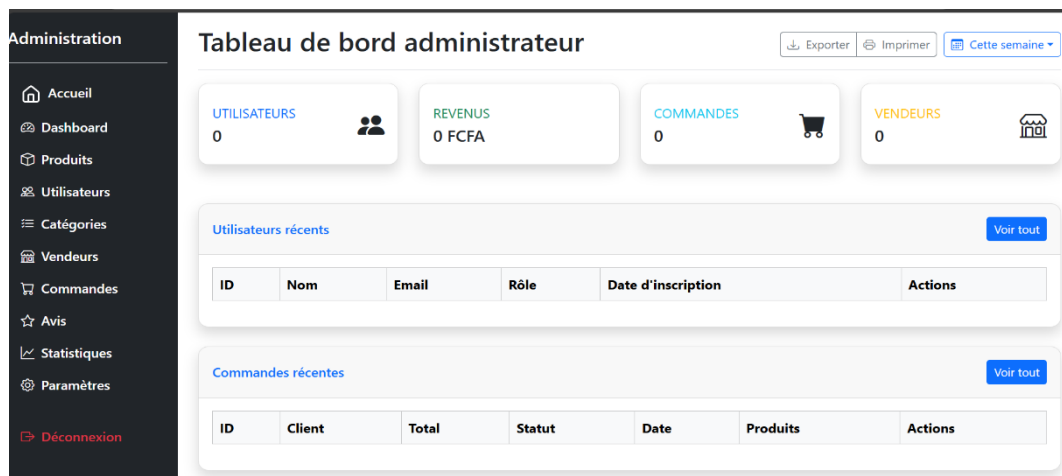
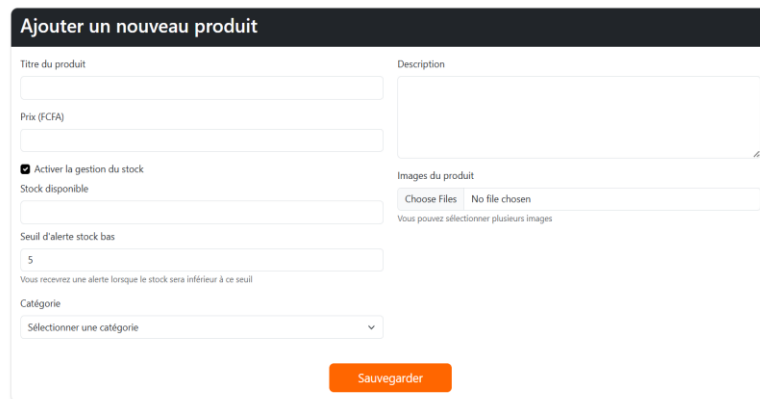


Figure 15 : Tableau de bord de l'administrateur

Sur cette interface web, l'administrateur peut gérer la plateforme notamment : créer les catégories, voir les produits créés par les vendeurs sur la plateforme, approuver les vendeurs ou rejeter les commandes de tous les clients.

7. Formulaire d'ajout des produits par les vendeurs



The screenshot shows a web form titled "Ajouter un nouveau produit" in a dark header. The form is divided into two main columns. The left column contains: a text input for "Titre du produit", a text input for "Prix (FCFA)", a checked checkbox for "Activer la gestion du stock", a text input for "Stock disponible", a text input for "Seuil d'alerte stock bas" with the value "5", and a dropdown menu for "Catégorie" with the placeholder "Sélectionner une catégorie". The right column contains: a large text area for "Description", a file upload section for "Images du produit" with a "Choose Files" button and "No file chosen" text, and a note "Vous pouvez sélectionner plusieurs images". An orange "Sauvegarder" button is at the bottom center.

Figure 16 : Formulaire d'ajout de produit

Sur ce formulaire le vendeur créer ses produits qui seront disponibles sur la plateforme, il peut également modifier les informations d'un produit depuis ce formulaire.

Pour conclure, ce chapitre nous a permis de présenter la solution que nous avons proposée pour résoudre le problème. Nous avons mis l'accent sur les différentes interfaces que nous avons développées et présenté les outils utilisés tout au long du processus, de la conception à la mise en œuvre, pour garantir l'efficacité et la qualité de notre solution.

CONCLUSION

Arrivés au terme de notre analyse, il a été question pour nous de présenter le contexte du projet ainsi que les enjeux identifiés dans le domaine du commerce électronique, en retraçant l'évolution des plateformes, depuis les systèmes traditionnels de vente physique avec interactions directes, jusqu'aux premières plateformes web où les commandes en ligne étaient traitées manuellement, pour arriver aux solutions actuelles intégrées capables de gérer automatiquement les transactions en temps réel grâce aux technologies web modernes et protocoles sécurisés. Cette évolution a apporté de nombreux avantages, notamment une meilleure flexibilité, une accessibilité accrue, une expérience utilisateur optimisée, ainsi que des fonctionnalités avancées telles que la personnalisation des recommandations, la gestion intelligente des promotions, la géolocalisation des services, l'analyse comportementale et la gestion centralisée des vendeurs et consommateurs. Nous avons ensuite réalisé une plateforme digitale de mise en relation, en commençant par l'analyse des besoins et l'identification des fonctionnalités clés, suivies du choix des technologies adaptées, de la conception de l'architecture, puis de l'implémentation d'un système de promotion intégré. En définitive, une plateforme bien conçue et optimisée offre une navigation fluide, une promotion ciblée des produits, une expérience utilisateur de qualité, tout en garantissant la sécurité des transactions et la protection des données personnelles, ce qui constitue un levier puissant pour améliorer les interactions commerciales, faciliter les échanges et contribuer à la croissance économique ainsi qu'à la satisfaction des acteurs du marché.

REFERENCES BIBLIOGRAPHIQUES

- Rapport de stage.

NGNOUZOU PA SA'A DE MEZZI Rodrigue (2023/2024). VULGARISATION DES PAIEMENTS MARCHANDS DANS LE SECTEUR INFORMEL : CAS DE MAMONI FINANCE, rapport de stage Licence, IUT Douala, 57p.

- **Rehaïem Abdelmoumen**, "Conception, réalisation et d'une application de gestion des emplois de temps mobile", Mémoire de master, Université Kasdi Merbah Ouargla, 2016.
- <https://www.youtube.com/playlist?list=PLMS9Cy4Enq5LUTdetDXnCwf8Q2j0xeMGD> : Graven tuto
- <https://www.udemy.com/course/laravel-9-construire-4-projets-professionnels-de-a-a-z/learn/lecture/31126462?start=0#overview> : projet laravel
- <https://laracasts.com/series> : Apprenons Laravel
- <https://www.definitions.net/definition/XAMPP>
- <https://laravel.com/docs/11.x/readme> : Documentation de Laravel
- www.stackoverflow.com
- <https://chatgpt.com>