|  |  |  |
| --- | --- | --- |
| C:\wamp\www\PROJET\fsts.png |  |  |

**UNIVERSITE HASSAN 1ER**

**FACULTE DES SCIENCES ET TECHNIQUES**

**-SETTAT-**

**Rapport de Projet de fin d’études**

Pour l’Obtention du Diplôme

**LICENCE SCIENCES ET TECHNIQUES**

**Spécialité : Génie Informatique**

**Conception d'un site web pour l'académie de Football ACOS Academy**

Réalisé par :

**Omar EL KHAIRI**

Sous la direction de :

**M. Said EL KAFHALI,** professeur à laFST de Settat

**M. Noureddine DARKAOUI,** ACOS Academy

*Soutenu le :***12 juin 2025**

**JURY**

**M. Said EL KAFHALI**, professeur à la FST de Settat àé

**M. Abderrahim MARZOUK**, professeur à la FST de Settat

**M. Mohamed HANINI**, professeur à la FST de Settat

**M. Jamal JABALI**, professeur à la FST de Settat

**Année Universitaire : 2024-2025**

## ****Sommaire****

Dédicace .................................................................................................................................... 3  
Remerciements ......................................................................................................................... 4  
Résumé ..................................................................................................................................... 5  
Abstract .................................................................................................................................... 6  
Sommaire ................................................................................................................................. 7  
Liste des tableaux ................................................................................................................... 10  
Liste des figures ...................................................................................................................... 11  
Liste des acronymes ............................................................................................................... 12  
Introduction générale ............................................................................................................ 13

### ****Chapitre I : Contexte général du projet**** ..................................................................... 15

Introduction ............................................................................................................................ 15  
I. Présentation de l’organisme d’accueil ............................................................................... 16

1. Présentation du projet personnel ............................................................................... 16
2. Organigramme simplifié du travail ......................................................................…...17  
   II. Contexte du projet ............................................................................................................. 18
3. Critique de l’existant ......................................................................................................... 18
4. Problématique ................................................................................................................... 19
5. Objectif(s) du projet .......................................................................................................... 19
6. Solution(s) proposée(s) ..................................................................................................... 20  
   III. Planification et conduite du projet .................................................................................. 21
7. Méthodologie de travail ..................................................................................................... 21
8. Planning prévisionnel ........................................................................................................ 22
9. Diagramme de Gantt ......................................................................................................... 23  
   Conclusion dué chapitre ........................................................................................................... 24

### ****Chapitre II : Analyse et conception**** .......................................................................... 25

Introduction ............................................................................................................................ 25  
I. Besoins fonctionnels .......................................................................................................... 26  
II. Identification et description des acteurs ........................................................................... 27  
III. Diagramme de contexte statique .................................................................................... 28  
IV. Cas d’utilisation / Fonctionnalités attendues .................................................................. 29  
V. Diagrammes de séquences ................................................................................................ 30  
VI. Diagrammes d’activités .................................................................................................... 32  
VII. Diagramme de classes ...................................................................................................... 34  
VIII. Besoins non fonctionnels / techniques ........................................................................... 35

* Performance ........................................................................................................................ 35
* Sécurité ............................................................................................................................... 35
* Portabilité ............................................................................................................................ 36
* Maintenance ....................................................................................................................... 36

### ****Chapitre III : Étude technique**** ................................................................................... 37

Introduction ............................................................................................................................ 37  
I. Architecture logicielle du système .................................................................................... 38  
II. Technologies utilisées ....................................................................................................... 39  
III. Frameworks et bibliothèques .......................................................................................... 40  
IV. Outils de développement .................................................................................................. 41  
V. Choix du SGBD .................................................................................................................. 42

### ****Chapitre IV : Réalisation et mise en œuvre**** .............................................................. 43

Introduction ............................................................................................................................ 43  
I. Structure du projet ............................................................................................................. 44  
II. Schéma de la base de données ......................................................................................... 45  
III. Présentation des interfaces / Captures d’écran ............................................................... 46  
IV. Tests réalisés ..................................................................................................................... 50  
V. Qualité du code .................................................................................................................. 51  
VI. Diagramme de Gantt réel .................................................................................................. 52  
Conclusion du chapitre ........................................................................................................... 53

### ****Conclusion générale et perspectives**** ......................................................................... 54

Webographie / Bibliographie ................................................................................................ 55  
Annexes .................................................................................................................................. 56

# **CHAPITRE 1 :** **Contexte générale du projet**

## Presentation et context de project :

#### Presenation :

Le projet consiste en la conception et le développement d'une application intelligente

visant à optimiser l'employabilité des étudiants en informatique au Maroc. Cette

application permettra aux professeurs d'ajouter les étudiants en fournissant leurs

données. Grâce à un modèle de machine Learning intégré, les données ajoutées seront

analysées pour déterminer si l'étudiant est employable ou non.

L'importance de l'employabilité des étudiants en informatique au Maroc réside dans la

nécessité de les préparer efficacement pour le marché du travail compétitif. En

fournissant une évaluation des données des étudiants et en les aidant à identifier leurs

lacunes, cette application vise à renforcer leurs chances de trouver un emploi

correspondant à leurs qualifications

## Context :

Avec la croissance continue du commerce en ligne, les entreprises ont besoin d’interfaces de gestion efficaces, ergonomiques et sécurisées. Les dashboards e- commerce permettent aux administrateurs de superviser les commandes, gérer les stocks, suivre les ventes et avoir une vue d’ensemble sur l’activité.

Cependant, les solutions existantes sont souvent :

* + complexes à utiliser pour des petites structures,
  + peu personnalisables,
  + ou payantes.

Ainsi, j’ai souhaité développer une solution **moderne, intuitive et personnalisable**, intégrant :

* + une architecture **full stack**,
  + un design responsive basé sur **React.js** et **Tailwind CSS** (avec **ShadCN UI….**),
  + un backend avec **PHP, Larvel**,
  + une base de données relationnelle **MySQL** .

L’objectif de ce projet est double :

* 1. **Mettre en pratique** les compétences acquises durant la formation (conception, développement, architecture logicielle, UX/UI…
  2. **Proposer une base solide** pour une future application e-commerce réelle ou comme produit open-source.

## ****Organigramme :****

Le projet ayant été réalisé dans le cadre d’un travail personnel validé par un encadrant académique, l’organigramme du projet est simplifié comme suit :

* **Encadrant universitaire** : Responsable du suivi pédagogique du projet, émet des recommandations, valide les étapes du projet.
* **Étudiant développeur (moi-même)** : Responsable de l’ensemble des tâches liées au projet, à savoir :
  + L’analyse des besoins fonctionnels et techniques,
  + La conception de l’architecture logicielle et de la base de données,
  + Le développement du frontend et du backend,
  + La mise en œuvre des interfaces utilisateur,
  + La rédaction de la documentation technique et du rapport final

## ****Critique de l’existant****

Il existe aujourd’hui plusieurs plateformes e-commerce comme **Shopify**, **WooCommerce** ou **Prestashop** qui proposent des interfaces d’administration. Cependant, ces solutions ne sont pas toujours adaptées pour :

* Les petits projets ou les développeurs débutants,
* Les besoins simples et personnalisés,
* L’apprentissage et la mise en pratique des compétences techniques.

De plus, ces outils sont souvent :

* Complexes à utiliser,
* Difficiles à personnaliser,
* Payants ou avec des fonctionnalités limitées en version gratuite.

Dans le cadre d’un projet de fin d’études, utiliser ces plateformes ne permet pas de montrer ses compétences en développement. C’est pourquoi j’ai choisi de créer **mon propre dashboard e-commerce** pour gérer les produits, les clients et les commandes.

## ****Problématique****

Dans un contexte où le commerce en ligne occupe une place de plus en plus importante, les entreprises ont besoin d’**outils de gestion efficaces** pour piloter leurs activités. Les tableaux de bord proposés par les solutions e-commerce existantes sont souvent **complexes**, **coûteux**, ou **difficiles à personnaliser** pour des besoins spécifiques, notamment dans les petites structures ou les projets pédagogiques.

Face à cette situation, se pose la question suivante :

**Comment développer une solution web personnalisée et accessible, permettant à un administrateur de gérer de manière centralisée les produits, les clients et les commandes d’une boutique en ligne, tout en assurant une bonne ergonomie, une visualisation claire des données, et une architecture logicielle robuste ?**

Cette problématique implique des défis à la fois techniques (architecture full-stack, sécurité, performance) et fonctionnels (interface intuitive, gestion fluide des entités, visualisation de statistiques).

## Objectifs du projet

L’objectif principal de ce projet est de concevoir et développer un **dashboard e-commerce d’administration** complet et performant, permettant à un administrateur de gérer efficacement les produits, clients et commandes d’une boutique en ligne.

Les objectifs spécifiques sont :

* + **Concevoir une interface utilisateur ergonomique et robuste** avec **React** en **TypeScript**, pour bénéficier d’un typage statique, d’une meilleure maintenabilité et d’une expérience développeur améliorée. Le design est moderne et responsive, avec l’utilisation de **Tailwind CSS** et **Shadcn UI**
  + **Développer un backend sécurisé et performant** avec **Laravel**, qui expose une **API RESTful** pour gérer la logique métier, l’authentification, la validation des données, et la persistance dans une base de données relationnelle **MySQL**.
  + **Assurer une séparation claire entre frontend et backend** via des API REST, facilitant la maintenance et la scalabilité.
  + **Implémenter les fonctionnalités essentielles** : gestion complète des produits (création, modification, suppression, inventaire), gestion des commandes (suivi, statuts, filtres) et administration des clients (profil, historique d’achats).
  + **Afficher des indicateurs clés de performance (KPI)** dans un tableau de bord interactif pour une visualisation synthétique des ventes, commandes, et autres données importantes.
  + **Appliquer les bonnes pratiques** de développement web full stack, notamment en termes de sécurité (authentification JWT ou Sanctum), validation, tests unitaires et fonctionnels, ainsi que documentation technique.

Ce projet permet ainsi de mettre en œuvre des compétences avancées en développement full stack moderne, avec des technologies largement utilisées en industrie.

* + **Respecter les bonnes pratiques de développement**, notamment en matière de sécurité (authentification, gestion des sessions, validation des données), de tests et de documentation.

Ce projet vise à la fois un usage pédagogique pour valider les compétences acquises et un prototype fonctionnel pouvant servir de base à un futur produit ou projet personnel.

## Solution proposée :

Pour répondre à la problématique posée, j’ai développé une solution complète basée sur une architecture **full stack moderne** qui sépare clairement la partie frontend et backend, afin d’assurer modularité, maintenance et évolutivité.

#### Frontend :

* + - **Technologie** : React avec **TypeScript**  
      L’utilisation de TypeScript permet d’améliorer la robustesse du code grâce à un typage statique, facilitant la détection d’erreurs dès la phase de développement.
    - **Design** : Interface utilisateur moderne, responsive et intuitive, réalisée avec Tailwind CSS, offrant une expérience utilisateur fluide sur desktop et mobile.
    - **Fonctionnalités** :
    - Gestion des produits, clients et commandes via des formulaires dynamiques.
    - Tableau de bord affichant les indicateurs clés de performance (ventes, commandes en cours, statistiques clients).
    - Recherche, filtres et pagination pour faciliter la navigation dans les listes.
    - Authentification sécurisée avec gestion des sessions utilisateur.

#### Backend

* + **Technologie** : Laravel (PHP)  
    Laravel fournit un cadre structuré et sécurisé pour créer une API RESTful, responsable de la logique métier, la gestion des utilisateurs, la validation des données et la communication avec la base de données.
  + **API RESTful** :  
    Le backend expose des endpoints pour le frontend, permettant les opérations CRUD (Create, Read, Update, Delete) sur les entités principales (produits, clients, commandes).
  + **Sécurité** :  
    Authentification sécurisée via Laravel Sanctum ou JWT, contrôle d’accès basé sur les rôles, protection contre les attaques courantes (injection SQL, etc.).
  + **Base de données** :  
    Utilisation d’une base de données relationnelle **MySQL** pour stocker les données de façon structurée et fiable.

#### Architecture

* + - **Séparation Frontend / Backend** :  
      Le frontend et le backend communiquent via des appels API, permettant une indépendance technologique et facilitant la maintenance et les évolutions futures.
    - **Modularité et évolutivité** :  
      Le code est organisé en modules clairs, facilitant l’ajout de nouvelles fonctionnalités (ex : gestion des promotions, notifications, etc.).

#### Avantages de la solution

* + - Adaptée aux besoins réels d’un administrateur e-commerce.
    - Solution personnalisée, légère et évolutive.
    - Utilisation de technologies modernes et reconnues, permettant un apprentissage approfondi et une application professionnelle.
    - Interface utilisateur claire et agréable, simplifiant la gestion quotidienne.

## Conduite du projet / Méthodologie de travail

Pour mener à bien ce projet, j’ai adopté une méthodologie agile et itérative permettant de livrer un produit fonctionnel rapidement tout en restant flexible face aux évolutions et aux retours.

#### 1. Analyse des besoins

* + - Identification des fonctionnalités principales (gestion produits, commandes, clients).
    - Rédaction d’un cahier des charges simplifié pour cadrer les objectifs et le périmètre.
    - Priorisation des fonctionnalités selon leur valeur métier.

#### 2. Conception

* + - Modélisation de la base de données schéma relationnel MySQL.
    - Conception des maquettes UI / UX sous forme de wireframes pour valipder l’ergonomie.
    - Définition de l’architecture technique (séparation frontend/backend, choix des technologies).

#### 3. Développement itératif

* + - Découpage du projet en sprints de 1 à 2 semaines.
    - Développement du backend avec Laravel pour exposer les APIs REST nécessaires.
    - Développement parallèle du frontend avec React + TypeScript, consommation des APIs.
    - Intégration progressive des fonctionnalités, tests unitaires et fonctionnels.

#### 4. Tests et validation

* + - Tests manuels pour vérifier la conformité aux besoins.
    - Correction des bugs et ajustements UI/UX.
    - Tests de sécurité et validation des accès (authentification, rôles).

#### 5. Documentation

* + - Rédaction d’une documentation technique pour faciliter la maintenance.
    - Guide utilisateur simple pour la prise en main de l’application.

#### 6. Déploiement

* + - Mise en production sur un serveur adapté (hébergement Laravel + frontend build).
    - Configuration des environnements (production, développement).
    - Surveillance post-déploiement pour détecter d’éventuels problèmes.

#### 7. Suivi et amélioration continue

* + - Recueil des retours utilisateurs pour identifier des pistes d’amélioration.
    - Planification d’évolutions futures selon les besoins métiers.

## Planification du projet / Planning prévisionnel

Le projet a été découpé en 5 grandes phases, chacune avec des tâches précises et une durée estimée. Ce découpage m’a permis de suivre une progression logique et de garder une bonne visibilité sur l’avancement.

|  |  |  |
| --- | --- | --- |
| ****Phase**** | **Tâches principales** | **Tâches principales** |
| **1. Préparation** | - Analyse des besoins  - Cahier des charges  - Choix des technologies | 1 semain |
| **2 . Conception** | - Maquettes UI (figma ou autre, Dribble)  - Modélisation de la base de données  - Architecture de l’application (front + back) | 1 semaine |
| **3**. **Développement Backend** | - Création du projet Laravel  - Mise en place de l’authentification  - Création des routes et contrôleurs API  - Gestion base de données (migrations, seeders) | 2 semaines |
| **4. Développement Frontend** | - Initialisation du projet React avec TypeScript  - Connexion aux API Laravel  - Création des composants (produits, clients, commandes, tableau de bord, etc.)  - Navigation, validations, messages d’erreur | 2 à 3 semaines |
| **5. Tests & Déploiement** | - Tests manuels et débogage  - Améliorations UI/UX  - Documentation utilisateur + technique  - Déploiement final | 1 semaine |

### Outils de suivi

* **Git** : suivi des versions du code, branches pour chaque fonctionnalité.

## ****Conclusion****

Ce premier chapitre a permis de situer le projet dans son environnement global. Nous avons présenté l’organisme d’accueil ainsi que son organisation, ses activités principales et son secteur d’intervention. Ensuite, nous avons détaillé le contexte dans lequel s’inscrit le projet, en mettant en évidence les limites de l’existant, les besoins identifiés, ainsi que la problématique à résoudre.

À partir de cette analyse, nous avons défini les objectifs à atteindre et proposé une solution adaptée, fondée sur une architecture moderne combinant Laravel pour le backend et React avec TypeScript pour le frontend. La conduite du projet s’est appuyée sur une méthodologie agile et un découpage en phases bien structurées, accompagnées d’un planning prévisionnel clair et réaliste.

Ce cadre général nous permettra d’aborder dans le chapitre suivant les aspects techniques liés à la conception et au développement de l’application, en assurant une continuité logique dans la présentation du travail effectué.

# **Chapitre II : Analyse et conception**

## Besoins fonctionnels :

Les besoins fonctionnels décrivent les services que le système doit offrir à ses utilisateurs. Dans le cadre de notre projet, les besoins fonctionnels identifiés sont les suivants :

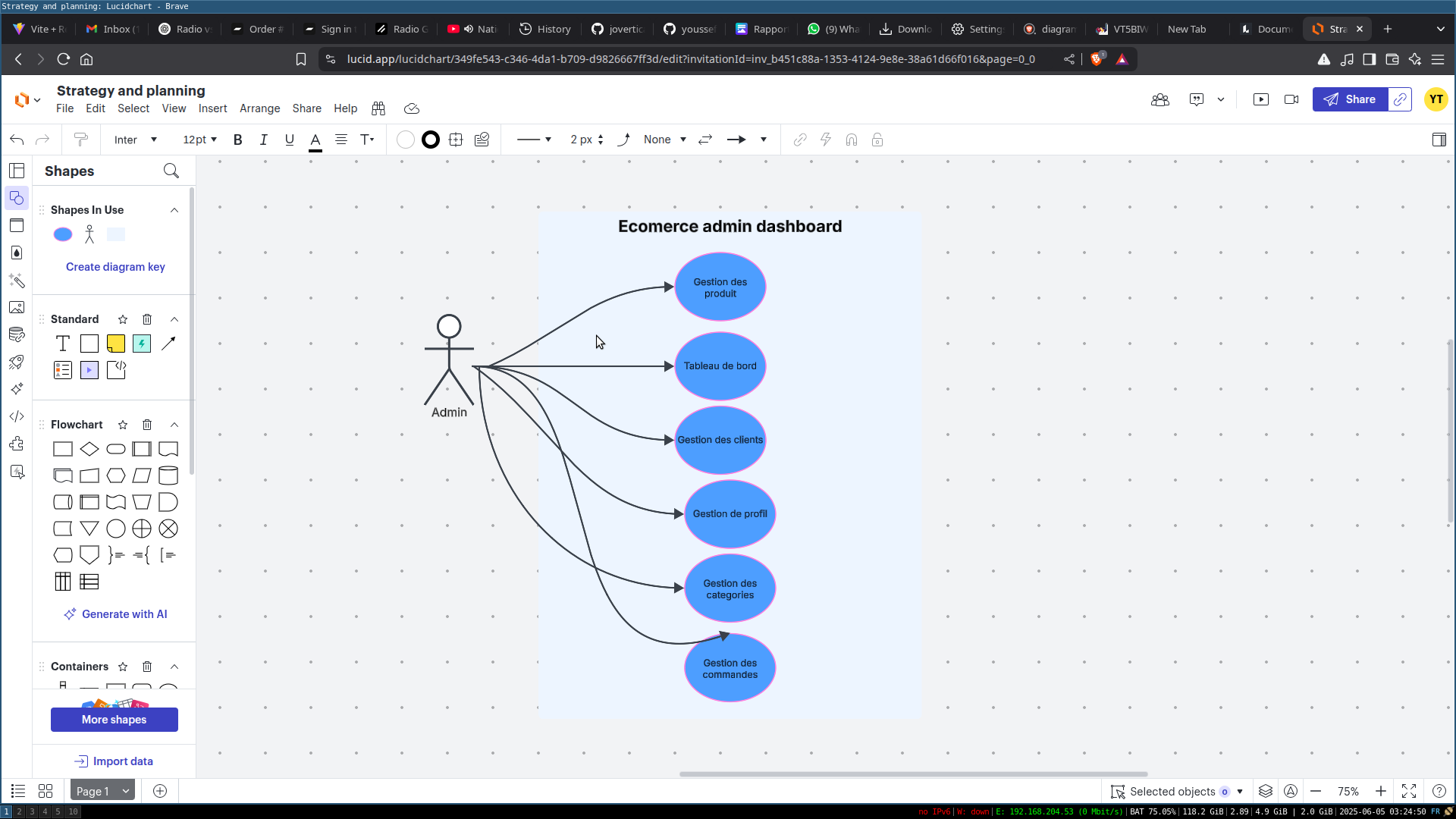
1. Authentification et gestion des rôles (Admin, Utilisateur…).
2. Gestion des produits : ajout, modification, suppression, visualisation.
3. Gestion des commandes : suivi des commandes, changement de statut.
4. Gestion des clients : informations personnelles, historique d’achat.
5. Tableau de bord pour l’administration avec des statistiques (ventes, utilisateurs actifs…).
6. Interface utilisateur fluide et responsive.

#### Identification et description des acteurs :

|  |  |
| --- | --- |
| Acteur | Description |
| **Administrateur** | Gère les utilisateurs, produits, commandes, statistiques… |

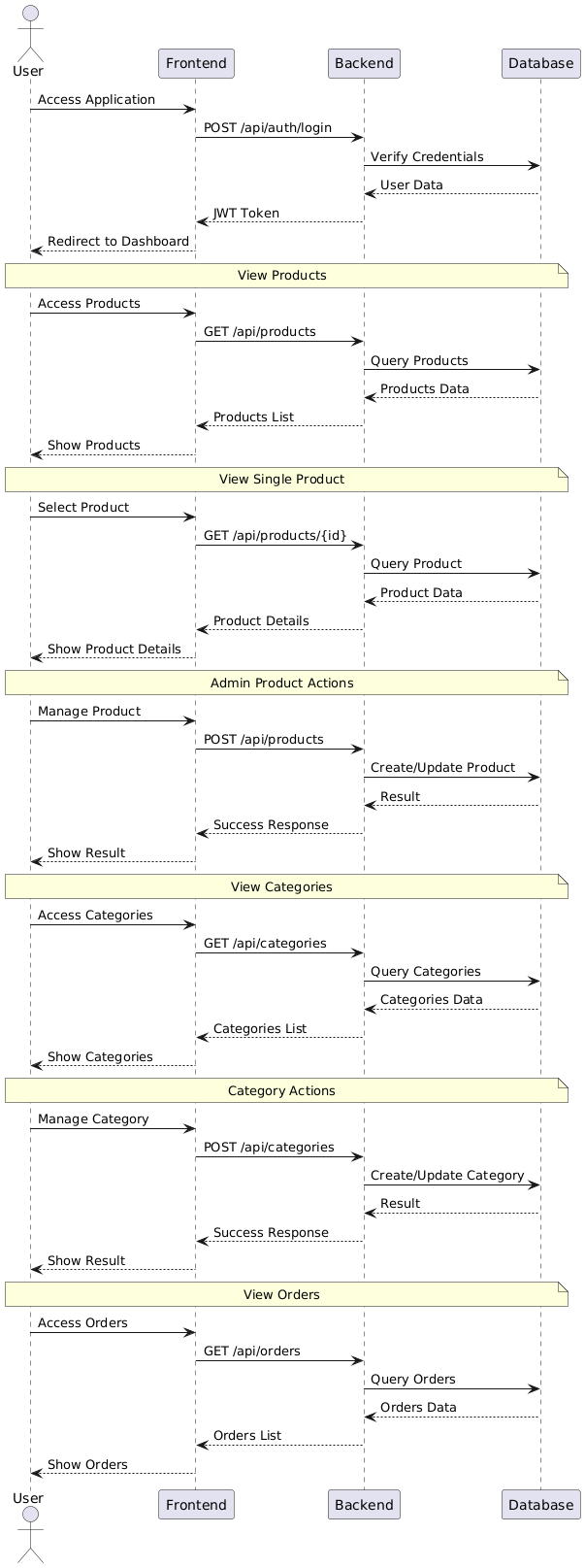
#### ****Diagramme de Cas d’Utilisation****

Le **diagramme de cas d'utilisation** (Use Case Diagram) est un outil de modélisation UML (Unified Modeling Language) permettant de représenter les interactions entre les **acteurs** du système et les **fonctionnalités (cas d'utilisation)** qu’ils peuvent exécuter. Il permet de visualiser **qui fait quoi** dans le système de manière simple et intuitive.



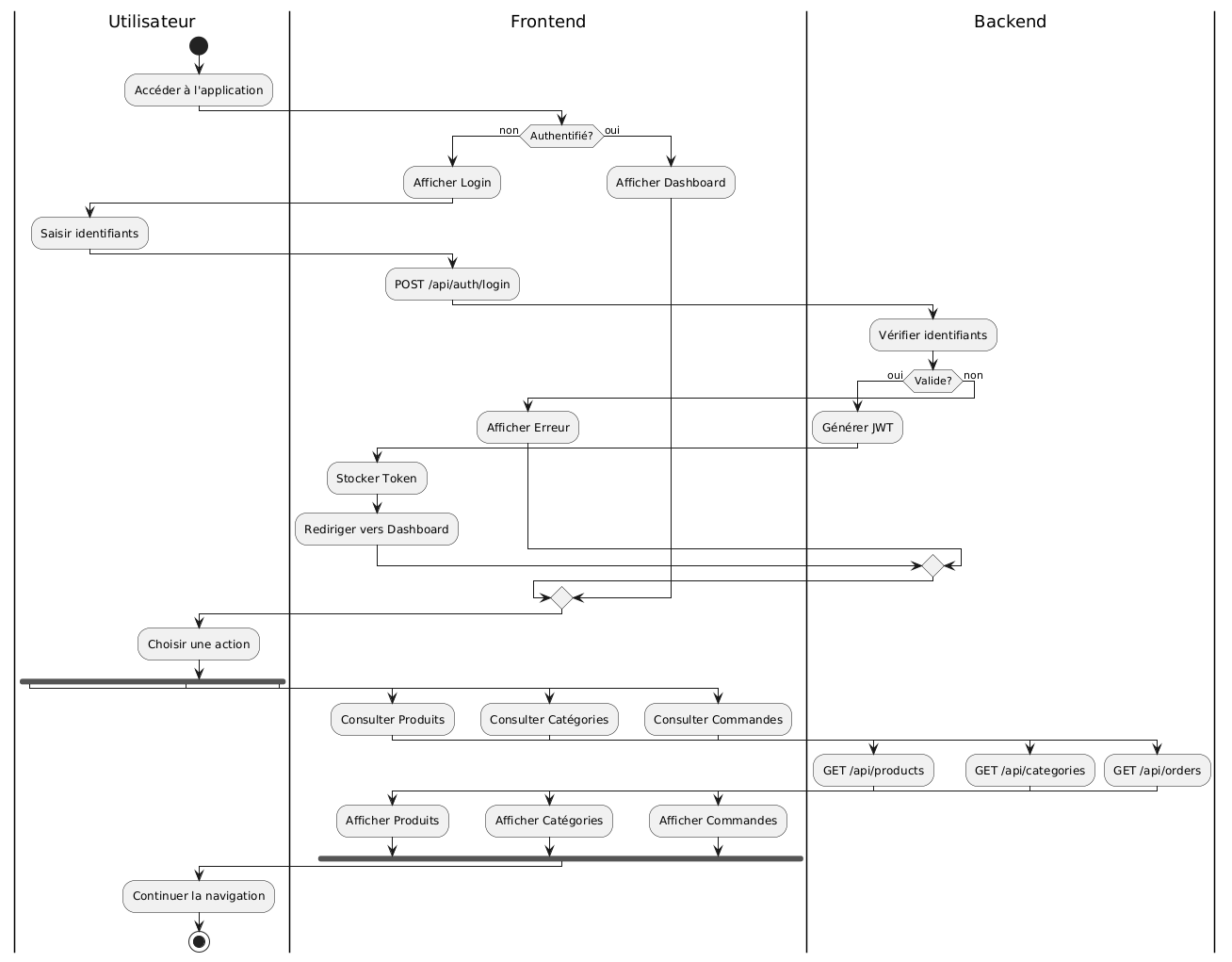
#### ****Diagrammes de Séquences****

**diagramme de séquence** est un diagramme UML qui décrit **l’enchaînement des messages échangés entre les objets ou composants d’un système** au cours d’unscénario précis. Il illustre **l’ordre chronologique des interactions** et permet de comprendre la logique du traitement d’un cas d’utilisation.



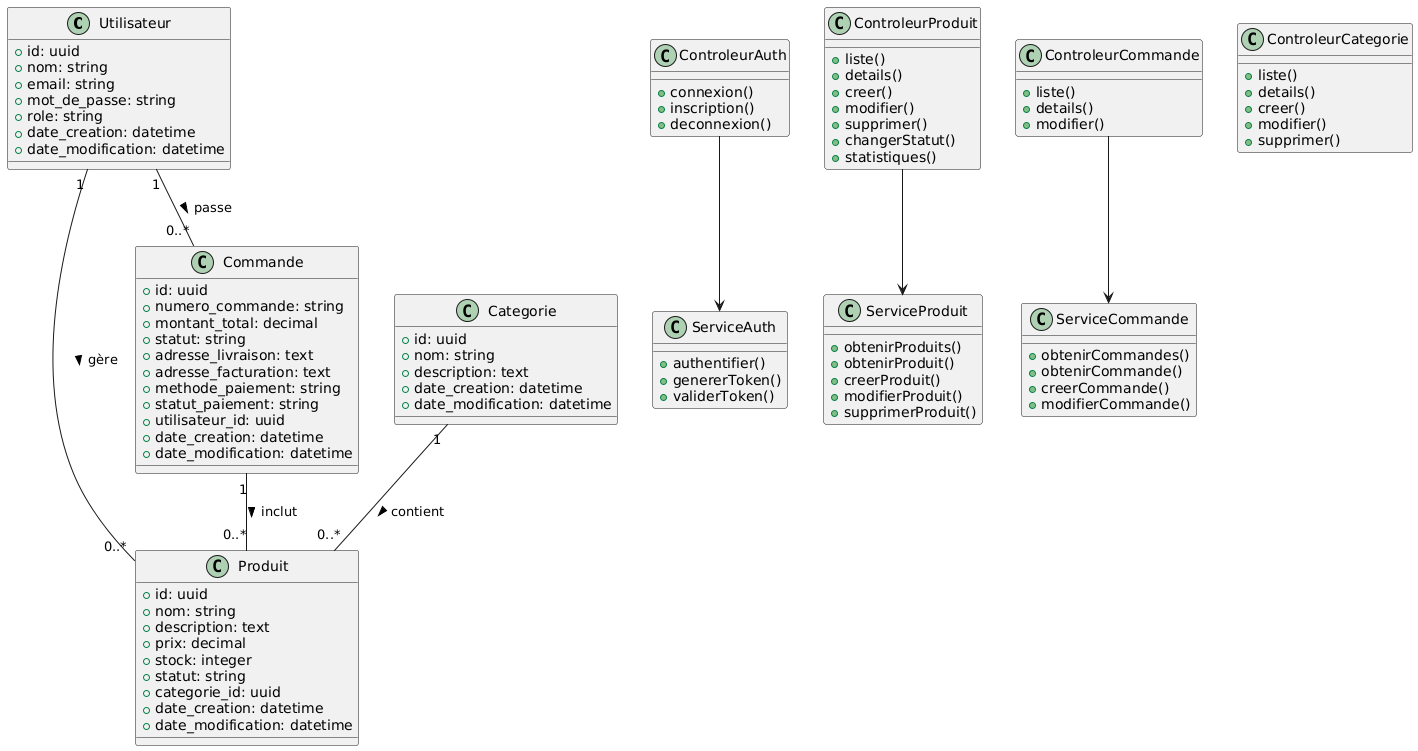
#### Diagramme d’activités – « Accès et utilisation du Dashboard Admin »

Ce diagramme montre les différentes étapes et décisions liées à l'accès et à l'utilisation du dashboard par l'administrateur.



#### Diagrame de class

Le **diagramme de classes** est un diagramme structurel de l'UML (Unified Modeling Language) qui permet de représenter **les différentes classes du système**, ainsi que **leurs relations**, **attributs** et **méthodes**.  
Il décrit **la structure statique du système** et sert de base pour la conception orientée objet.



## Besoins non fonctionnels / techniques

#### ****Performance****

Le système doit être rapide et réactif :

* + Le tableau de bord admin doit afficher les données (utilisateurs, commandes, produits, etc.) en **moins de 2 secondes**.
  + L’utilisation de React avec TypeScript garantit une **interface dynamique**, réactive et fluide grâce au DOM virtuel.
  + L’API Laravel est optimisée avec un **chargement paresseux** (lazy loading) et la **pagination** pour réduire les requêtes massives en base de données.

#### ****Sécurité****

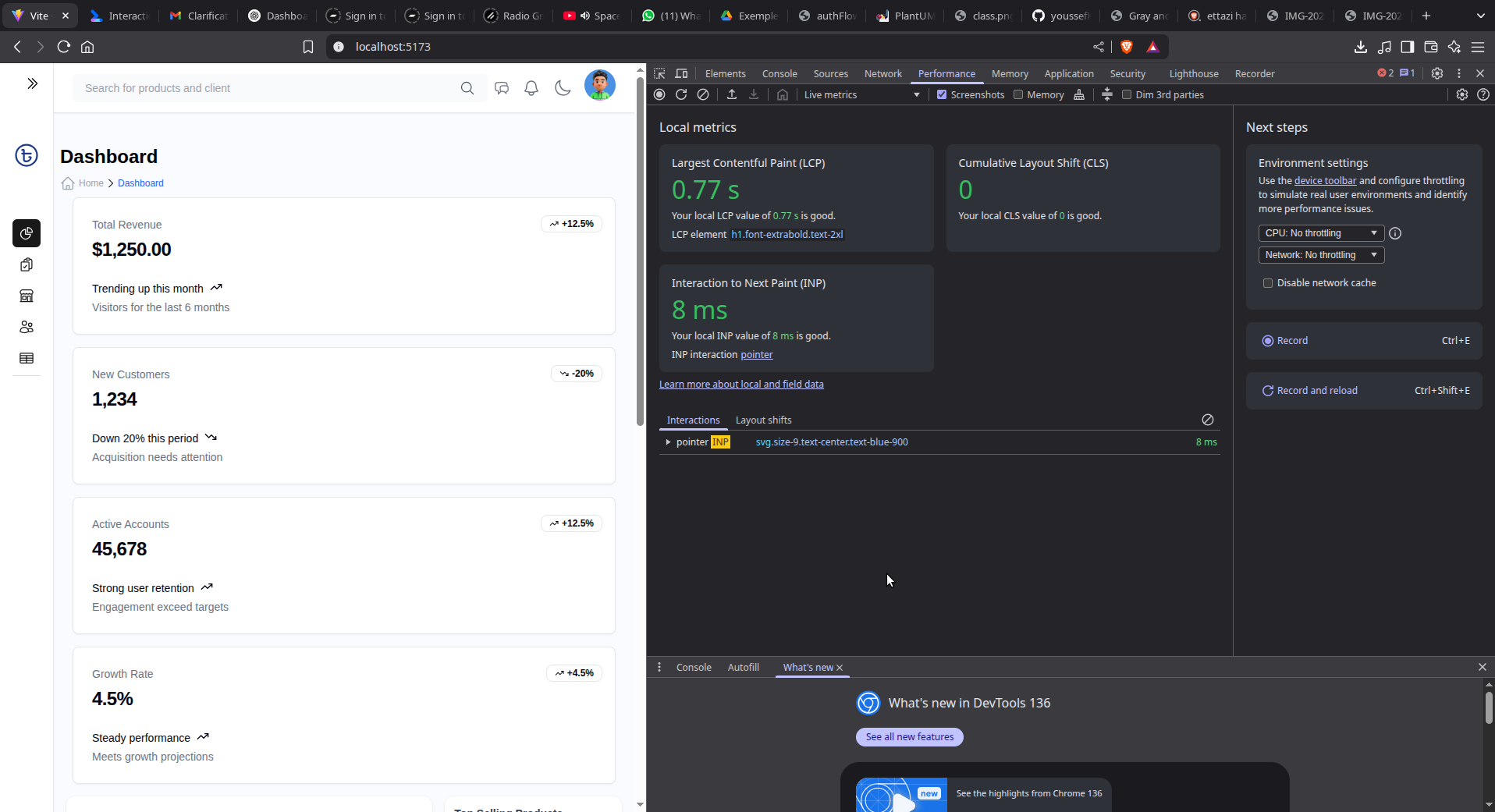
La sécurité est essentielle pour un système manipulant des données sensibles :

* + Authentification sécurisée via **tokens JWT** .
  + Gestion des rôles et permissions : **seuls les administrateurs** peuvent accéder au dashboard.
  + Données sensibles chiffrées (ex. : mots de passe avec bcrypt).
  + Protection contre les attaques injection SQL assurée par les mécanismes de Laravel et les bonnes pratiques front-end.

#### ****Maintenance****

Pour faciliter la maintenance et les évolutions futures :

* + Architecture en **composants réutilisables** côté React.
  + Utilisation de **principes SOLID** et d’un **framework MVC** (Laravel) pour un code clair et organisé.
  + Documentation du code et des API.
  + Facilité de mise à jour des dépendances grâce à **npm** et **composer**



#### ****Portabilité****

Le système doit pouvoir être déployé sur différents environnements :

* Compatible avec les navigateurs récents (Chrome, Firefox, Edge…).
* Déploiement possible sur tout serveur supportant **PHP 8+**, **MySQL**.

# **Chapitre III : Etude technique**

## ****Présentation des technologies utilisées :****

Dans le cadre du développement de cette application web, plusieurs technologies modernes ont été adoptées pour assurer la performance, la sécurité, la maintenabilité et l’ergonomie du projet. Ces technologies se divisent en deux grandes catégories : **le front-end** (interface utilisateur) et **le back-end** (logique métier, base de données et API).

#### ****Front-end :****

##### ****React avec TypeScript****

* + - * **React** est une bibliothèque JavaScript développée par Facebook permettant de créer des interfaces utilisateur dynamiques et réactives.
      * **TypeScript** est un sur-ensemble de JavaScript qui ajoute le typage statique, ce qui permet d’éviter de nombreuses erreurs lors du développement.
      * Ensemble, ils facilitent la création de composants réutilisables, typés, maintenables et faciles à tester

##### ****Tailwind CSS****

* + - * Un framework CSS utilitaire permettant de concevoir des interfaces rapidement sans écrire de styles CSS classiques.
      * Il permet un **design responsive** et **modulaire**, tout en maintenant une base de code propre et organisée.

##### ****shadcn/ui****

* + - * Une bibliothèque de composants UI accessible, construite sur Tailwind CSS et Radix UI.
      * Elle fournit des composants modernes, stylés et facilement personnalisables comme des **modals**, **toasts**, **formulaires**, etc.

##### ****Yup****

* + - * Une bibliothèque JavaScript utilisée pour la **validation des schémas de données**, notamment pour les formulaires.
      * Elle est intégrée avec des outils comme React Hook Form pour valider facilement les champs côté client.

##### ****Autres bibliothèques****

* + - * **React Hook Form** : pour la gestion des formulaires.
      * **Axios** : pour effectuer des appels HTTP vers l’API Laravel

#### ****Back-en****

##### ****Laravel (PHP)****

Un framework web robuste basé sur PHP, orienté objet et structuré en MVC (Modèle-Vue-Contrôleur).

Laravel propose :

Un système d’authentification,

Une gestion simplifiée des bases de données avec Eloquent ORM,

Des routes API RESTful simples à mettre en œuvre,

Un environnement sécurisé avec middleware, validation, etc.

##### ****MySQL****

* + - * Un système de gestion de base de données relationnelle (SGBDR).
      * Utilisé pour stocker toutes les données liées aux utilisateurs, étudiants, résultats, etc.

##### ****Cloudinary****

* + - * Une plateforme de **stockage et gestion d’images** dans le cloud.
      * Elle permet :
        + L’upload sécurisé des images via API,
        + Le redimensionnement et l’optimisation des images,
        + La récupération rapide via une URL unique.

## **Choix du SGBDR : **MySQL****

Le **Système de Gestion de Base de Données Relationnelle (SGBDR)** choisi pour ce projet est **MySQL**. Ce choix a été motivé par plusieurs raisons techniques et pratiques :

### Simplicité d’utilisation

MySQL est simple à installer, configurer et utiliser. Il est parfaitement adapté aux projets web de taille petite à moyenne, comme celui développé dans ce projet.

### Compatibilité avec Laravel

MySQL est **nativement supporté** par le framework Laravel. L’intégration avec **Eloquent ORM** est fluide, ce qui facilite la gestion des modèles, migrations et requêtes.

### Performance et fiabilité

MySQL est reconnu pour sa **rapidité d'exécution** des requêtes et sa **fiabilité** dans les environnements de production. Il convient bien aux applications nécessitant une bonne performance en lecture et en écriture.

### Documentation abondante

Il existe une **grande communauté** et une **documentation riche**, ce qui facilite la résolution des problèmes éventuels et accélère le développement.

### Portabilité

MySQL est **open source**, multiplateforme, et peut être utilisé aussi bien en local que sur des serveurs cloud. Cela permet une migration facile entre les environnements de développement et de production.

## **Les outils de développement utilisés**

Le développement de l’application s’est appuyé sur une combinaison d’outils modernes côté front-end et back-end pour optimiser la productivité, la qualité du code et la maintenance.

#### ****Éditeurs et Environnement de développement****

* + - **Neovim** : éditeur de code léger, hautement personnalisable avec support LSP, idéal pour coder rapidement en React, TypeScript et PHP.
    - **LAMP** (Linux, Apache, MySQL, PHP) : environnement local utilisé pour faire tourner Laravel.
    - **phpMyAdmin** : interface graphique pour la gestion de la base de données MySQL.

### ****Outils de développement back-end****

* + - **Laravel** : framework PHP utilisé pour créer l’API RESTful, la logique métier et l’authentification.
    - **Artisan** : outil en ligne de commande fourni par Laravel pour générer des fichiers, exécuter des migrations, lancer le serveur local, etc.
    - **Composer** : gestionnaire de dépendances PHP, utilisé pour installer Laravel et les packages tiers.

#### ****Outils de développement front-end****

* + - **React (avec TypeScript)** : bibliothèque JavaScript utilisée pour construire une interface utilisateur réactive et typée.
    - **Tailwind CSS** : framework CSS utilitaire permettant un développement rapide et propre.
    - **shadcn/ui** : bibliothèque de composants UI construite sur Tailwind CSS.
    - **Yup** : bibliothèque de validation de formulaires côté client.
    - **Vite** : outil de bundling utilisé pour le front-end React, avec rechargement à chaud rapide.
    - **npm** : gestionnaire de packages Node.js utilisé pour installer et gérer les dépendances front-end.

#### ****Outils d’API et de test****

* + - **Insomnia** : outil de test d’API pour simuler les requêtes HTTP vers l’API Laravel.
    - **Cloudinary** : plateforme de gestion d’images en ligne utilisée pour stocker et optimiser les images envoyées par les utilisateurs.

### ****Outils de versionnage****

* + - **Git** : pour la gestion du code source, le suivi des modifications et la création de branches.
    - **GitHub** : pour héberger le projet, suivre les issues, collaborer et documenter.