

Une solution web pour la vente en ligne de pièces détachées automobiles

AutoPartsHub

Esprit School Of Business

Préparé par :  
• Linda Karou  
• Rayen Nassrallah  
• Ismail Aouini  
• Sara Hamrouni  
• Zied Neji

Encadré Par :

Mr. Ahmed Ben Mansour

Mme Hanene Chettaoui

Année Universitaire : 2024/2025

****Table Des Matières****

1. **Introduction**  
    1.1. Présentation du projet  
    1.2. Présentation de l’organisme d’accueil  
    1.3. Problème et solution proposée  
    1.4. Étude de l’existant  
    1.5. Méthodologie de travail  
    1.6. Conclusion
2. **Sprint 0 – Conception et Architecture**  
    2.1. Introduction  
    2.2. Analyse des besoins  
     2.2.1. Besoins fonctionnels  
     2.2.2. Besoins non fonctionnels  
    2.3. Diagrammes UML  
     2.3.1. Cas d’utilisation général  
     2.3.2. Diagramme de classes  
     2.3.3. Diagrammes de séquence  
    2.4. Backlog Produit  
    2.5. Planification des sprints  
    2.6. Environnement de travail  
    2.7. Conclusion
3. **Sprint 1 – Authentification et Catalogue**  
    3.1. Introduction  
    3.2. Backlog Sprint 1  
    3.3. Diagrammes du sprint  
     3.3.1. Cas d’utilisation  
     3.3.2. Diagramme de séquence (MVC)  
     3.3.3. Diagramme de classes  
    3.4. Réalisation  
    3.5. Conclusion
4. **Sprint 2 – Commandes**  
    4.1. Introduction  
    4.2. Backlog Sprint 2  
    4.3. Diagrammes du sprint  
     4.3.1. Cas d’utilisation  
     4.3.2. Diagramme de séquence (MVC)  
     4.3.3. Diagramme de classes  
    4.4. Réalisation  
    4.5. Conclusion
5. **Sprint 3 – Paiement et Suivi**  
    5.1. Introduction  
    5.2. Backlog Sprint 3  
    5.3. Diagrammes du sprint  
     5.3.1. Cas d’utilisation  
     5.3.2. Diagramme de séquence  
     5.3.3. Diagramme de classes  
    5.4. Réalisation  
    5.5. Conclusion
6. **Sprint 4 – Optimisation et Finalisation**   
    6.1. Introduction  
    6.2. Optimisations apportées  
    6.3. Intégration finale et tests  
    6.4. Conclusion
7. **Conclusion Générale et Perspectives**  
    - Résumé des résultats obtenus  
    - Limites et axes d’amélioration  
    - Perspectives futures

**CHAPITRE 1**

* 1. Introduction

AutoPartsHub est une plateforme innovante dédiée à l’achat, la recherche et la traçabilité des pièces détachées automobiles, avec une approche à la fois B2B et B2C. Dans un secteur automobile en constante évolution, la gestion des pièces détachées demeure un défi majeur pour les professionnels et les consommateurs. Les plateformes en ligne actuelles peinent à répondre aux besoins spécifiques des utilisateurs, notamment en matière de sécurité, de traçabilité des produits et de gestion des stocks.

Le projet AutoPartsHub vise à résoudre ces problèmes à travers une plateforme intuitive, alimentée par des technologies de pointe telles que la blockchain, l’intelligence artificielle (IA) et des systèmes de paiement sécurisés. Ce système vise à offrir aux utilisateurs une expérience fluide et sécurisée, tout en garantissant une traçabilité maximale des pièces et une gestion optimisée des stocks.

En réponse aux défis de la contrefaçon, de la fragmentation des chaînes logistiques et du manque de digitalisation dans le secteur, AutoPartsHub se positionne comme une solution moderne et adaptable aux exigences du marché. Ce chapitre fournit une vue d'ensemble du projet, présente l’organisme d’accueil, et détaille la méthodologie adoptée pour la réalisation du projet.

1.2 Présentation de l’organisme d’accueil

**Nom fictif : AutoPartsHub SARL**

AutoPartsHub SARL est une entreprise dynamique, fondée en 2020, spécialisée dans le secteur du e-commerce automobile. L'entreprise opère principalement en Europe et dispose de trois entrepôts logistiques répartis sur le continent, ce qui lui permet de gérer un large volume de pièces détachées et de répondre rapidement à la demande croissante du marché.

Depuis sa création, AutoPartsHub connaît une croissance annuelle de 40%, attestant de son succès dans un secteur hautement concurrentiel. Cette croissance est soutenue par un investissement continu dans les technologies avancées et par une forte orientation client, permettant à l'entreprise d’offrir un service à la fois rapide, fiable et sécurisé.

L’entreprise a pour objectif de se différencier sur le marché en proposant une solution intégrée et sécurisée pour la gestion des pièces détachées, en particulier en s'attaquant aux problèmes liés à la contrefaçon, aux délais de livraison et à l'inefficacité des chaînes logistiques actuelles.

1.3 Problème et solution proposée

**Problématique**

Dans le secteur des pièces détachées automobiles, plusieurs défis persistent, notamment :

* **Contrefaçon massive** : Environ 20% des pièces détachées sur le marché sont contrefaites, ce qui représente un risque majeur pour la sécurité des utilisateurs finaux.
* **Chaînes logistiques fragmentées** : Les processus de gestion des stocks et de livraison sont souvent fragmentés, avec des fournisseurs multiples et des délais de livraison peu fiables.
* **Manque de digitalisation** : De nombreuses entreprises du secteur ne tirent pas parti des technologies modernes, ce qui entraîne des inefficacités dans la gestion des stocks, le traitement des commandes et la traçabilité des pièces.

**Solutions proposées par AutoPartsHub :**

AutoPartsHub a conçu une solution globale pour répondre à ces défis :

* **Blockchain pour la traçabilité** : L’utilisation de la blockchain permet de garantir une traçabilité totale des pièces détachées, depuis leur fabrication jusqu’à leur vente finale. Chaque pièce est enregistrée de manière sécurisée et immuable, assurant ainsi son authenticité et sa provenance.
* **Intelligence Artificielle pour la prévision de stock** : L’IA est utilisée pour anticiper les besoins en stock en fonction des tendances de consommation et des ventes passées. Cela permet d'optimiser les stocks et d'éviter les ruptures, tout en réduisant les coûts d’entreposage.
* **Paiement sécurisé (Stripe)** : Le système de paiement de la plateforme repose sur Stripe, garantissant des transactions sécurisées et conformes aux standards PCI-DSS, pour protéger les informations sensibles des utilisateurs.
* **Recherche par VIN (Numéro d’identification du véhicule)** : Les utilisateurs peuvent rechercher des pièces spécifiques en utilisant le VIN de leur véhicule, garantissant ainsi la compatibilité des pièces avec les véhicules des clients.

**Étude de l’existant**

Pour mieux comprendre les défis à surmonter, une comparaison avec les principales plateformes existantes dans le domaine a été réalisée :

| **Solution** | **Traçabilité** | **Rapidité de livraison** | **Coût** | **Sécurité** |
| --- | --- | --- | --- | --- |
| **Oscaro** | Moyenne | 3 jours | Élevé | Faible |
| **Mister-Auto** | Faible | 2 jours | Moyen | Moyenne |
| **Amazon Auto** | Bonne | 1 jour | Variable | Moyenne |

AutoPartsHub se distingue par une **traçabilité améliorée** grâce à la blockchain, une **livraison rapide** et une **sécurité accrue**, tout en maintenant un **coût compétitif**.

1.4 Méthodologie de travail

Afin de répondre à ces enjeux, AutoPartsHub a opté pour une approche de développement **agile** en utilisant la méthodologie **SCRUM**. Cette méthode permet une grande flexibilité et une adaptation rapide aux besoins du marché et aux retours des utilisateurs. Elle est particulièrement adaptée aux projets complexes où des ajustements constants sont nécessaires.

* **Sprint de 2 semaines** : Chaque itération dure deux semaines, permettant une planification régulière, des ajustements rapides et une validation constante des fonctionnalités.
* **Revues et rétrospectives** : À la fin de chaque sprint, une revue des résultats est réalisée avec l’équipe, suivie d’une rétrospective pour identifier les axes d’amélioration.
* **Rôles définis** :

**Product Owner (Linda)** : Responsable de la gestion du produit et de la priorisation des fonctionnalités.

**Scrum Master (Rayen)** : Garant de la méthodologie SCRUM et du bon déroulement des sprints.

**Modèles de développement**

* **Modèle prédictif (en cascade)** : Ce modèle est très rigide et ne permet pas d’adaptation aux changements en cours de projet. Il repose sur une planification détaillée dès le début et un suivi strict des étapes du projet.
* **Modèle adaptatif (Agile/SCRUM)** : Le modèle Agile, adopté par AutoPartsHub, permet une gestion plus flexible et réactive des demandes des utilisateurs, en permettant des ajustements réguliers du produit en fonction des feedbacks reçus.

1.5 Conclusion

Ce chapitre présente l'introduction au projet **AutoPartsHub**, en exposant ses objectifs, les problèmes rencontrés dans le secteur des pièces détachées automobiles, ainsi que les solutions innovantes apportées. La méthodologie agile SCRUM a été choisie pour assurer une flexibilité maximale et une adaptation rapide aux besoins changeants du marché.

Chapitre 2 :

Sprint 0 – Conception et Architecture

2.1 Introduction

Le **Sprint 0** est la phase de lancement du projet, au cours de laquelle les bases techniques, fonctionnelles et organisationnelles sont posées. Il s’agit d’une étape cruciale qui permet de définir les grandes lignes de la plateforme, d'analyser les besoins des utilisateurs, et de commencer à modéliser le système selon les exigences du projet AutoPartsHub.

Dans ce sprint, l’objectif principal est de bien cerner l’architecture du système et de préparer les outils et environnements nécessaires pour le développement à venir. Ce chapitre décrit les étapes franchies durant ce premier sprint, les choix technologiques effectués, ainsi que la structure fondamentale du projet.

2.2 Analyse des besoins

La première étape de la conception d’AutoPartsHub a consisté en une analyse approfondie des besoins, tant fonctionnels que non fonctionnels.

2.2.1 Besoins fonctionnels

Les besoins fonctionnels se rapportent aux actions principales que les utilisateurs pourront accomplir sur la plateforme. Voici les principaux besoins identifiés :

1. **Recherche de pièces** : Les utilisateurs (particuliers et professionnels) doivent pouvoir rechercher facilement des pièces détachées à l’aide de différents filtres, notamment par **VIN** (Numéro d’identification du véhicule).
2. **Gestion des commandes** : Les utilisateurs doivent pouvoir commander des pièces, suivre leurs commandes et recevoir des notifications sur l’état d’avancement.
3. **Paiement en ligne** : Un système de paiement sécurisé est indispensable, permettant aux utilisateurs d’effectuer des paiements pour leurs commandes de manière fluide et sécurisée.
4. **Authentification utilisateurs** : Un système d'authentification sécurisé (via JWT) permet aux utilisateurs de se connecter à leur compte et d'accéder à des fonctionnalités spécifiques selon leur rôle (client, administrateur, technicien).

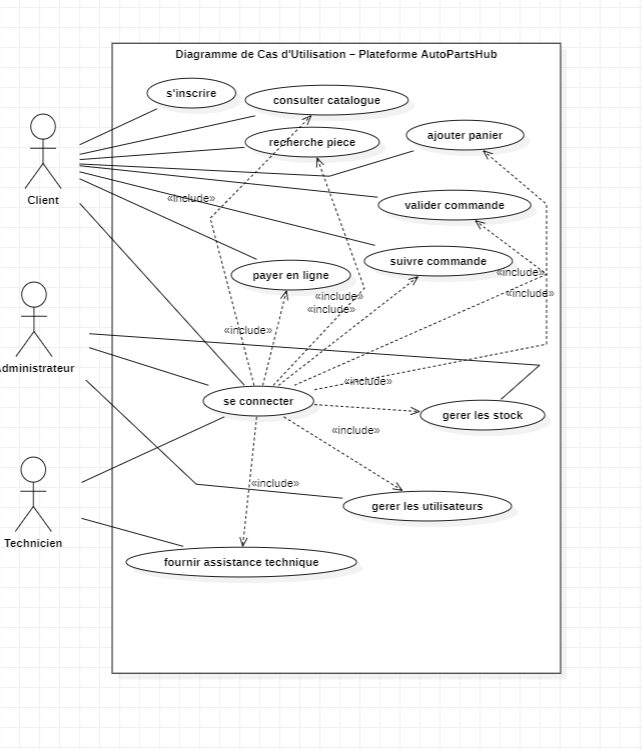
2.2.2 Besoins non fonctionnels

Les besoins non fonctionnels sont liés à la performance, la sécurité et la fiabilité du système :

1. **Temps de réponse** : Le système doit répondre aux requêtes des utilisateurs en moins de 2 secondes.
2. **Uptime** : Le système doit garantir une disponibilité de 99,9% pour assurer un service sans interruption.
3. **Sécurité** : La plateforme doit respecter les normes **PCI-DSS** pour garantir la sécurité des informations sensibles et des paiements des utilisateurs.

2.3.1 Cas d’utilisation général

Les diagrammes de cas d'utilisation permettent de visualiser les interactions entre les différents acteurs du système (utilisateurs, administrateurs, techniciens, etc.). Voici un aperçu des cas d’utilisation définis pour AutoPartsHub.



**Acteurs** :

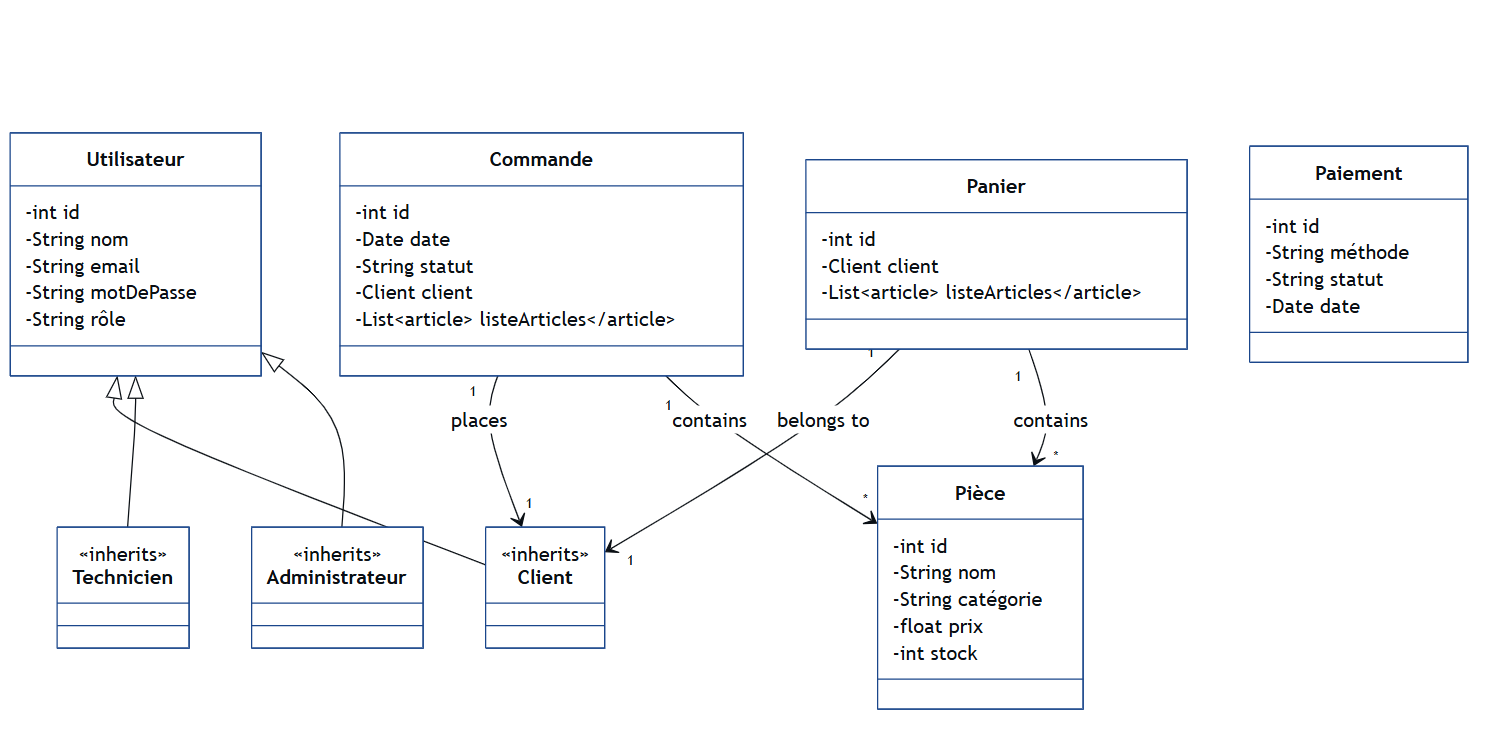
1. **Client** : Recherche des pièces, passe des commandes, effectue des paiements.
2. **Admin** : Gère les stocks, valide les commandes, suit les transactions.
3. **Technicien** : Vérifie l’authenticité des pièces, assure la gestion technique du catalogue.

**Cas** :

1. **Commander** : Le client choisit une pièce, l'ajoute à son panier et procède au paiement.
2. **Vérifier authenticité** : Le technicien utilise la blockchain pour vérifier l’origine et l’authenticité des pièces.
3. **Gérer stock** : L’admin ajuste les niveaux de stock et effectue les réapprovisionnements.

2.3.2 Diagramme de classes

Le diagramme de classes illustre les principales entités du système et leurs relations. Les entités principales sont :

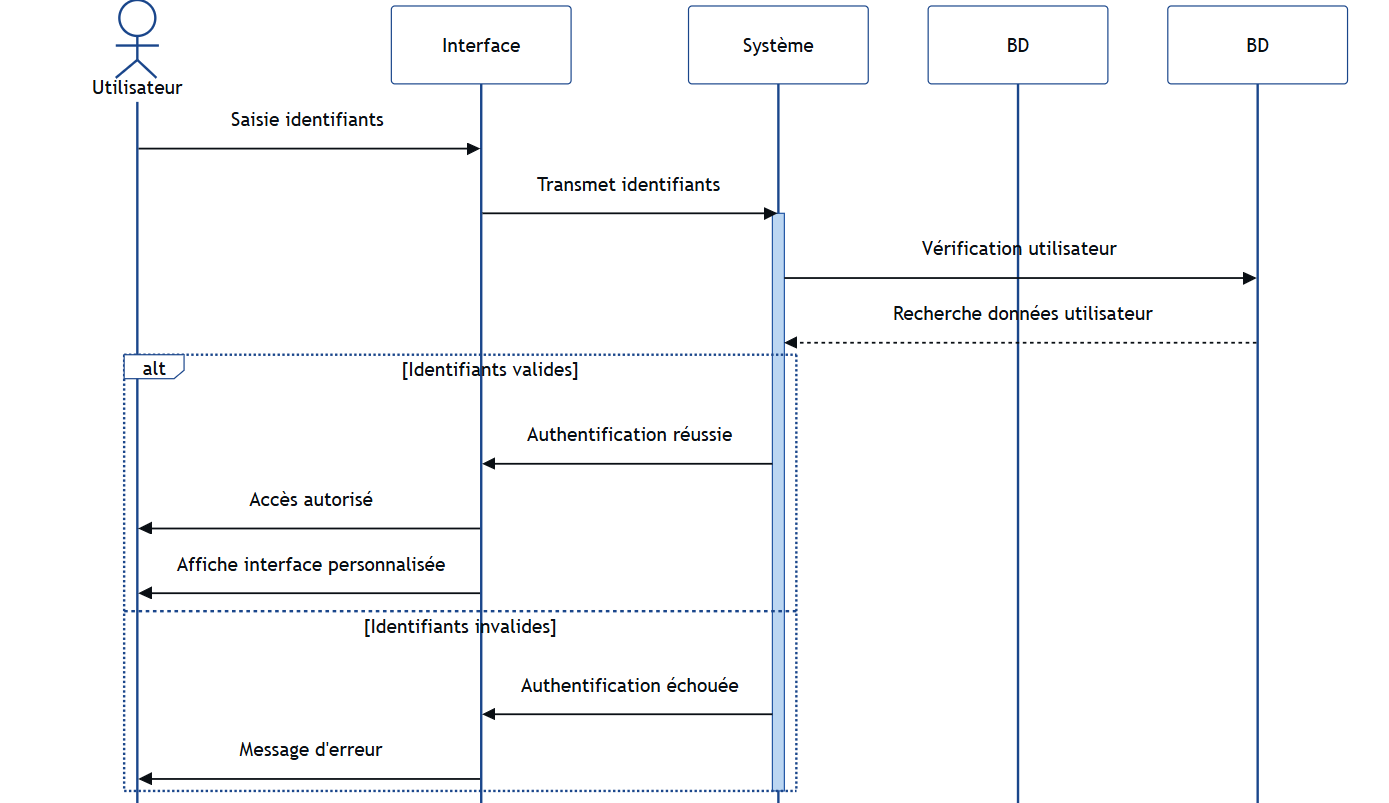


1. **Utilisateur** : Représente un client, un administrateur ou un technicien. Cette classe contient des attributs comme le nom, l'email, le rôle, etc.
2. **Commande** : Représente une commande effectuée par un client. Cette classe est liée à des pièces et des paiements.
3. **Pièce** : Représente une pièce détachée automobile. Elle possède des attributs comme le nom, la description, le prix et la disponibilité.
4. **Paiement** : Représente une transaction de paiement effectuée via Stripe. Cette classe inclut des informations sur le montant, la date, et l’état du paiement.

Les relations entre ces classes incluent des relations de type **1-n** (par exemple, un utilisateur peut avoir plusieurs commandes), ainsi que des héritages pour définir les rôles des utilisateurs (Client, Admin, Technicien).

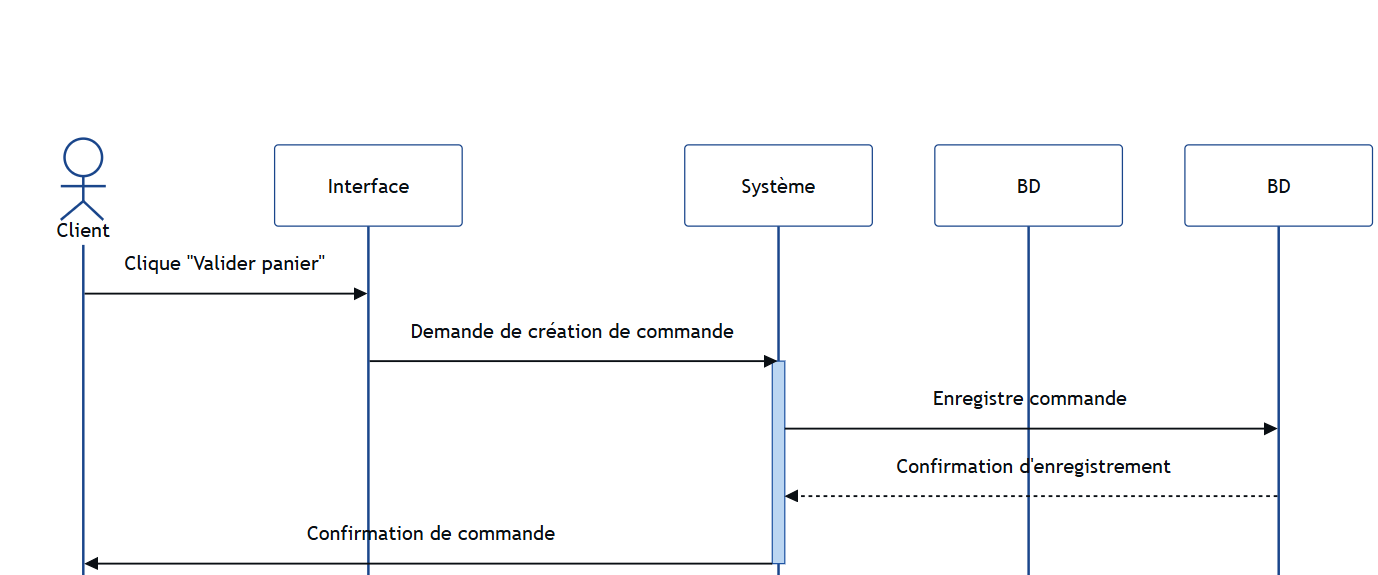
2.3.3 Diagramme de séquence (connexion)

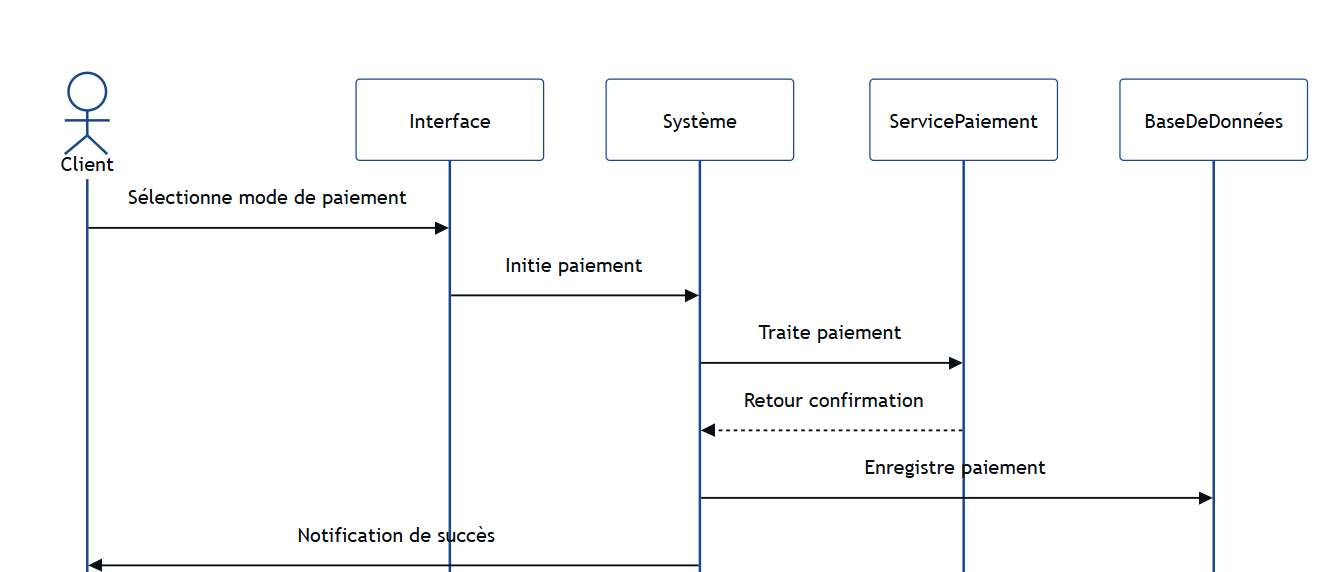
Le diagramme de séquence présente les interactions entre les différentes couches du système (Frontend, Backend, Base de données, etc.) pour un cas d’utilisation spécifique, comme la connexion d'un utilisateur :



1. **[Utilisateur]** → **Frontend** : L’utilisateur entre ses identifiants.
2. **Frontend** → **Backend** : Le frontend envoie une requête d’authentification.
3. **Backend** → **Base de données** : Le backend vérifie les informations d’identification dans la base de données.
4. **Backend** → **Frontend** : Si les identifiants sont valides, un token JWT est envoyé au frontend.
5. **[Utilisateur]** → **Frontend** : L’utilisateur est redirigé vers son tableau de bord avec accès sécurisé.

2.3.4 Diagramme de séquence (commander) :



2.3.6 Diagramme de séquence (paiement)  

2.4 Backlog Produit

Le **Backlog Produit** est une liste des fonctionnalités à développer, priorisée selon leur importance pour le projet. Voici quelques exemples de User Stories pour les fonctionnalités principales de la plateforme :

| **ID** | **User Story** | **Priorité** | **Critères d’acceptation** |
| --- | --- | --- | --- |
| US1 | Rechercher des pièces par VIN | Haute | Résultats obtenus en moins de 1 seconde |
| US2 | Authentification via JWT | Critique | Accès sécurisé avec RBAC (Role-Based Access Control) |
| US3 | Commander une pièce | Haute | Le panier est valide et le paiement peut être effectué |
| US4 | Tableau de bord pro | Moyenne | L’admin peut voir et gérer le stock |

Chaque élément du backlog est décomposé en tâches détaillées qui seront réalisées durant les sprints suivants.

2.5 Planification des sprints

La planification des sprints a été élaborée pour organiser le développement de la plateforme en étapes incrémentales. Chaque sprint dure deux semaines et est dédié à des fonctionnalités spécifiques.

| **Sprint** | **Durée** | **Objectif** |
| --- | --- | --- |
| Sprint 1 | 01/10 – 15/10 | Authentification et traçabilité des pièces |
| Sprint 2 | 16/10 – 30/10 | Catalogue de pièces et tableau de bord |
| Sprint 3 | 01/11 – 15/11 | Système de commande et suivi des commandes |
| Sprint 4 | 16/11 – 30/11 | Optimisation, tests finaux et finalisation |

2.6 Environnement de travail

Les choix technologiques sont essentiels pour assurer la performance et la scalabilité du projet.

Voici un aperçu de l’environnement de travail :

1. **Frontend** : Angular + Tailwind
2. **Backend** : Spring Boot
3. **Base de données** : PostgreSQL
4. **CI/CD** : Jenkins + Docker
5. **Outils** : Jira (gestion de projet), Git (contrôle de version), Postman (tests d’API)

Ces technologies ont été choisies pour leur robustesse, leur flexibilité et leur capacité à répondre aux exigences du projet en termes de performance, de sécurité et d’évolutivité.

2.7 Conclusion

Le **Sprint 0** a permis de poser les fondations du projet **AutoPartsHub**. La définition des besoins fonctionnels et non fonctionnels, la modélisation du système avec les diagrammes UML, et la planification des sprints ont permis d’établir une feuille de route claire pour les étapes à venir. Ce travail préparatoire garantit que la plateforme sera développée de manière structurée et cohérente, tout en répondant aux besoins des utilisateurs finaux.

sprints ont permis d’établir une feuille de route claire pour les étapes à venir. Ce travail préparatoire garantit que la plateforme sera développée de manière structurée et cohérente, tout en répondant aux besoins des utilisateurs finaux.

**CHAPITRE 3**

Chapitre 3 : Sprint 1 – Authentification et Catalogue

3.1 Introduction

Le Sprint 1 est consacré à la mise en place de la sécurité de la plateforme et à la construction du catalogue dynamique de pièces détachées. Ce sprint est essentiel pour fournir les premières fonctionnalités clés de la plateforme, en particulier l’authentification des utilisateurs et la gestion du catalogue de pièces.

L’objectif est de garantir une base sécurisée pour les utilisateurs, tout en permettant une navigation fluide dans le catalogue. L’authentification des utilisateurs via un système sécurisé basé sur JWT (JSON Web Token), ainsi que la possibilité pour les utilisateurs de visualiser et de filtrer les pièces, sont les deux principales fonctionnalités mises en place dans ce sprint.

3.2 Backlog Sprint

Le Backlog Sprint regroupe les tâches qui doivent être accomplies dans ce sprint, avec une estimation de la complexité, du temps nécessaire et du statut de chaque tâche. Voici les tâches principales du Sprint 1 :

| Tâche ID | Description | Complexité | Temps estimé | Statut |
| --- | --- | --- | --- | --- |
| AUTH-01 | Formulaire de connexion | Moyenne | 6h | ✔ Terminé |
| AUTH-02 | Authentification via JWT | Élevée | 10h | ✔ Terminé |
| CAT-01 | Interface catalogue | Moyenne | 8h | ✔ Terminé |
| CAT-02 | CRUD catalogue (ajouter/modifier/supprimer des pièces) | Élevée | 12h | 🕓 En cours |

Description détaillée des tâches :

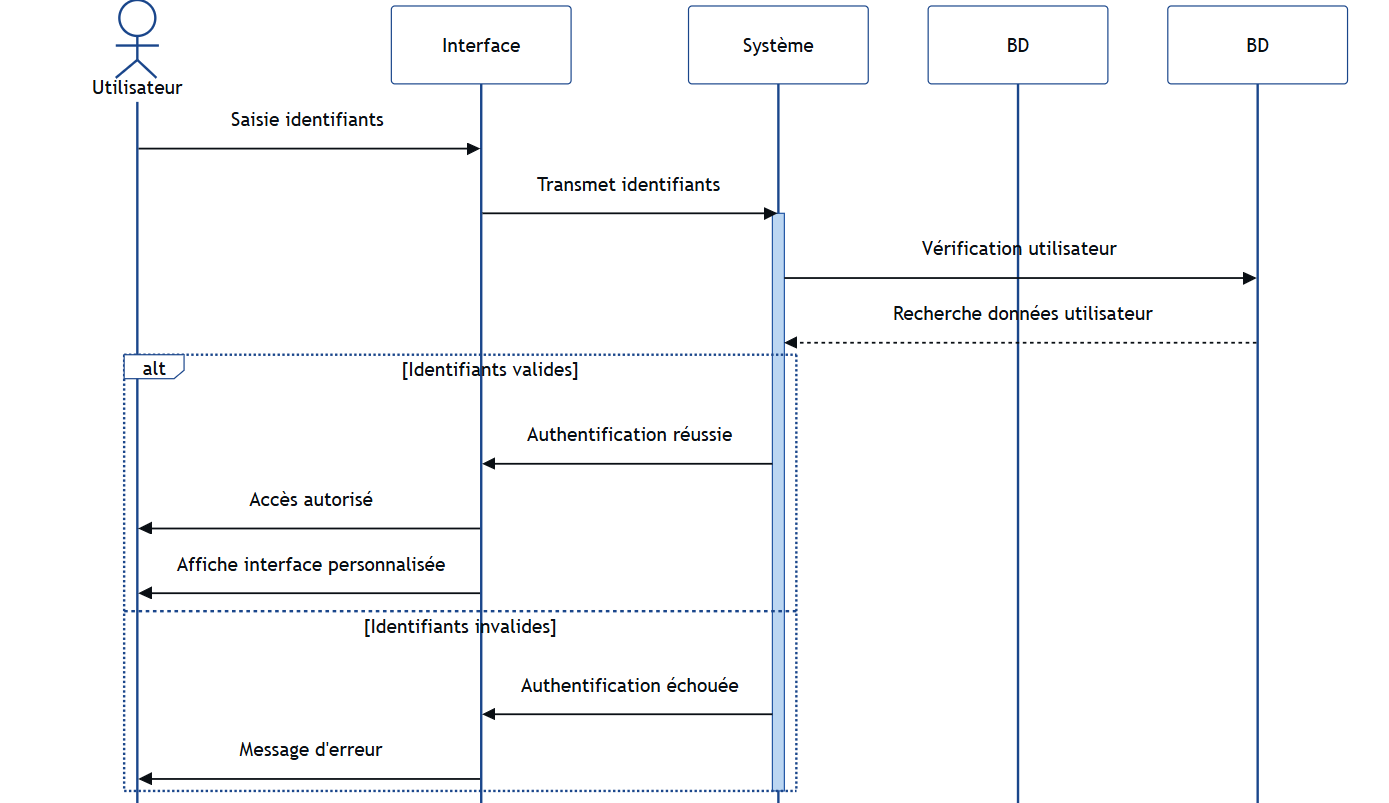
1. Formulaire de connexion (AUTH-01) : Création de l'interface utilisateur pour la connexion, permettant aux utilisateurs de saisir leur nom d’utilisateur et leur mot de passe. Ce formulaire intègre des mécanismes de validation des champs.
2. Authentification via JWT (AUTH-02) : Mise en place de l’authentification par token JSON Web Token pour garantir la sécurité de la plateforme. Cette tâche consiste à créer un mécanisme de génération et de validation des tokens, ainsi que l’intégration avec le backend.
3. Interface catalogue (CAT-01) : Développement de l’interface utilisateur pour le catalogue de pièces détachées, permettant aux utilisateurs de rechercher des pièces et de filtrer les résultats en fonction de différents critères.
4. CRUD catalogue (CAT-02) : Développement des fonctionnalités CRUD (Créer, Lire, Mettre à jour, Supprimer) pour la gestion des pièces dans le catalogue. Cela permet aux administrateurs de gérer l’ensemble des pièces disponibles à la vente sur la plateforme.

3.3 Diagrammes

3.3.1 Cas d’utilisation : Se connecter et Voir le catalogue

Le diagramme de cas d’utilisation pour le Sprint 1 illustre les actions principales des utilisateurs de la plateforme. Voici les principaux cas d'utilisation de ce sprint :

1. Se connecter : L'utilisateur entre ses identifiants dans le formulaire de connexion, et le système vérifie ses informations. Si elles sont correctes, un token JWT est généré et envoyé au client pour l'authentifier.
2. Voir catalogue : L'utilisateur consulte le catalogue de pièces détachées, avec la possibilité de filtrer les résultats par différents critères (marque, type de pièce, etc.).
   * 1. Diagramme de séquence (MVC)



Le diagramme de séquence suivant montre le processus de connexion d’un utilisateur et d’affichage du catalogue, selon le modèle MVC (Modèle-Vue-Contrôleur) :

1. [Utilisateur] → Frontend : L'utilisateur saisit ses identifiants dans le formulaire de connexion.
2. Frontend → Backend : Le frontend envoie les identifiants au backend pour validation.
3. Backend → Base de données : Le backend vérifie les identifiants dans la base de données.
4. Backend → Frontend : Si les identifiants sont valides, le backend génère un token JWT et l'envoie au frontend.
5. [Utilisateur] → Frontend : L'utilisateur est connecté et peut accéder au catalogue de pièces.
6. Frontend → Backend : Le frontend envoie une requête pour récupérer les pièces du catalogue.
7. Backend → Base de données : Le backend interroge la base de données pour récupérer les pièces.
8. Backend → Frontend : Le backend renvoie les données des pièces au frontend, qui les affiche à l'utilisateur.

3.3.3 Diagramme de classe : Authentification et Catalogue

Les principales classes de ce sprint incluent les entités suivantes :

1. Utilisateur : Cette classe contient des informations sur les utilisateurs (nom, email, mot de passe, rôle) et gère l’authentification via JWT.
2. Pièce : Représente une pièce détachée avec des attributs tels que le nom, la référence, le prix, la disponibilité, et les catégories associées.
3. Catalogue : Cette classe est responsable de l’affichage du catalogue des pièces et de la gestion des opérations CRUD.

Les relations entre ces classes sont les suivantes :

1. Un Utilisateur peut être lié à plusieurs Commandes.
2. Une Pièce appartient à un Catalogue.
3. Un Catalogue contient plusieurs Pièces.

3.4 Réalisation

Pendant ce sprint, plusieurs fonctionnalités clés ont été réalisées :

1. Authentification via JWT : Le système d'authentification a été mis en place avec succès, garantissant une communication sécurisée entre le frontend et le backend. Chaque utilisateur reçoit un token JWT lors de sa connexion, qu'il utilise pour accéder à ses informations personnelles et passer des commandes.
2. Interface catalogue : Un catalogue dynamique a été créé, permettant aux utilisateurs de voir les pièces disponibles et de filtrer les résultats par différents critères. La vue du catalogue est responsive et optimisée pour une navigation fluide.
3. Tests : Des tests automatisés ont été réalisés pour vérifier la bonne fonctionnalité de l’authentification et de l’affichage du catalogue. Les tests incluent des vérifications d’authentification (avec JWT) et des tests d’affichage du catalogue avec les filtres de recherche. Des tests unitaires ont également été effectués sur le backend, avec une couverture de 85 %.
   1. Tests d'authentification : Vérification du bon fonctionnement du mécanisme d'authentification JWT.
   2. Tests du catalogue : Vérification de l'affichage des pièces et de la capacité de filtrer les résultats en fonction des critères définis.

3.5 Conclusion

Le Sprint 1 a permis de poser les bases sécurisées pour la plateforme avec la mise en place de l’authentification via JWT. De plus, un catalogue fonctionnel a été développé, offrant une interface utilisateur simple et fluide pour rechercher et afficher les pièces détachées.

Les tâches prévues ont été réalisées dans les temps et avec une bonne qualité de code. Le système est désormais prêt à être étendu dans les sprints suivants, avec la mise en place des fonctionnalités de commande et de paiement qui seront abordées dans le Sprint 2.

Chapitre 4 :

Sprint 2 – Commandes

4.1 Introduction

Le **Sprint 2** se concentre sur la mise en place du système de commande, couvrant le processus du panier à la validation de la commande, ainsi que l’intégration du paiement en ligne sécurisé. Ce sprint est crucial pour permettre aux utilisateurs de passer des commandes en toute sécurité et suivre l'évolution de leurs achats.

L’objectif principal est de créer un flux de commande fluide, avec la possibilité pour les utilisateurs de sélectionner des produits, les ajouter à leur panier, passer à la validation et effectuer un paiement sécurisé. Ce sprint inclut également la gestion de l'historique des commandes pour les utilisateurs et l'intégration de la notification en temps réel de l’état de la commande.

4.2 Backlog Sprint

Le **Backlog Sprint** pour ce sprint présente les tâches à accomplir, avec les niveaux de complexité, les estimations de temps et le statut de chaque tâche. Voici les principales tâches :

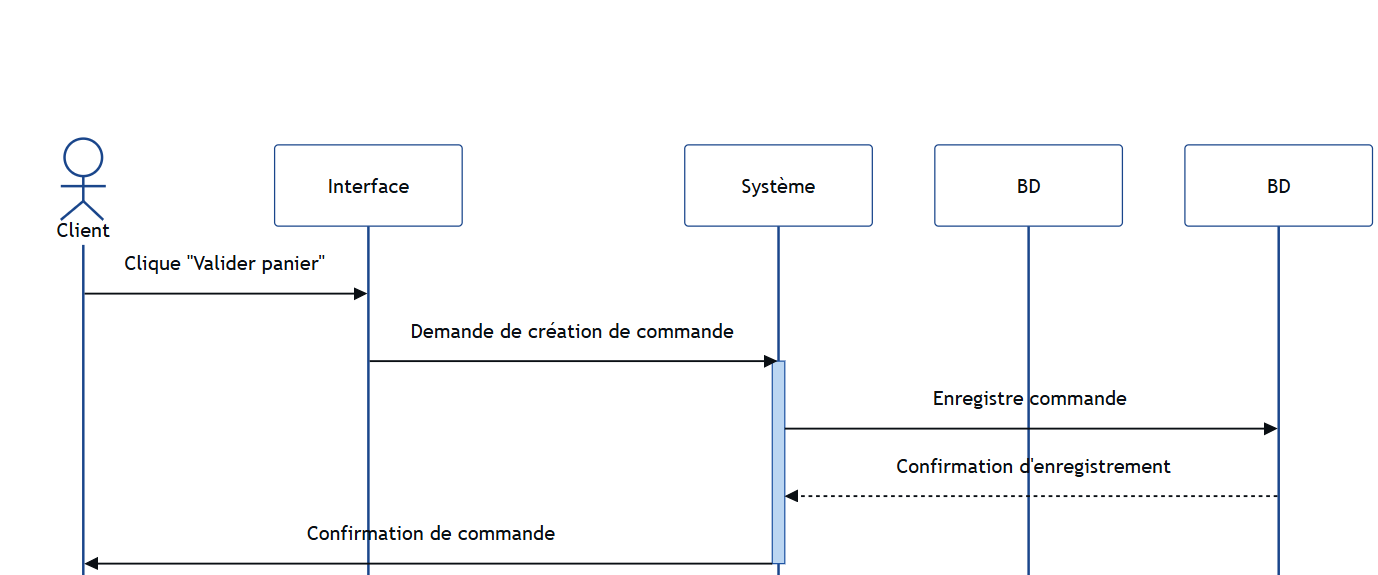
| **Tâche ID** | **Description** | **Complexité** | **Temps estimé** | **Statut** |
| --- | --- | --- | --- | --- |
| **ORD-01** | Création du panier | Moyenne | 7h | ✔ Terminé |
| **ORD-02** | Paiement en ligne | Élevée | 12h | ✔ Terminé |
| **ORD-03** | Historique client | Moyenne | 6h | ✔ Terminé |
| **ORD-04** | Notification statut de commande | Élevée | 10h | 🕓 En cours |

**Description détaillée des tâches :**

1. **Création du panier (ORD-01)** : Développement de la fonctionnalité permettant à l'utilisateur d'ajouter des pièces à un panier virtuel, avec la possibilité de modifier la quantité ou de supprimer des éléments.
2. **Paiement en ligne (ORD-02)** : Intégration d'un système de paiement sécurisé avec **Stripe**, permettant aux utilisateurs de régler leurs commandes en ligne. Cette fonctionnalité est conforme aux exigences de sécurité PCI-DSS pour garantir la protection des informations de paiement.
3. **Historique client (ORD-03)** : Mise en place d'une fonctionnalité permettant à l'utilisateur de consulter l'historique de ses commandes passées, avec un affichage des détails de chaque commande.
4. **Notification statut de commande (ORD-04)** : Intégration d’un système de notification en temps réel pour informer les utilisateurs du statut de leur commande (ex. : traitement, expédition, livraison). Ce système utilise des notifications **WebSocket** pour la mise à jour instantanée.

4.3 Diagrammes

4.3.1 Cas d’utilisation : Passer commande



Le diagramme de cas d’utilisation montre l’interaction entre l’utilisateur et le système pour passer une commande. Voici les principaux cas d’utilisation dans ce sprint :

1. **Sélectionner une pièce** : L'utilisateur navigue dans le catalogue, sélectionne une pièce et l’ajoute au panier.
2. **Passer à la validation** : Une fois les pièces sélectionnées, l'utilisateur passe à la validation de la commande, où il entre ses informations de livraison et de paiement.
3. **Effectuer un paiement** : L’utilisateur entre ses informations de paiement pour effectuer l'achat.
4. **Recevoir la confirmation de commande** : Après le paiement, l'utilisateur reçoit une confirmation de commande avec les détails de celle-ci.

4.3.2 Diagramme de séquence (MVC)

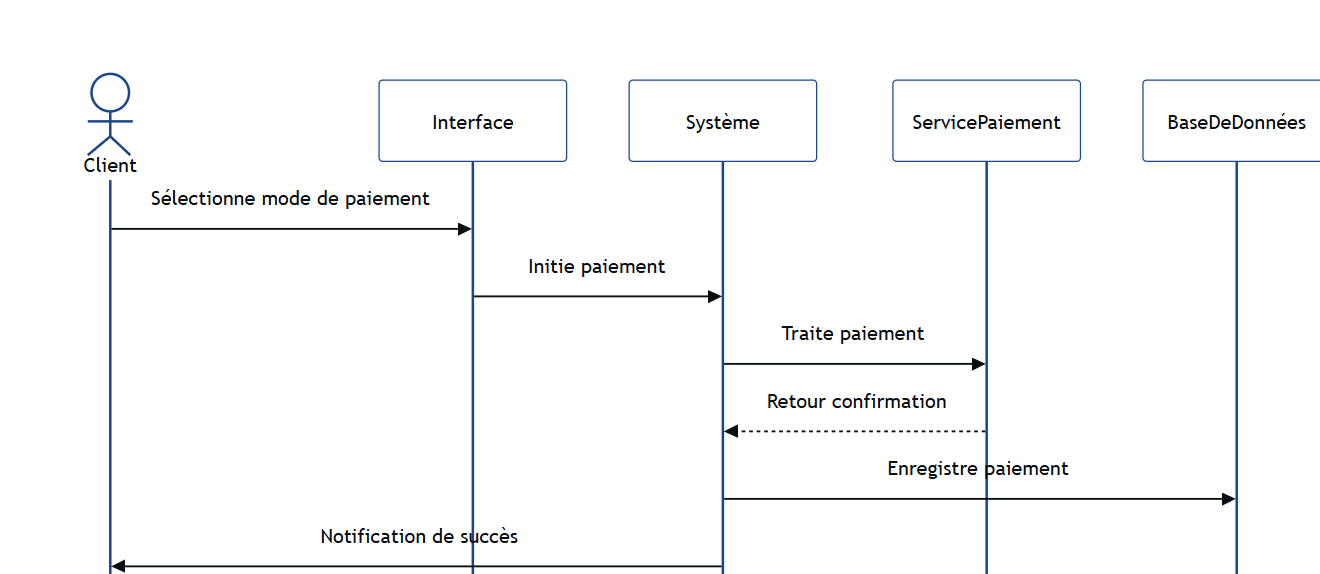
Voici le diagramme de séquence pour l’action de passer commande, selon le modèle **MVC** (Modèle-Vue-Contrôleur) :

1. **[Utilisateur]** → **Frontend** : L’utilisateur sélectionne une ou plusieurs pièces et les ajoute au panier.
2. **Frontend** → **Backend** : Le frontend envoie une requête au backend pour mettre à jour le panier.
3. **Backend** → **Base de données** : Le backend met à jour la base de données pour enregistrer les modifications du panier.
4. **[Utilisateur]** → **Frontend** : L’utilisateur passe à la validation de la commande.
5. **Frontend** → **Backend** : Le frontend envoie la requête pour récupérer les informations de la commande (produits, prix, etc.).
6. **Backend** → **Base de données** : Le backend interroge la base de données pour obtenir les détails de la commande.
7. **[Utilisateur]** → **Frontend** : L’utilisateur procède au paiement.
8. **Frontend** → **Stripe** : Le frontend transmet les informations de paiement à **Stripe** pour traiter la transaction.
9. **Stripe** → **Frontend** : Stripe confirme la réussite ou l'échec de la transaction.
10. **Frontend** → **Backend** : En cas de paiement réussi, le frontend envoie une confirmation au backend pour finaliser la commande.
11. **Backend** → **Base de données** : Le backend met à jour le statut de la commande dans la base de données.
12. **[Utilisateur]** → **Frontend** : L’utilisateur reçoit la confirmation de commande et les détails de l’expédition.

4.3.3 Diagramme de classe : Commande et Paiement

Les principales classes liées aux commandes sont les suivantes :

1. **Commande** : Représente une commande passée par un utilisateur, avec des attributs tels que la date de la commande, le statut, et la liste des articles achetés.
2. **Panier** : Contient la liste des articles que l'utilisateur a ajoutés à son panier avant de finaliser la commande.
3. **Paiement** : Contient les informations relatives au paiement, y compris le statut du paiement, le montant et le mode de paiement utilisé (par exemple, **Stripe**).
4. **Utilisateur** : Un utilisateur peut être lié à plusieurs commandes, et chaque commande contient une ou plusieurs pièces.



4.4 Réalisation

Pendant ce sprint, plusieurs fonctionnalités clés ont été développées :

1. **Création du panier** : Un système de panier fonctionnel a été mis en place, permettant aux utilisateurs d'ajouter, de modifier et de supprimer des produits. Chaque utilisateur a une session unique pour son panier, ce qui permet de suivre ses choix même s’il n’a pas encore finalisé la commande.
2. **Paiement en ligne sécurisé** : L’intégration de **Stripe** a été réalisée avec succès. Les utilisateurs peuvent maintenant effectuer un paiement en ligne en toute sécurité. Le système est conforme aux exigences de sécurité PCI-DSS, garantissant que les informations de paiement sont traitées de manière sécurisée et cryptée.
3. **Historique des commandes** : Les utilisateurs peuvent désormais consulter un historique détaillé de leurs commandes passées. Ce système affiche des informations telles que les articles achetés, le statut de la commande, ainsi que la date de commande et de livraison estimée.
4. **Notification en temps réel** : Un système de notifications en temps réel a été mis en place pour informer les utilisateurs du statut de leur commande. Les notifications sont envoyées via **WebSocket**, permettant des mises à jour instantanées.

4.5 Conclusion

Le **Sprint 2** a permis de construire l'un des éléments les plus critiques du projet : le système de commande. Les utilisateurs peuvent désormais sélectionner des pièces, les ajouter à leur panier, passer une commande, effectuer un paiement sécurisé et recevoir une confirmation.

Le système est entièrement sécurisé et conforme aux normes de sécurité nécessaires. Le travail effectué dans ce sprint pose les bases pour les fonctionnalités de suivi et de gestion des paiements dans les prochains sprints.

Les tâches prévues ont été complétées dans les délais impartis, et le système est opérationnel pour les utilisateurs. Le prochain sprint (**Sprint 3**) se concentrera sur la gestion des paiements et le suivi des transactions en temps réel.

Chapitre 5 :

Sprint 3 – Paiement et Suivi

5.1 Introduction

Le **Sprint 3** se concentre sur la mise en place du système de **paiement sécurisé** et du **suivi transactionnel**. L'objectif est d'assurer une expérience d'achat fluide et fiable pour l'utilisateur, en intégrant une solution de paiement conforme aux normes de sécurité (PCI-DSS) et en offrant un mécanisme de suivi des commandes après paiement.

Le système de **paiement en ligne** intégré dans ce sprint permet aux utilisateurs de finaliser leurs commandes de manière sécurisée, tandis que le **suivi des commandes** permet aux utilisateurs de suivre l’état de leur commande en temps réel. Ce sprint est essentiel pour garantir que les utilisateurs peuvent effectuer des transactions en toute confiance et suivre leur commande après l'achat.

5.2 Backlog Sprint

Le **Backlog Sprint** pour ce sprint présente les tâches à accomplir, avec les niveaux de complexité, les estimations de temps et le statut de chaque tâche. Voici les principales tâches :

| **Tâche ID** | **Description** | **Complexité** | **Temps estimé** | **Statut** |
| --- | --- | --- | --- | --- |
| **PAY-01** | Intégration de Stripe pour le paiement | Élevée | 12h | ✔ Terminé |
| **PAY-02** | Logs de transaction | Moyenne | 6h | ✔ Terminé |
| **PAY-03** | Timeline de suivi des commandes | Moyenne | 7h | 🕓 En cours |

**Description détaillée des tâches :**

1. **Intégration de Stripe pour le paiement (PAY-01)** : Mise en place d'une solution de paiement en ligne via **Stripe**, qui permet aux utilisateurs de régler leurs commandes de manière sécurisée. Cette tâche comprend l'implémentation du formulaire de paiement, le traitement de la transaction, ainsi que la gestion des erreurs possibles.
2. **Logs de transaction (PAY-02)** : Implémentation d'un système de logs pour suivre toutes les transactions effectuées sur la plateforme. Chaque transaction est enregistrée, y compris les détails du paiement (montant, mode de paiement, date), afin de garantir la transparence et la traçabilité des paiements.
3. **Timeline de suivi des commandes (PAY-03)** : Développement d'une interface de suivi des commandes permettant à l'utilisateur de visualiser l’état de sa commande à chaque étape du processus (ex. : en préparation, expédiée, livrée). Cette fonctionnalité utilise des notifications en temps réel via **WebSocket** pour informer instantanément l'utilisateur des changements de statut.

5.3 Diagrammes

5.3.1 Cas d’utilisation : Payer et suivre commande

Le diagramme de cas d’utilisation pour le **paiement et suivi des commandes** décrit les interactions entre l'utilisateur et le système pour effectuer un paiement et suivre la commande :

1. **Passer commande** : L'utilisateur finalise la commande et entre ses informations de paiement.
2. **Effectuer un paiement** : Le système traite le paiement via **Stripe** et confirme la transaction.
3. **Suivre la commande** : L'utilisateur peut consulter l'état actuel de la commande via l'interface de suivi.

5.3.2 Diagramme de séquence : Paiement et Suivi

Le diagramme de séquence montre l'interaction entre l'utilisateur et le système pour effectuer un paiement et suivre l'état de la commande :

1. **[Utilisateur]** → **Frontend** : L’utilisateur choisit un mode de paiement et soumet les informations nécessaires (numéro de carte bancaire, etc.).
2. **Frontend** → **Stripe** : Le frontend transmet les informations de paiement à **Stripe** pour traitement.
3. **Stripe** → **Frontend** : Stripe répond avec le résultat de la transaction (réussie ou échouée).
4. **Frontend** → **Backend** : En cas de transaction réussie, le frontend envoie une demande pour mettre à jour l'état de la commande dans la base de données.
5. **Backend** → **Base de données** : Le backend met à jour l’état de la commande et la marque comme « Payée ».
6. **Backend** → **[Utilisateur]** : L’utilisateur reçoit la confirmation de paiement et l’état actuel de la commande.
7. **[Utilisateur]** → **Frontend** : L’utilisateur consulte l'interface de suivi des commandes.
8. **Frontend** → **Backend** : Le frontend interroge le backend pour obtenir les informations actuelles sur la commande.
9. **Backend** → **Base de données** : Le backend récupère les détails de la commande et son statut actuel.
10. **Backend** → **[Utilisateur]** : L'utilisateur reçoit les informations mises à jour concernant l'état de sa commande (par exemple, « En préparation », « Expédiée », etc.).

5.3.3 Diagramme de classe : Transaction et Suivi

Les principales classes liées au système de paiement et de suivi sont :

1. **Transaction** : Représente une transaction de paiement, incluant des informations telles que le montant, la date de la transaction, le statut (réussie, échouée) et le mode de paiement (ex. carte bancaire, PayPal).
2. **Commande** : Liée à la transaction, une commande contient des informations sur les articles achetés, le statut actuel de la commande, et une référence à la transaction effectuée.
3. **Utilisateur** : Un utilisateur peut passer plusieurs commandes, et chaque commande peut être suivie grâce à un système de notifications en temps réel.

5.4 Réalisation

Au cours de ce sprint, plusieurs fonctionnalités ont été développées pour gérer le paiement et le suivi des commandes :

1. **Intégration du paiement Stripe** : L'intégration de **Stripe** a été réalisée avec succès pour permettre un paiement sécurisé en ligne. Les informations de paiement sont traitées conformément aux normes **PCI-DSS** et sont cryptées lors de leur transmission pour garantir la confidentialité des données.
2. **Logs de transaction** : Un système de logs a été mis en place pour enregistrer les détails de chaque transaction de paiement. Cela permet de suivre l’historique des paiements, de garantir la traçabilité des transactions et d'identifier rapidement tout problème éventuel.
3. **Timeline de suivi des commandes** : Le développement de la fonctionnalité de suivi des commandes a été réalisé avec l'implémentation de notifications en temps réel. Grâce aux **WebSockets**, les utilisateurs sont informés instantanément de l'état de leur commande, ce qui améliore l'expérience utilisateur et la transparence.

5.5 Conclusion

Le **Sprint 3** a permis de finaliser deux éléments clés de la plateforme : le **paiement sécurisé** et le **suivi des commandes**. Ces fonctionnalités sont essentielles pour assurer la sécurité des transactions financières et fournir aux utilisateurs une transparence sur l'état de leurs commandes.

Le système de paiement via **Stripe** fonctionne de manière fluide et sécurisée, et le suivi des commandes via des notifications en temps réel offre une expérience utilisateur enrichie.

Haut du formulaire

Bas du formulaire