

République Tunisienne

\*\*\*\*\*

**ESIP-GAFSA**

\*\*\*\*\*



Réf. : 202...../N° .....

Etablissement d'Enseignement  
Supérieur Privé Agréé par  
l'Etat sous N° 05-2013

**Ecole Supérieure Privée d'Ingénieurs de Gafsa**  
**ESIP-Gafsa**

# **RAPPORT DE STAGE**

**Spécialité : Génie Informatique**

Réalisé Par

**Ismail Mabrouki**

---

## **STAGE TECHNICIEN**

---

*Du : 01/06/2024 Au 01/07/2024*

**Encadrant professionnel : Ahmed Neffati**

**Organisme : Beecoders**



**Année Universitaire : 2023/2024**

# REMERCIEMENTS

*Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réussite de mon projet de stage technicien. Tout d'abord, j'adresse mes sincères remerciements à mon encadreur, **Mr. Ahmed Neffati**, ainsi qu'à **Mr. Mohamed Aziz Charrada**, pour leur accueil chaleureux, le temps précieux qu'ils ont consacré à mon encadrement, et le partage généreux de leurs expertises. Leur confiance en moi a été un moteur essentiel pour mon épanouissement dans la réalisation de ce projet. Ils ont été d'une aide précieuse, surtout dans les moments les plus délicats. Je voudrais également exprimer mon profond respect envers ceux qui ont bien voulu consulter ce rapport. Je les remercie chaleureusement d'avoir pris le temps de s'intéresser à mon travail, d'apporter des critiques constructives et de partager leurs précieux conseils. Leurs retours ont contribué à enrichir mon expérience et à améliorer ce rapport.*

*Enfin, je tiens à exprimer ma reconnaissance envers toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce rapport de stage technicien. Merci à tous pour votre soutien et votre contribution à la réussite de ce projet.*

# Table des matières

|  |           |
|--|-----------|
| <b>Introduction Générale</b>                       | <b>9</b>  |
| <b>1 Contexte général</b>                          | <b>11</b> |
| 1.1 Introduction . . . . .                         | 11        |
| 1.2 Présentation de la société d'accueil . . . . . | 11        |
| 1.3 Présentation du Projet . . . . .               | 12        |
| 1.3.1 Problématique . . . . .                      | 12        |
| 1.3.2 Travail demandé . . . . .                    | 12        |
| 1.4 Etude préalable . . . . .                      | 13        |
| 1.4.1 Etude de l'existant . . . . .                | 13        |
| 1.4.2 Critique de l'existant . . . . .             | 14        |
| 1.4.3 Solution Proposée . . . . .                  | 17        |
| 1.5 Méthodologie de travail . . . . .              | 18        |
| 1.5.1 Choix méthodologique . . . . .               | 18        |
| 1.5.2 Pourquoi Scrum . . . . .                     | 18        |
| 1.6 Langage de conception . . . . .                | 20        |
| 1.7 Conclusion . . . . .                           | 21        |
| <b>2 Analyse et planification du projet</b>        | <b>22</b> |
| 2.1 Introduction . . . . .                         | 22        |
| 2.2 Spécification des besoins . . . . .            | 22        |
| 2.2.1 Identification des acteurs . . . . .         | 22        |
| 2.2.2 Besoins fonctionnels . . . . .               | 23        |
| 2.2.3 Besoins non fonctionnels . . . . .           | 24        |
| 2.2.4 Maintenabilité et extensibilité . . . . .    | 25        |
| 2.3 Backlog du produit . . . . .                   | 25        |

---

|          |   |           |
|----------|---|-----------|
| 2.4      | Identification des sprints . . . . .                        | 27        |
| 2.4.1    | Sprint 1 (Jours 1 à 7) . . . . .                            | 27        |
| 2.4.2    | Sprint 2 (Jours 8 à 14) . . . . .                           | 27        |
| 2.4.3    | Sprint 3 (Jours 15 à 21) . . . . .                          | 27        |
| 2.4.4    | Sprint 4 (Jours 22 à 28) . . . . .                          | 28        |
| 2.5      | Diagrammes des cas d'utilisation . . . . .                  | 29        |
| 2.6      | Diagramme des classes . . . . .                             | 29        |
| 2.7      | Architecture du système . . . . .                           | 31        |
| 2.7.1    | Architecture globale . . . . .                              | 31        |
| 2.7.2    | Architecture logique . . . . .                              | 33        |
| 2.8      | Environnement de développement . . . . .                    | 35        |
| 2.8.1    | Environnement matériel . . . . .                            | 35        |
| 2.8.2    | Environnement logiciel . . . . .                            | 35        |
| 2.9      | Choix des logiciels . . . . .                               | 35        |
| 2.9.1    | Choix : Java Spring Boot . . . . .                          | 35        |
| 2.9.2    | Choix : Angular . . . . .                                   | 36        |
| 2.9.3    | Choix : MySQL . . . . .                                     | 37        |
| 2.9.4    | Choix : Docker (Maildev) . . . . .                          | 38        |
| 2.9.5    | Choix : Visual Studio Code (VS Code) . . . . .              | 39        |
| 2.9.6    | Choix : IntelliJ IDEA Community . . . . .                   | 40        |
| 2.9.7    | Choix : Git . . . . .                                       | 41        |
| <b>3</b> | <b>Sprint 1</b>   | <b>43</b> |
| 3.1      | Spécification et analyse des besoins . . . . .              | 43        |
| 3.1.1    | Backlog du Sprint 1 . . . . .                               | 44        |
| 3.1.2    | Diagramme de Cas d'Utilisation Raffiné : Sprint 1 . . . . . | 44        |
| 3.2      | Description détaillée des cas d'utilisation . . . . .       | 45        |
| 3.3      | Diagrammes de séquence . . . . .                            | 46        |
| 3.3.1    | Diagramme de séquences " S'inscrire " . . . . .             | 46        |
| 3.3.2    | Diagramme de séquences " Se Connecter " . . . . .           | 47        |
| 3.4      | Réalisation . . . . .                                       | 48        |
| <b>4</b> | <b>Sprint 2</b>   | <b>53</b> |
| 4.0.1    | Backlog du Sprint 2 . . . . .                               | 54        |
| 4.0.2    | Diagramme de Cas d'Utilisation Raffiné : Sprint 2 . . . . . | 54        |

---

|          |  |           |
|----------|--|-----------|
| 4.1      | Description détaillée des cas d'utilisation . . . . .          | 55        |
| 4.2      | Diagrammes de séquence . . . . .                               | 55        |
| 4.2.1    | Diagramme de séquence "Gérer les commandes" . . . . .          | 56        |
| 4.3      | Réalisation . . . . .  | 57        |
| <b>5</b> | <b>Sprint 3</b>  | <b>60</b> |
| 5.1      | Spécification et analyse des besoins . . . . .                 | 60        |
| 5.1.1    | Backlog du Sprint 3 . . . . .                                  | 61        |
| 5.1.2    | Diagramme de Cas d'Utilisation Raffiné : Sprint 3 . . . . .    | 61        |
| 5.2      | Description détaillée des cas d'utilisation . . . . .          | 62        |
| 5.3      | Diagrammes de séquence . . . . .                               | 63        |
| 5.3.1    | Diagramme de séquence "Gérer les avis" . . . . .               | 63        |
| 5.3.2    | Diagramme de séquence "Paiement sécurisé" . . . . .            | 65        |
| 5.3.3    | Diagramme de séquence "Recherche de produits" . . . . .        | 66        |
| 5.4      | Réalisation . . . . .  | 66        |
| <b>6</b> | <b>Sprint 4</b>  | <b>73</b> |
| 6.1      | Spécification et analyse des besoins . . . . .                 | 73        |
| 6.1.1    | Backlog du Sprint 4 . . . . .                                  | 74        |
| 6.1.2    | Diagramme de Cas d'Utilisation Raffiné : Sprint 4 . . . . .    | 74        |
| 6.2      | Description détaillée des cas d'utilisation . . . . .          | 75        |
| 6.3      | Diagrammes de séquence . . . . .                               | 76        |
| 6.3.1    | Diagramme de séquence "Filtrer les produits" . . . . .         | 76        |
| 6.3.2    | Diagramme de séquence "Voir l'historique d'activité" . . . . . | 77        |
| 6.4      | Réalisation . . . . .  | 79        |
|          | <b>Conclusion générale</b>                                     | <b>85</b> |

# Liste des tableaux

|     |   |    |
|-----|---|----|
| 2.1 | Backlog du produit de TunisiCart . . . . .                              | 26 |
| 3.1 | Backlog du Sprint 1 basé sur les User Stories pour le projet TunisiCart | 44 |
| 3.2 | Cas d'utilisation : S'Inscrire . . . . .                                | 45 |
| 3.3 | Cas d'utilisation : Se Connecter . . . . .                              | 45 |
| 3.4 | Cas d'utilisation : Gérer son Profil . . . . .                          | 46 |
| 4.1 | Backlog du Sprint 2 basé sur les User Stories pour le projet TunisiCart | 54 |
| 4.2 | Cas d'utilisation : Ajouter des produits . . . . .                      | 55 |
| 4.3 | Cas d'utilisation : Gérer les commandes . . . . .                       | 55 |
| 5.1 | Backlog du Sprint 3 basé sur les User Stories pour le projet TunisiCart | 61 |
| 5.2 | Cas d'utilisation : Gérer les avis . . . . .                            | 62 |
| 5.3 | Cas d'utilisation : Paiement sécurisé . . . . .                         | 62 |
| 5.4 | Cas d'utilisation : Recherche de produits . . . . .                     | 63 |
| 6.1 | Backlog du Sprint 4 basé sur les User Stories pour le projet TunisiCart | 74 |
| 6.2 | Cas d'utilisation : Filtrer les produits . . . . .                      | 75 |
| 6.3 | Cas d'utilisation : Trier les produits . . . . .                        | 75 |
| 6.4 | Cas d'utilisation : Voir l'historique d'activité . . . . .              | 76 |

# Table des figures

|      |   |    |
|------|---|----|
| 1.1  | Logo of Beecoders . . . . .   | 11 |
| 1.2  | représente le processus Scrum et ses principales phases. . . . .        | 19 |
| 1.3  | QR code Github : Tunicart . . . . .                                     | 21 |
| 2.1  | Diagramme de cas d'utilisation global . . . . .                         | 29 |
| 2.2  | figure diagramme de classe . . . . .                                    | 30 |
| 2.3  | figure architecture multicouches . . . . .                              | 32 |
| 2.4  | ModeleMVC . . . . .   | 34 |
| 2.5  | Logo de java Spring boot . . . . .                                      | 36 |
| 2.6  | Logo d'Angular . . . . .  | 37 |
| 2.7  | Logo de mySQL . . . . .   | 38 |
| 2.8  | maildev/maildev - Docker Image . . . . .                                | 39 |
| 2.9  | VsCode Logo . . . . .   | 40 |
| 2.10 | Intellij Logo . . . . .   | 41 |
| 2.11 | Git Logo . . . . .  | 42 |
| 3.1  | Diagramme de Cas d'Utilisation Raffiné pour le Sprint 1 . . . . .       | 44 |
| 3.2  | Diagramme de séquence pour UC1 : Inscription d'un utilisateur . . . . . | 47 |
| 3.3  | Diagramme de séquence pour UC2 : Connexion d'un utilisateur . . . . .   | 48 |
| 3.4  | Interface d'inscription . . . . .                                       | 49 |
| 3.5  | Interface d'authentification . . . . .                                  | 50 |
| 3.6  | Interface de profil utilisateur . . . . .                               | 51 |
| 3.7  | Interface de mise à jour de profil utilisateur . . . . .                | 52 |
| 4.1  | Diagramme de Cas d'Utilisation Raffiné pour le Sprint 2 . . . . .       | 54 |
| 4.2  | Diagramme de séquence pour UC4 : Gérer les commandes . . . . .          | 56 |
| 4.3  | Interface d'ajout de produits . . . . .                                 | 57 |

---

|      |  |    |
|------|--|----|
| 4.4  | Interface de produit Ajouté . . . . .                                  | 58 |
| 4.5  | Interface de gestion des commandes . . . . .                           | 59 |
| 5.1  | Diagramme de Cas d'Utilisation Raffiné pour le Sprint 3 . . . . .      | 61 |
| 5.2  | Diagramme de séquence pour UC7 : Gérer les avis . . . . .              | 64 |
| 5.3  | Diagramme de séquence pour UC8 : Paiement sécurisé . . . . .           | 65 |
| 5.4  | Diagramme de séquence pour UC9 : Laisser un avis sur un produit . . .  | 66 |
| 5.5  | Interface d'ajout des avis . . . . .                                   | 67 |
| 5.6  | Interface de gestion des avis . . . . .                                | 68 |
| 5.7  | Interface de cart . . . . .  | 69 |
| 5.8  | Interface de checkout . . . . .  | 69 |
| 5.9  | Interface de paiement sécurisé . . . . .                               | 70 |
| 5.10 | Interface de stripe dashboard . . . . .                                | 70 |
| 5.11 | Interface de recherche de produits 1 . . . . .                         | 71 |
| 5.12 | Interface de recherche de produits 2 . . . . .                         | 72 |
| 6.1  | Diagramme de Cas d'Utilisation Raffiné pour le Sprint 4 . . . . .      | 74 |
| 6.2  | Diagramme de séquence pour UC10 : Filtrer les produits" . . . . .      | 77 |
| 6.3  | Diagramme de séquence pour UC11 : Voir l'historique d'activité . . . . | 78 |
| 6.4  | Interface de filtres 1 . . . . .                                       | 79 |
| 6.5  | Interface de filtres 2 . . . . .                                       | 80 |
| 6.6  | Interface de filtres 3 . . . . .                                       | 80 |
| 6.7  | Interface de tri 1 . . . . .   | 81 |
| 6.8  | Interface de tri 2 . . . . .   | 82 |
| 6.9  | Interface de tri 3 . . . . .   | 83 |
| 6.10 | Interface d'historique des activités : Authentification . . . . .      | 84 |
| 6.11 | Interface d'historique des activités : Achat . . . . .                 | 84 |



# Introduction Générale

Le commerce en ligne connaît une croissance exponentielle, devenant une composante essentielle de l'économie moderne. Notre projet, Tunicart, se concentre sur le développement d'une plateforme e-commerce innovante, conçue pour faciliter l'achat et la vente de produits en ligne, tout en offrant une expérience utilisateur fluide et sécurisée.

Tunicart vise à connecter les vendeurs et les acheteurs sur une seule plateforme, leur offrant des fonctionnalités avancées pour gérer leurs transactions, des outils de marketing intégrés, et des options de paiement variées. De plus, notre plateforme permet aux vendeurs de présenter leurs produits de manière attrayante, tandis que les acheteurs peuvent parcourir, comparer et acheter des articles avec facilité.

Les utilisateurs bénéficient également de recommandations personnalisées basées sur leurs préférences et historiques d'achats, grâce à un moteur de recherche intelligent et un système de filtrage efficace. Tunicart propose une interface intuitive où chaque utilisateur peut gérer son profil, suivre ses commandes, et interagir avec le support client en cas de besoin.

Ce rapport présente les étapes de mise en œuvre de notre projet, organisées en sept chapitres :

1. **Contexte général :** Le premier chapitre introduit le projet Tunicart, en exposant le contexte général, les objectifs, et les enjeux auxquels répond notre solution. Nous y abordons également la méthodologie adoptée pour la réalisation du projet.
2. **Analyse des besoins :** Le deuxième chapitre se concentre sur l'analyse détaillée des besoins fonctionnels et non fonctionnels, les acteurs du système, ainsi que les choix technologiques et architecturaux. Des diagrammes de cas d'utilisation et de classe y sont également présentés.
3. **Sprint 1 : Gestion des utilisateurs :** Le troisième chapitre est consacré à la première phase de développement, qui porte sur la gestion des utilisateurs. Cela inclut la création de comptes, la gestion des profils et les fonctionnalités de connexion.
4. **Sprint 2 : Ajout de produits et gestion des catégories :** Le quatrième chapitre détaille la deuxième phase de développement, qui consiste en l'ajout de produits, la gestion des catégories, ainsi que la gestion des stocks pour permettre aux vendeurs d'ajouter et de modifier leurs offres.

- 
5. **Sprint 3 : Gestion des avis et paiements sécurisés :** Le cinquième chapitre présente la troisième phase de développement. Celle-ci inclut la gestion des avis clients, ainsi que la mise en place d'un système de paiement sécurisé pour finaliser les transactions.
  6. **Sprint 4 : Recherche de produits et recommandations :** Enfin, le sixième chapitre aborde la dernière phase de développement, qui se concentre sur la recherche de produits par les utilisateurs ainsi que sur les recommandations personnalisées en fonction de leurs préférences et interactions sur la plateforme.

Ces défis font obstacle à la croissance du secteur et nuisent à la compétitivité de l'économie tunisienne. Face à ces défis, il est nécessaire de mettre en place des solutions innovantes et efficaces pour simplifier et digitaliser le transport et la logistique.

# Premier Chapitre

## Contexte général

### 1.1 Introduction

Dans ce premier chapitre, nous nous focaliserons sur une présentation approfondie du contexte de notre projet Tunicart. Nous explorerons en détail les divers éléments qui entourent notre initiative, notamment les objectifs, l'analyse de la situation actuelle, la proposition de solution, ainsi que l'approche méthodologique que nous adopterons.

### 1.2 Présentation de la société d'accueil

Beecoders est une entreprise spécialisée dans le domaine du développement web et mobile. Nous mettons à votre disposition des services de conseil visant à améliorer les performances de votre entreprise dans le domaine de l'informatique. En outre, nous proposons des formations en ligne pour renforcer les compétences de vos collaborateurs.



**Figure 1.1** – Logo of Beecoders

## Historique

Bee coders, fondée en 2020, est une entreprise de services du numérique spécialisée dans le développement web et mobile (Android et IOS), le design web, le consulting IT et les formations à distance. [2]

## 1.3 Présentation du Projet

Dans cette section, nous aborderons la problématique de notre projet et donnons une description détaillée du travail qui nous est demandé.

### 1.3.1 Problématique

Le marché tunisien de la vente en ligne est marqué par une fragmentation des offres, rendant difficile pour les consommateurs de trouver et comparer des produits provenant de divers vendeurs en un seul endroit. Cette dispersion complique l'expérience d'achat en ligne, entraînant une perte de temps et une frustration pour les utilisateurs.

De plus, les petites entreprises locales peinent à accéder à des plateformes de vente en ligne qui leur permettent de toucher une large audience. Cela limite leur visibilité et leur capacité à concurrencer les grandes enseignes, réduisant ainsi leur part de marché et leur croissance potentielle.

Enfin, l'absence de solutions intégrées pour gérer les aspects logistiques, tels que les paiements sécurisés, la gestion des stocks, et les livraisons, représente un obstacle majeur pour les vendeurs souhaitant se lancer dans le commerce en ligne. Ces défis logistiques peuvent entraîner des erreurs, des retards, et une insatisfaction des clients, compromettant ainsi la réputation des vendeurs.

TunisiCart vise à résoudre ces problématiques en offrant une plateforme centralisée qui rassemble des vendeurs diversifiés, permet une gestion simplifiée des opérations de vente, et facilite l'accès des petites entreprises au marché en ligne tunisien. Grâce à cette solution, TunisiCart ambitionne de transformer l'expérience d'achat en ligne en Tunisie, tout en soutenant la croissance des entreprises locales.

### 1.3.2 Travail demandé

Pour répondre aux problématiques identifiées, notre projet se concentre sur la création et l'optimisation de la plateforme TunisiCart. Nos objectifs seront :

- **Mettre en place une plateforme multi-seller centralisée** : L'objectif principal de notre projet est de créer une plateforme qui rassemble divers vendeurs tunisiens, permettant ainsi aux consommateurs de comparer et d'acheter des produits de différentes boutiques en un seul endroit. Cette plateforme simplifiera l'expérience d'achat en ligne en offrant un accès facile à une variété de produits.

- **Soutenir les petites entreprises locales** : TunisiCart vise à faciliter l'accès des petites entreprises au marché en ligne en leur offrant une vitrine numérique. Nous souhaitons créer un espace où les petites entreprises peuvent augmenter leur visibilité et atteindre une clientèle plus large, tout en réduisant les obstacles liés à la technologie et à la logistique.
- **Intégrer des solutions logistiques efficaces** : Nous mettrons en place des outils pour aider les vendeurs à gérer leurs stocks, suivre les livraisons, et assurer des paiements sécurisés. Ces solutions seront conçues pour améliorer l'efficacité opérationnelle des vendeurs et garantir la satisfaction des clients.
- **Assurer la croissance et l'amélioration continue de la plateforme** : Après le lancement de TunisiCart, nous suivrons de près les performances de la plateforme et recueillerons les retours des utilisateurs pour identifier les domaines à améliorer. Nous nous engagerons à faire évoluer la plateforme en fonction des besoins du marché et des nouvelles tendances du e-commerce.

## 1.4 Etude préalable

Dans cette section, nous allons examiner les différentes solutions d'apprentissage en ligne actuellement proposées sur le marché. Ensuite, nous analyserons les défis et les limitations auxquels elles sont confrontées.

### 1.4.1 Etude de l'existant

#### Plateformes Multi-Seller en Tunisie

- **Jumia Tunisie** :
  - **Description** : Jumia est un géant du e-commerce en Afrique avec une présence notable en Tunisie. C'est une plateforme multi-seller qui permet à divers vendeurs de proposer leurs produits sur le site.
  - **Caractéristiques** : Large éventail de produits, intégration de solutions logistiques et de paiement, interface conviviale.
- **Tunisianet** :
  - **Description** : Tunisianet est une plateforme de vente en ligne tunisienne qui propose une variété de produits, de l'électronique aux vêtements.
  - **Caractéristiques** : Permet la vente par plusieurs fournisseurs, offre des options de livraison locales.
- **KechExpress** :
  - **Description** : Bien que plus petite, KechExpress est une autre plateforme de commerce électronique qui permet à plusieurs vendeurs de proposer leurs produits.
  - **Caractéristiques** : Focalisée sur la livraison rapide et le service client.

## Plateformes Multi-Seller Internationales

- **Amazon :**
  - **Description :** Amazon est l'une des plus grandes plateformes de commerce électronique multi-seller au monde. Elle permet à des millions de vendeurs de proposer leurs produits à une audience mondiale.
  - **Caractéristiques :** Infrastructure logistique massive, options de publicité, services de fulfilment, support client étendu.
- **eBay :**
  - **Description :** eBay est une plateforme de vente aux enchères et de commerce électronique qui permet aux vendeurs de lister leurs produits dans diverses catégories.
  - **Caractéristiques :** Vente aux enchères, options de mise en vente immédiate, système de feedback pour évaluer les vendeurs.
- **Etsy :**
  - **Description :** Etsy est une plateforme spécialisée dans les produits faits main, vintage, et créatifs. Elle permet aux artisans et créateurs de vendre leurs produits.
  - **Caractéristiques :** Ciblée sur les produits artisanaux et uniques, outils de personnalisation pour les boutiques, communauté de créateurs.
- **Alibaba / AliExpress :**
  - **Description :** Alibaba et AliExpress sont des plateformes de commerce électronique basées en Chine, offrant un vaste marché pour les vendeurs du monde entier.
  - **Caractéristiques :** Focus sur les achats en gros pour Alibaba et les achats au détail pour AliExpress, options de livraison internationale, outils pour les vendeurs.
- **Rakuten :**
  - **Description :** Rakuten est une plateforme de commerce électronique japonaise avec une présence internationale, offrant une large gamme de produits.
  - **Caractéristiques :** Intégration de services financiers, programme de fidélité, outils pour les vendeurs.

### 1.4.2 Critique de l'existant

L'analyse critique des plateformes existantes nous permet de comprendre leurs points forts et leurs faiblesses, ainsi que les opportunités d'amélioration pour notre propre projet. Nous avons examiné plusieurs plateformes de commerce électronique multi-seller, tant locales qu'internationales, pour identifier les aspects à améliorer.

## Plateformes Multi-Seller en Tunisie

- **Jumia Tunisie**
  - **Points forts :**
    - Large éventail de produits disponibles.
    - Infrastructure logistique établie avec des options de livraison efficaces.
    - Interface utilisateur conviviale.
  - **Faiblesses :**
    - Limitations dans le support pour les petites entreprises locales.
    - Problèmes de disponibilité de certains produits.
    - Manque de personnalisation des options de recherche.
- **Tunisianet**
  - **Points forts :**
    - Bonne couverture des besoins locaux en produits divers.
    - Support pour les vendeurs locaux.
    - Options de livraison adaptées au marché tunisien.
  - **Faiblesses :**
    - Interface utilisateur moins moderne.
    - Moins d'options pour le suivi des commandes et la gestion des stocks.
    - Problèmes de service client et de gestion des retours.
- **KechExpress**
  - **Points forts :**
    - Concentration sur la livraison rapide.
    - Service client réactif.
  - **Faiblesses :**
    - Catalogue de produits limité.
    - Manque de fonctionnalités avancées pour les vendeurs.
    - Absence de solutions intégrées pour la gestion des paiements.

## Plateformes Multi-Seller Internationales

- **Amazon**
  - **Points forts :**
    - Large réseau logistique et choix de produits.
    - Outils avancés pour les vendeurs, y compris le fulfilment.
    - Plateforme bien établie avec une grande audience mondiale.
  - **Faiblesses :**

- Concurrence féroce entre les vendeurs.
- Frais élevés pour l'utilisation des services de fulfilment.
- Complexité dans la gestion des comptes de vendeur.
- **eBay**
  - **Points forts :**
    - Flexibilité avec les enchères et les ventes immédiates.
    - Système de feedback détaillé pour les acheteurs et les vendeurs.
    - Variété de produits et catégories.
  - **Faiblesses :**
    - Interface utilisateur vieillissante.
    - Moins de support pour les vendeurs internationaux.
    - Complexité dans la gestion des enchères et des paiements.
- **Etsy**
  - **Points forts :**
    - Spécialisation dans les produits artisanaux et uniques.
    - Communauté active de créateurs.
    - Outils de personnalisation pour les boutiques.
  - **Faiblesses :**
    - Limité aux produits faits main et vintage.
    - Moins d'options pour la gestion des stocks et des livraisons.
    - Concurrence entre les créateurs sur une plateforme similaire.
- **Alibaba / AliExpress**
  - **Points forts :**
    - Vaste marché pour les achats en gros et au détail.
    - Options de livraison internationale.
    - Plateforme établie avec une large base d'utilisateurs.
  - **Faiblesses :**
    - Problèmes potentiels de qualité des produits.
    - Délais de livraison plus longs pour les commandes internationales.
    - Moins de personnalisation pour les petites entreprises.
- **Rakuten**
  - **Points forts :**
    - Programme de fidélité attractif.
    - Large gamme de produits.
    - Services financiers intégrés.
  - **Faiblesses :**
    - Moins connu en dehors du Japon.
    - Moins d'options pour les vendeurs internationaux.
    - Interface utilisateur moins intuitive.



### 1.4.3 Solution Proposée

Afin de répondre efficacement aux défis identifiés lors de l'analyse des besoins et d'apporter une valeur ajoutée significative, notre proposition consiste à développer une plateforme de commerce en ligne innovante, TunisiCart, qui répondra à plusieurs besoins clés du marché. Voici les principaux aspects de notre solution :

1. **Plateforme Multi-Vendeurs** : TunisiCart sera conçue pour permettre à plusieurs vendeurs de proposer leurs produits sur une seule plateforme, offrant ainsi une variété étendue de produits aux consommateurs. Cette fonctionnalité favorisera la diversification des offres et améliorera l'accessibilité pour les acheteurs.
2. **Interface Intuitive** : Nous développerons une interface utilisateur moderne et facile à utiliser, permettant une navigation fluide entre les différents produits et vendeurs. Les fonctionnalités de filtrage avancé, les options de tri, et la recherche intuitive garantiront une expérience d'achat agréable.
3. **Système de Gestion des Commandes** : La plateforme inclura un système robuste de gestion des commandes, facilitant le traitement des achats, le suivi des expéditions, et la gestion des retours. Les vendeurs pourront gérer leurs stocks et suivre leurs ventes en temps réel.
4. **Support et Feedback** : TunisiCart intégrera un système de feedback et d'évaluation pour que les acheteurs puissent évaluer leurs expériences avec les produits et les vendeurs. Ce système aidera à maintenir la qualité des produits et à améliorer le service client.
5. **Sécurité et Confidentialité** : Nous garantirons un niveau élevé de sécurité pour protéger les données des utilisateurs et les transactions financières. Des mesures de sécurité robustes seront mises en place pour prévenir les fraudes et les violations de données.
6. **Intégration de Paiement** : La plateforme inclura des options de paiement sécurisées et variées pour faciliter les transactions. Les utilisateurs pourront choisir parmi plusieurs méthodes de paiement, y compris les cartes bancaires, les paiements électroniques et les virements bancaires.
7. **Personnalisation des Offres** : TunisiCart proposera des fonctionnalités de personnalisation pour offrir des recommandations de produits basées sur les préférences et les comportements d'achat des utilisateurs. Les offres et promotions seront également adaptées aux besoins individuels des clients.
8. **Support aux Vendeurs** : Nous offrirons des outils et des ressources pour aider les vendeurs à optimiser leur présence en ligne et à maximiser leurs ventes. Des options de formation et de support technique seront disponibles pour les aider à gérer efficacement leurs boutiques.

Cette approche intégrée vise à offrir une plateforme de commerce en ligne complète et efficace qui répond aux besoins des consommateurs et des vendeurs, tout en contribuant au développement du marché local et en offrant une expérience utilisateur enrichissante.

## 1.5 Méthodologie de travail

L'une des étapes les plus importantes pour mener à bien un travail est de choisir la bonne façon de procéder. C'est pourquoi nous considérons le choix de la méthodologie de travail la mieux adaptée à notre travail comme une étape importante pour assurer une bonne organisation des tâches.

### 1.5.1 Choix méthodologique

Le choix de la méthodologie pour un projet dépend de divers facteurs, notamment la nature et l'envergure du projet. Les objectifs principaux de la gestion de projet sont d'améliorer la satisfaction des clients tout en simplifiant le processus de développement.

On distingue généralement deux grandes approches de gestion de projet :

#### Méthodes classiques

Ces méthodes sont caractérisées par leur prévisibilité. Elles suivent des étapes pré-établies du cycle de vie du développement logiciel, avec une planification rigide et une documentation abondante, comme le cycle en cascade. Elles sont également marquées par une absence de communication continue avec le client et une difficulté à intégrer de nouveaux besoins une fois le projet en cours.

#### Méthodes agiles

Les méthodes agiles regroupent des pratiques variées en développement logiciel. Contrairement aux approches traditionnelles, elles sont conçues pour être flexibles et adaptées. Des méthodes telles que SCRUM et XP favorisent la collaboration avec le client, permettant des ajustements tout au long du projet. Elles offrent une meilleure visibilité en tenant compte des besoins et des priorités du client, tout en facilitant l'acceptation des changements.

Pour notre projet, nous avons choisi une méthode du type AGILE, plus particulièrement SCRUM.

### 1.5.2 Pourquoi Scrum

Parmi les méthodes agiles, Scrum est particulièrement populaire. Contrairement à XP, qui peut rendre le délai de livraison incertain, Scrum offre une approche structurée avec des délais de livraison clairs et prévisibles.

## Définition

Scrum est un cadre méthodologique utilisé dans le développement logiciel et la gestion de projets pour favoriser la collaboration, l'adaptabilité et l'efficacité. Il s'inscrit dans les pratiques agiles et est conçu pour aider les équipes à travailler ensemble de manière plus flexible et itérative.

## Les rôles dans Scrum

- **Product Owner** : Responsable de la gestion du backlog produit, il définit les priorités des fonctionnalités à développer et veille à ce que l'équipe de développement crée des produits qui répondent aux besoins des utilisateurs et des parties prenantes.
- **Scrum Master** : Facilite l'application de Scrum, aide l'équipe à surmonter les obstacles et à améliorer ses pratiques. Il veille également à ce que les processus Scrum soient suivis correctement.
- **Équipe de Développement** : Groupe de professionnels qui travaillent ensemble pour créer le produit. L'équipe est auto-organisée et interdisciplinaire, ce qui signifie qu'elle possède toutes les compétences nécessaires pour réaliser le produit sans dépendre d'autres équipes.

Dans notre cas, les rôles sont répartis comme suit :

- **Product Owner** : Ahmed NEFFATI
- **Scrum Master** : Ahmed NEFFATI
- **Équipe de Développement** : Ismail MABROUKI

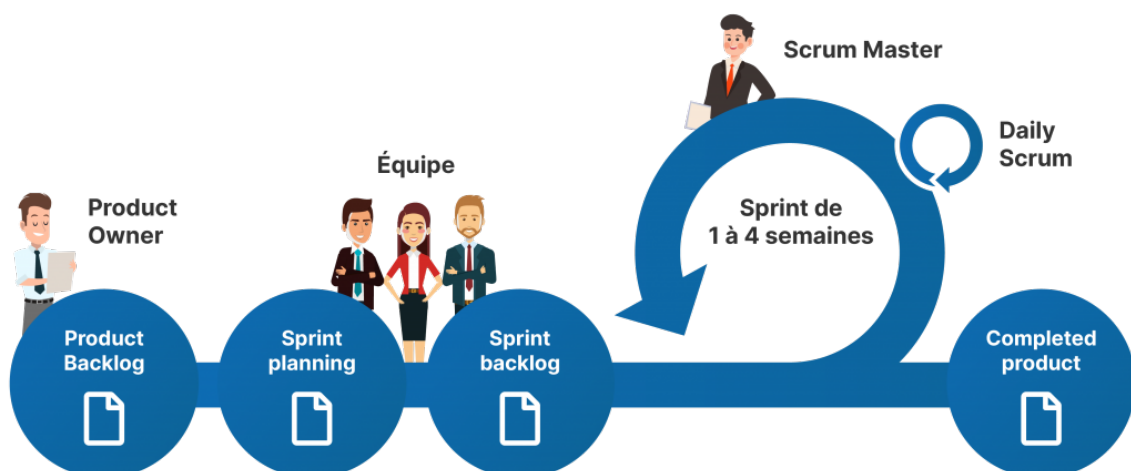


Figure 1.2 – représente le processus Scrum et ses principales phases.

**Outils de SCRUM :**

- **Google Meet** : Pour la communication et les réunions.
- **Jira** : Pour la gestion du projet.
- **GIT/GITHUB** : Pour la gestion des travaux.
- **LaTeX** : Outil de rédaction du rapport.

## 1.6 Langage de conception

### UML (Unified Modeling Language)

UML, qui signifie Unified Modeling Language (Langage de Modélisation Unifié), est un langage graphique standardisé utilisé pour visualiser la conception de systèmes logiciels. Il permet de créer des plans directeurs pour les applications logicielles en représentant les objets, leurs interactions et leurs comportements. Voici quelques aspects

clés d'UML :

- Représentation visuelle : UML utilise des diagrammes pour représenter différents aspects d'un système, facilitant la compréhension des relations et interactions complexes entre les composants.
- Orientation objet : Il s'aligne bien avec les principes de la programmation orientée objet (POO), permettant aux développeurs de modéliser des entités du monde réel en tant qu'objets au sein du système.
- Polyvalence : UML propose une variété de diagrammes pour divers aspects de la conception de systèmes, notamment :
  - Diagrammes de classes : Ils représentent les classes, leurs attributs, leurs méthodes et leurs relations telles que l'héritage et l'association.
  - Diagrammes d'objets : Ils illustrent les instances de classes et leurs relations à un moment donné.
  - Diagrammes de séquence : Ils capturent la séquence de messages échangés entre les objets au cours d'une interaction spécifique.
  - Diagrammes de cas d'utilisation : Ils décrivent les interactions entre les acteurs (utilisateurs) et le système d'un point de vue fonctionnel.
  - Diagrammes d'états : Ils modélisent le comportement d'un objet lorsqu'il passe entre différents états en réponse à des événements.
  - Diagrammes d'activités : Ils illustrent le flux des activités au sein d'un système, y compris l'exécution séquentielle et parallèle.

## 1.7 Conclusion

En conclusion, ce chapitre introductif a posé les bases de notre projet, en définissant sa mission, en présentant notre site web et en établissant les objectifs à atteindre. Nous avons également mené une analyse critique de l'existant afin de mieux comprendre les défis et les opportunités du marché. Enfin, nous avons décrit la méthodologie de développement qui guidera la réalisation de notre projet.

Le prochain chapitre s'attaquera à la phase cruciale d'analyse et de spécification des besoins.

[1]



Figure 1.3 – QR code Github : Tunicart

## Deuxième Chapitre

# Analyse et planification du projet

### 2.1 Introduction

Pour une meilleure conception du projet TunisiCart, l'étude des besoins est une étape primordiale. Dans ce chapitre, nous nous intéressons à l'identification des différents acteurs du projet et des besoins fonctionnels et non fonctionnels demandés par la société. Par la suite, nous exposerons notre backlog du produit, l'architecture du projet, ainsi que l'environnement de travail.

### 2.2 Spécification des besoins

Ces sections présentent une spécification détaillée des besoins pour l'application TuniCart. Il définit les fonctionnalités attendues de l'application, les contraintes techniques et les exigences non fonctionnelles. Cette spécification servira de base à la conception et au développement de l'application TuniCart.

#### 2.2.1 Identification des acteurs

L'identification des acteurs est une étape cruciale pour comprendre les interactions entre les différentes parties prenantes du projet TunisiCart. Ces acteurs sont les utilisateurs et systèmes qui interagiront avec l'application, chacun ayant des rôles et des responsabilités spécifiques.

Les principaux acteurs identifiés dans le cadre du projet TunisiCart sont :

- **Visiteur :**

Le visiteur est un utilisateur non enregistré qui peut parcourir les produits disponibles sur la plateforme sans avoir accès aux fonctionnalités réservées aux utilisateurs inscrits, telles que l'achat de produits ou la gestion du compte.

- **Client :**

Le client est l'utilisateur final de la plateforme, qui parcourt les produits, passe des commandes, laisse des avis, et effectue des paiements.

- **Vendeur :**

Le vendeur est un utilisateur enregistré qui propose ses produits à la vente sur la plateforme. Il gère les produits, les stocks, les commandes, et interagit avec les clients via la plateforme.

- **Administrateur :**

L'administrateur est responsable de la gestion globale de la plateforme. Il gère les comptes utilisateurs, modère les contenus, supervise les catégories et les produits, et assure le bon fonctionnement du système.

- **Système de paiement :**

Le système de paiement est une entité externe intégrée à la plateforme pour faciliter les transactions financières entre les clients et les vendeurs de manière sécurisée.

Chaque acteur interagit avec la plateforme TunisiCart d'une manière spécifique, contribuant ainsi à l'expérience globale offerte par le système.

## 2.2.2 Besoins fonctionnels

L'application à réaliser doit offrir un ensemble de fonctionnalités qui doivent être mises en relation avec un ensemble de besoins utilisateur. Ces derniers définissent les services que les utilisateurs s'attendent à voir fournis par cette application. Les principales exigences métiers de notre application se résument dans les fonctionnalités suivantes :

- **Inscription à la plateforme :**

Permettre aux utilisateurs de créer un compte sur la plateforme.

- **Authentification :**

Authentification par un login et un mot de passe pour accéder aux différentes fonctionnalités.

- **Gestion des utilisateurs :**

Permettre aux administrateurs de gérer les comptes des clients et des vendeurs.

- **Contrôle d'accès aux fonctionnalités :**

Différencier les fonctionnalités disponibles selon le rôle de l'utilisateur (client, vendeur, administrateur).

- **Gestion des produits :**

Consulter la liste des produits. Ajouter de nouveaux produits par catégorie, modifier et supprimer des produits.

- **Gestion des commandes :**

Consulter la liste des commandes. Ajouter de nouvelles commandes, modifier et supprimer des commandes.

- **Gestion des avis :**  
Consulter la liste des avis. Ajouter de nouveaux avis pour les produits, modifier et supprimer des avis.
- **Gestion des paiements :**  
Consulter la liste des paiements effectués. Ajouter de nouvelles options de paiement, modifier et supprimer des paiements.

### 2.2.3 Besoins non fonctionnels

En plus des besoins fonctionnels décrits précédemment, il est essentiel de définir les besoins non fonctionnels de TuniCart. Ces besoins décrivent les contraintes et les exigences non liées aux fonctionnalités spécifiques de l'application, mais qui impactent sa conception, son développement et son utilisation.

#### Ergonomie

- **Simplicité et facilité d'utilisation :** L'interface utilisateur de Tunicart doit être intuitive et facile à utiliser pour tous les types d'utilisateurs, quelle que soit leur expérience technique. L'application doit être navigable de manière fluide et logique, permettant aux utilisateurs de trouver rapidement les informations et fonctionnalités dont ils ont besoin.
- **Confort et satisfaction de l'utilisateur :** L'interaction avec Tunicart doit être agréable et sans frustration. L'application doit offrir une expérience utilisateur cohérente et positive, en minimisant les erreurs et en fournissant des messages clairs et utiles.

#### Fiabilité

- **Stabilité et absence de bugs :** Tunicart doit fonctionner de manière fiable et sans bugs majeurs. Les utilisateurs ne doivent pas rencontrer de problèmes techniques qui entravent leur utilisation de l'application.
- **Disponibilité et performances constantes :** L'application doit être disponible en permanence et offrir des performances acceptables, même en cas de forte charge d'utilisateurs. Les temps de réponse doivent être courts et les processus fluides.

#### Sécurité

- **Protection des données :** Les données personnelles et sensibles des utilisateurs doivent être protégées contre tout accès non autorisé, divulgation ou altération. Tunicart doit implémenter des mesures de sécurité robustes, telles que le chiffrement des données, l'authentification forte et la gestion des autorisations d'accès strictes.



- **Authentification et contrôle d'accès** : L'accès aux fonctionnalités et informations de Tunicart doit être contrôlé en fonction des rôles et des privilèges des utilisateurs. Un processus d'authentification sécurisé doit être mis en place pour garantir que seuls les utilisateurs autorisés peuvent accéder aux données et fonctionnalités appropriées.

### 2.2.4 Maintenabilité et extensibilité

- **Code propre et modulaire** : Le code source de Tunicart doit être bien structuré, documenté et facile à maintenir. L'utilisation de pratiques de programmation modernes et de principes de conception modulaire facilitera la modification, l'ajout de nouvelles fonctionnalités et la correction de bugs à l'avenir.
- **Capacité d'adaptation et d'évolution** : L'architecture de Tunicart doit être flexible et extensible pour permettre l'intégration de nouvelles fonctionnalités, l'adaptation à de nouveaux cas d'utilisation et la croissance de la base d'utilisateurs. L'application doit pouvoir évoluer sans compromettre sa performance, sa stabilité ou sa sécurité.

## 2.3 Backlog du produit

Après avoir présenté les besoins de notre solution, nous allons détailler dans cette section le backlog du produit, qui décrit la liste des besoins sous forme de user stories.

Nous présentons le tableau 2.1, qui résume le backlog du produit relatif à notre solution et qui énumère les champs suivants :

- **ID** : C'est un nombre unique et auto-incrémenté pour chaque User Story.
- **Titre** : C'est le résumé de la User Story.
- **User Story** : C'est une description courte de la tâche à réaliser, qui se définit de la manière suivante : *En tant que "acteur", je souhaite ...*
- **E** : Évaluation initiale de la charge de travail nécessaire pour l'implémentation de cette tâche. L'unité est en jours.
- **P** : Priorité.

Pour prioriser nos besoins, nous avons opté pour la méthode «MoSCoW», qui se base sur les acronymes suivants :

- **M** – *Must have* : Indique les besoins essentiels qui doivent absolument être implémentés pour que le projet soit considéré comme un succès.
- **S** – *Should have* : Indique les besoins importants mais non essentiels. Leur absence n'empêchera pas le projet d'atteindre ses objectifs, mais ils devraient idéalement être implémentés.
- **C** – *Could have* : Indique les besoins qui seraient agréables à avoir, mais qui sont moins prioritaires.

— **W** – *Won't have* : Indique les besoins qui ne seront pas implémentés dans cette phase du projet.

| ID | Titre                    | User Story   | E (jours) | P (MoSCoW) |
|----|--------------------------|--|-----------|------------|
| 1  | Inscription              | En tant que visiteur, je souhaite m'inscrire sur la plateforme afin de pouvoir accéder aux fonctionnalités réservées aux utilisateurs enregistrés. | 2         | M          |
| 2  | Authentification         | En tant qu'utilisateur enregistré, je souhaite me connecter à la plateforme pour accéder à mon compte.   | 2         | M          |
| 3  | Profil utilisateur       | En tant qu'utilisateur enregistré, je souhaite modifier mon profil (adresse, numéro de téléphone, etc.) pour maintenir mes informations à jour.    | 3         | M          |
| 4  | Historique des activités | En tant qu'utilisateur, je souhaite voir mon historique d'achat et de navigation pour suivre mes activités sur la plateforme.                      | 3         | S          |
| 5  | Ajout de produits        | En tant que vendeur, je souhaite ajouter des produits à ma boutique pour les vendre sur la plateforme.   | 3         | M          |
| 6  | Gestion des commandes    | En tant que client, je souhaite gérer mes commandes pour suivre leur statut et historique.   | 2         | M          |
| 7  | Gestion des avis         | En tant que client, je souhaite laisser un avis sur un produit pour partager mon expérience avec les autres utilisateurs.                          | 2         | S          |
| 8  | Paiement sécurisé        | En tant que client, je souhaite effectuer des paiements en ligne de manière sécurisée pour finaliser mes achats.                                   | 3         | M          |
| 9  | Recherche de produits    | En tant qu'utilisateur, je souhaite rechercher des produits spécifiques pour trouver ce que je cherche rapidement.                                 | 2         | M          |
| 10 | Filtres et tri           | En tant qu'utilisateur, je souhaite filtrer et trier les produits par prix, popularité, avis, etc., pour affiner mes choix.                        | 2         | S          |

**Table 2.1** – Backlog du produit de TunisiCart

## 2.4 Identification des sprints

Pour organiser le développement de notre projet TunisiCart, nous avons divisé le travail en sprints. Chaque sprint représente une période de temps pendant laquelle un ensemble de fonctionnalités doit être développé et livré. L'objectif est de garantir une gestion efficace du projet et de permettre des ajustements en cours de route. Nous avons planifié les sprints en fonction des fonctionnalités prioritaires et des estimations de temps, en respectant un minimum de 7 jours par sprint.

### 2.4.1 Sprint 1 (Jours 1 à 7)

- **Inscription** : En tant que visiteur, je souhaite m'inscrire sur la plateforme afin de pouvoir accéder aux fonctionnalités réservées aux utilisateurs enregistrés.  
*Durée : 3 jours*
- **Authentification** : En tant qu'utilisateur enregistré, je souhaite me connecter à la plateforme pour accéder à mon compte.  
*Durée : 2 jours*
- **Profil utilisateur** : En tant qu'utilisateur enregistré, je souhaite modifier mon profil (adresse, numéro de téléphone, etc.) pour maintenir mes informations à jour.  
*Durée : 2 jours*

### 2.4.2 Sprint 2 (Jours 8 à 14)

- **Ajout de produits** : En tant que vendeur, je souhaite ajouter des produits à ma boutique pour les vendre sur la plateforme.  
*Durée : 4 jours*
- **Gestion des commandes** : En tant que client, je souhaite gérer mes commandes pour suivre leur statut et historique.  
*Durée : 3 jours*

### 2.4.3 Sprint 3 (Jours 15 à 21)

- **Gestion des avis** : En tant que client, je souhaite laisser un avis sur un produit pour partager mon expérience avec les autres utilisateurs.  
*Durée : 2 jours*
- **Paiement sécurisé** : En tant que client, je souhaite effectuer des paiements en ligne de manière sécurisée pour finaliser mes achats.  
*Durée : 4 jours*
- **Recherche de produits** : En tant qu'utilisateur, je souhaite rechercher des produits spécifiques pour trouver ce que je cherche rapidement.  
*Durée : 1 jour*

#### 2.4.4 Sprint 4 (Jours 22 à 28)

- **Filtres et tri** : En tant qu'utilisateur, je souhaite filtrer et trier les produits par prix, popularité, avis, etc., pour affiner mes choix.  
*Durée : 2 jours*
- **Historique des activités** : En tant qu'utilisateur, je souhaite voir mon historique d'achat et de navigation pour suivre mes activités sur la plateforme.  
*Durée : 5 jours*

## 2.5 Diagrammes des cas d'utilisation

Dans le domaine du développement logiciel, le diagramme de cas d'utilisation global est un outil fondamental pour comprendre les subtilités de la fonctionnalité d'un système du point de vue de ses utilisateurs. Il offre une vue d'ensemble des interactions entre les différents acteurs et des fonctionnalités essentielles du système. Essentiellement, il définit le « qui fait quoi » dans le système. La figure ci-dessous illustre le diagramme du cas d'utilisation global du système

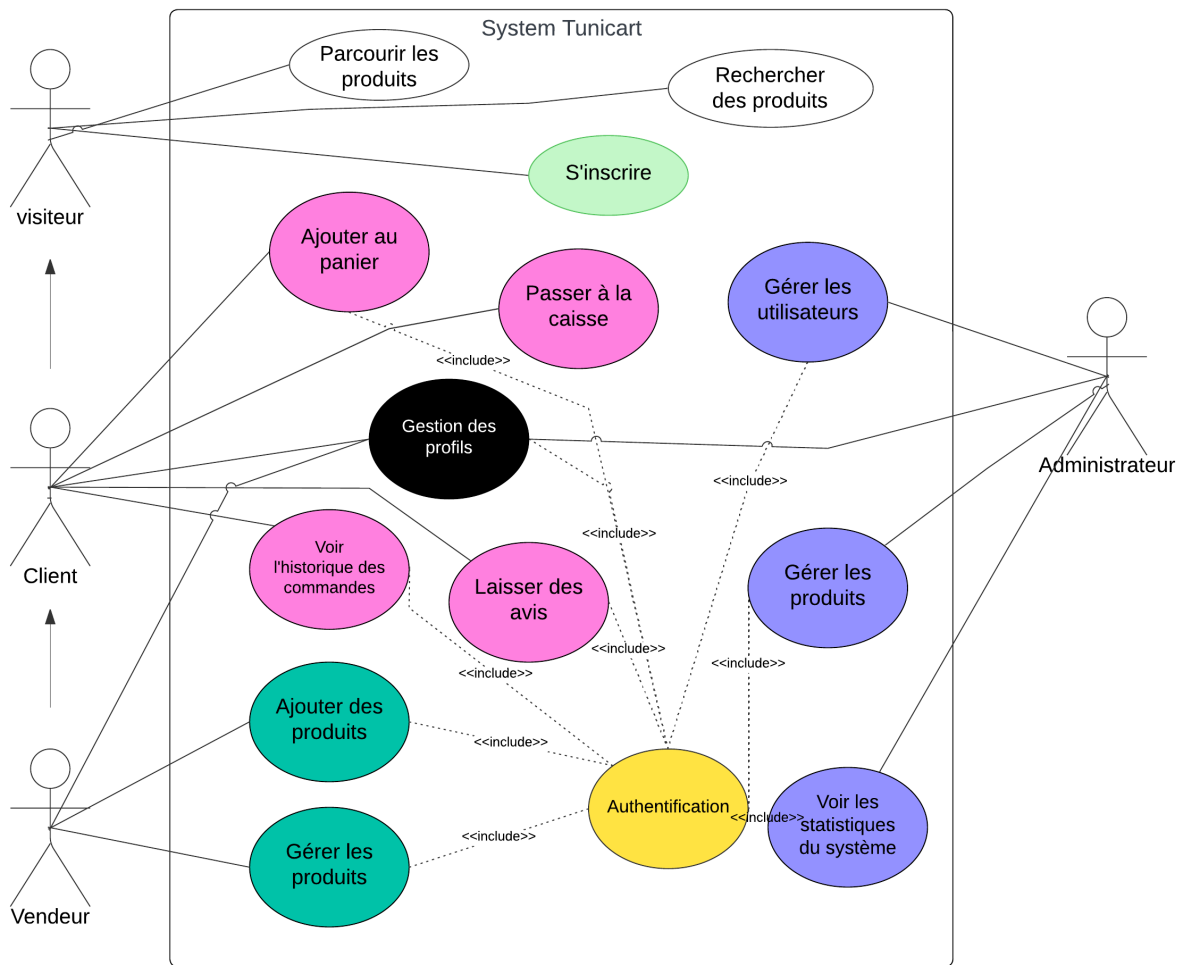


Figure 2.1 – Diagramme de cas d'utilisation global

## 2.6 Diagramme des classes

Le diagramme des classes est considéré comme le plus important dans la modélisation orienté objet. Il s'agit d'une vue statique, du fait qu'on ne tient pas compte

au facteur temporelle dans le comportement du système. Indépendamment d'un langage de programmation particulier, le diagramme des classes permet de modéliser les classes du système et leurs relations, illustré par la figure 2.2

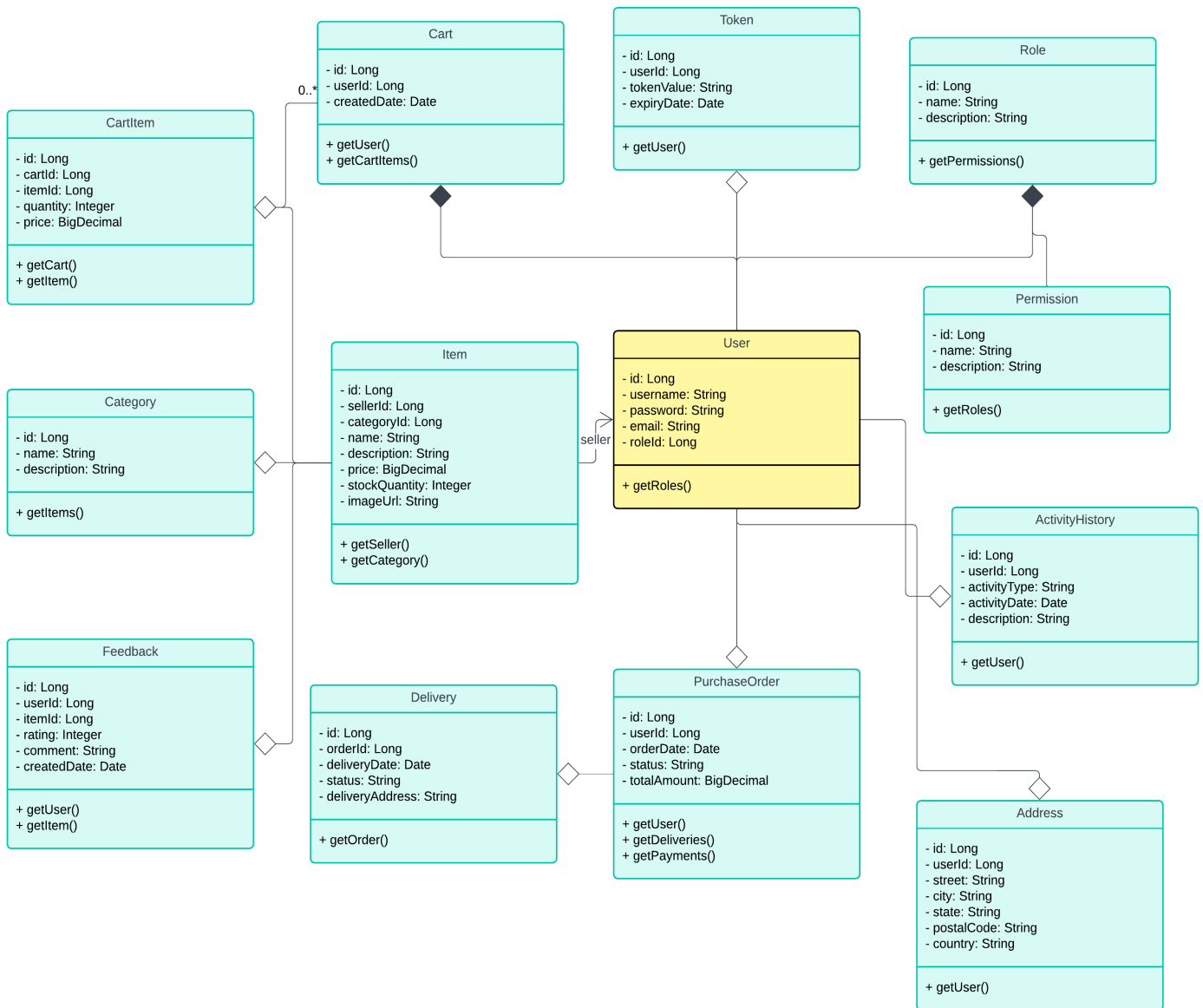


Figure 2.2 – figure diagramme de classe

## 2.7 Architecture du système

### 2.7.1 Architecture globale

Tunicart adoptera une architecture logicielle multicouche, composée de trois couches principales :

**1. Couche présentation :**

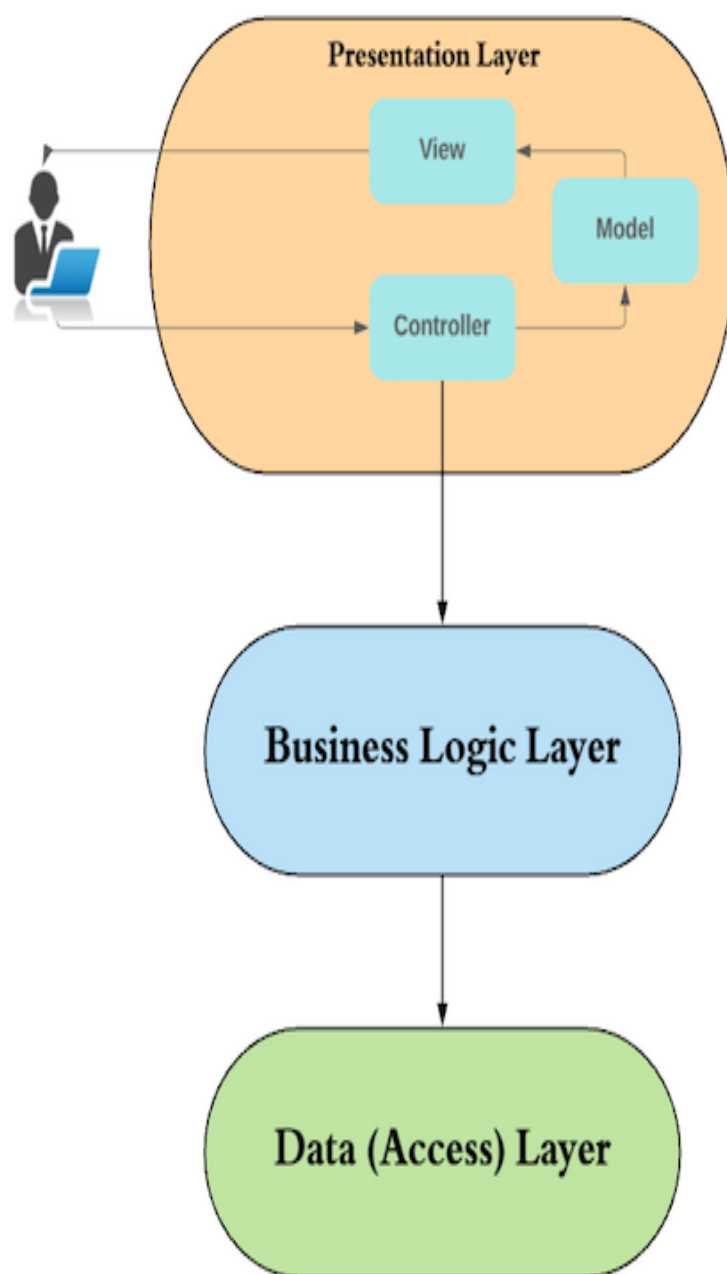
- Cette couche sera responsable de l'interaction avec l'utilisateur, fournissant des interfaces utilisateur web et mobiles intuitives et réactives.
- Elle sera construite à l'aide de frameworks modernes tels que React ou Angular pour le web et Flutter ou React Native pour les applications mobiles.

**2. Couche métier :**

- Cette couche encapsulera la logique métier du système, gérant les fonctionnalités principales telles que la gestion des expéditions, la gestion des flottes, le suivi en temps réel, les rapports et l'analyse.
- Elle sera implémentée en utilisant des langages de programmation tels que Java ou Python, en s'appuyant sur des frameworks de développement tels que Spring Boot ou Django.

**3. Couche d'accès aux données :**

- Cette couche fournira une abstraction pour accéder aux données persistantes du système, telles que les informations sur les expéditions, les transporteurs, les véhicules, les clients et les utilisateurs.
- Elle utilisera une base de données relationnelle robuste telle que MySQL ou PostgreSQL, avec des frameworks d'accès aux données tels que Hibernate ou SQLAlchemy.



**Figure 2.3** – figure architecture multicouches



## 2.7.2 Architecture logique

L'architecture logique du projet TunisiCart est basée sur le modèle de conception MVC, qui permet de structurer l'application en séparant les préoccupations. La figure suivante illustre le schéma de l'architecture logique du projet.

- **Couche Modèle :**

La couche Modèle est responsable de l'accès aux données. Elle permet le stockage et la récupération des données traitées par la couche applicative. Cette couche contient les entités de l'application ainsi que les classes de manipulation des données.

*Exemples de classes : Product, User, Order, Feedback*

- **Couche Vue :**

La couche Vue correspond à la couche de présentation, offrant l'interface utilisateur de l'application. Elle permet aux utilisateurs d'interagir avec le système en affichant les données issues de la couche applicative.

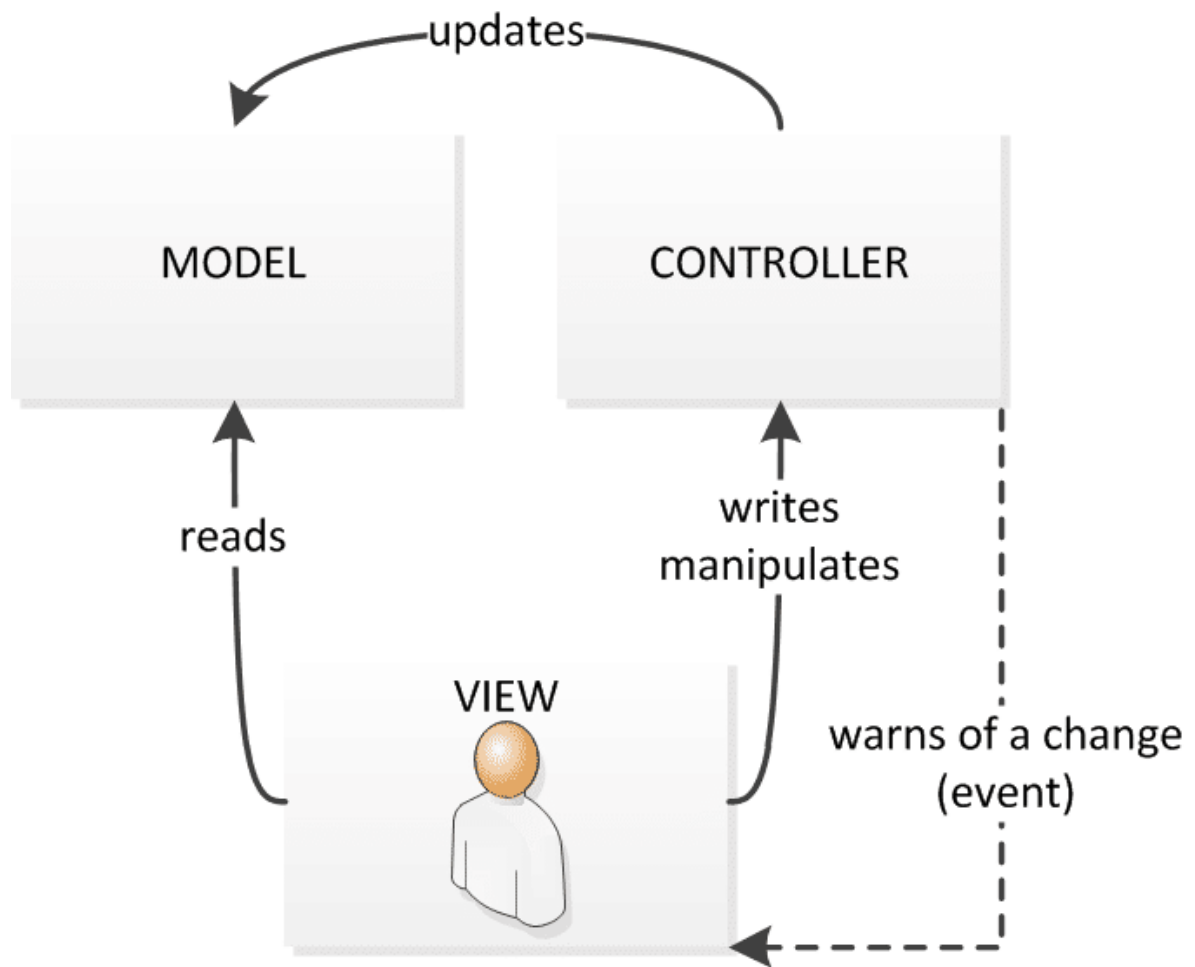
*Technologie utilisée : Angular*

*Exemples de fonctionnalités : Affichage des produits, formulaire de commande, gestion du panier*

- **Couche Contrôleur :**

La couche Contrôleur gère la logique métier de l'application. Elle traite les requêtes des utilisateurs, applique les règles de gestion, et coordonne les actions entre les couches Modèle et Vue. Cette couche est divisée en plusieurs sous-couches pour mieux structurer les responsabilités.

- **Couche d'accès aux données (DAO) :** Contient les classes responsables de l'accès à la base de données et du traitement des requêtes SQL.
- **Couche de service métier :** Contient les classes qui implémentent la logique métier spécifique de l'application.
- **Couche d'exposition des services :** Gère l'exposition des API REST, permettant la communication entre le frontend et le backend.
- **Couche de sécurité :** Gère l'authentification et l'autorisation des utilisateurs, protégeant les accès aux ressources sensibles.

**Figure 2.4** – ModeleMVC

## 2.8 Environnement de développement

L'environnement de développement est l'ensemble des outils et des ressources matériels et logiciels utilisés pour concevoir, développer et tester une application web. Il joue un rôle crucial dans la réussite du processus de développement, en fournissant aux développeurs les outils nécessaires pour créer un produit de qualité.

### 2.8.1 Environnement matériel

Device Name : ASUS ExpertBook AiselDev  
Processor : 11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz  
Installed RAM : 24.0 GB (23.7 GB usable)  
Device ID : 29D589CF-2364-4E33-9F23-6DAF16D9D018  
Product ID : 00331-10000-00001-AA514  
System Type : 64-bit operating system, x64-based processor

### 2.8.2 Environnement logiciel

- Java Spring Boot
- Angular
- PostgreSQL
- Docker (maildev)
- Visual Studio Code (VS Code)
- IntelliJ IDEA Community
- Git

## 2.9 Choix des logiciels

### 2.9.1 Choix : Java Spring Boot

Lors de la conception de notre application web, nous avons décidé d'utiliser Java Spring Boot comme framework de développement. Spring Boot est un framework Java basé sur Spring qui facilite le développement d'applications Java, en offrant une configuration par défaut et des fonctionnalités de démarrage rapide.

Les raisons de notre choix sont les suivantes :

- **Facilité de configuration** : Spring Boot offre une configuration par défaut, ce qui permet de démarrer rapidement un projet sans avoir à configurer manuellement de nombreux paramètres.

- **Productivité accrue** : Grâce à son système de gestion des dépendances intégré et à ses fonctionnalités de développement, Spring Boot permet d'accélérer le processus de développement et d'améliorer la productivité des développeurs.
- **Support de la communauté** : Spring Boot bénéficie d'une large communauté de développeurs et d'une documentation exhaustive, ce qui facilite la résolution des problèmes et la recherche de solutions.
- **Intégration avec d'autres technologies** : Spring Boot s'intègre facilement avec d'autres technologies Java, ce qui nous permettra d'exploiter pleinement les fonctionnalités de notre application web.

En résumé, nous sommes convaincus que Java Spring Boot est le choix optimal pour le développement de notre application web, en raison de sa simplicité, de sa productivité et de son intégration avec d'autres technologies Java.



Figure 2.5 – Logo de java Spring boot

### 2.9.2 Choix : Angular

Pour la conception de notre application web, nous avons délibérément opté pour Angular en tant que framework front-end. Angular est un framework JavaScript développé par Google, largement utilisé pour créer des applications web modernes et dynamiques.

Nous avons choisi Angular pour les raisons suivantes :

- **Structure robuste** : Angular offre une structure de projet solide et bien définie, facilitant l'organisation et la maintenance du code source de notre application.
- **Performances optimisées** : Angular est conçu pour offrir des performances élevées, avec des mécanismes tels que la détection des modifications, le lazy loading et le pré-chargement des modules, ce qui garantit une expérience utilisateur fluide.
- **Binding de données bidirectionnel** : Angular offre une liaison de données bidirectionnelle, permettant une synchronisation efficace entre les vues et les modèles de données, ce qui simplifie le développement et améliore la réactivité de l'interface utilisateur.
- **Large écosystème** : Angular bénéficie d'un vaste écosystème d'outils, de bibliothèques et de modules complémentaires, ce qui facilite l'intégration de fonctionnalités avancées et la résolution des problèmes.

- **Support de la communauté** : Angular est soutenu par une communauté active de développeurs et dispose d'une documentation exhaustive, ce qui facilite l'apprentissage et la résolution des problèmes rencontrés.

En conclusion, nous sommes convaincus qu'Angular est le choix idéal pour développer notre application web en raison de sa structure robuste, de ses performances optimisées et de son vaste écosystème de développement.



Figure 2.6 – Logo d'Angular

### 2.9.3 Choix : MySQL

Pour le stockage des données de notre application web, nous avons choisi d'utiliser MySQL comme système de gestion de base de données. MySQL est un système de base de données relationnelle open source largement adopté, reconnu pour sa simplicité, ses performances et sa flexibilité.

Nous avons opté pour MySQL en raison des avantages suivants :

- **Simplicité et facilité d'utilisation** : MySQL est réputé pour sa facilité d'installation et de configuration, ce qui permet une prise en main rapide, particulièrement adaptée pour les développeurs et les équipes de développement agiles.
- **Performances élevées** : MySQL est optimisé pour offrir des performances rapides, même dans les environnements à grande échelle. Il est couramment utilisé dans des applications web à fort trafic grâce à sa capacité à gérer des volumes de données importants tout en maintenant des temps de réponse rapides.
- **Large compatibilité** : MySQL est compatible avec de nombreux langages de programmation et plateformes, ce qui en fait un choix flexible et adapté à différents types d'architectures web et mobiles.
- **Sécurité** : MySQL propose des fonctionnalités de sécurité robustes, telles que l'authentification basée sur des rôles, la gestion des utilisateurs et des privilèges,

ainsi que la gestion des connexions SSL, garantissant un traitement sécurisé des données.

- **Communauté et support commercial :** MySQL bénéficie d'une large communauté d'utilisateurs, ainsi que du support commercial d'Oracle, ce qui assure un suivi régulier, des mises à jour fréquentes et une documentation riche.

En conclusion, nous avons choisi MySQL pour le stockage des données de notre application web, en raison de sa simplicité, de ses performances optimisées et de sa compatibilité avec divers environnements de développement.



Figure 2.7 – Logo de mySQL

#### 2.9.4 Choix : Docker (Maildev)

Pour la gestion de notre environnement de développement et de test des emails, nous avons opté pour l'utilisation de Docker avec Maildev. Docker est une plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs légers et isolés, tandis que Maildev est un serveur de messagerie SMTP et IMAP simulé conçu pour le développement et le test d'applications web.

Nous avons choisi Docker avec Maildev pour les raisons suivantes :

- **Facilité de configuration :** Docker simplifie la configuration et le déploiement de Maildev en encapsulant tous les composants nécessaires dans un conteneur unique, ce qui facilite la mise en place de l'environnement de développement.
- **Isolation :** Docker utilise des conteneurs légers et isolés pour exécuter des applications, ce qui garantit que Maildev fonctionne de manière cohérente et fiable, sans interférence avec d'autres composants du système.
- **Reproductibilité :** En utilisant Docker, nous pouvons reproduire facilement l'environnement de développement sur différentes machines, assurant ainsi une cohérence dans le processus de développement et de test des emails.

- **Flexibilité** : Docker offre une grande flexibilité dans la gestion des conteneurs, ce qui nous permet d'ajuster rapidement les paramètres de configuration de Maildev selon les besoins spécifiques de notre application web.
- **Efficacité des tests** : Maildev permet de simuler l'envoi et la réception d'emails dans un environnement de développement, ce qui facilite les tests d'intégration et de validation des fonctionnalités liées aux emails.

En résumé, Docker avec Maildev est un choix judicieux pour la gestion de notre environnement de développement et de test des emails, offrant une configuration facile, une isolation sécurisée, une reproductibilité garantie et une efficacité accrue des tests.



Figure 2.8 – maildev/maildev - Docker Image

### 2.9.5 Choix : Visual Studio Code (VS Code)

Pour le développement de notre application web, nous avons choisi d'utiliser Visual Studio Code (VS Code) comme environnement de développement intégré (IDE). VS Code est un éditeur de code source léger, puissant et extensible développé par Microsoft, largement utilisé par les développeurs pour sa polyvalence et sa richesse en fonctionnalités.

Nous avons opté pour VS Code pour les raisons suivantes :

- **Polyvalence** : VS Code prend en charge de nombreux langages de programmation et frameworks, ce qui en fait un outil polyvalent pour le développement d'applications web, mobiles et cloud.

- **Performance** : VS Code est connu pour sa légèreté et sa réactivité, offrant une expérience utilisateur fluide même sur des machines moins puissantes.
- **Extension et personnalisation** : VS Code propose une large gamme d'extensions et de thèmes, ainsi que des fonctionnalités de personnalisation avancées, permettant aux développeurs de configurer l'IDE selon leurs préférences et leurs besoins spécifiques.
- **Intégration avec les outils** : VS Code s'intègre facilement avec de nombreux outils de développement populaires, tels que Git, Docker, et les frameworks comme Angular et React, ce qui facilite la collaboration et l'intégration de fonctionnalités tierces.
- **Communauté active** : VS Code bénéficie d'une large communauté de développeurs et d'utilisateurs, offrant un support continu, des mises à jour régulières et une documentation exhaustive.

En conclusion, nous sommes convaincus que Visual Studio Code est le choix optimal pour le développement de notre application web, en raison de sa polyvalence, de sa performance et de sa large gamme de fonctionnalités.

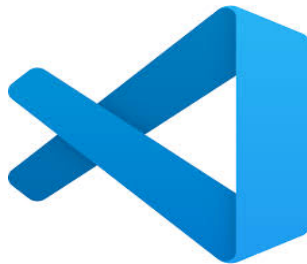


Figure 2.9 – VsCode Logo

### 2.9.6 Choix : IntelliJ IDEA Community

Pour le développement de notre application web, nous avons décidé d'utiliser IntelliJ IDEA Community Edition comme notre environnement de développement intégré (IDE). IntelliJ IDEA est un IDE développé par JetBrains, réputé pour sa richesse en fonctionnalités, sa productivité et sa facilité d'utilisation.

Nous avons opté pour IntelliJ IDEA Community pour les raisons suivantes :

- **Richesse en fonctionnalités** : IntelliJ IDEA offre une large gamme de fonctionnalités avancées, telles que l'analyse statique du code, la complétion intelligente, le refactoring automatique et le débogage avancé, ce qui améliore la productivité des développeurs.
- **Intégration avec les technologies Java** : IntelliJ IDEA est spécialement conçu pour le développement Java, offrant une prise en charge complète des technologies



Java, y compris Spring, Hibernate, Maven et Gradle, ce qui facilite le développement d'applications web Java.

- **Interface utilisateur intuitive** : IntelliJ IDEA propose une interface utilisateur intuitive et conviviale, avec des fonctionnalités telles que la navigation rapide, les raccourcis clavier personnalisables et la gestion des projets, ce qui améliore l'expérience de développement.
- **Extensions et plugins** : IntelliJ IDEA dispose d'un écosystème d'extensions et de plugins étendu, permettant aux développeurs d'ajouter des fonctionnalités supplémentaires selon leurs besoins spécifiques.
- **Support de la communauté** : IntelliJ IDEA bénéficie d'une large communauté de développeurs et d'utilisateurs, offrant un support continu, des mises à jour régulières et une documentation exhaustive.

En résumé, nous sommes convaincus que IntelliJ IDEA Community Edition est le choix idéal pour le développement de notre application web, en raison de sa richesse en fonctionnalités, de son intégration avec les technologies Java et de son interface utilisateur intuitive.

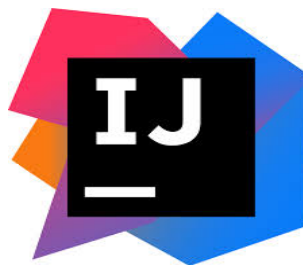


Figure 2.10 – IntelliJ Logo

### 2.9.7 Choix : Git

Pour la gestion du code source de notre application web, nous avons opté pour Git comme système de contrôle de version. Git est un système de gestion de versions distribué, largement utilisé par les développeurs pour suivre les modifications apportées au code source et collaborer efficacement sur des projets logiciels.

Nous avons choisi Git pour les raisons suivantes :

- **Flexibilité et polyvalence** : Git offre une grande flexibilité dans la gestion du code source, avec des fonctionnalités telles que les branches, les fusions et les rebasages, permettant aux développeurs de travailler de manière efficace sur des fonctionnalités distinctes et d'expérimenter de nouvelles idées.
- **Distribution décentralisée** : Git est un système de contrôle de version distribué, ce qui signifie que chaque développeur dispose d'une copie complète de

l'historique du projet, facilitant le travail hors ligne et la collaboration avec des équipes distribuées.

- **Performances élevées** : Git est conçu pour être rapide et efficace, même avec des projets de grande taille, ce qui garantit des opérations de gestion du code source rapides et fluides.
- **Communauté active** : Git bénéficie d'une large communauté de développeurs et d'utilisateurs, offrant un support continu, des mises à jour régulières et une documentation exhaustive.
- **Intégration avec les plateformes de développement** : Git s'intègre facilement avec des plateformes de développement populaires telles que GitHub, GitLab et Bitbucket, offrant des fonctionnalités avancées telles que le suivi des problèmes, les demandes de tirage et les intégrations continues.

En conclusion, nous sommes convaincus que Git est le choix optimal pour la gestion du code source de notre application web, en raison de sa flexibilité, de sa distribution décentralisée et de sa communauté active.



Figure 2.11 – Git Logo

# Troisième Chapitre

## Sprint 1

### Introduction

Après avoir analysé et déterminé les objectifs globaux de notre solution, nous présentons en détail les différentes phases exécutées durant ce premier sprint, ayant pour objectif la réalisation des fonctionnalités suivantes : Gestion des utilisateurs et mise en place des fonctionnalités de base pour la plateforme TunisiCart.

Nous commençons par la spécification et l'analyse des besoins du sprint 1. Par la suite, nous faisons une description détaillée des cas d'utilisation, des diagrammes de séquence, et nous terminons par la réalisation technique.

### 3.1 Spécification et analyse des besoins

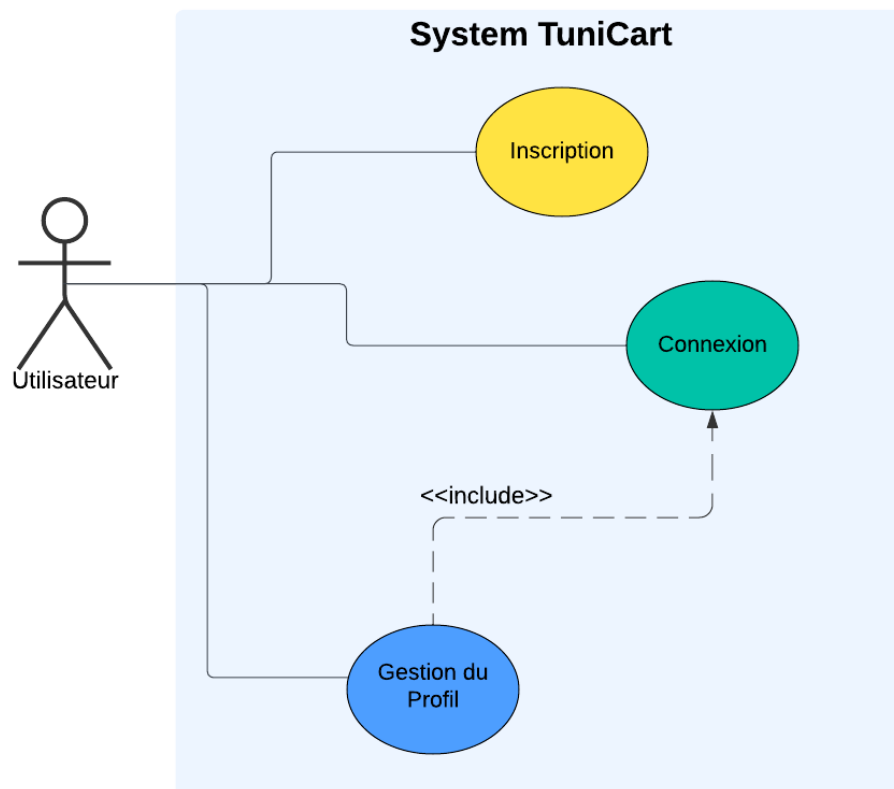
Dans cette partie, nous allons entamer l'analyse et la spécification des besoins de ce sprint. Nous allons commencer par présenter le backlog, ensuite établir le diagramme de cas d'utilisation raffiné, et finir par détailler la description des cas d'utilisation.

### 3.1.1 Backlog du Sprint 1

| ID  | User Story   | Priorité (MoSCoW) |
|-----|--|-------------------|
| 1.0 | En tant que nouvel utilisateur, je veux pouvoir m'inscrire sur la plateforme en fournissant mes informations personnelles, afin de pouvoir utiliser les fonctionnalités du site. | M                 |
| 1.1 | En tant qu'utilisateur inscrit, je veux pouvoir me connecter à la plateforme en utilisant mon identifiant et mon mot de passe, afin d'accéder à mon compte.                      | M                 |
| 1.2 | En tant qu'utilisateur connecté, je veux pouvoir modifier et mettre à jour mon profil, afin de garder mes informations personnelles à jour.                                      | S                 |

**Table 3.1** – Backlog du Sprint 1 basé sur les User Stories pour le projet TunisiCart

### 3.1.2 Diagramme de Cas d'Utilisation Raffiné : Sprint 1



**Figure 3.1** – Diagramme de Cas d'Utilisation Raffiné pour le Sprint 1

## 3.2 Description détaillée des cas d'utilisation

### 1. S'Inscrire

**Description :** Un nouvel utilisateur souhaite s'inscrire sur la plateforme TunisiCart pour accéder aux fonctionnalités de la plateforme. Lors de l'inscription, l'utilisateur fournit ses informations personnelles, telles que son nom, prénom, adresse e-mail et mot de passe.

| Élément             | Détails  |
|---------------------|--|
| Titre               | S'Inscrire   |
| Acteurs             | Nouvel Utilisateur   |
| Préconditions       | Aucun compte existant  |
| Postconditions      | Un compte utilisateur est créé avec succès   |
| Scénario Nominal    | <ol style="list-style-type: none"> <li>1. L'utilisateur accède à la page d'inscription.</li> <li>2. L'utilisateur saisit ses informations personnelles dans le formulaire.</li> <li>3. L'utilisateur soumet le formulaire.</li> <li>4. Le système vérifie les informations et crée un nouveau compte.</li> <li>5. L'utilisateur reçoit une confirmation de l'inscription.</li> </ol> |
| Scénario Alternatif | Si les informations sont invalides (e.g., e-mail déjà utilisé, mot de passe trop faible), le système affiche un message d'erreur et demande des corrections.   |

**Table 3.2** – Cas d'utilisation : S'Inscrire

### 2. Se Connecter

**Description :** Un utilisateur inscrit souhaite se connecter à son compte TunisiCart pour accéder à ses fonctionnalités personnalisées. Lors de la connexion, l'utilisateur entre son identifiant et son mot de passe.

| Élément             | Détails  |
|---------------------|--|
| Titre               | Se Connecter   |
| Acteurs             | Utilisateur Inscrit  |
| Préconditions       | Compte utilisateur existant  |
| Postconditions      | L'utilisateur est authentifié et accède à son espace personnel   |
| Scénario Nominal    | <ol style="list-style-type: none"> <li>1. L'utilisateur accède à la page de connexion.</li> <li>2. L'utilisateur entre son identifiant et mot de passe.</li> <li>3. L'utilisateur soumet les informations de connexion.</li> <li>4. Le système vérifie les informations et authentifie l'utilisateur.</li> <li>5. L'utilisateur est redirigé vers son tableau de bord ou la page d'accueil.</li> </ol> |
| Scénario Alternatif | <p>Si les informations de connexion sont incorrectes, le système affiche un message d'erreur et demande à réessayer.</p> <p>Si l'utilisateur oublie son mot de passe, il peut utiliser la fonction de réinitialisation.</p>  |

**Table 3.3** – Cas d'utilisation : Se Connecter

### 3. Gérer son Profil

**Description :** Un utilisateur connecté souhaite gérer son profil sur la plateforme TunisiCart. Cela inclut la modification et la mise à jour de ses informations personnelles, telles que son nom, prénom, adresse e-mail et mot de passe.

| Élément             | Détails  |
|---------------------|--|
| Titre               | Gérer son Profil   |
| Acteurs             | Utilisateur Connecté   |
| Préconditions       | L'utilisateur doit être connecté   |
| Postconditions      | Les informations du profil de l'utilisateur sont mises à jour avec succès  |
| Scénario Nominal    | <ol style="list-style-type: none"> <li>1. L'utilisateur accède à la page de gestion du profil.</li> <li>2. L'utilisateur visualise ses informations actuelles.</li> <li>3. L'utilisateur modifie les informations souhaitées (nom, prénom, e-mail, mot de passe).</li> <li>4. L'utilisateur soumet les modifications.</li> <li>5. Le système vérifie les informations et met à jour le profil.</li> <li>6. L'utilisateur reçoit une confirmation de la mise à jour du profil.</li> </ol> |
| Scénario Alternatif | Si les informations fournies sont invalides (e.g., format d'e-mail incorrect, mot de passe faible), le système affiche un message d'erreur et demande des corrections.   |

**Table 3.4** – Cas d'utilisation : Gérer son Profil

## 3.3 Diagrammes de séquence

Les diagrammes de séquence ci-dessous montrent l'interaction entre les différents acteurs et le système pour les cas d'utilisation définis :

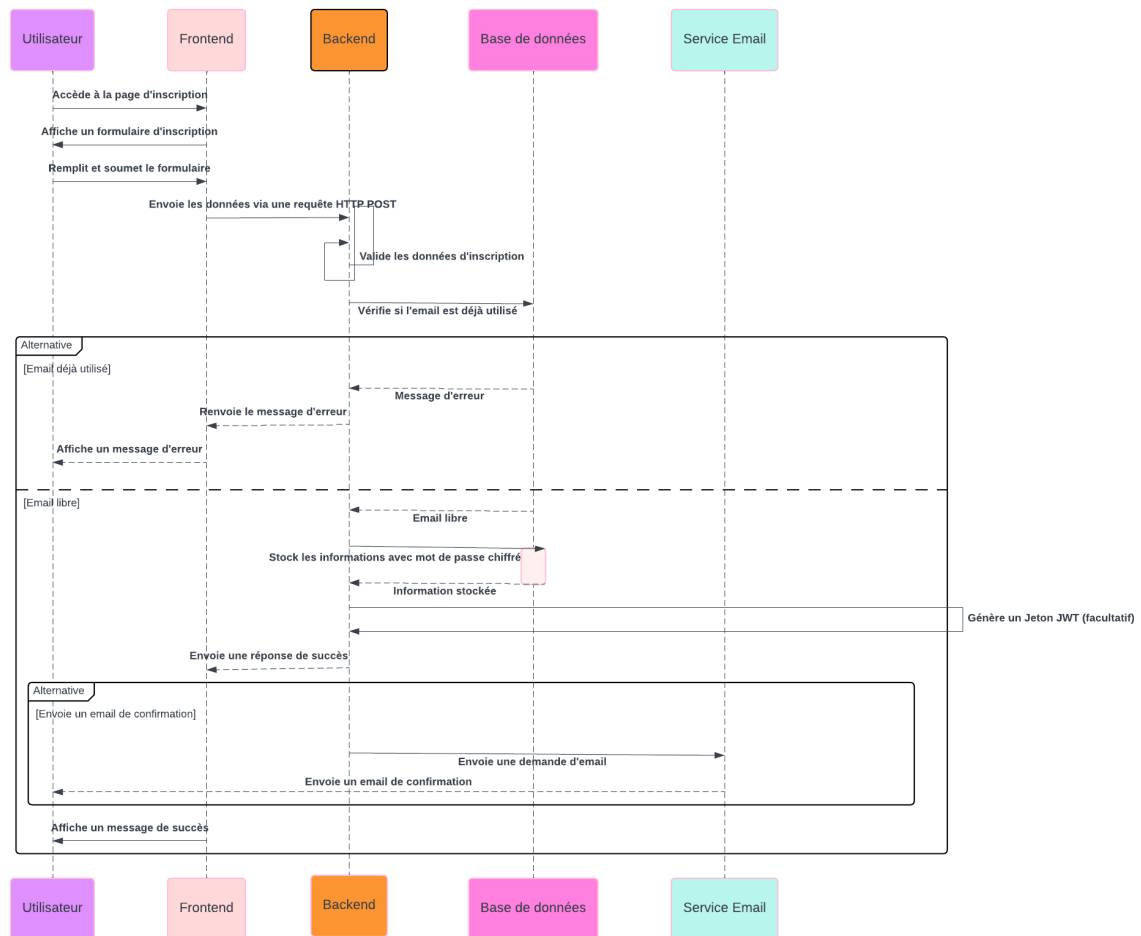
### 3.3.1 Diagramme de séquences " S'inscrire "

Pour avoir la sécurité voulue dans notre projet dans les informations personnelles des utilisateurs, nous avons conçu une authentification et autorisation basées sur les jetons JSON Web token (JWT).

La figure 3.2 illustre le diagramme de séquence et les interactions entre les objets qui se produisent lors du cas d'utilisation « S'inscrire ».

Ce diagramme de séquence illustre les étapes du processus d'inscription, en montrant les interactions entre l'utilisateur, le frontend, le backend, la base de données, et éventuellement un service d'email pour confirmation.

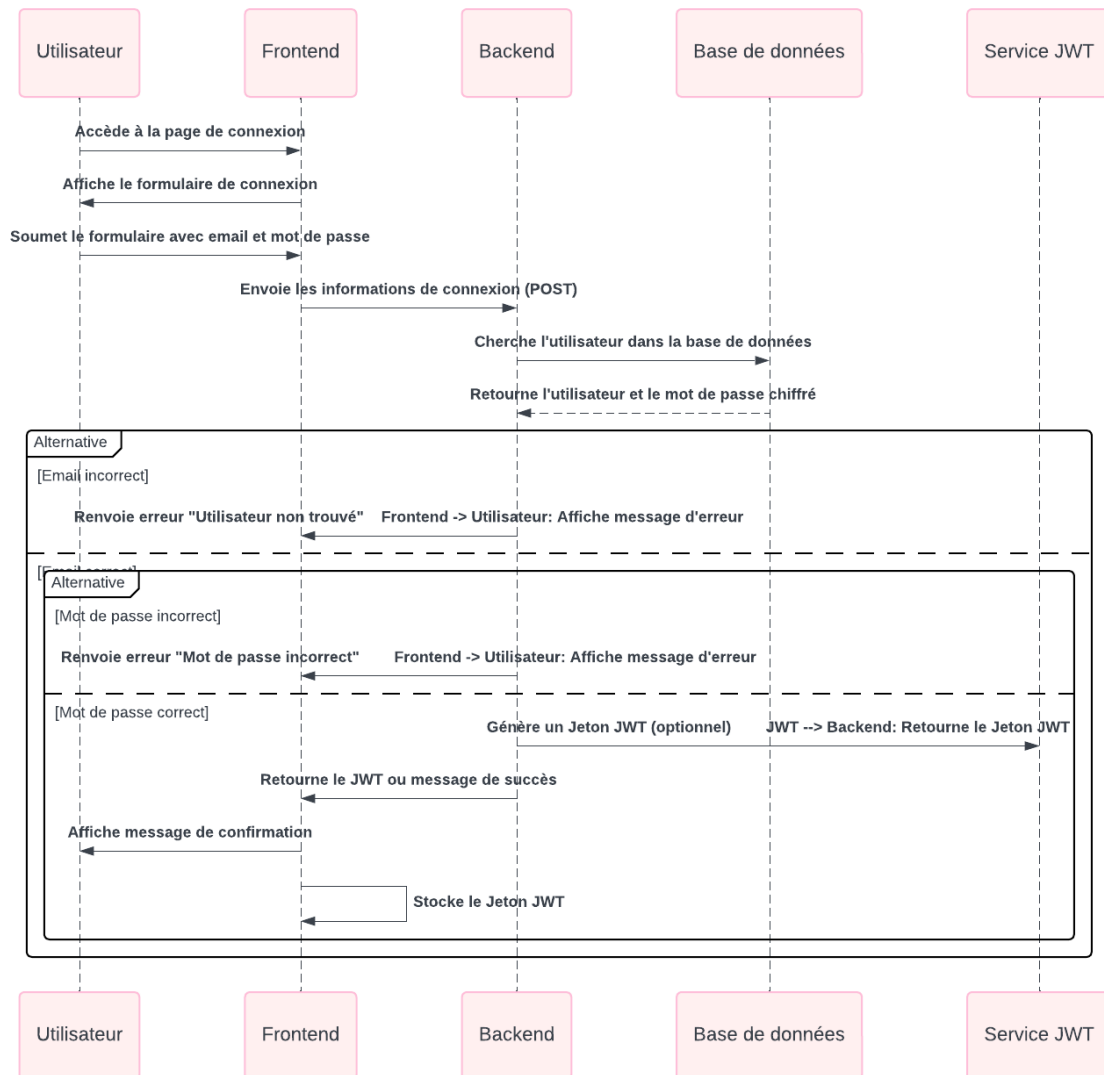
Il inclut également la validation des données et la gestion des erreurs lorsque les informations fournies ne sont pas correctes.



**Figure 3.2** – Diagramme de séquence pour UC1 : Inscription d'un utilisateur

### 3.3.2 Diagramme de séquences " Se Connecter"

Ce diagramme de séquence montre les interactions principales lors du processus de connexion, depuis la soumission du formulaire de connexion jusqu'à la validation des informations dans la base de données et la génération d'un jeton JWT (si utilisé). Il inclut également la gestion des erreurs lorsque les informations fournies sont incorrectes, garantissant une réponse claire à l'utilisateur en cas d'échec.



**Figure 3.3** – Diagramme de séquence pour UC2 : Connexion d'un utilisateur

## 3.4 Réalisation

Dans cette partie, nous allons présenter les interfaces développées dans ce sprint.



— **Inscription :**

- *Introduction* : L'interface d'inscription permet aux visiteurs de créer un compte sur la plateforme TunisiCart. Elle recueille les informations nécessaires telles que le nom, le prénom, l'email et le mot de passe.
- *Interface* :

The screenshot displays the 'Create an account' form on the TunisiCart website. The header includes the 'Tunicart' logo, navigation links for 'Home', 'Products', and 'About', a search bar with a 'Search' button, and 'Sign In' and 'Sign Up' buttons. The form itself is titled 'Create an account' and features a blue arrow icon. It contains several input fields: 'Firstname' (with 'John' entered), 'Lastname' (with 'Doe' entered), 'Email address' (with 'name@example.com' entered), 'Password' (with 'Password' entered), and 'Date of Birth' (with 'jj/mm/aaaa' entered and a calendar icon). There is also a dropdown menu for 'Account type' with the text 'Select account type'. At the bottom of the form, there is a blue button labeled '➔ Create an account' and a link that says 'Already have an account? [Login](#)'.

Figure 3.4 – Interface d'inscription

— **Authentication :**

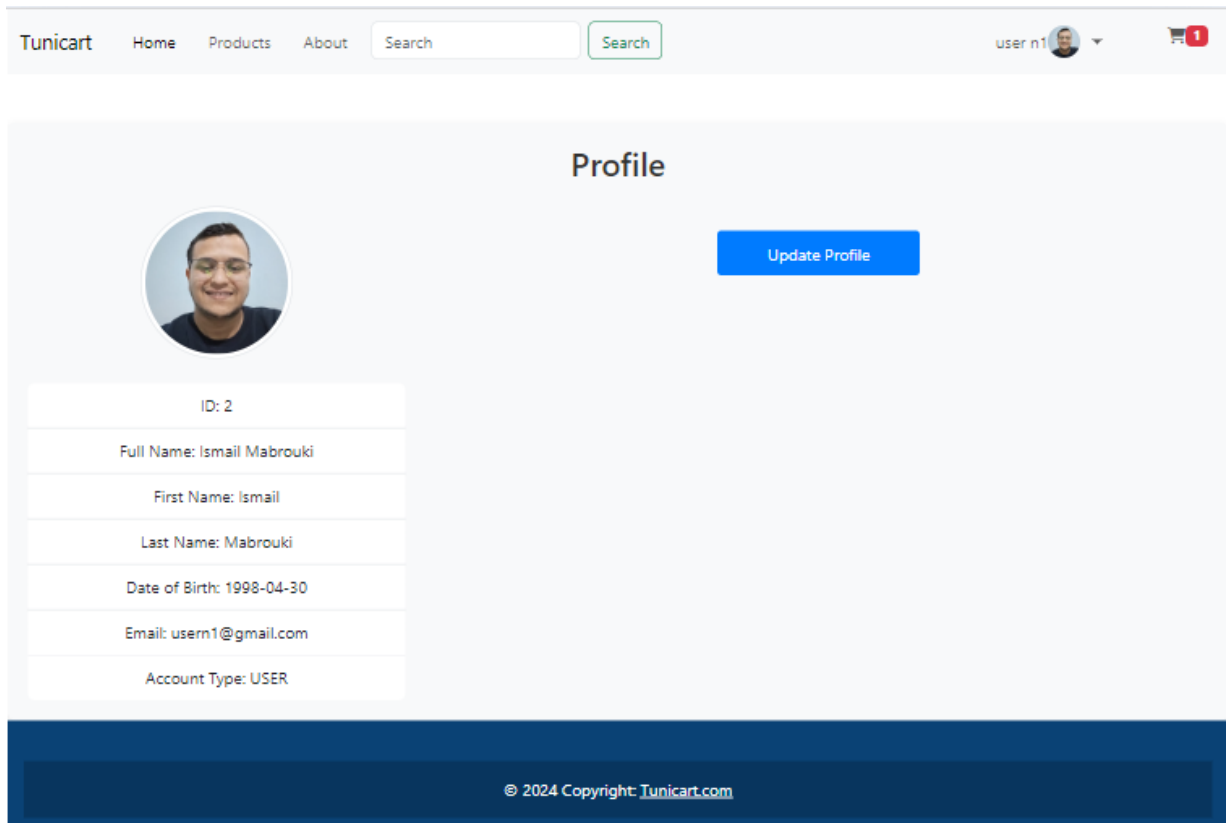
- *Introduction* : L'interface d'authentification permet aux utilisateurs enregistrés de se connecter à leur compte. Les utilisateurs doivent entrer leur email et mot de passe pour accéder aux fonctionnalités réservées.
- *Interface* :

The screenshot displays the Tunicart website's login interface. At the top, a navigation bar includes the 'Tunicart' logo, links for 'Home', 'Products', and 'About', a search bar with a 'Search' button, and 'Sign In' and 'Sign Up' buttons. The main content area features a central white login box titled 'Login to TunisiCart'. This box contains an 'Email address' field with the placeholder 'name@example.com', a 'Password' field with the placeholder 'Password', and a blue 'SIGN IN' button. To the right of the button is a link that says 'Don't have an account? Register'. The footer consists of a dark blue bar with a 'Register for free' link and a 'Sign up!' button, followed by a copyright notice: '© 2024 Copyright: Tunicart.com'.

**Figure 3.5** – Interface d'authentification

— **Profil utilisateur :**

- *Introduction* : L'interface de gestion du profil utilisateur permet aux utilisateurs de mettre à jour leurs informations personnelles, telles que l'adresse et le numéro de téléphone, pour maintenir leurs données à jour.
- *Interface 1* :



**Figure 3.6** – Interface de profil utilisateur

— *Interface 2 :*

The screenshot displays the 'Tunicart' web application interface. At the top, a navigation bar includes the 'Tunicart' logo, links for 'Home', 'Products', and 'About', a search bar with a 'Search' button, and a user profile section labeled 'user n1' with a dropdown arrow and a shopping cart icon showing '1' item. Below the navigation bar, the 'Upload Profile Image' section features a 'Profile Image:' label, a file selection button 'Choisir un fichier', a text field 'Aucun fichier choisi', and a blue 'Upload Image' button. The main content area is titled 'Profile' and contains a form with the following fields: 'ID' (with a cursor), 'First Name' (filled with 'Ismail'), 'Last Name' (filled with 'Mabrouki'), 'Password' (filled with '\*\*\*\*\*'), 'Date of Birth' (filled with '30/04/1998' and a calendar icon), and 'Email' (filled with 'usern1@gmail.com'). At the bottom of the form are 'Update' and 'Cancel' buttons. A dark blue footer bar is visible at the very bottom.

**Figure 3.7** – Interface de mise à jour de profil utilisateur

Ce sprint a permis de poser les bases de la gestion des utilisateurs sur la plateforme TunisiCart, préparant le terrain pour les fonctionnalités à venir dans les prochains sprints.

# Quatrième Chapitre

## Sprint 2

### Introduction

Après le succès du premier sprint, le deuxième sprint de TunisiCart se concentre sur l'ajout de nouvelles fonctionnalités clés, telles que l'ajout de produits par les vendeurs et la gestion des commandes par les clients. Ce sprint a pour objectif d'améliorer l'expérience utilisateur en permettant une gestion efficace des produits et un suivi transparent des commandes.

Dans ce sprint, nous allons commencer par la spécification et l'analyse des besoins pour ces nouvelles fonctionnalités. Ensuite, nous détaillerons les cas d'utilisation associés, illustrerons les interactions à travers des diagrammes de séquence, et enfin, nous présenterons la réalisation technique de ces fonctionnalités. Ce travail permettra d'élargir les capacités de la plateforme et d'améliorer sa robustesse face aux exigences croissantes des utilisateurs.

### Spécification et analyse des besoins

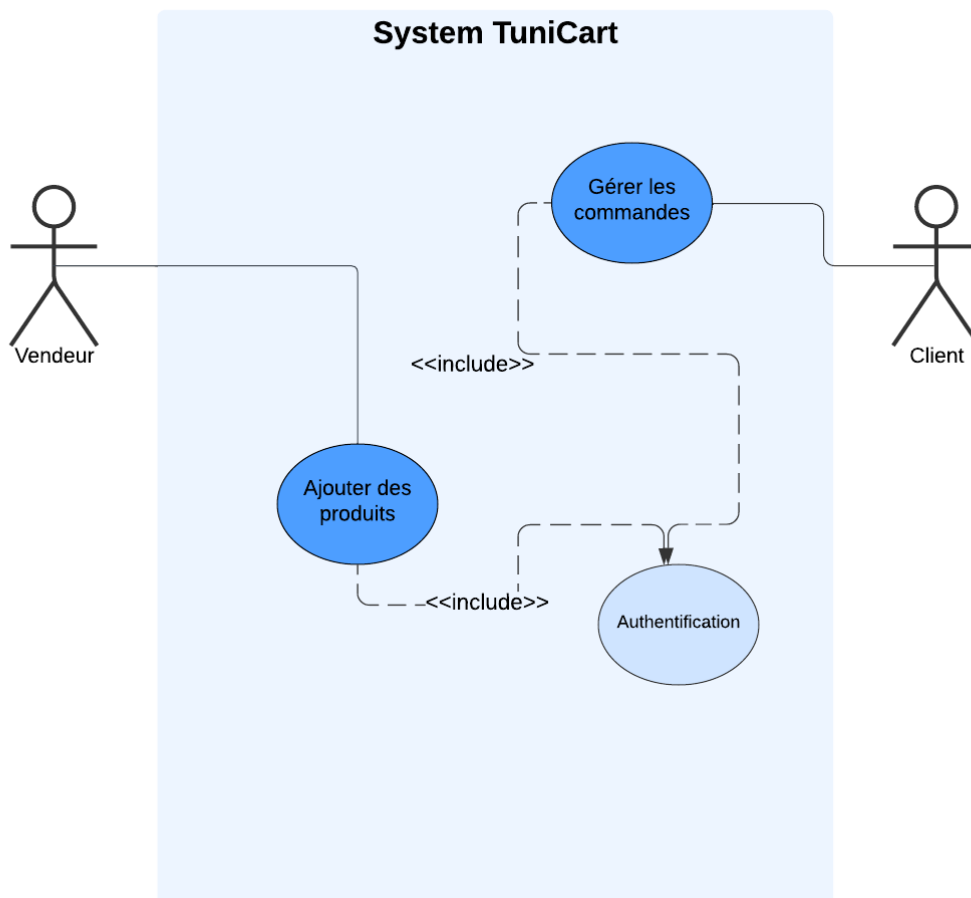
Dans cette partie, nous allons entamer l'analyse et la spécification des besoins de ce sprint. Nous allons commencer par présenter le backlog, ensuite établir le diagramme de cas d'utilisation raffiné, et finir par détailler la description des cas d'utilisation.

### 4.0.1 Backlog du Sprint 2

| ID  | User Story   | Priorité (MoSCoW) |
|-----|--|-------------------|
| 2.0 | En tant que vendeur, je souhaite ajouter des produits à ma boutique pour les vendre sur la plateforme. | M                 |
| 2.1 | En tant que client, je souhaite gérer mes commandes pour suivre leur statut et historique.             | M                 |

**Table 4.1** – Backlog du Sprint 2 basé sur les User Stories pour le projet TunisiCart

### 4.0.2 Diagramme de Cas d’Utilisation Raffiné : Sprint 2



**Figure 4.1** – Diagramme de Cas d’Utilisation Raffiné pour le Sprint 2

## 4.1 Description détaillée des cas d'utilisation

### 1. Ajouter des produits

**Description :** Un vendeur souhaite ajouter de nouveaux produits à sa boutique sur la plateforme TunisiCart pour les vendre aux clients.

| Élément                    | Détails  |
|----------------------------|--|
| <b>Titre</b>               | Ajouter des produits   |
| <b>Acteurs</b>             | Vendeur  |
| <b>Préconditions</b>       | Le vendeur doit être authentifié et avoir un compte valide sur la plateforme.  |
| <b>Postconditions</b>      | Le produit est ajouté avec succès à la boutique du vendeur et est visible pour les clients.  |
| <b>Scénario Nominal</b>    | <ol style="list-style-type: none"> <li>1. Le vendeur accède à la section "Ajouter un produit".</li> <li>2. Le vendeur saisit les détails du produit (nom, description, prix, etc.).</li> <li>3. Le vendeur télécharge des images du produit.</li> <li>4. Le vendeur soumet le formulaire pour ajouter le produit.</li> <li>5. Le système enregistre les informations et rend le produit visible dans la boutique.</li> </ol> |
| <b>Scénario Alternatif</b> | Si des informations sont manquantes ou invalides, le système affiche un message d'erreur demandant de les corriger avant de soumettre à nouveau.   |

**Table 4.2** – Cas d'utilisation : Ajouter des produits

### 2. Gérer les commandes

**Description :** Un client souhaite gérer ses commandes sur la plateforme TunisiCart, incluant la visualisation du statut des commandes et l'historique des commandes passées.

| Élément                    | Détails  |
|----------------------------|--|
| <b>Titre</b>               | Gérer les commandes  |
| <b>Acteurs</b>             | Client   |
| <b>Préconditions</b>       | Le client doit être authentifié et avoir passé au moins une commande.  |
| <b>Postconditions</b>      | Le client peut visualiser le statut actuel et l'historique de ses commandes.   |
| <b>Scénario Nominal</b>    | <ol style="list-style-type: none"> <li>1. Le client accède à la section "Mes commandes".</li> <li>2. Le client voit la liste de ses commandes récentes.</li> <li>3. Le client sélectionne une commande pour voir les détails.</li> <li>4. Le système affiche le statut actuel de la commande et l'historique des statuts.</li> <li>5. Le client peut filtrer et rechercher dans l'historique des commandes.</li> </ol> |
| <b>Scénario Alternatif</b> | Si le client n'a pas encore passé de commande, le système affiche un message indiquant qu'aucune commande n'est disponible.  |

**Table 4.3** – Cas d'utilisation : Gérer les commandes

## 4.2 Diagrammes de séquence

Les diagrammes de séquence ci-dessous montrent l'interaction entre les différents acteurs et le système pour les cas d'utilisation définis :

### 4.2.1 Diagramme de séquence "Gérer les commandes"

Le diagramme de séquence décrit ici représente les interactions entre le client, le frontend, le backend et la base de données lors de la gestion des commandes. Le client peut voir l'état actuel de ses commandes, accéder aux détails et consulter l'historique. Un scénario alternatif est prévu si aucune commande n'a été passée, et un message en informe le client

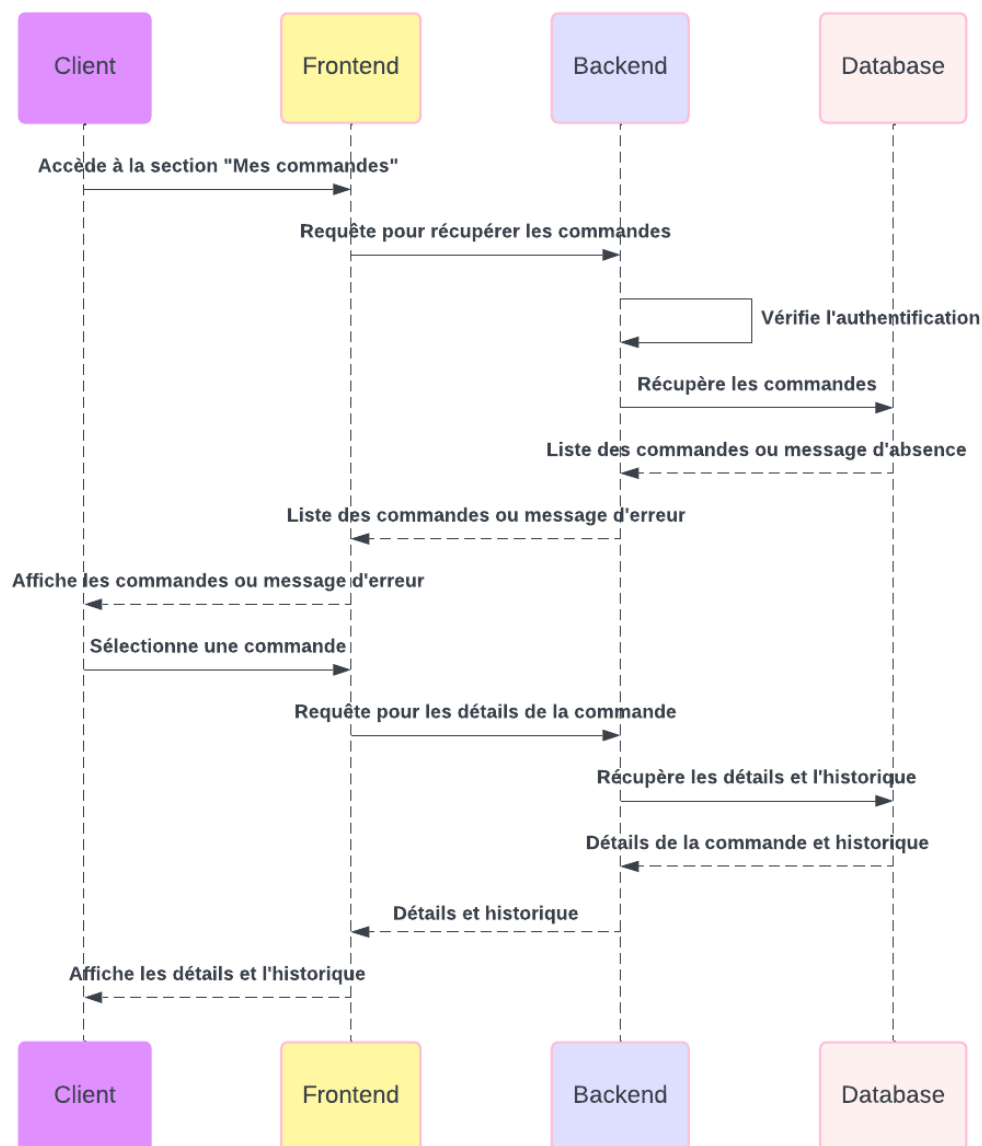


Figure 4.2 – Diagramme de séquence pour UC4 : Gérer les commandes



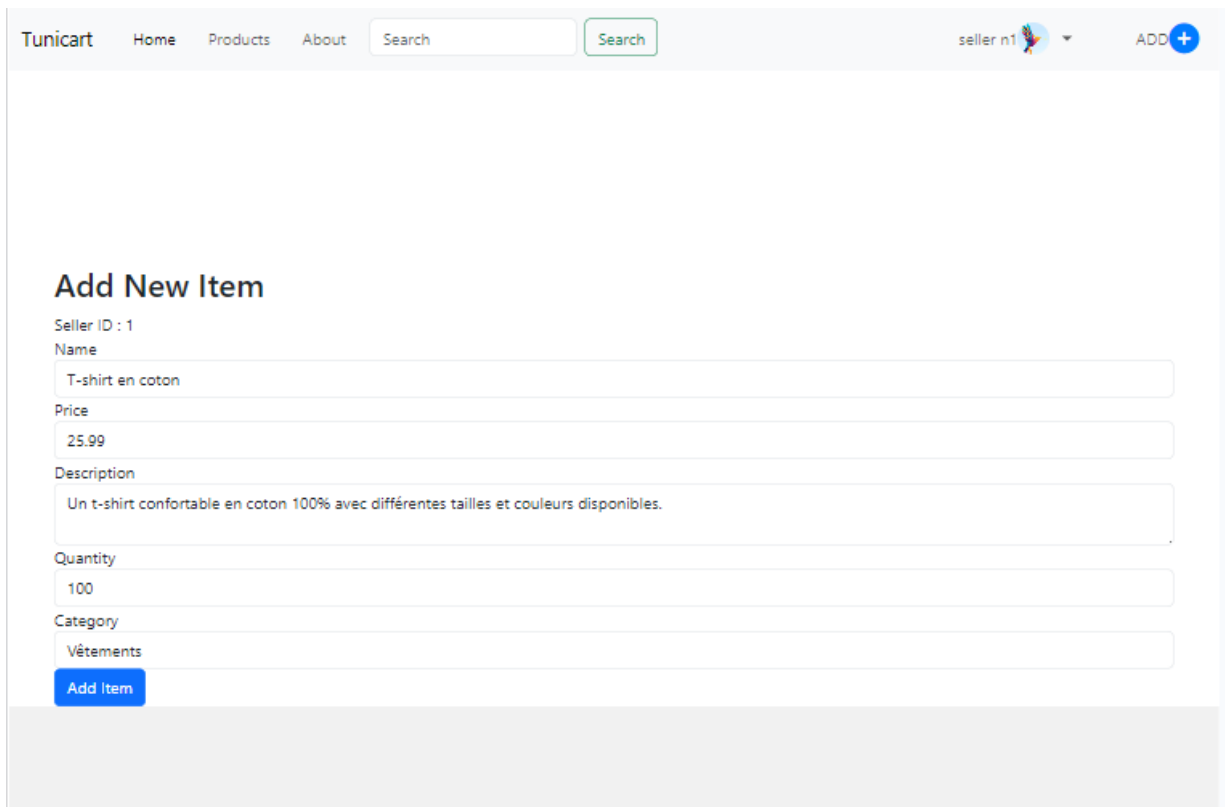
## 4.3 Réalisation

Dans cette partie, nous allons présenter les interfaces développées dans ce sprint.

— **Ajout de produits :**

- *Introduction* : L'interface d'ajout de produits permet aux vendeurs de créer de nouvelles fiches produits pour les vendre sur la plateforme. Les vendeurs peuvent saisir des informations telles que le nom, la description, le prix et télécharger des images du produit.

- *Interface1* :



The screenshot shows the 'Add New Item' form in the Tunicart application. The header includes the Tunicart logo, navigation links (Home, Products, About), a search bar, and user information (seller n1, ADD button). The form fields are as follows:

- Seller ID :** 1
- Name:** T-shirt en coton
- Price:** 25.99
- Description:** Un t-shirt confortable en coton 100% avec différentes tailles et couleurs disponibles.
- Quantity:** 100
- Category:** Vêtements

An 'Add Item' button is located at the bottom of the form.

**Figure 4.3** – Interface d'ajout de produits

— *Interface2 :*

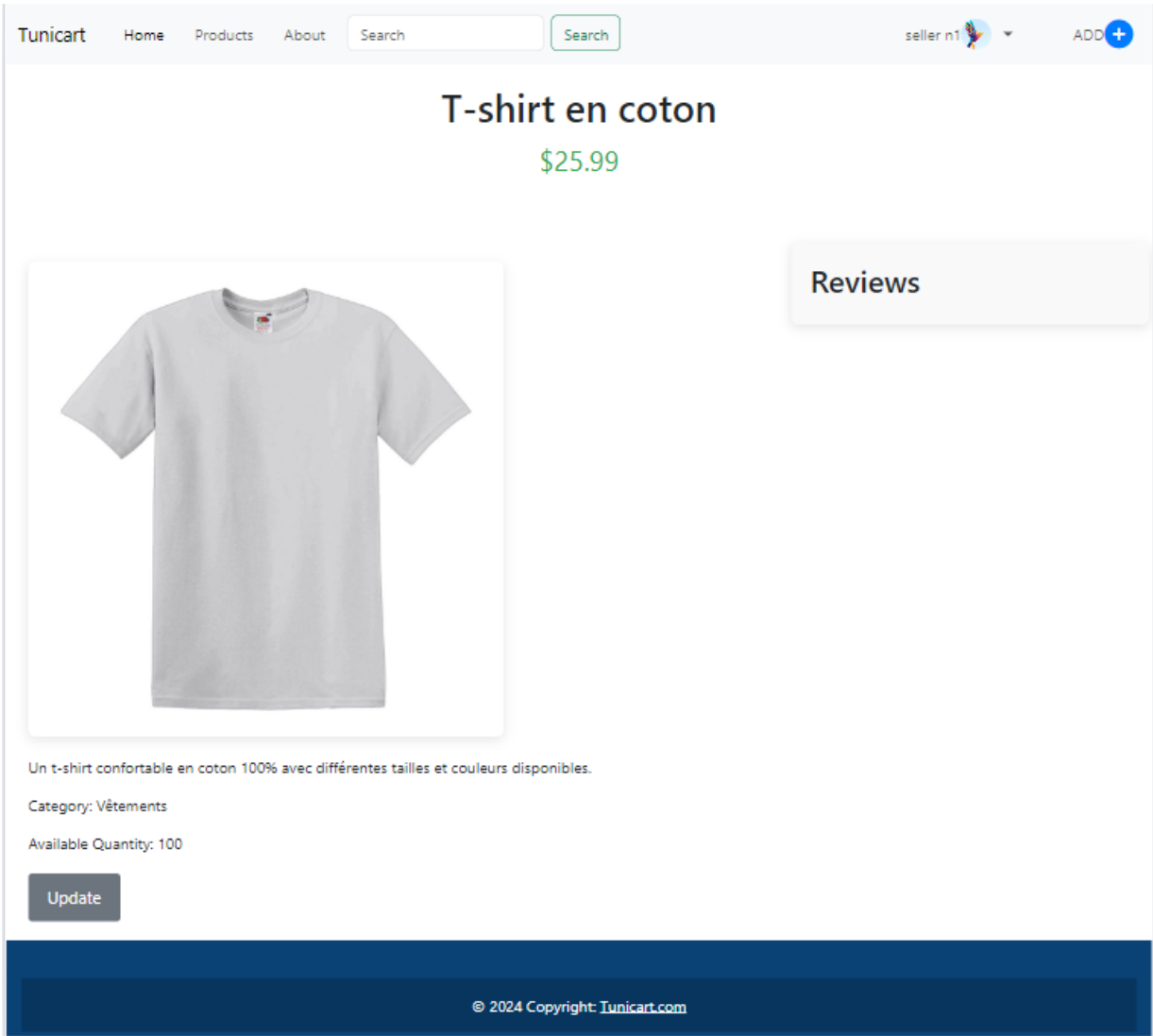
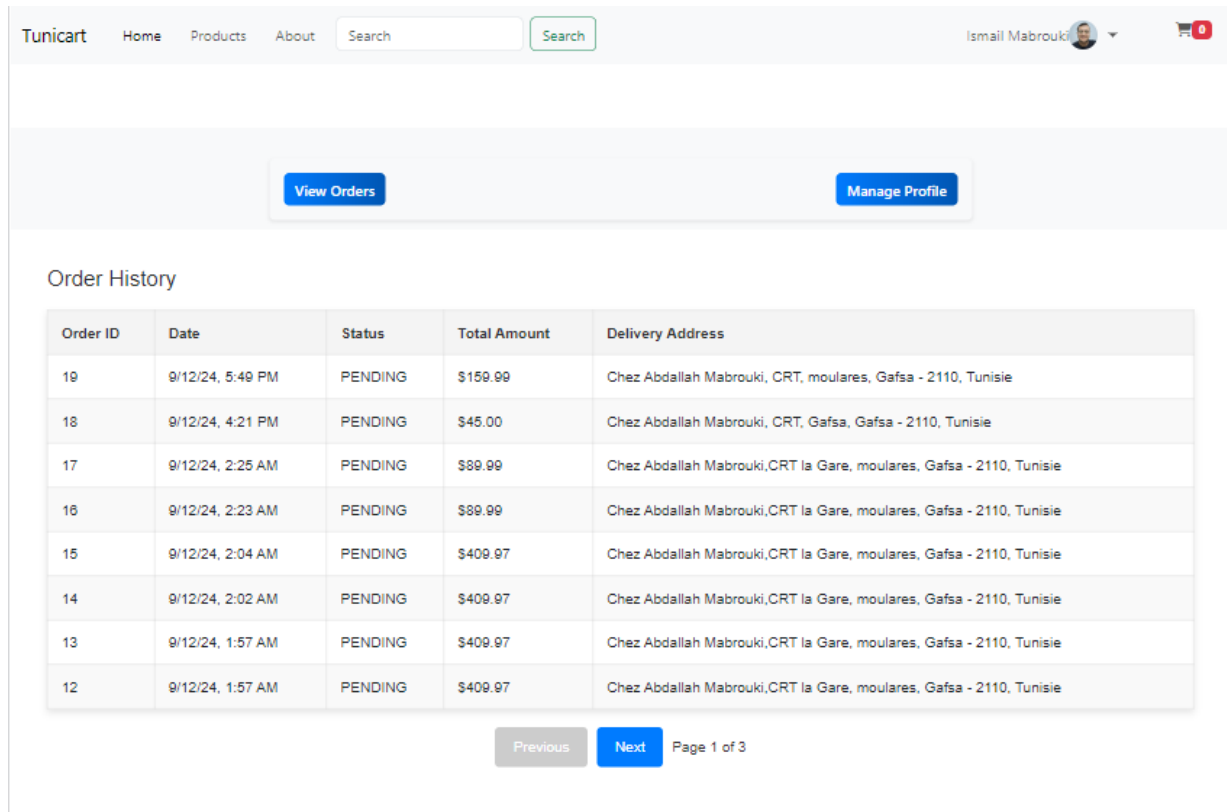


Figure 4.4 – Interface de produit Ajouté

— **Gestion des commandes :**

- *Introduction* : L'interface de gestion des commandes permet aux clients de suivre leurs commandes passées, de voir les détails et le statut de chaque commande, et de gérer les retours ou échanges.
- *Interface* :



**Figure 4.5** – Interface de gestion des commandes

Ce sprint a permis d'enrichir la plateforme TunisiCart avec des fonctionnalités cruciales pour la gestion des produits et a préparé la base pour les futures améliorations du système.

# Cinquième Chapitre

## Sprint 3

### Introduction

Le troisième sprint du projet TunisiCart se concentre sur l'amélioration de l'expérience utilisateur à travers l'ajout de fonctionnalités cruciales. Ces fonctionnalités incluent la gestion des avis, permettant aux clients de partager leurs expériences, l'intégration d'un système de paiement sécurisé pour assurer la sécurité des transactions en ligne, et l'implémentation de la recherche de produits pour faciliter la navigation sur la plateforme. Ce sprint vise à affiner la plateforme en améliorant l'interaction utilisateur et la fiabilité des services offerts.

Nous commencerons par spécifier et analyser les besoins pour ces nouvelles fonctionnalités, suivis des cas d'utilisation détaillés, des diagrammes de séquence, et enfin de la réalisation technique.

### 5.1 Spécification et analyse des besoins

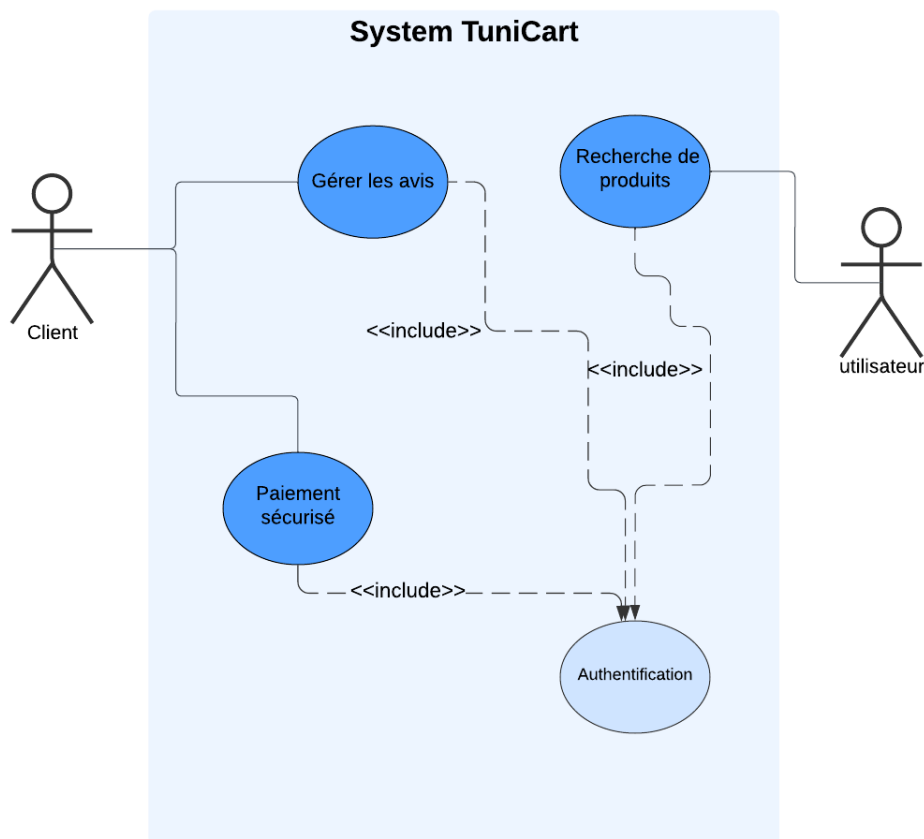
Dans cette partie, nous allons entamer l'analyse et la spécification des besoins de ce sprint. Nous allons commencer par présenter le backlog, ensuite établir le diagramme de cas d'utilisation raffiné, et finir par détailler la description des cas d'utilisation.

### 5.1.1 Backlog du Sprint 3

| ID  | User Story  | Priorité (MoSCoW) |
|-----|---|-------------------|
| 3.0 | En tant que client, je souhaite laisser un avis sur un produit pour partager mon expérience avec les autres utilisateurs. | C                 |
| 3.1 | En tant que client, je souhaite effectuer des paiements en ligne de manière sécurisée pour finaliser mes achats.          | M                 |
| 3.2 | En tant qu'utilisateur, je souhaite rechercher des produits spécifiques pour trouver ce que je cherche rapidement.        | S                 |

**Table 5.1** – Backlog du Sprint 3 basé sur les User Stories pour le projet TunisiCart

### 5.1.2 Diagramme de Cas d'Utilisation Raffiné : Sprint 3



**Figure 5.1** – Diagramme de Cas d'Utilisation Raffiné pour le Sprint 3

## 5.2 Description détaillée des cas d'utilisation

### 1. Gérer les avis

**Description :** Un client souhaite laisser un avis sur un produit pour partager son expérience avec les autres utilisateurs. Ce cas d'utilisation permet aux clients d'exprimer leur opinion et d'évaluer les produits achetés.

| Élément             | Détails  |
|---------------------|--|
| Titre               | Gérer les avis   |
| Acteurs             | Client   |
| Préconditions       | Le client doit être connecté à son compte  |
| Postconditions      | Un avis est ajouté au produit avec succès  |
| Scénario Nominal    | <ol style="list-style-type: none"> <li>1. Le client accède à la page du produit.</li> <li>2. Le client sélectionne l'option pour laisser un avis.</li> <li>3. Le client saisit son avis et sa note dans le formulaire.</li> <li>4. Le client soumet l'avis.</li> <li>5. Le système enregistre l'avis et le rend visible pour les autres utilisateurs.</li> </ol> |
| Scénario Alternatif | Si le formulaire d'avis est incomplet ou incorrect (e.g., note manquante), le système affiche un message d'erreur et demande des corrections.  |

**Table 5.2** – Cas d'utilisation : Gérer les avis

### 2. Paiement sécurisé

**Description :** Un client souhaite effectuer un paiement en ligne de manière sécurisée pour finaliser ses achats sur la plateforme. Ce cas d'utilisation garantit que les informations de paiement sont traitées en toute sécurité.

| Élément             | Détails  |
|---------------------|--|
| Titre               | Paiement sécurisé  |
| Acteurs             | Client   |
| Préconditions       | Le client doit avoir un panier avec des articles à acheter et être connecté à son compte   |
| Postconditions      | Le paiement est effectué avec succès et la commande est confirmée  |
| Scénario Nominal    | <ol style="list-style-type: none"> <li>1. Le client accède à la page de paiement.</li> <li>2. Le client saisit les informations de paiement dans le formulaire sécurisé.</li> <li>3. Le client soumet les informations de paiement.</li> <li>4. Le système vérifie et traite les informations de paiement.</li> <li>5. Le système confirme la transaction et envoie une confirmation de commande au client.</li> </ol> |
| Scénario Alternatif | Si les informations de paiement sont incorrectes ou rejetées (e.g., carte de crédit expirée), le système affiche un message d'erreur et demande de ressaisir les informations.   |

**Table 5.3** – Cas d'utilisation : Paiement sécurisé

### 3. Recherche de produits

**Description :** Un utilisateur souhaite rechercher des produits spécifiques sur la plateforme pour trouver rapidement ce qu'il cherche. Ce cas d'utilisation permet d'affiner les résultats de recherche selon les critères définis par l'utilisateur.

| Élément             | Détails  |
|---------------------|--|
| Titre               | Recherche de produits  |
| Acteurs             | Utilisateur  |
| Préconditions       | L'utilisateur doit être connecté ou non pour accéder à la fonctionnalité de recherche  |
| Postconditions      | Les résultats de recherche sont affichés selon les critères spécifiés  |
| Scénario Nominal    | <ol style="list-style-type: none"><li>1. L'utilisateur accède à la barre de recherche sur la plateforme.</li><li>2. L'utilisateur saisit ses critères de recherche dans le champ prévu à cet effet.</li><li>3. L'utilisateur soumet la recherche.</li><li>4. Le système affiche les résultats de recherche correspondant aux critères spécifiés.</li></ol> |
| Scénario Alternatif | Si aucun produit ne correspond aux critères de recherche, le système affiche un message indiquant qu'aucun résultat n'a été trouvé.  |

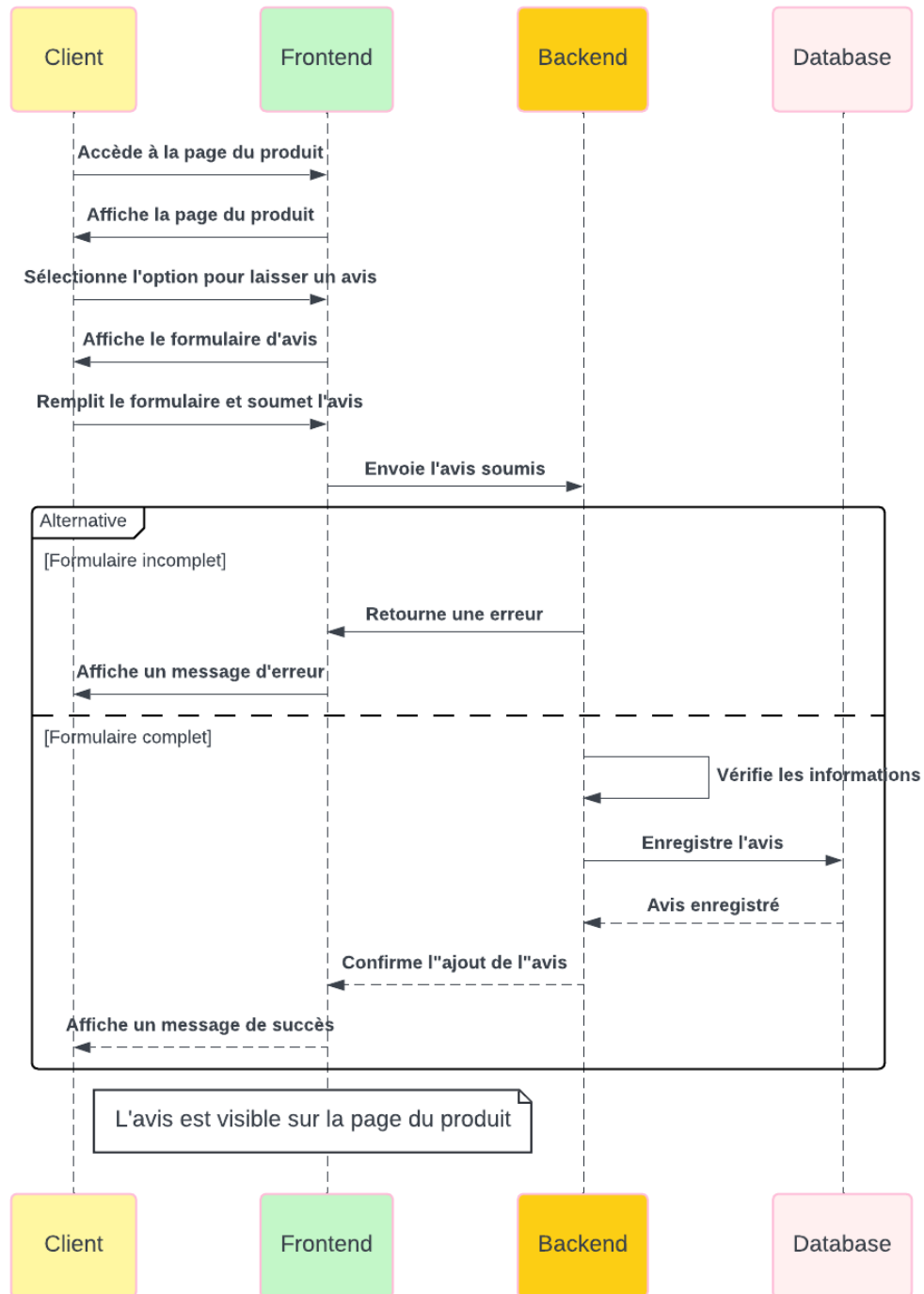
**Table 5.4** – Cas d'utilisation : Recherche de produits

## 5.3 Diagrammes de séquence

Les diagrammes de séquence ci-dessous montrent l'interaction entre les différents acteurs et le système pour les cas d'utilisation définis :

### 5.3.1 Diagramme de séquence "Gérer les avis"

Ce diagramme de séquence décrit les interactions lors de la gestion des avis sur la plateforme. Le client peut soumettre un avis sur un produit, et le backend vérifie les informations avant de les enregistrer dans la base de données. Si des informations sont manquantes, un message d'erreur est affiché pour que le client puisse corriger le formulaire.

**Figure 5.2** – Diagramme de séquence pour UC7 : Gérer les avis



### 5.3.2 Diagramme de séquence "Paiement sécurisé"

Ce diagramme de séquence décrit les interactions lors d'un paiement sécurisé. Le client saisit ses informations de paiement, qui sont vérifiées par un service tiers sécurisé. En cas de succès, la commande est confirmée et le paiement est enregistré. Si le paiement échoue, un message d'erreur est renvoyé pour que le client puisse corriger ses informations.

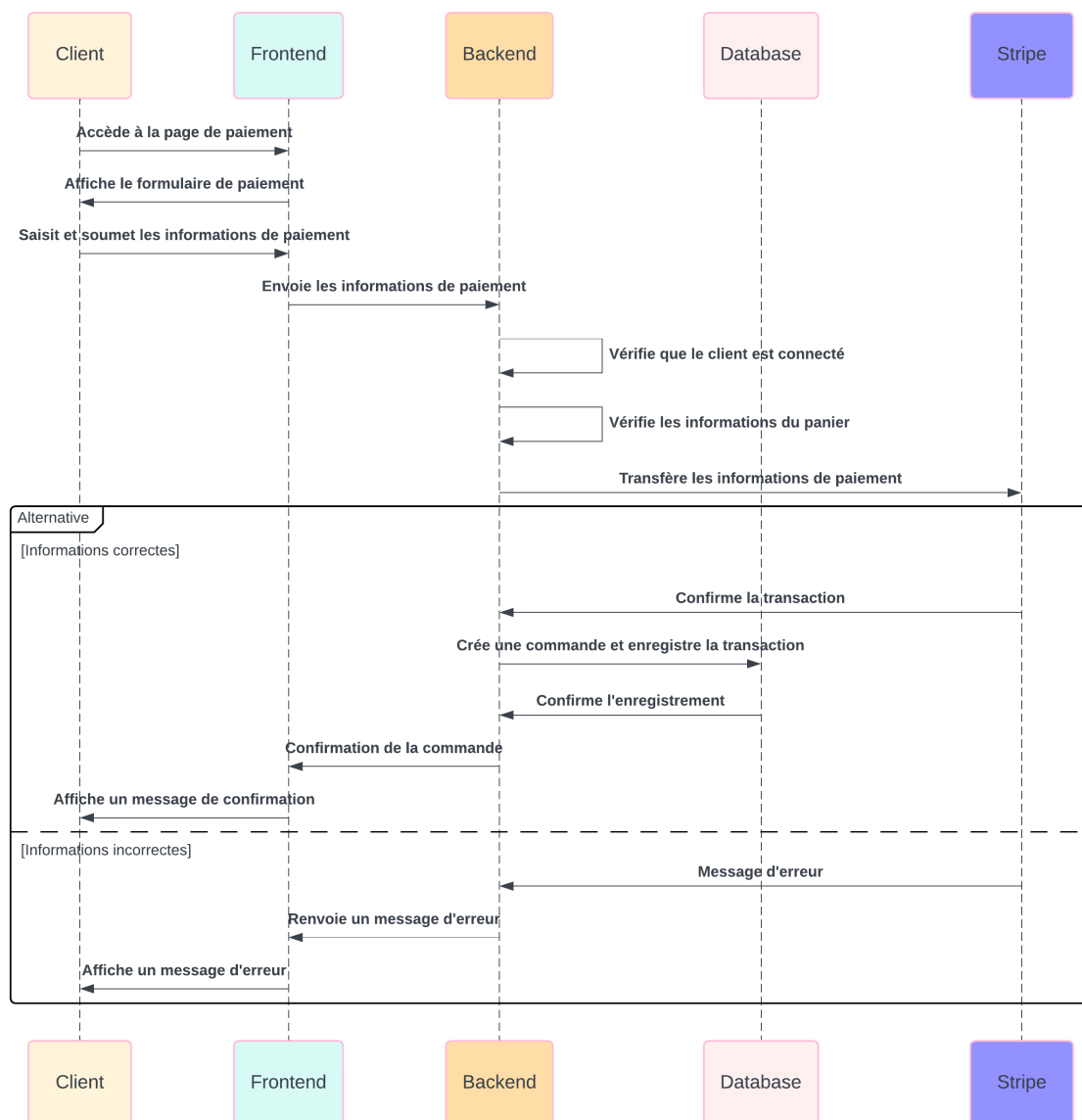


Figure 5.3 – Diagramme de séquence pour UC8 : Paiement sécurisé

### 5.3.3 Diagramme de séquence "Recherche de produits"

Ce diagramme de séquence décrit les interactions lors de la recherche de produits sur la plateforme. L'utilisateur saisit les critères de recherche dans la barre dédiée, les résultats sont ensuite affichés selon les critères définis, ou un message d'erreur apparaît s'il n'y a aucun produit correspondant.

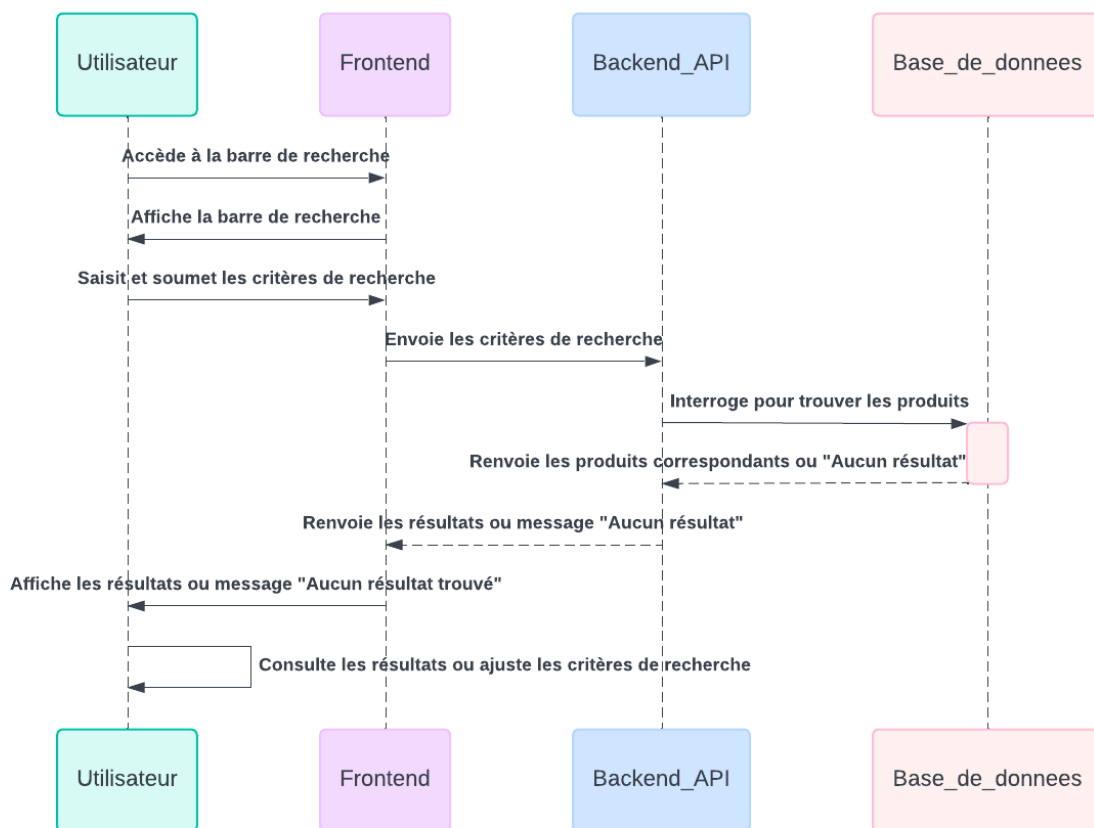


Figure 5.4 – Diagramme de séquence pour UC9 : Laisser un avis sur un produit

## 5.4 Réalisation


Ce sprint a été marqué par l'implémentation des fonctionnalités de gestion des commandes, de suivi des livraisons, et d'avis utilisateurs, avec une attention particulière à l'intégration de ces fonctionnalités dans le système existant.

— **Gestion des avis :**

- *Introduction* : L'interface de gestion des avis permet aux clients de laisser des avis et des notes sur les produits qu'ils ont achetés. Cette fonctionnalité aide à partager les expériences et à guider les autres utilisateurs dans leurs choix.
- *Interface 1* :

### T-shirt en coton

\$25.99



#### Reviews

##### Add a Review

Rating:

4 / 5

Comment:

good t-shirt.

Submit Review

Un t-shirt confortable en coton 100% avec différentes tailles et couleurs disponibles.

Category: Vêtements

Available Quantity: 100

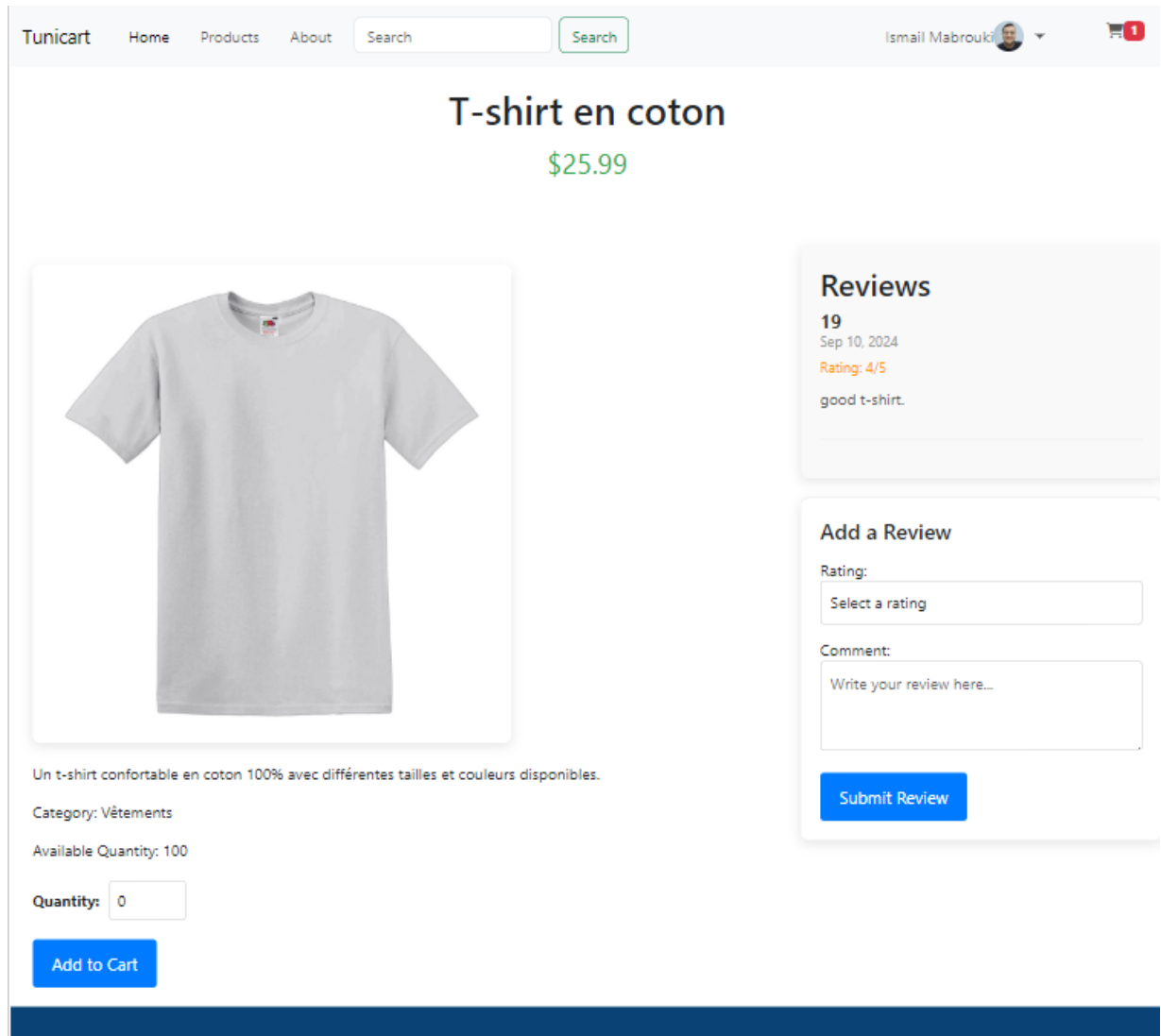
Quantity:

Add to Cart

© 2024 Copyright: [Tunicart.com](#)

Figure 5.5 – Interface d'ajout des avis

— *Interface 2 :*



**Figure 5.6** – Interface de gestion des avis

— **Paiement sécurisé :**

— *Introduction :* L'interface de paiement sécurisé permet aux clients de régler leurs achats en ligne en toute sécurité. Elle prend en charge les paiements par carte bancaire et autres méthodes sécurisées.

— *Interface 1 :*

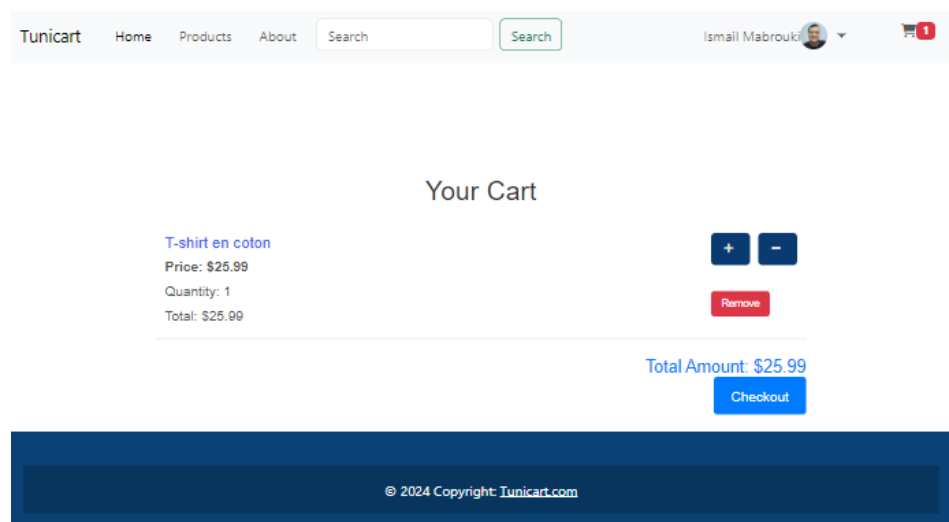


Figure 5.7 – Interface de cart

— *Interface 2 :*

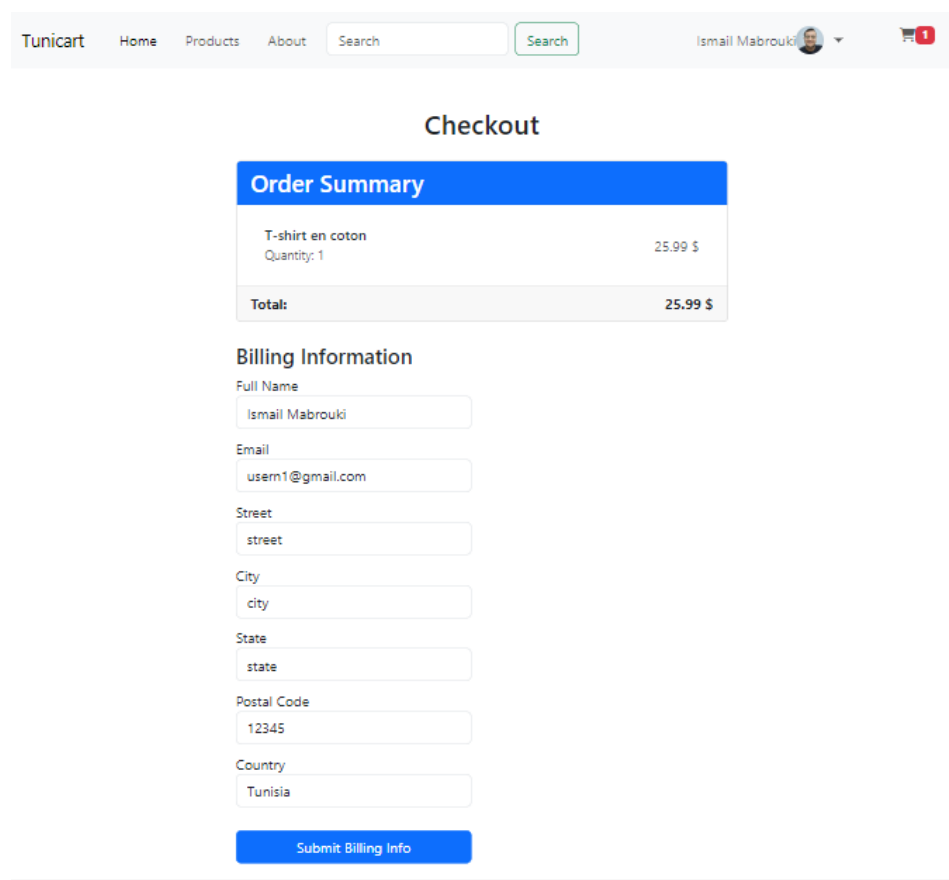


Figure 5.8 – Interface de checkout

— *Interface 3 :*

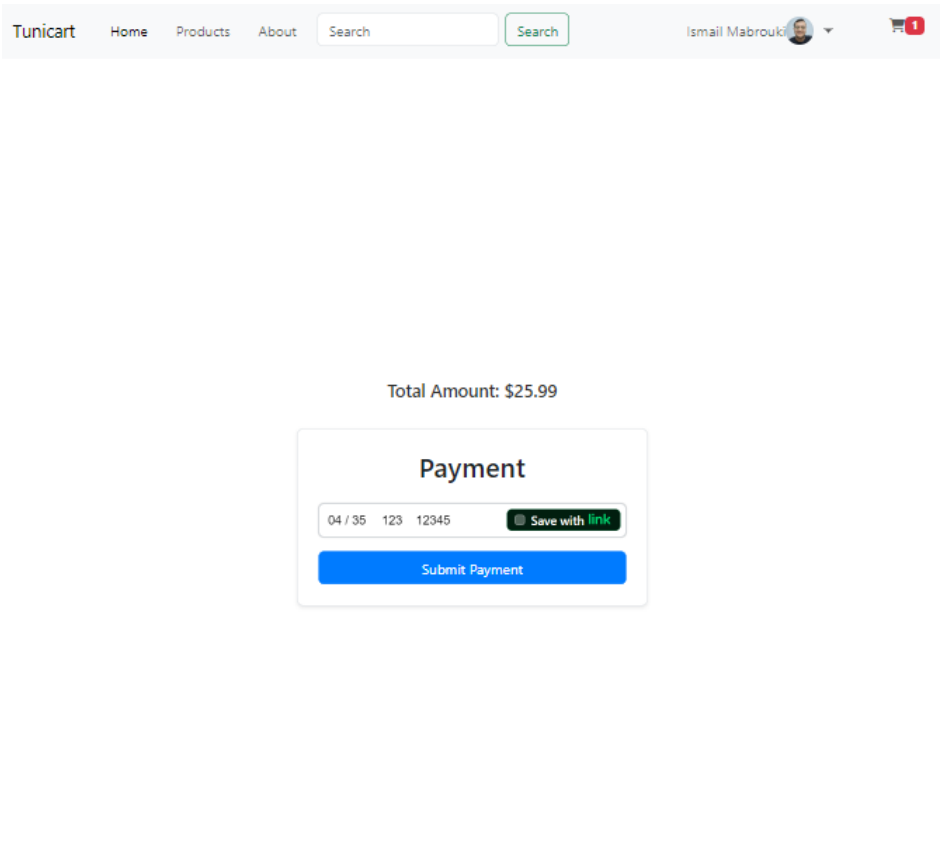


Figure 5.9 – Interface de paiement sécurisé

— *Interface 4 :*

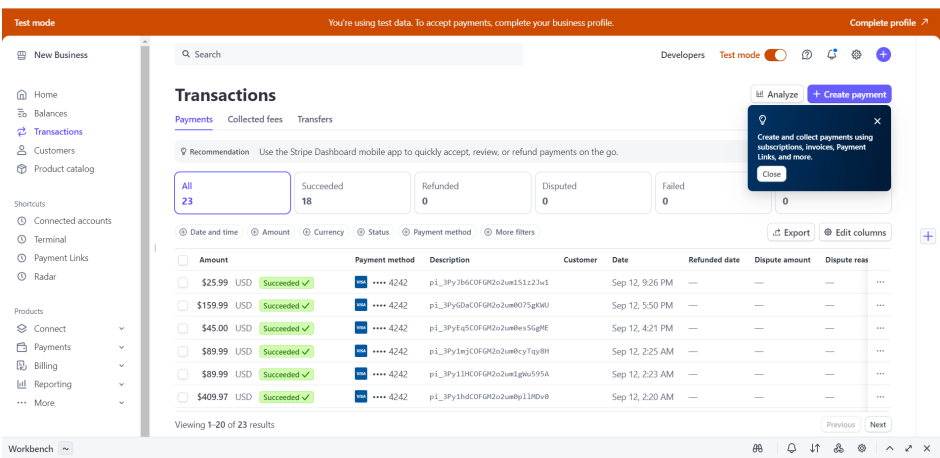


Figure 5.10 – Interface de stripe dashboard

— Recherche de produits :

— *Introduction* : L'interface de recherche de produits permet aux utilisateurs de rechercher des articles spécifiques à l'aide de mots-clés.

— *Interface 1* :

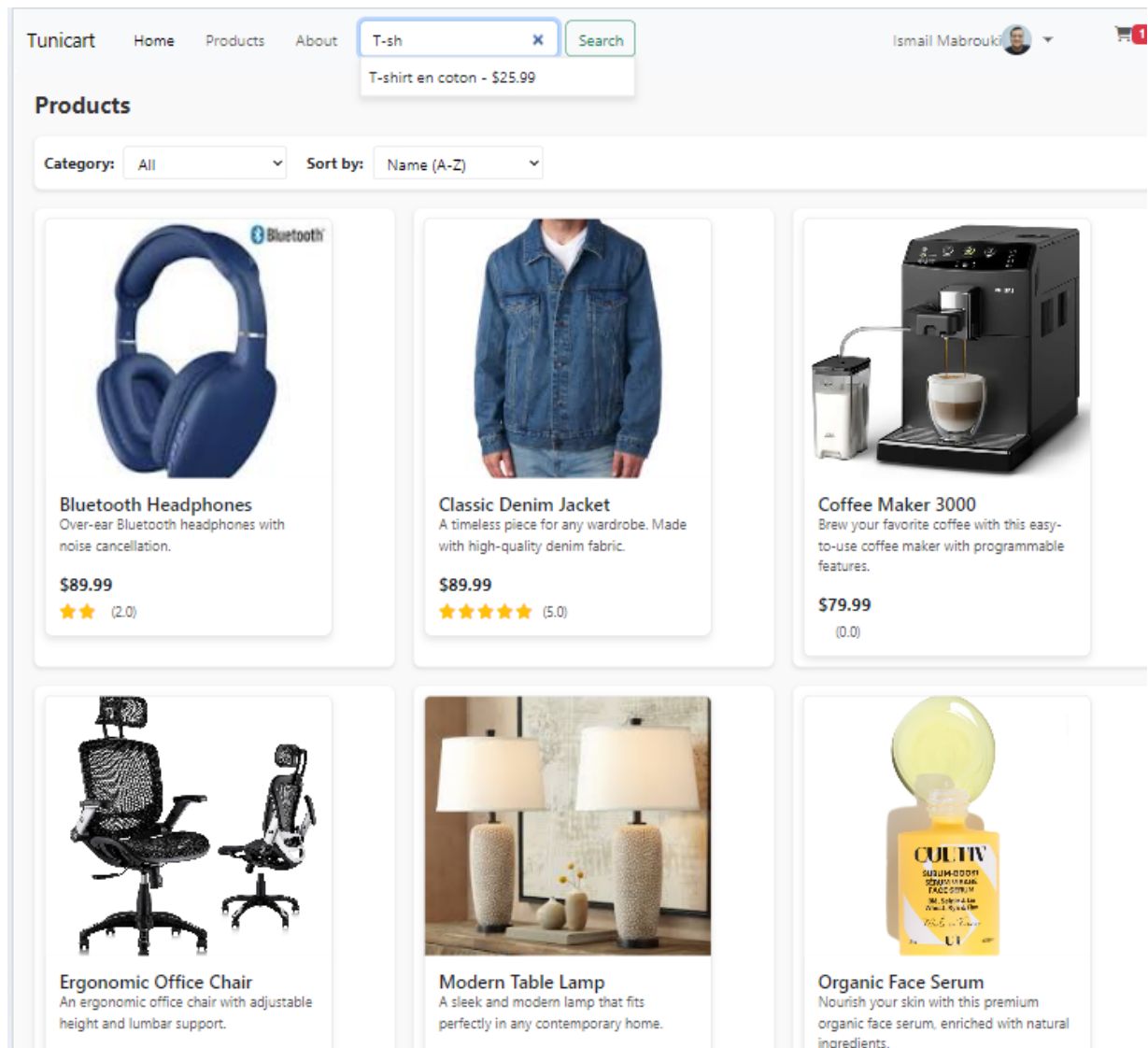
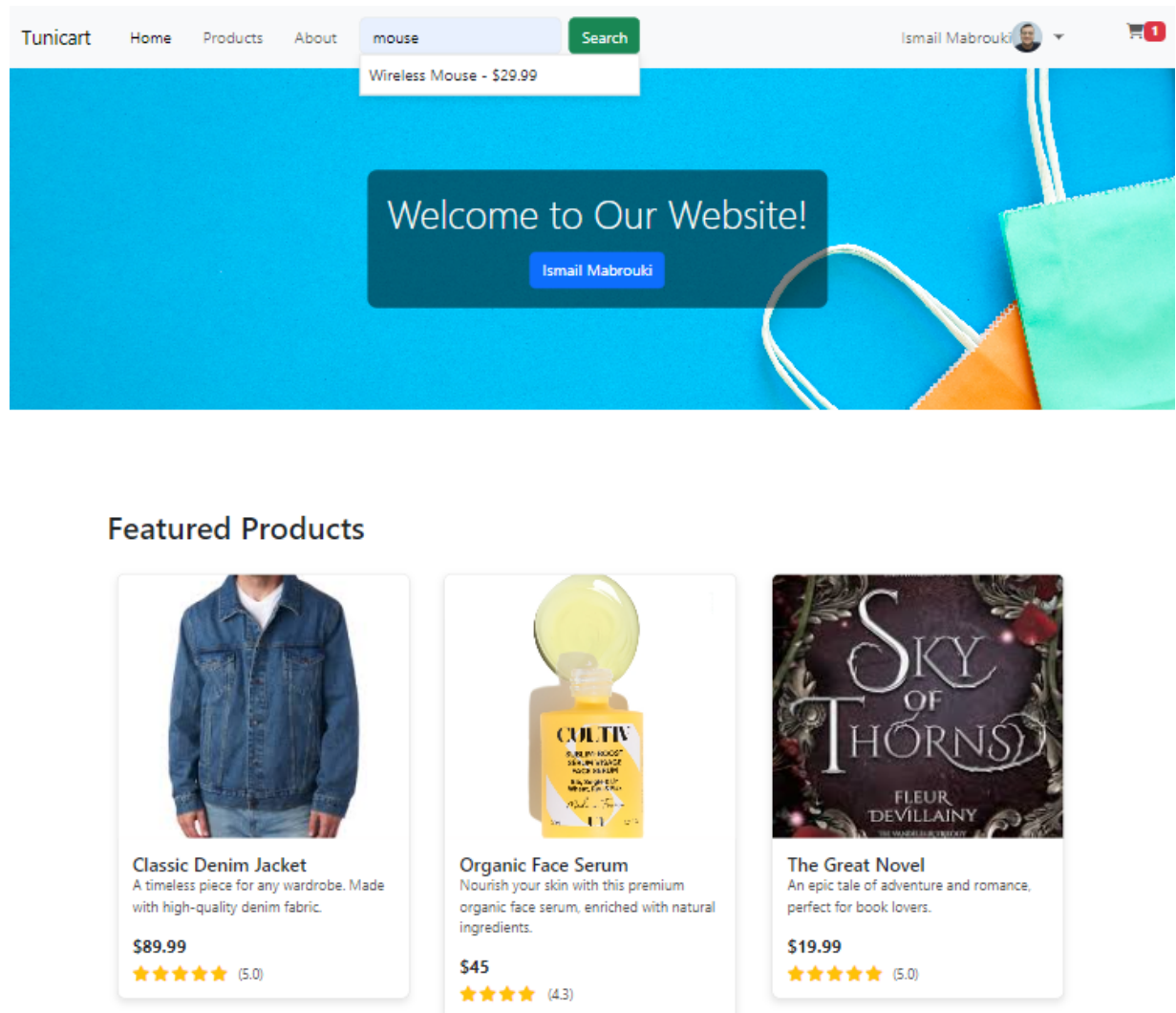


Figure 5.11 – Interface de recherche de produits 1

— *Interface 2 :*



**Figure 5.12** – Interface de recherche de produits 2

Ce sprint a permis de finaliser les fonctionnalités critiques pour l'expérience utilisateur, rendant la plateforme Tunicart prête à gérer l'intégralité du cycle d'achat, du passage de commande à la livraison et au feedback.



# Sixième Chapitre

## Sprint 4

### Introduction

Le quatrième sprint se concentre sur l'amélioration de l'expérience utilisateur en ajoutant des fonctionnalités clés qui permettent de mieux gérer et affiner les interactions avec les produits sur la plateforme TunisiCart. Ce sprint vise à introduire des outils pour filtrer et trier les produits, ainsi qu'à fournir aux utilisateurs un historique détaillé de leurs activités sur la plateforme.

Dans cette phase, nous avons défini et analysé les besoins pour les fonctionnalités suivantes : le filtrage et le tri des produits, ainsi que l'affichage de l'historique des activités. Ces ajouts sont essentiels pour enrichir l'expérience utilisateur en facilitant la recherche de produits et en offrant un suivi détaillé des interactions avec la plateforme.

Nous commencerons par spécifier et analyser les besoins pour ces fonctionnalités. Ensuite, nous décrirons les cas d'utilisation détaillés, fournirons les diagrammes de séquence correspondants, et terminerons par la mise en œuvre technique de ces fonctionnalités.

### 6.1 Spécification et analyse des besoins

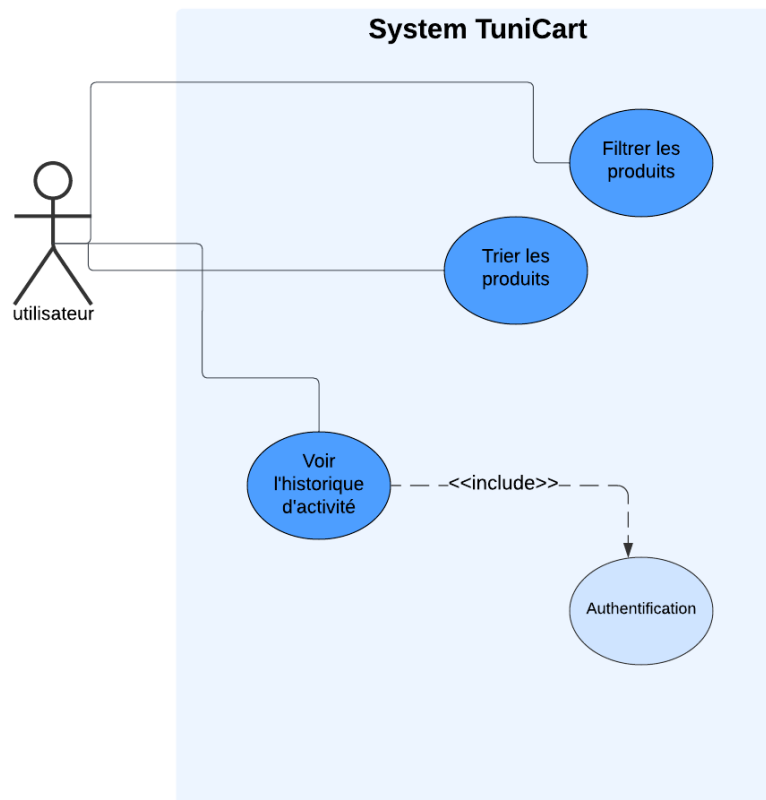
Dans cette partie, nous allons entamer l'analyse et la spécification des besoins de ce sprint. Nous allons commencer par présenter le backlog, ensuite établir le diagramme de cas d'utilisation raffiné, et finir par détailler la description des cas d'utilisation.

### 6.1.1 Backlog du Sprint 4

| ID  | User Story   | Priorité (MoSCoW) |
|-----|--|-------------------|
| 4.0 | En tant qu'utilisateur, je souhaite filtrer les produits par prix, catégorie, ordre alphabétique, etc., afin d'affiner mes choix et trouver plus facilement ce que je cherche.         | M                 |
| 4.1 | En tant qu'utilisateur, je souhaite trier les produits par différentes caractéristiques (prix, description, avis..) pour voir les produits les mieux adaptés à mes besoins en premier. | M                 |
| 4.2 | En tant qu'utilisateur, je souhaite voir mon historique d'achat et de navigation pour suivre mes activités et avoir un aperçu de mes interactions avec la plateforme.                  | S                 |

**Table 6.1** – Backlog du Sprint 4 basé sur les User Stories pour le projet TunisiCart

### 6.1.2 Diagramme de Cas d'Utilisation Raffiné : Sprint 4



**Figure 6.1** – Diagramme de Cas d'Utilisation Raffiné pour le Sprint 4

## 6.2 Description détaillée des cas d'utilisation

### 1. Filtrer les produits

**Description :** Les utilisateurs souhaitent filtrer les produits selon divers critères (prix, popularité, avis, etc.) pour affiner leur recherche et trouver plus facilement ce qu'ils cherchent.

| Élément             | Détails  |
|---------------------|--|
| Titre               | Filtrer les produits   |
| Acteurs             | Utilisateur  |
| Préconditions       | L'utilisateur est connecté et sur la page de produits  |
| Postconditions      | Les produits sont affichés en fonction des filtres appliqués   |
| Scénario Nominal    | 1. L'utilisateur accède à la page de produits.<br>2. L'utilisateur sélectionne les critères de filtrage (e.g., prix, popularité).<br>3. Le système applique les filtres sélectionnés et affiche les produits correspondants. |
| Scénario Alternatif | Si aucun produit ne correspond aux critères,<br>le système affiche un message indiquant qu'aucun produit n'est disponible pour ces filtres.  |

**Table 6.2** – Cas d'utilisation : Filtrer les produits

### 2. Trier les produits

**Description :** Les utilisateurs souhaitent trier les produits selon divers critères (prix, popularité, avis, etc.) pour organiser leur affichage selon leurs préférences.

| Élément             | Détails   |
|---------------------|---|
| Titre               | Trier les produits  |
| Acteurs             | Utilisateur   |
| Préconditions       | L'utilisateur est connecté et sur la page de produits   |
| Postconditions      | Les produits sont affichés dans l'ordre spécifié par l'utilisateur  |
| Scénario Nominal    | 1. L'utilisateur accède à la page de produits.<br>2. L'utilisateur sélectionne un critère de tri (e.g., prix croissant).<br>3. Le système trie les produits selon le critère sélectionné et met à jour l'affichage. |
| Scénario Alternatif | Si le tri échoue en raison d'un problème technique,<br>le système affiche un message d'erreur et demande à l'utilisateur de réessayer.  |

**Table 6.3** – Cas d'utilisation : Trier les produits

### 3. Voir l'historique d'activité

**Description :** Les utilisateurs souhaitent voir leur historique d'achats et de navigation pour suivre leurs activités sur la plateforme.

| Élément             | Détails  |
|---------------------|--|
| Titre               | Voir l'historique d'activité   |
| Acteurs             | Utilisateur  |
| Préconditions       | L'utilisateur est connecté et a un historique d'activité   |
| Postconditions      | L'utilisateur visualise son historique d'achats et de navigation   |
| Scénario Nominal    | 1. L'utilisateur accède à la page de son compte.<br>2. L'utilisateur sélectionne l'option pour voir son historique.<br>3. Le système affiche l'historique d'achats et de navigation. |
| Scénario Alternatif | Si aucun historique n'est disponible ou si un problème survient, le système affiche un message approprié indiquant l'absence d'historique ou une erreur.                             |

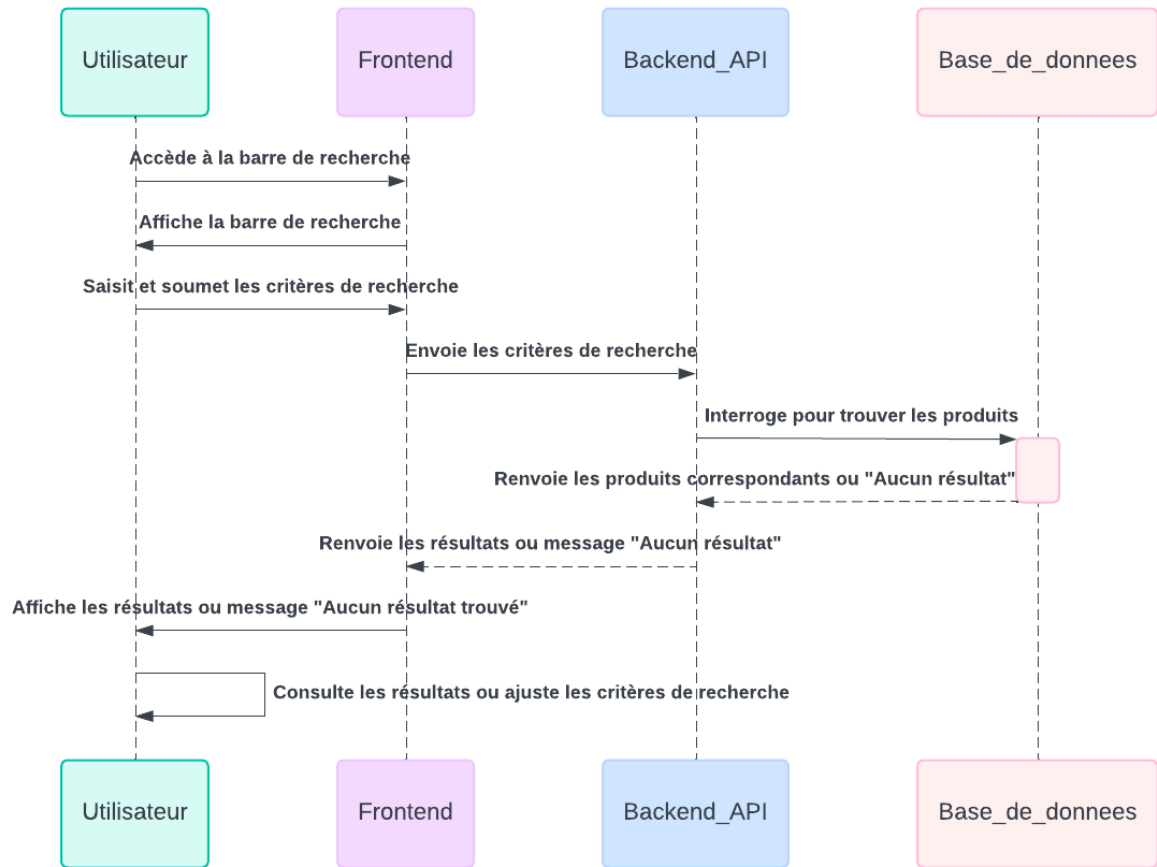
**Table 6.4** – Cas d'utilisation : Voir l'historique d'activité

## 6.3 Diagrammes de séquence

Les diagrammes de séquence ci-dessous montrent l'interaction entre les différents acteurs et le système pour les cas d'utilisation définis :

### 6.3.1 Diagramme de séquence "Filtrer les produits"

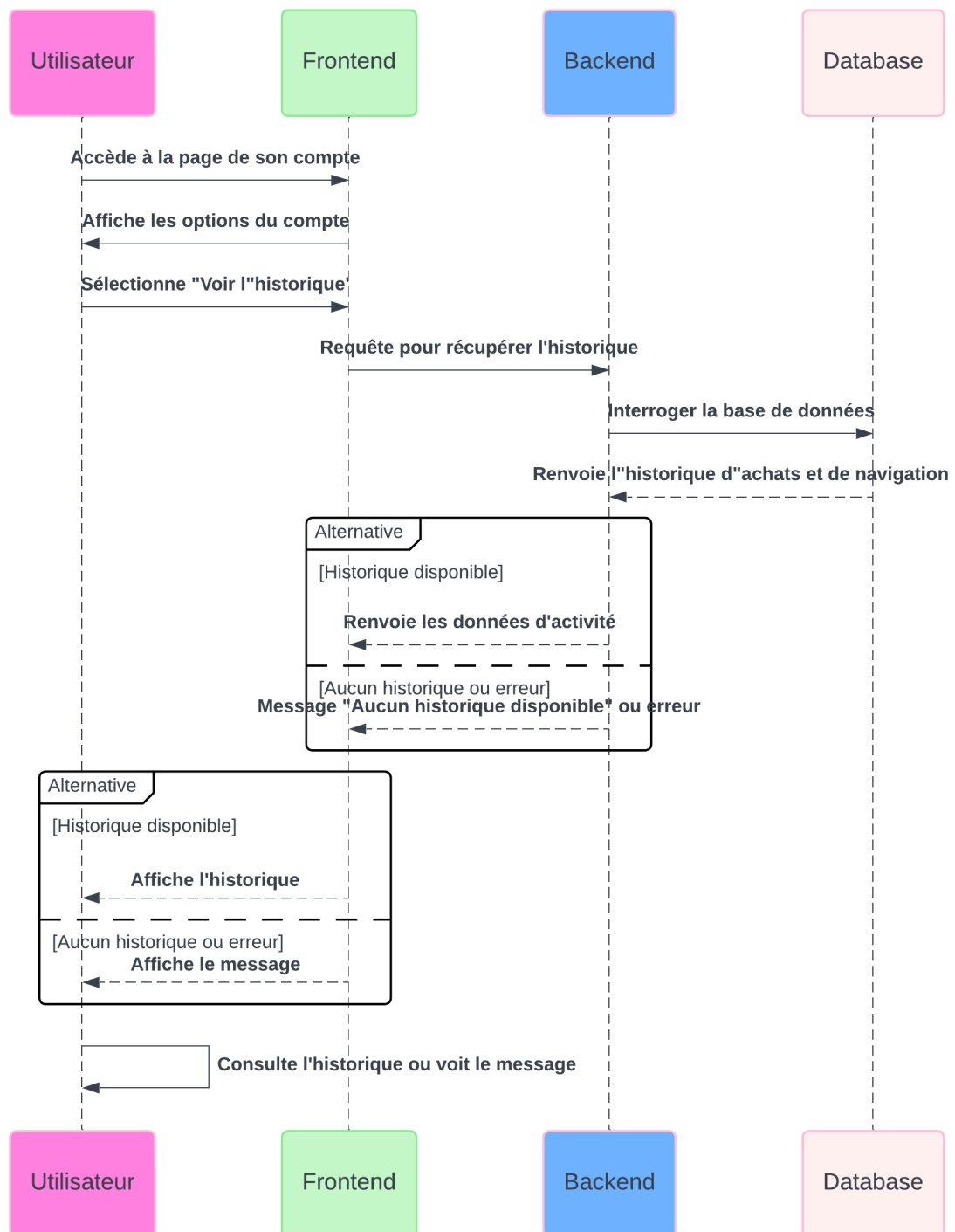
Ce diagramme de séquence décrit les interactions pour filtrer les produits sur TunisiCart. L'utilisateur sélectionne les filtres souhaités, et le système affiche uniquement les produits qui correspondent aux critères appliqués. Si aucun produit n'est trouvé, un message d'erreur est affiché pour informer l'utilisateur.



**Figure 6.2** – Diagramme de séquence pour UC10 : Filtrer les produits"

### 6.3.2 Diagramme de séquence "Voir l'historique d'activité"

Ce diagramme de séquence montre comment un utilisateur connecté sur Tunisi-Cart peut consulter l'historique de ses achats et de sa navigation. Si les données sont disponibles, elles sont affichées, sinon un message approprié est montré.

**Figure 6.3** – Diagramme de séquence pour UC11 : Voir l'historique d'activité

## 6.4 Réalisation

Dans cette section, nous présentons les interfaces développées lors du Sprint 4 du projet TunisiCart. Ces interfaces permettent aux utilisateurs d'améliorer leur expérience sur la plateforme en affinant leurs recherches de produits et en consultant leurs activités passées.

— **Filtres et tri :**

- *Introduction* : L'interface de filtres et tri permet aux utilisateurs d'affiner leur recherche de produits en utilisant divers critères comme le prix, la popularité, et les avis. Cela améliore l'expérience de navigation en facilitant la découverte des produits.
- *Interface 1* :

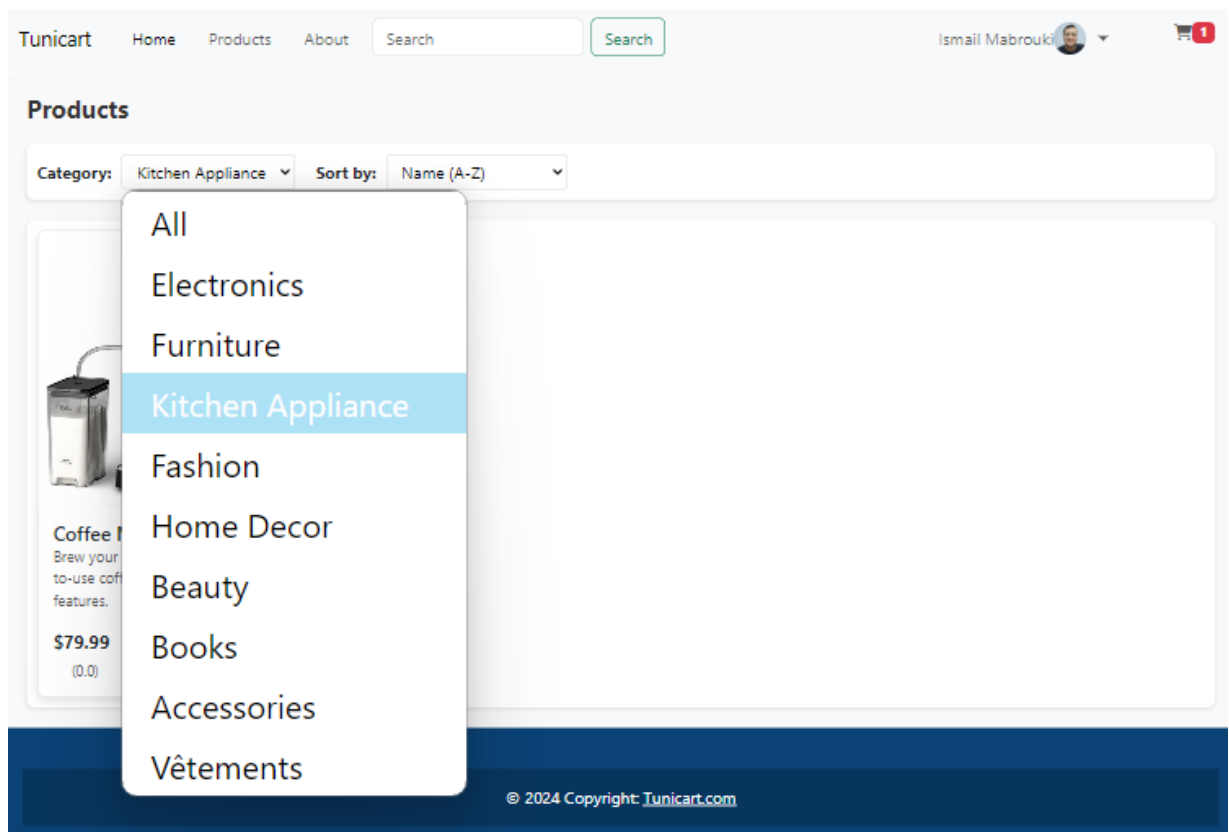


Figure 6.4 – Interface de filtres 1

— *Interface 2 :*

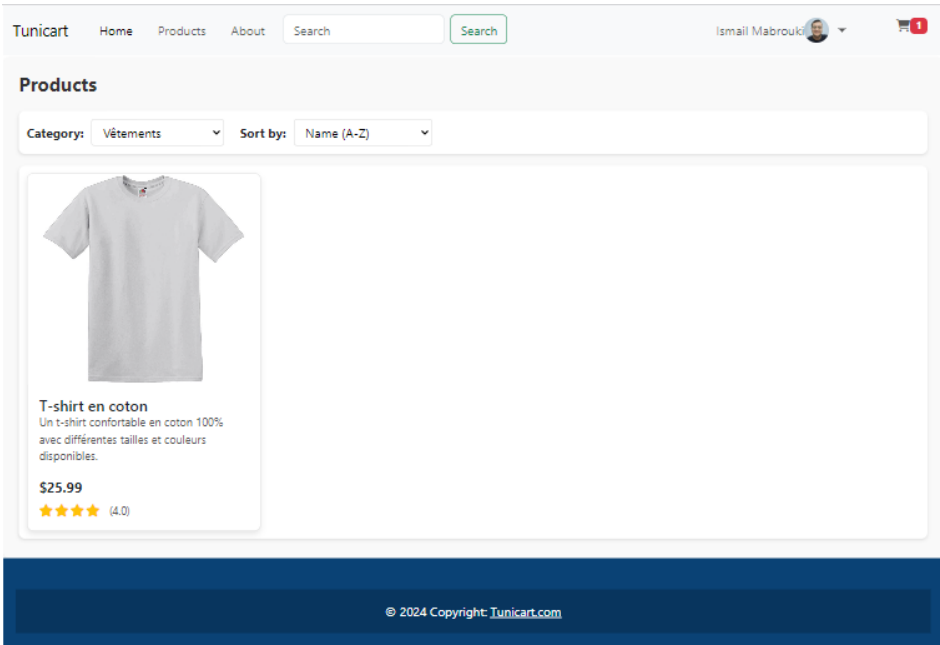


Figure 6.5 – Interface de filtres 2

— *Interface 3 :*

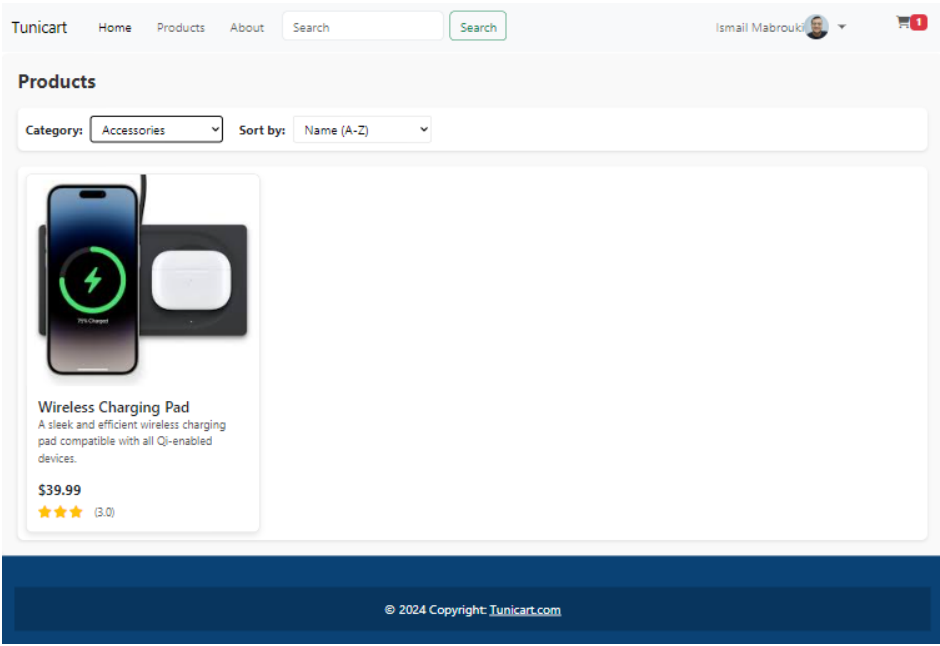


Figure 6.6 – Interface de filtres 3



— *Interface 4 :*

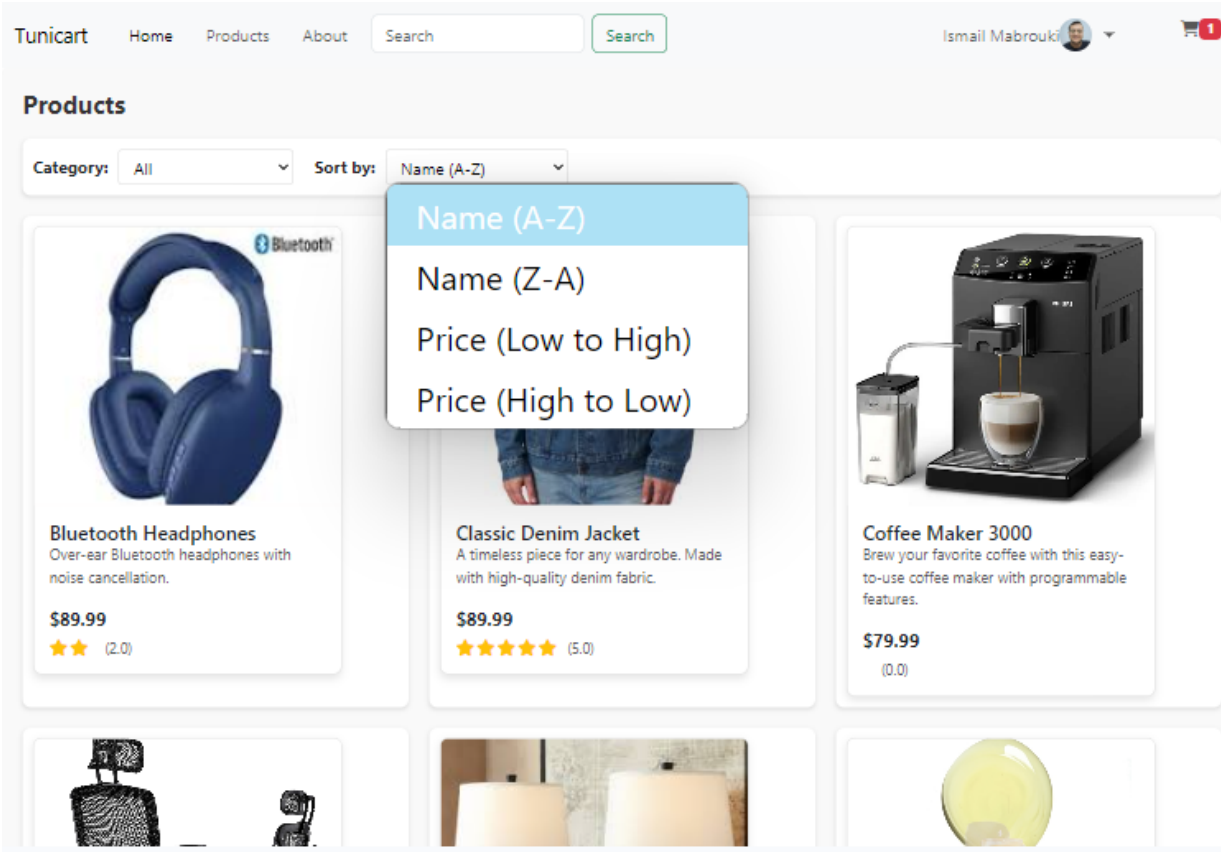


Figure 6.7 – Interface de tri 1

— Interface 5 :

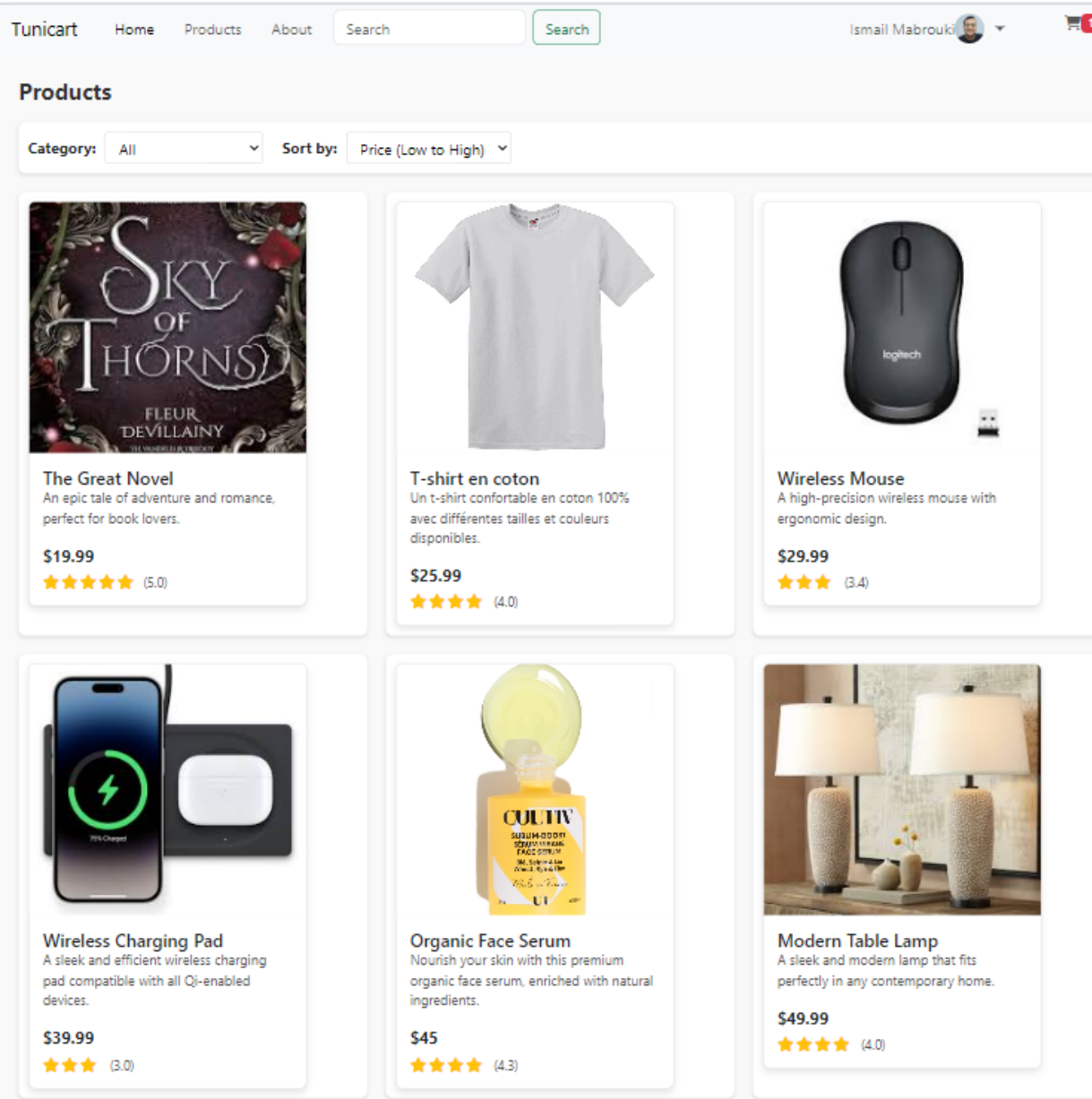


Figure 6.8 – Interface de tri 2

— *Interface 5 :*

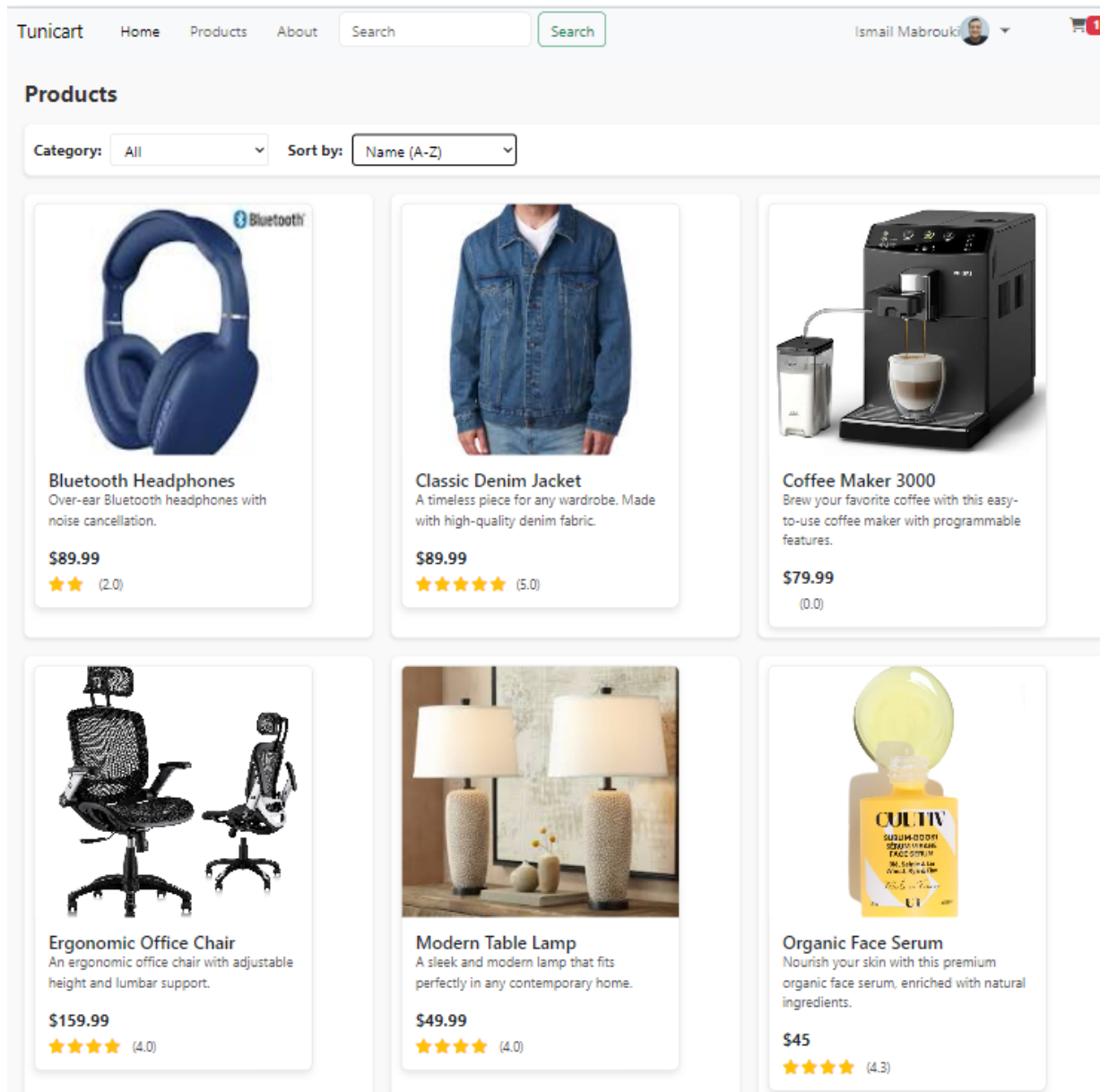


Figure 6.9 – Interface de tri 3

— **Historique des activités :**

— *Introduction :* L'interface d'historique des activités permet aux utilisateurs de consulter leurs achats passés ainsi que leur activité de navigation sur la plateforme. Cela aide les utilisateurs à garder une trace de leurs interactions et achats.

— *Interface 1 :*

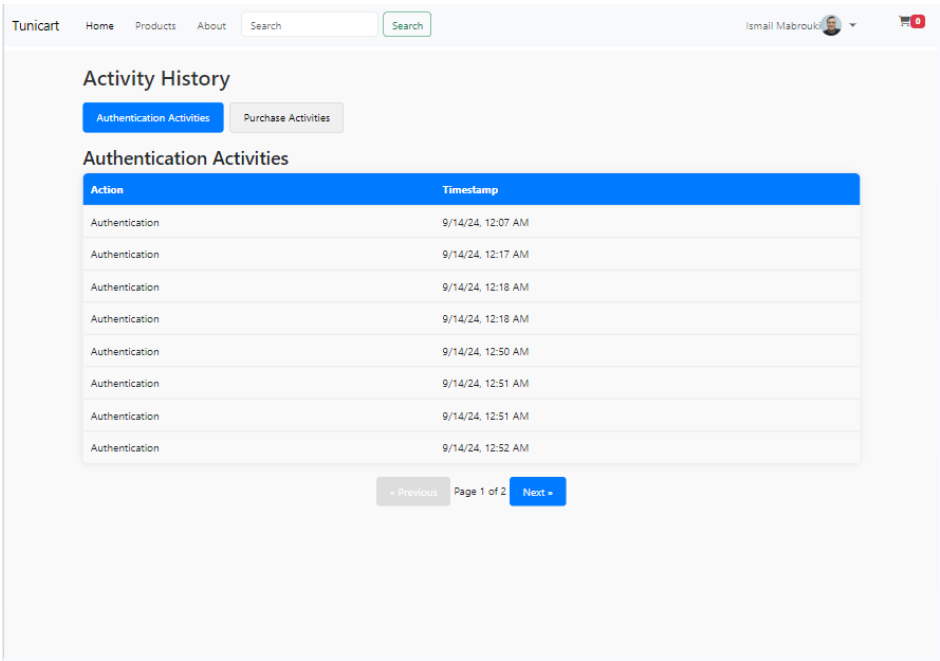


Figure 6.10 – Interface d’historique des activités : Authentification

— *Interface 2 :*

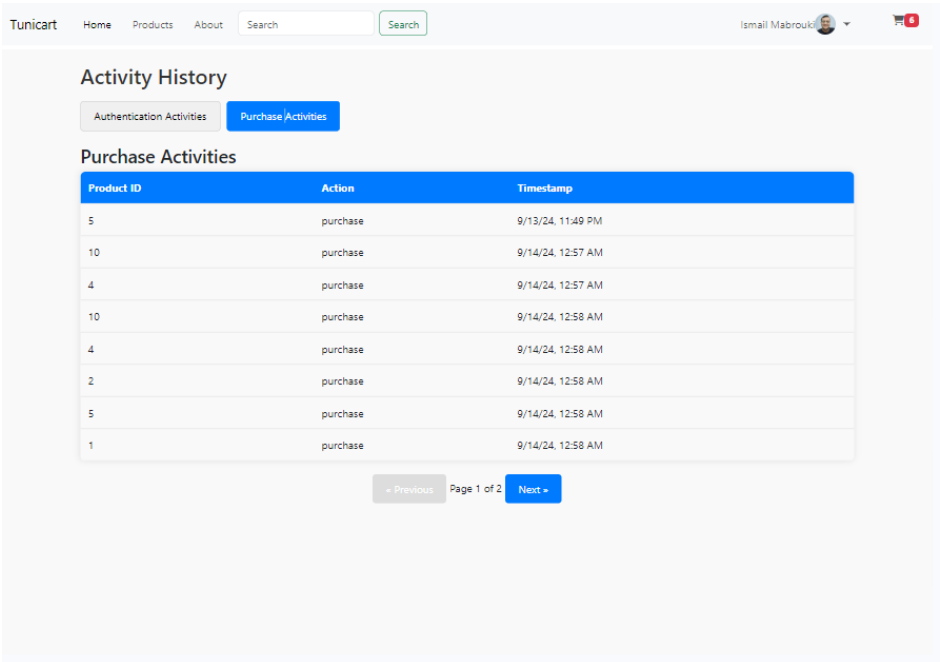


Figure 6.11 – Interface d’historique des activités : Achat

# Conclusion générale

Ce projet de développement de la plateforme TunisiCart, réalisé au sein de BeeCoders, visait à créer une solution e-commerce robuste et dynamique, répondant aux besoins variés des utilisateurs dans le contexte tunisien et au-delà.

L'objectif principal était de concevoir et de déployer une plateforme multi-vendeurs qui offre une expérience utilisateur fluide et efficace. Nous avons mis en place des fonctionnalités essentielles telles que la gestion des utilisateurs, l'ajout de produits, la gestion des commandes, ainsi que des fonctionnalités avancées telles que le filtrage et le tri des produits, et la gestion des avis.

Le processus de développement s'est articulé autour de la méthodologie Agile Scrum, permettant une gestion itérative et incrémentale du projet. Cette approche a facilité l'adaptation continue aux besoins évolutifs des utilisateurs et a permis des ajustements en temps réel pour améliorer la qualité du produit final.

À travers ce projet, nous avons pu approfondir nos compétences techniques en développement front-end et back-end. La mise en œuvre des technologies telles que Angular et Spring Boot, combinée avec des pratiques modernes de gestion de projet, a enrichi notre expérience et a consolidé nos connaissances acquises tout au long de notre parcours académique.

Ce projet représente une étape significative dans notre transition vers le monde professionnel, offrant une opportunité précieuse pour appliquer des concepts théoriques à des problèmes pratiques et réels. Il a également permis de développer des compétences en analyse de besoins, en conception de systèmes, et en gestion de projet.

En perspective, TunisiCart pourrait évoluer vers l'intégration de fonctionnalités supplémentaires pour améliorer encore l'expérience utilisateur. Par exemple, l'ajout de recommandations personnalisées basées sur les préférences des utilisateurs, ou l'implémentation de modules d'analyse avancée pour optimiser la gestion des stocks et les stratégies de marketing. Ces ajouts pourraient renforcer la position de TunisiCart sur le marché et offrir une valeur accrue aux utilisateurs.

En conclusion, ce projet a été une aventure enrichissante, marquée par des défis stimulants et des succès significatifs. Il représente un pas important vers la réalisation de notre vision d'une plateforme e-commerce innovante et performante.

# Bibliographie

- [1] Ismail Mabrouki. Repository : Tunicart. <https://github.com/IsmailMabrouki/Tunicart>, July 2024. Accessed : 2024-08 -25.
- [2] A. Neffati. beecoders.tn. <https://www.beecoders.tn/>, January 2018. Accessed : 2022-02-05.