**Rapport de Projet**

**Conception et réalisation d'une application de covoiturage moderne et flexible**

**Réalisé par :** Tbibzat Charaf Eddine   
**Encadré par :** Abdeladim Abid

**Promotion :** 2025-2026 **YouCode-UMP6-Simplon**  
**Date :** 03/05/2025

Une image contenant Police, logo, Graphique, symbole

Le contenu généré par l’IA peut être incorrect.

**Remerciements :**

Il m'est agréable de m'acquitter d'une reconnaissance envers toutes les personnes qui m'ont soutenu et qui sont intervenues dans l'accomplissement de mon Projet Fill Rouge.

Toute ma gratitude à mon encadrant AbdAdimAbid qui n'a pas cessé de m'encourager et de me guider avec ses précieux conseils.

Je remercie également l'ensemble du corps professoral de YouCode Safi et tous ceux qui m'ont aidé de près ou de loin pour la réalisation du présent travail.

Finalement, je remercie toute personne ayant contribué à la réalisation de ce projet.

**Résumé :**

Dans le cadre pédagogique de ma formation à YouCode Safi, j'ai été amené à élaborer un travail dans lequel je présenterais mes connaissances cognitives et mettrais en pratique les compétences acquises durant ma formation. C'est pour cela que j'ai créé mon propre projet "SwiftCar".

Mon projet consiste à créer une application de covoiturage nommée "SwiftCar" qui connecte directement les conducteurs et les passagers avec une flexibilité accrue et une interface conviviale, en utilisant les technologies: React.js, Laravel, PostgreSQL, et diverses APIs.

J'ai commencé mon projet par une phase d'analyse et conception où j'ai établi les diagrammes UML nécessaires (cas d'utilisation, diagramme de classes, etc.), puis j'ai créé les maquettes des interfaces sur un outil de design. Par la suite, j'ai développé le front-end avec React.js et MUI/TailwindCSS, et implémenté le back-end avec Laravel. J'ai ajouté des fonctionnalités comme la géolocalisation, la messagerie intégrée, et un système de paiement sécurisé pour offrir une expérience utilisateur complète et moderne.

La réalisation de ce projet est passée par quatre phases principales :

* Analyse des besoins et conception
* Développement front-end et back-end
* Tests et validation
* Déploiement et finalisation

**Sommaire :**

* Remerciements
* Résumé
* Sommaire
* Liste des figures
* Liste des abréviations
* Introduction Générale
* CHAPITRE I : Contexte du travail et cahier de charge
* CHAPITRE II : Analyse et conception
* CHAPITRE III : Réalisation et mise en œuvre du projet
* Conclusion Générale
* Webographie

**Liste des figures :**

* Figure 1 : Architecture MVC
* Figure 2 : Diagramme de cas d'utilisation - Passager
* Figure 3 : Diagramme de cas d'utilisation - Conducteur
* Figure 4 : Diagramme de cas d'utilisation - Administrateur
* Figure 5 : Diagramme de classes
* Figure 6 : Maquette - Page d'accueil
* Figure 7 : Maquette - Recherche de trajets
* Figure 8 : Maquette - Profil utilisateur
* Figure 9 : Maquette - Messagerie
* Figure 10 : Interface finale - Page d'accueil
* Figure 11 : Interface finale - Réservation de trajet

**Liste des abréviations :**

| **Abréviation** | **Signification** |
| --- | --- |
| API | Application Programming Interface |
| JS | JavaScript |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |
| MVC | Modèle, Vue, Contrôleur |
| UI | User Interface |
| UX | User Experience |
| SGBD | Système de Gestion de Base de Données |

**Introduction Générale**

Mon projet est réalisé dans le cadre d'un stage de fin d'études afin de valider les compétences acquises lors de ma formation à YouCode et approfondir ces connaissances dans une mise en situation réelle sur un projet lié à notre formation.

Ce projet sert à la conception et au développement d'une application de covoiturage nommée "SwiftCar". Cette plateforme révolutionnaire vise à établir une connexion directe et transparente entre les conducteurs et les passagers. Contrairement aux solutions classiques souvent rigides et standardisées, cette application se démarque par sa flexibilité accrue, sa capacité d'adaptation aux besoins individuels et son interface conviviale et moderne.

En mettant au cœur du projet une communication fluide et en temps réel, l'application permettra aux utilisateurs d'interagir directement pour négocier les tarifs, organiser des trajets personnalisés et partager des informations de manière instantanée.

Ce rapport est subdivisé en trois chapitres : le premier décrit l'idée générale du projet et le cahier de charge, le deuxième présente théoriquement la spécification des besoins et les diagrammes de modélisation, le troisième est consacré à la représentation d'une vue globale sur le développement et les interfaces finales de l'application.

**CHAPITRE I : Contexte du travail et cahier de charge**

Dans ce chapitre, je commencerai par une présentation de l'idée générale de mon projet "SwiftCar". Ensuite, j'exposerai le cahier de charge en précisant les problématiques qu'il résout ainsi que les périmètres de ce projet et ses principaux objectifs.

**1 – Idée générale du projet :**

Pour faciliter le covoiturage et répondre aux besoins modernes de mobilité partagée, l'application "SwiftCar" propose une plateforme de mise en relation directe entre conducteurs et passagers. Cette solution se distingue des applications traditionnelles par sa flexibilité et son approche centrée sur l'utilisateur.

L'application permettra aux passagers de rechercher des trajets disponibles, de contacter directement les conducteurs pour négocier les tarifs, et de suivre leur position en temps réel grâce à la géolocalisation. Les conducteurs pourront proposer leurs trajets, définir leurs disponibilités et leurs tarifs initiaux, et interagir avec les passagers via une messagerie intégrée.

**2 – Problématiques :**

Le projet vise à résoudre plusieurs problématiques identifiées dans le domaine du covoiturage :

* **Disponibilité des conducteurs** : Difficulté à trouver un conducteur dans les zones moins fréquentées ou à des heures de faible demande.
* **Sécurité pendant le trajet** : Inquiétudes des passagers concernant leur sécurité pendant le voyage.
* **Tarifs imprévisibles** : Frustration liée à des variations tarifaires inattendues.
* **Trajets longs et délais excessifs** : Problèmes d'itinéraires non optimaux prolongeant inutilement les trajets.
* **Mauvaise communication** : Difficultés d'échange d'informations entre passagers et conducteurs.
* **Qualité du service** : Préoccupations concernant le confort et la propreté des véhicules.
* **Accessibilité pour les personnes à mobilité réduite** : Manque de solutions adaptées pour les personnes ayant des besoins spécifiques.
* **Absence d'assistance en cas de problème** : Manque de support immédiat lors de situations problématiques.

**3 – Solutions proposées :**

En considérant toutes les problématiques mentionnées dans la phase précédente, je propose les solutions suivantes tout en respectant les règles d'une application web moderne :

* **Système de géolocalisation en temps réel** : Pour suivre le conducteur et optimiser les itinéraires.
* **Messagerie intégrée sécurisée** : Pour une communication directe et efficace.
* **Système de notation et d'évaluation** : Pour garantir la qualité du service et la confiance.
* **Tarification transparente et négociable** : Pour éviter les mauvaises surprises et permettre des ajustements.
* **Vérification des conducteurs** : Pour assurer la sécurité des passagers.
* **Bouton d'urgence et alertes de sécurité** : Pour une intervention rapide en cas de problème.
* **Options d'accessibilité** : Pour les personnes ayant des besoins spécifiques.
* **Paiements sécurisés intégrés** : Pour faciliter les transactions.

**4 – Étude des besoins :**

**Besoins fonctionnels :**

Les besoins fonctionnels représentent les actions que le système doit exécuter pour être opérationnel :

* **Inscription et gestion des profils** :
  + Inscription par email, téléphone ou réseau social
  + Profils personnalisés avec photos et historique des trajets
  + Vérification d'identité pour les conducteurs
* **Gestion des trajets** :
  + Publication, modification et suppression de trajets
  + Fixation des tarifs initiaux
  + Gestion des réservations
* **Messagerie et communication** :
  + Système de chat entre passagers et conducteurs
  + Notifications en temps réel
* **Géolocalisation et suivi** :
  + Suivi en temps réel des conducteurs
  + Optimisation des itinéraires
* **Système de paiement** :
  + Paiements en ligne sécurisés
  + Historique des transactions
* **Évaluation et notation** :
  + Système de feedback après chaque trajet
  + Gestion des commentaires et réclamations

**Besoins non-fonctionnels :**

Ces exigences concernent les contraintes et qualités du système :

* **Fiabilité** : Fonctionnement cohérent sans erreurs
* **Sécurité** : Protection des données personnelles et des transactions
* **Performance** : Temps de réponse rapide et optimisation des ressources
* **Ergonomie** : Interface intuitive et conviviale
* **Compatibilité** : Fonctionnement sur différents appareils et navigateurs
* **Évolutivité** : Capacité à s'adapter aux changements et à intégrer de nouvelles fonctionnalités

**5 – Acteurs :**

L'application comporte trois types d'acteurs principaux :

* **Passager** : Recherche et réserve des trajets, communique avec les conducteurs, évalue les services
* **Conducteur** : Propose des trajets, gère ses disponibilités, accepte les réservations
* **Administrateur** : Gère la plateforme, valide les comptes conducteurs, règle les litiges

**6 – Conclusion :**

Dans ce chapitre, j'ai présenté le contexte du projet "SwiftCar", la problématique qu'il résout, les solutions proposées et les différents acteurs concernés. Dans le chapitre suivant, je vais aborder la phase de conception déployée dans ce projet.

**CHAPITRE II : ANALYSE ET CONCEPTION**

La conception est la phase créative d'un projet. Le but premier de la conception est de permettre de créer un système répondant à un besoin en tenant compte des contraintes.

Dans ce chapitre, j'utiliserai la notation UML pour modéliser l'architecture de l'application et j'expliquerai le modèle MVC que j'ai choisi pour structurer le développement.

**1 – Modèle de conception MVC :**

Le modèle MVC (Modèle-Vue-Contrôleur) est un modèle d'architecture logicielle qui sépare les données (Modèle), l'interface utilisateur (Vue) et la logique de contrôle (Contrôleur). Cette architecture favorise une meilleure organisation du code et facilite la maintenance.

**A – Rôles des composants :**

* **Modèle** : Gère les données et la logique métier. Dans notre application, il s'agit des classes qui représentent les utilisateurs, les trajets, les réservations, etc.
* **Vue** : S'occupe de l'interface utilisateur et de la présentation des données. Notre front-end développé avec React.js correspond à cette partie.
* **Contrôleur** : Gère les interactions entre le Modèle et la Vue. Notre API Laravel joue ce rôle en recevant les requêtes du front-end et en manipulant les modèles appropriés.

**B – Interactions entre les composants :**

Le diagramme ci-dessous résume les relations entre les composants d'une architecture MVC :

*(Imaginez un diagramme MVC ici)*

Les requêtes des utilisateurs sont reçues par le Contrôleur qui utilise les services du Modèle pour préparer les données. Ensuite, ces données sont transmises à la Vue pour être présentées à l'utilisateur.

**C – Avantages et inconvénients :**

**Avantages** :

* Séparation claire des responsabilités
* Code plus organisé et plus facile à maintenir
* Possibilité de travailler sur différentes parties de l'application indépendamment

**Inconvénients** :

* Complexité initiale accrue
* Nécessité d'une bonne planification

**2 – Modélisation UML :**

**A – Diagrammes de cas d'utilisation :**

**Diagramme de cas d'utilisation - Passager :**

*(Imaginez un diagramme de cas d'utilisation ici)*

Les principales fonctionnalités pour le passager incluent :

* S'inscrire et se connecter
* Rechercher des trajets
* Réserver un trajet
* Communiquer avec le conducteur
* Payer pour le trajet
* Évaluer le conducteur

**Diagramme de cas d'utilisation - Conducteur :**

*(Imaginez un diagramme de cas d'utilisation ici)*

Les principales fonctionnalités pour le conducteur incluent :

* S'inscrire et se connecter
* Créer et gérer des trajets
* Accepter ou refuser des réservations
* Communiquer avec les passagers
* Recevoir des paiements
* Évaluer les passagers

**Diagramme de cas d'utilisation - Administrateur :**

*(Imaginez un diagramme de cas d'utilisation ici)*

Les principales fonctionnalités pour l'administrateur incluent :

* Gérer les utilisateurs
* Valider les comptes conducteurs
* Gérer les trajets
* Résoudre les litiges
* Consulter les statistiques

**B – Diagramme de classes :**

Le diagramme de classes représente la structure du système en termes de classes, d'attributs, de méthodes et de relations entre les classes.

*(Imaginez un diagramme de classes ici)*

Les principales classes de notre application sont :

* User (Utilisateur)
* Driver (Conducteur)
* Passenger (Passager)
* Trip (Trajet)
* Booking (Réservation)
* Message (Message)
* Payment (Paiement)
* Review (Évaluation)
* Vehicle (Véhicule)

**C – Charte graphique :**

La charte graphique de SwiftCar a été conçue pour refléter les valeurs de l'application : modernité, confiance et simplicité.

**Palette de couleurs** :

* Couleur principale : Bleu (#1E88E5) - Représente la confiance et la sécurité
* Couleur secondaire : Vert (#4CAF50) - Évoque l'écologie et le partage
* Couleur d'accentuation : Orange (#FF9800) - Pour l'énergie et la convivialité
* Couleurs neutres : Blanc (#FFFFFF) et Gris (#F5F5F5) - Pour la clarté et la lisibilité

**Typographie** :

* Titres : Poppins (Sans-serif)
* Corps de texte : Roboto (Sans-serif)

**3 – Conclusion :**

La phase de conception est cruciale pour le succès du projet. En utilisant le modèle MVC et la modélisation UML, j'ai pu définir clairement l'architecture de l'application et les interactions entre ses différents composants. Dans le chapitre suivant, je vais détailler la mise en œuvre du projet et présenter les interfaces finales.

**CHAPITRE III : RÉALISATION ET MISE EN ŒUVRE DU PROJET**

Après avoir terminé la conception détaillée et la planification de notre application, nous traitons dans le présent chapitre les détails liés à la réalisation de l'application SwiftCar.

**1 – Les Outils de Développement :**

**A – Langages de programmation :**

**Frontend :**

* **JavaScript/TypeScript** : Langages de programmation principaux pour le développement frontend
* **HTML5** : Pour la structure des pages web
* **CSS3/SCSS** : Pour le style et la mise en page

**Backend :**

* **PHP** : Langage principal pour le développement backend avec Laravel
* **SQL** : Pour les requêtes de base de données

**B – Frameworks et bibliothèques :**

**Frontend :**

* **React.js** : Bibliothèque JavaScript pour la création d'interfaces utilisateur
* **Redux** : Pour la gestion de l'état de l'application
* **MUI (Material-UI)** : Framework CSS pour le design des composants
* **TailwindCSS** : Framework CSS utilitaire pour des styles personnalisés

**Backend :**

* **Laravel** : Framework PHP pour le développement d'API RESTful
* **Laravel Passport** : Pour l'authentification OAuth2

**C – Logiciels utilisés :**

* **Visual Studio Code** : Éditeur de code
* **Figma** : Pour la conception des maquettes
* **Postman** : Pour tester les API
* **StarUML** : Pour la création des diagrammes UML
* **Git/GitHub** : Pour le contrôle de version

**D – Base de données :**

* **PostgreSQL** : Système de gestion de base de données relationnelle

**E – APIs et services tiers :**

* **Google Maps API** : Pour la géolocalisation et le calcul d'itinéraires
* **Stripe/PayPal** : Pour les paiements en ligne
* **Firebase** : Pour les notifications push

**2 – Mise en œuvre du développement :**

**A – Architecture technique :**

L'application a été développée selon une architecture client-serveur :

* **Frontend** : Application React.js déployée sur un serveur statique
* **Backend** : API Laravel déployée sur un serveur PHP
* **Base de données** : PostgreSQL hébergée sur un serveur dédié

**B – Mise en œuvre du frontend :**

Le développement frontend a été réalisé avec React.js, en suivant une approche de développement par composants :

* Création de composants réutilisables (boutons, cartes, formulaires)
* Mise en place du routage avec React Router
* Gestion de l'état global avec Redux
* Intégration des API de géolocalisation et de paiement
* Implémentation de la messagerie en temps réel

**C – Mise en œuvre du backend :**

Le développement backend a été réalisé avec Laravel :

* Création des modèles et des migrations pour la base de données
* Développement des contrôleurs pour gérer les requêtes HTTP
* Mise en place des routes API RESTful
* Implémentation de l'authentification et de l'autorisation
* Gestion des transactions et des paiements

**D – Tests et validation :**

* Tests unitaires pour le frontend et le backend
* Tests d'intégration pour vérifier les interactions entre les composants
* Tests de performance pour assurer une bonne expérience utilisateur
* Tests d'utilisabilité avec un panel d'utilisateurs potentiels

**3 – Les Interfaces :**

Cette partie présente les principales interfaces de l'application SwiftCar.

**A – Page d'accueil :**

La page d'accueil présente une interface épurée avec :

* Une barre de recherche pour trouver rapidement un trajet
* Une carte interactive montrant les trajets disponibles à proximité
* Des sections présentant les avantages de SwiftCar
* Un appel à l'action pour s'inscrire ou se connecter

**B – Recherche et réservation de trajets :**

Cette interface permet aux passagers de :

* Rechercher des trajets selon différents critères
* Visualiser les résultats sur une carte
* Consulter les détails d'un trajet (horaires, tarifs, évaluations)
* Contacter le conducteur
* Réserver un trajet

**C – Profil utilisateur :**

Cette interface permet aux utilisateurs de :

* Consulter et modifier leurs informations personnelles
* Voir leur historique de trajets
* Consulter leurs évaluations
* Gérer leurs moyens de paiement

**D – Messagerie :**

Cette interface permet aux utilisateurs de :

* Communiquer en temps réel
* Partager leur position
* Négocier les détails du trajet
* Recevoir des notifications

**4 – Conclusion :**

Dans ce chapitre, j'ai présenté les outils et technologies utilisés pour développer l'application SwiftCar, ainsi que les principales interfaces réalisées. L'application répond aux besoins identifiés dans le cahier des charges et offre une expérience utilisateur moderne et intuitive.

**Conclusion Générale**

Le projet SwiftCar a été une opportunité enrichissante pour mettre en pratique les connaissances acquises durant ma formation à YouCode et explorer de nouvelles technologies dans un contexte réel.

J'ai pu mener à bien ce projet en suivant une méthodologie structurée, en commençant par l'analyse des besoins, puis la conception de l'architecture et des interfaces, et enfin le développement et les tests de l'application.

Les principales réalisations de ce projet sont :

* Une application de covoiturage moderne et flexible
* Une interface utilisateur intuitive et responsive
* Un système de communication en temps réel entre les utilisateurs
* Des fonctionnalités de géolocalisation et de paiement sécurisé
* Une architecture évolutive permettant l'ajout de futures fonctionnalités

Ce projet m'a permis d'acquérir de nouvelles compétences techniques, notamment dans le développement frontend avec React.js, le développement backend avec Laravel, et l'intégration d'APIs tierces. J'ai également renforcé mes compétences en gestion de projet, en conception d'interfaces utilisateur et en résolution de problèmes.

Pour l'avenir, plusieurs axes d'amélioration sont envisagés :

* Développement d'une version mobile native de l'application
* Intégration de l'intelligence artificielle pour l'optimisation des trajets
* Ajout de fonctionnalités sociales pour renforcer la communauté
* Extension à d'autres modes de transport partagé

**Webographie**

**Sites internet consultés :**

* Uber
* Pip Pip yala
* Indrive