Rapport du Projet
FindMyTech

Mouadh Belgoumri (Team Leader, Frontend)
Abdelouadoude Sahnoune (Frontend)
Khaled Houari (Backend)
Arab Mehdi (Backend)
Cheurfa Aimed (UI/UX)
Zinnedine Zitouni (UI/UX)
Encadré par : Mdm. Feriel Khennouche

17 mars 2025

Table des matières

1 Titre du projet

FindMyTech

2 Contexte et justification

L'achat d'équipements informatiques peut être complexe en raison de la diversité des produits et des spécifications techniques. FindMyTech vise à simplifier ce processus en fournissant un comparateur de produits performant avec une boutique intégrée permettant l'achat direct.

3 Objectifs du projet

— Développer une plateforme intuitive permettant aux utilisateurs de comparer facilement des équipements informatiques.
— Offrir une base de données centralisée regroupant des fiches techniques détaillées.
— Permettre un filtrage avancé selon les catégories et budgets.
— Intégrer un magasin en ligne facilitant l'achat direct des produits comparés.

4 Acteurs du système

4.1 Administrateur

— Gestion des produits et des catégories.
— Supervision des commandes et des transactions.
— Maintenance de la plateforme et gestion des utilisateurs.

4.2 Acheteur

— Recherche et comparaison de produits.
— Ajout de produits favoris et suivi des prix.
— Achat direct via le magasin intégré.

4.3 Vendeur

— Gestion du stock et des prix.
— Réception et gestion des commandes clients.
— Interaction avec l'administration pour la mise en ligne des produits.

5 Technologies utilisées

— **Frontend** : HTML, CSS, JavaScript.
— **Backend** : Php, Laravel.
— **Base de données** : SQL.
— **UI/UX** : Figma.
— **API externes** : récupération des informations techniques des produits.

6 Architecture du projet

FindMyTech suit une architecture MVC (Modèle-Vue-Contrôleur) garantissant une séparation claire entre la logique métier, l'affichage et la gestion des données.

7 Contraintes et exigences

7.1 Contraintes Techniques

— Compatibilité avec les navigateurs modernes.
— Temps de réponse optimisé.
— Interface utilisateur fluide et réactive.

7.2 Contraintes de Sécurité

— Sécurisation des paiements et des transactions.
— Protection des données des utilisateurs.
— Authentification sécurisée avec JWT.

7.3 Contraintes Ergonomiques

— Interface intuitive et accessible pour tous les utilisateurs.
— Navigation fluide et structurée.

8 Phases du projet

— **Phase 1** : Analyse et conception (schéma de base de données, maquettes UI/UX).
— **Phase 2** : Développement des fonctionnalités principales.
— **Phase 3** : Tests et corrections des bugs.
— **Phase 4** : Présentation finale et soumission du rapport.

9 Livrables attendus

— Code source documenté.
— Rapport technique du projet.
— Présentation orale et démonstration de la plateforme.

10 Diagrammes UML

10.1 Diagramme de cas d'utilisation

10.1.1 Cas d'utilisation principaux

10.2 Pour l'Acheteur

- **Créer un compte** : L'utilisateur peut s'inscrire sur la plateforme.
- **Se connecter** : Il doit s'authentifier avant d'accéder à certaines fonctionnalités.
- **Changement des données personnelles** : Il peut modifier ses informations personnelles après connexion.
- **Recherche et filtrage** : Il peut rechercher un produit en utilisant des filtres spécifiques.
- **Comparaison** : Il peut comparer plusieurs produits en fonction de leurs caractéristiques.
- **Paiement** : Après avoir sélectionné un produit, il peut procéder au paiement.
  - ***«include» Vérifier la suffisance des fonds*** : Avant la validation du paiement, le système vérifie que l'utilisateur dispose des fonds nécessaires.

    Détails de la vérification des fonds

    - Vérification du solde disponible.
    - Vérification des limites de transaction.
- **Afficher une erreur d'authentification** : Si l'identification échoue, un message d'erreur s'affiche.

10.3 Pour le Vendeur

- **Mettre un nouveau produit** : Le vendeur peut ajouter un produit à vendre sur la plateforme.
- **Retirer son produit** : Il peut supprimer ses propres produits.
- **Consulter l'état du stock** : Il peut vérifier l'état du stock des produits qu'il vend.

10.4 Pour l'Admin

- **Consulter les statistiques** : L'admin a accès aux statistiques globales de la plateforme.
- **Consulter la liste des vendeurs** : Il peut voir les vendeurs enregistrés.
- **Modifier les produits (suppression et ajout)** : Il peut gérer les produits disponibles sur la plateforme.

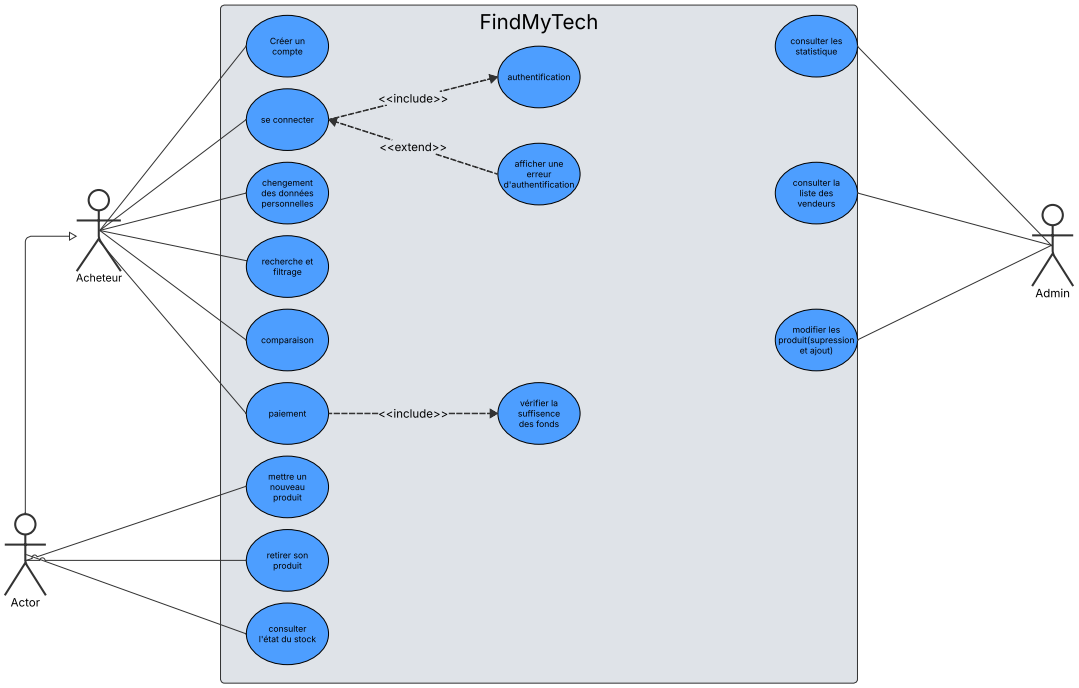


FIGURE 1 – Diagramme de cas d'utilisation

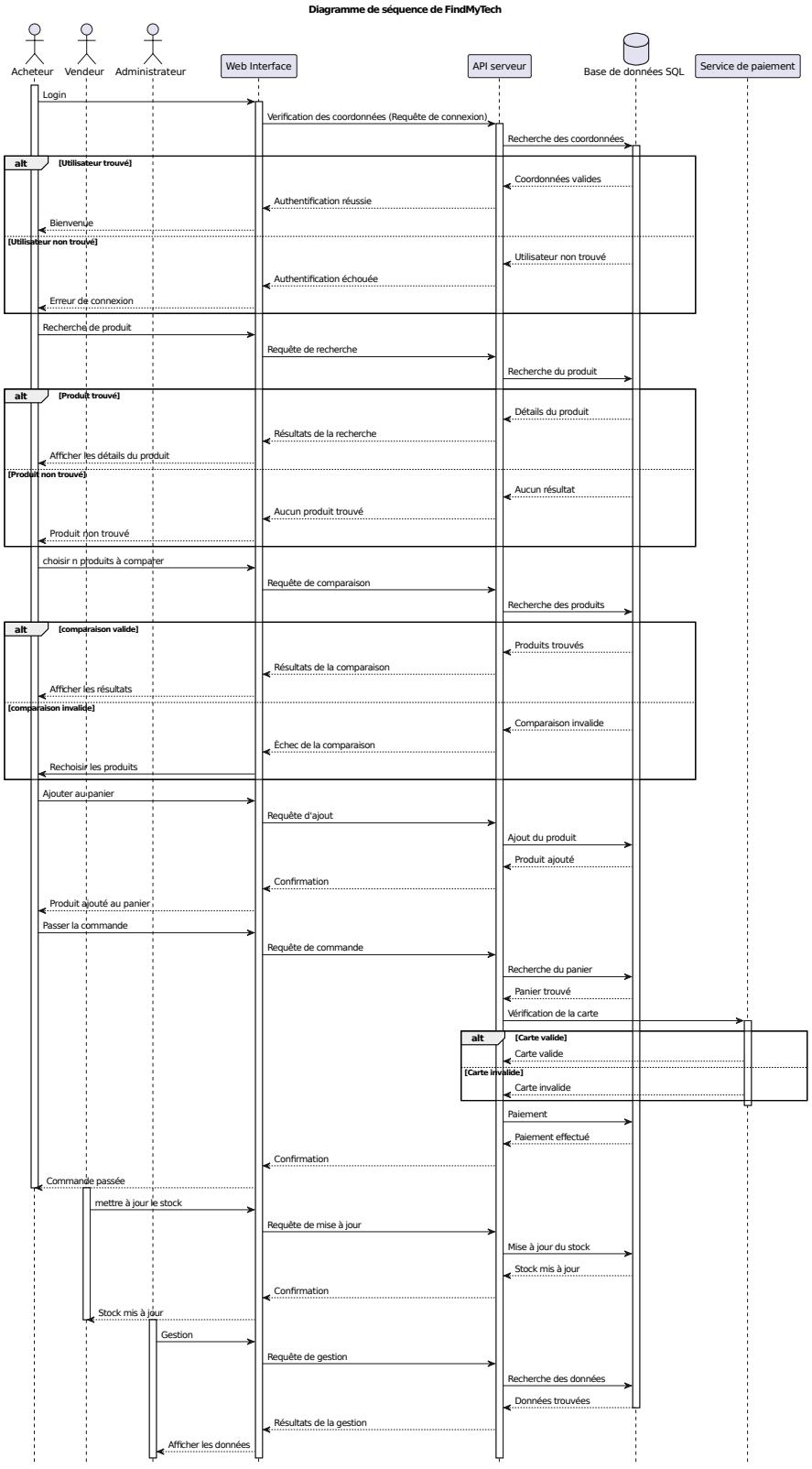
11 Relations entre les cas d'utilisation

— **«include» Vérifier la suffisance des fonds** est inclus dans le cas d'utilisation **Paiement**, car il s'agit d'une étape obligatoire pour valider une transaction.

— **«extend» Afficher une erreur d'authentification** est une extension du cas **Se connecter**, activée uniquement en cas d'échec de l'authentification.

11.1 Diagramme de séquence

Ce diagramme illustre les interactions dynamiques entre les utilisateurs, le système et la base de données, mettant en évidence le processus de comparaison et d'achat.



8

FIGURE 2 – Diagramme de séquence

11.2 Diagramme de classes

Il détaille la structure du projet en mettant en évidence les classes principales et leurs relations, incluant les classes *Produit*, *Utilisateur*, *Commande*, et *Panier*.

12 Description des classes

12.1 Classe Utilisateur

Attributs :

— `UserId` : int – Identifiant unique de l'utilisateur.

— `Password` : string – Mot de passe pour l'authentification.

Méthodes :

— `Créer un compte()` – Permet de créer un compte utilisateur.

— `se connecter()` – Authentifie l'utilisateur.

— `mettre_à_jour_le_profil()` – Met à jour les informations personnelles.

12.2 Classe Acheteur (Hérite de Utilisateur)

Attributs :

— `Nom` : string – Nom de l'acheteur.

— `Prénom` : string – Prénom de l'acheteur.

— `Adresse` : string – Adresse de livraison.

Méthodes :

— `ajouter_au_panier()` – Ajoute un produit au panier.

— `passer_une_commande()` – Finalise une commande.

12.3 Classe Vendeur (Hérite de Utilisateur)

Attributs :

— `adresse_de_magasin` : string – Adresse du magasin du vendeur.

Méthodes :

— `mettre_à_jour_son_stock()` – Permet de modifier les stocks disponibles.

12.4 Classe Administrateur (Hérite de Utilisateur)

Attributs :

— `SuperID` : int – Identifiant spécifique de l'administrateur.

Méthodes :

— `gestion_de_BD()` – Gestion de la base de données.

— `consulter_liste_vendeurs()` – Affiche la liste des vendeurs enregistrés.

— `consulter_ventes()` – Affiche les ventes réalisées.

12.5 Classe Panier

Attributs :

— `listeDesIdProduits[]` : int – Liste des identifiants des produits ajoutés.

— `prixUnit` : int – Prix unitaire d'un produit.

— `total` : int – Prix total du panier.

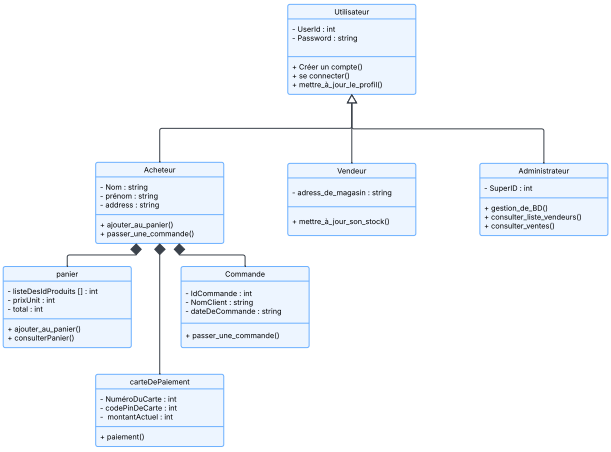


FIGURE 3 – Diagramme de classes

Méthodes :

— `ajouter_au_panier()` – Ajoute un produit au panier.

— `consulterPanier()` – Affiche le contenu du panier.

12.6 Classe Commande

Attributs :

— `IdCommande` : int – Identifiant unique de la commande.

— `NomClient` : string – Nom du client ayant passé la commande.

— `dateDeCommande` : string – Date de la commande.

Méthodes :

— `passer_une_commande()` – Permet de finaliser une commande.

12.7 Classe CarteDePaiement

Attributs :

— `NuméroDuCarte` : int – Numéro unique de la carte bancaire.

— `codePinDeCarte` : int – Code PIN de la carte.

— `montantActuel` : int – Solde disponible sur la carte.

Méthodes :

— `paiement()` – Effectue un paiement à partir de la carte.

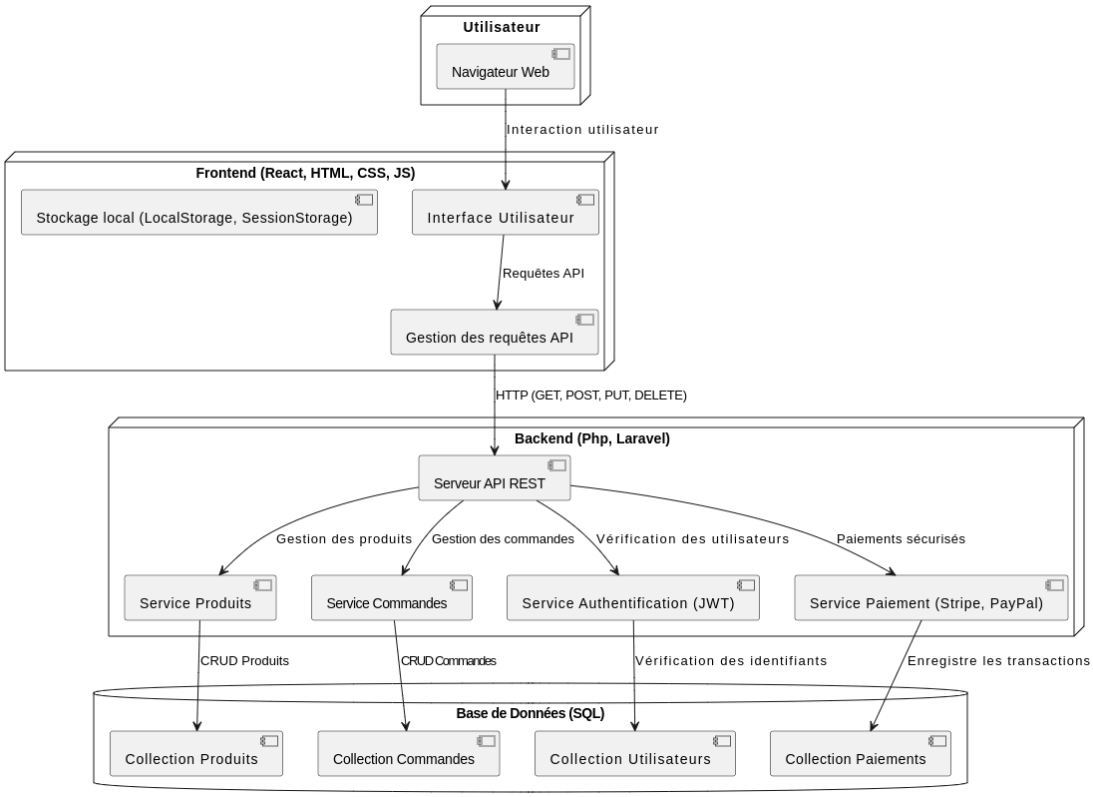


FIGURE 4 – Diagramme d'architecture

12.8 Diagramme d'architecture

Ce diagramme présente la structure logicielle du projet, mettant en avant la séparation entre l'interface utilisateur, la logique métier et la gestion des données.

13 Composants principaux

Le système est divisé en trois parties principales :

— **Frontend** (React, HTML, CSS, JS) : Interface utilisateur qui permet l'interaction avec le système.

— **Backend** (PHP, Laravel) : Serveur qui traite les requêtes et gère la logique métier.

— **Base de données** (SQL) : Stocke les informations essentielles comme les produits, commandes, utilisateurs et paiements.

14 Description détaillée

14.1 Utilisateur

L'utilisateur interagit avec le système via un **navigateur web**. Toutes les actions passent par l'interface utilisateur du frontend.

14.2 Frontend

Le frontend est développé avec **React, HTML, CSS et JavaScript**. Il contient plusieurs composants :

- **Interface utilisateur** : Affiche les pages web et gère l'interaction avec l'utilisateur.
- **Stockage local (LocalStorage, SessionStorage)** : Permet de stocker temporairement certaines données utilisateur.
- **Gestion des requêtes API** : Envoie des requêtes au backend via HTTP (`GET, POST, PUT, DELETE`).

14.3 Backend

Le backend est développé en **PHP (Laravel)** et est responsable de la gestion des données et des requêtes. Il est composé de :

- **Serveur API REST** : Point central de communication entre le frontend et la base de données.
- **Service Produits** : Gère les produits et leurs caractéristiques (CRUD Produits).
- **Service Commandes** : Gère les commandes des utilisateurs (CRUD Commandes).
- **Service Authentification (JWT)** : Vérifie les utilisateurs et gère la connexion sécurisée.
- **Service Paiement (Stripe, PayPal)** : Assure la gestion des transactions financières sécurisées.

14.4 Base de Données (SQL)

La base de données stocke toutes les informations nécessaires :

- **Collection Produits** : Stocke les informations sur les produits disponibles.
- **Collection Commandes** : Contient les détails des commandes effectuées.
- **Collection Utilisateurs** : Enregistre les comptes utilisateurs et leurs informations.
- **Collection Paiements** : Archive les transactions et paiements réalisés.

15 Conclusion

Cette architecture garantit une séparation claire entre les différentes couches du système et assure une communication fluide entre le frontend, le backend et

la base de données. Grâce à cette structuration, **FindMyTech** peut offrir une expérience utilisateur fluide, sécurisée et efficace.

16 Justification Des Technologies utilisées

16.1 Frontend

Le frontend est la partie visible du site web avec laquelle l'utilisateur interagit. Pour garantir une interface fluide, moderne et réactive, nous avons opté pour les technologies suivantes :

— **React.js** : Bibliothèque JavaScript pour la construction d'interfaces utilisateur.

— **Bootstrap** : Framework CSS permettant de créer des designs responsifs rapidement.

— **Sass** : Préprocesseur CSS facilitant l'écriture et la gestion des styles.

16.2 Backend

Pour la gestion des requêtes et des données, nous avons choisi :

— **PHP** : Langage de programmation utilisé pour le développement backend.

— **Laravel** : Framework PHP facilitant le développement d'applications web robustes et sécurisées.

— **SQL** : Système de gestion de bases de données relationnelles.

17 Justification des choix technologiques

17.1 Pourquoi React.js ?

React.js est une bibliothèque JavaScript développée par Facebook et largement utilisée pour créer des interfaces utilisateur dynamiques et performantes.

Avantages :

— **Composants réutilisables** : Permet de structurer l'interface en éléments indépendants et réutilisables.

— **Performance optimisée** : Grâce au *Virtual DOM*, React réduit le nombre de mises à jour réelles du DOM, améliorant ainsi la fluidité de l'application.

— **Grande communauté et support** : Un écosystème riche avec de nombreuses bibliothèques et ressources disponibles.

— **Facilité d'intégration** : Compatible avec d'autres technologies comme *Redux*, *React Router* et *Axios*.

17.2 Pourquoi Bootstrap ?

Bootstrap est un framework CSS populaire qui permet de créer rapidement des interfaces responsives et attrayantes.

Avantages :

— **Développement rapide** : Contient des composants prêts à l'emploi (boutons, formulaires, alertes, etc.).
— **Responsive par défaut** : Utilise une grille flexible qui s'adapte aux différents écrans.
— **Compatibilité élevée** : Fonctionne sur tous les navigateurs modernes sans nécessiter d'adaptations complexes.

17.3 Pourquoi Sass ?

Sass (Syntactically Awesome Stylesheets) est un préprocesseur CSS qui simplifie l'écriture et la gestion des styles.

Avantages :

— **Variables et mixins** : Permet d'utiliser des variables pour les couleurs, tailles et autres propriétés, rendant les styles plus cohérents et faciles à maintenir.
— **Imbrication (Nesting)** : Permet d'écrire du CSS plus propre et structuré.
— **Modularité** : Permet d'organiser le CSS en plusieurs fichiers pour un code plus lisible et réutilisable.

17.4 Pourquoi PHP et Laravel ?

PHP est un langage serveur largement utilisé pour le développement web, et **Laravel** est l'un de ses frameworks les plus populaires.

Avantages de Laravel :

— **Architecture MVC** : Favorise une séparation claire entre la logique métier et l'affichage.
— **Sécurité avancée** : Intègre des protections contre les injections SQL, XSS et CSRF.
— **ORM Eloquent** : Simplifie la gestion des bases de données.
— **API REST facile à implémenter** : Permet une communication efficace avec le frontend.

17.5 Pourquoi SQL ?

SQL est un langage utilisé pour gérer les bases de données relationnelles.

Avantages :

— **Intégrité des données** : Assure la cohérence et la fiabilité des informations stockées.
— **Puissance des requêtes** : Permet d'exécuter des opérations complexes de filtrage et d'agrégation.
— **Interopérabilité** : Compatible avec de nombreux systèmes comme MySQL, PostgreSQL et SQLite.

18 Conclusion

Le choix des technologies pour **FindMyTech** a été fait en fonction des besoins du projet et des meilleures pratiques du développement web. **React** permet d'avoir une interface interactive et performante, **Bootstrap** facilite la mise en page responsive, et **Sass** améliore la gestion des styles. Côté backend, **PHP et Laravel** assurent une gestion efficace des requêtes et de la sécurité, tandis que **SQL** garantit un stockage fiable des données. Cette combinaison garantit une application efficace, maintenable et évolutive.