REPUBLIQUE DU CAMEROUN
Paix – Travail – Patrie

REPUBLIC OF CAMEROON
Peace – Work – Fatherland

UNIVERSITÉ D'EBOLOWA

École Supérieure de Transport, de Logistique et de Commerce (ESTLC)
Département de Génie Informatique

CAHIER DES CHARGES

Projet Pim Pim

Application web de transport urbain multi-villes au Cameroun

Rédigé par :	BOUHGA NJAMPOU HERMANN
Filière :	Génie Informatique des Systèmes Logistiques Intelligents (GSI)
Structure d'accueil :	CTI (Connect Technology Institute) – GP Melen, Yaoundé
Encadreur Professionnel :	Ing. ANGOA MARCEL
Méthodologie :	Agile Scrum
Année académique :	2025 – 2026

17 juin 2026

Contents

1 Présentation générale du projet **2**
- 1.1 Contexte . . . 2
- 1.2 Objectifs du projet . . . 2
- 1.3 Périmètre du projet . . . 2
  - 1.3.1 Éléments inclus . . . 2
  - 1.3.2 Éléments exclus . . . 3

2 Présentation des acteurs **4**
- 2.1 Le Client . . . 4
- 2.2 Le Chauffeur . . . 4
- 2.3 L'Administrateur . . . 4

3 Expression des besoins **5**
- 3.1 Besoins fonctionnels . . . 5
  - 3.1.1 Pour le Client . . . 5
  - 3.1.2 Pour le Chauffeur . . . 5
  - 3.1.3 Pour l'Administrateur . . . 5
- 3.2 Besoins non-fonctionnels . . . 6

4 Contraintes du projet **7**
- 4.1 Contraintes techniques . . . 7
- 4.2 Contraintes temporelles . . . 7
- 4.3 Contraintes organisationnelles . . . 7
- 4.4 Contraintes budgétaires . . . 7

5 Méthodologie de gestion de projet **8**
- 5.1 Choix de la méthode Agile Scrum . . . 8
- 5.2 Adaptation au contexte solo . . . 8
- 5.3 Organisation des sprints . . . 8

6 Architecture technique envisagée **9**
- 6.1 Vue d'ensemble . . . 9
- 6.2 Justification des choix techniques . . . 9

7 Planning prévisionnel **10**
- 7.1 Diagramme de Gantt simplifié . . . 10

Chapitre 1
Présentation générale du projet

1.1 Contexte

Le secteur du transport urbain au Cameroun repose encore largement sur des solutions informelles (taxis traditionnels, motos-taxis) sans plateforme numérique locale véritablement adaptée aux réalités du marché national. Les solutions internationales existantes (Yango, Uber) ne couvrent pas l'ensemble des villes camerounaises et ne sont pas toujours pensées pour les spécificités locales.

C'est dans ce contexte que s'inscrit le projet **Pim Pim**, développé dans le cadre d'un stage au sein de CTI (Connect Technology Institute). L'objectif est de concevoir une plateforme web de réservation de courses, complète et évolutive, capable de couvrir progressivement plusieurs villes du Cameroun.

1.2 Objectifs du projet

- Permettre à un client de réserver une course de manière simple et rapide depuis une application web ;
- Permettre à un chauffeur de recevoir, accepter et effectuer des courses ;
- Permettre à un administrateur de superviser l'activité (chauffeurs, courses, villes, tarifs) ;
- Concevoir une architecture capable de gérer plusieurs villes du Cameroun de façon indépendante.

1.3 Périmètre du projet

1.3.1 Éléments inclus

- Application web responsive accessible par les clients, chauffeurs et administrateurs ;
- Gestion multi-villes (Yaoundé, Douala, et extension à d'autres villes) ;
- Réservation de course avec carte interactive et calcul de prix estimé ;
- Suivi de la position du chauffeur en temps réel ;

- Tableau de bord administrateur (gestion des chauffeurs, courses, tarifs, statistiques).

1.3.2 Éléments exclus

- Application mobile native (iOS/Android) ;
- Paiement en ligne réel (la gestion financière reste simulée pour la durée du stage) ;
- Système de facturation fiscale.

Chapitre 2
Présentation des acteurs

Le système Pim Pim met en relation trois types d'acteurs principaux, chacun disposant d'une interface et de droits d'accès spécifiques.

2.1 Le Client

Utilisateur final souhaitant se déplacer. Il réserve une course, suit son chauffeur en temps réel et évalue la prestation à la fin de la course.

2.2 Le Chauffeur

Conducteur partenaire de la plateforme. Il reçoit les demandes de course, les accepte ou les refuse, et effectue le trajet jusqu'à destination.

2.3 L'Administrateur

Gestionnaire de la plateforme. Il supervise l'ensemble de l'activité : gestion des villes desservies, des tarifs, validation des chauffeurs, et consultation des statistiques globales.

Chapitre 3
Expression des besoins

3.1 Besoins fonctionnels

3.1.1 Pour le Client

- S'inscrire et se connecter à la plateforme ;
- Sélectionner un point de départ et une destination sur une carte interactive ;
- Visualiser le prix estimé de la course avant validation ;
- Réserver une course et être mis en relation avec un chauffeur disponible ;
- Suivre en temps réel la position du chauffeur assigné ;
- Consulter l'historique de ses courses ;
- Évaluer le chauffeur et la course après le trajet.

3.1.2 Pour le Chauffeur

- S'inscrire et soumettre ses documents (véhicule, identité) ;
- Se déclarer disponible ou indisponible ;
- Recevoir et accepter ou refuser une demande de course ;
- Visualiser l'itinéraire vers le point de départ et la destination du client ;
- Mettre à jour le statut de la course (en cours, terminée) ;
- Consulter l'historique de ses courses effectuées.

3.1.3 Pour l'Administrateur

- Gérer la liste des villes desservies par la plateforme ;
- Définir et modifier la tarification par ville ;
- Valider ou bloquer les comptes chauffeurs ;
- Consulter les statistiques globales (nombre de courses, chauffeurs actifs par ville) ;
- Superviser l'ensemble des courses en cours sur une carte globale.

3.2 Besoins non-fonctionnels

- **Performance** : le système doit assurer un matching rapide entre client et chauffeur disponible ;
- **Sécurité** : authentification sécurisée via JWT, protection des données personnelles des utilisateurs ;
- **Disponibilité** : le service doit rester accessible en continu durant les heures d'utilisation ;
- **Ergonomie** : interface simple et intuitive, utilisable sur smartphone via navigateur web ;
- **Évolutivité** : l'architecture doit permettre l'ajout futur de nouvelles villes sans refonte majeure du système.

Chapitre 4
Contraintes du projet

4.1 Contraintes techniques

- Frontend développé en **React.js** (application unique avec routes différenciées selon le rôle) ;
- Backend développé en **Spring Boot** avec base de données **PostgreSQL** ;
- Cartographie interactive via la librairie **Leaflet** (OpenStreetMap) ;
- Communication temps réel via **WebSocket**.

4.2 Contraintes temporelles

Le projet doit être conçu, développé et testé sur une durée de **deux mois**, correspondant à la durée du stage au sein de CTI.

4.3 Contraintes organisationnelles

Le développement est assuré par un **unique développeur** (l'étudiant stagiaire), qui cumule les rôles de Product Owner, Scrum Master et équipe de développement dans le cadre de la méthodologie Agile Scrum retenue.

4.4 Contraintes budgétaires

Aucun outil ou service payant ne doit être utilisé, ce qui justifie le choix de Leaflet plutôt que Google Maps API.

Chapitre 5
Méthodologie de gestion de projet

5.1 Choix de la méthode Agile Scrum

La méthode Scrum a été retenue pour sa capacité à s'adapter à un contexte de développement court et évolutif. Elle permet un découpage du travail en cycles courts (Sprints), avec à chaque fin de cycle un incrément fonctionnel démontrable.

5.2 Adaptation au contexte solo

En l'absence d'équipe, les rôles Scrum sont cumulés par l'étudiant stagiaire :

Rôle Scrum	**Application au projet**
Product Owner	Définition et priorisation du besoin, en lien avec l'encadreur professionnel
Scrum Master	Organisation des sprints, suivi de l'avancement
Équipe de développement	Réalisation technique de l'ensemble du projet

5.3 Organisation des sprints

Le projet est découpé en quatre sprints de deux semaines chacun, précédés d'un sprint de cadrage.

Sprint	**Durée**	**Objectif**
Sprint 0	Cadrage	Cahier des charges, UML, architecture, backlog initial
Sprint 1	Semaines 1-2	Authentification, gestion des villes et tarifs
Sprint 2	Semaines 3-4	Réservation côté client (carte, création de course)
Sprint 3	Semaines 5-6	Interface chauffeur, matching, suivi temps réel
Sprint 4	Semaines 7-8	Tableau de bord administrateur, tests, finitions

Chapitre 6
Architecture technique envisagée

6.1 Vue d'ensemble

L'architecture retenue repose sur une séparation claire entre le frontend et le backend, communiquant via une API REST et un canal WebSocket pour le temps réel.

- **Frontend** : application React.js unique, avec des routes dédiées par rôle (`/client`, `/chauffeur`, `/admin`) ;
- **Backend** : API Spring Boot sécurisée par JWT, exposant les ressources REST et un point de communication WebSocket ;
- **Base de données** : PostgreSQL, structurée autour des entités Utilisateur, Ville, Course, Tarif et Avis.

6.2 Justification des choix techniques

Le choix de React.js et Spring Boot s'appuie sur la maîtrise déjà acquise de ces technologies par l'étudiant stagiaire lors de projets précédents, garantissant une montée en charge plus rapide du développement dans le délai imparti.

Chapitre 7
Planning prévisionnel

7.1 Diagramme de Gantt simplifié

