Structure de l’application YouGo

Table des matières

I- *Arborescence globale du projet*

II- *Outils recommandés et/ou à installer*

III- *Fonctionnalités de l’application*

IV- *Backlog de structuration du projet*

V- *Ooutils et technologies par phase du backlog*

VI- *Mapping complet - pages et fonctionnalités associées*

VII- *Modèle de base de données complet*

VIII- *Diagramme ERD (Entity-Relationship Diagram)*

I. Arborescence globale

```
/ifri_comotorage_project/          # Racine du projet Git
│
├── backend/                       # Partie Django Backend
│   ├── manage.py
│   ├── requirements.txt
│   ├── docker-compose.yml         # (optionnel pour setup local/prod)
│   ├── .env                       # (variables d'environnement sécurisées)
│   ├── core/                      # Projet Django
│   │   ├── __init__.py
│   │   ├── settings.py
│   │   ├── urls.py
│   │   ├── asgi.py                # Channels (WebSocket)
│   │   ├── wsgi.py
│   │   ├── routing.py             # Channels routing
│   │   ├── celery.py              # (optionnel, pour background tasks si besoin)
│   ├── apps/                      # Apps Django organisées
│   │   ├── accounts/              # Gestion des utilisateurs + Auth + Profil
│   │   ├── kyc/                   # Gestion KYC
│   │   ├── rides/                 # Offres / Demandes / Matching
│   │   ├── messaging/             # Messagerie instantanée (Channels + Redis)
│   │   ├── tracking/              # Tracking GPS
│   │   ├── notifications/         # Notifications Push (FCM)
│   │   ├── billing/               # Factures / Historique
│   │   ├── support/               # Chatbot + Aide
│   │   ├── common/                # Utils génériques, helpers
│   ├── media/                     # Uploads utilisateurs (photo profil, KYC, factures PDF)
│   ├── static/                    # Fichiers statiques si besoin
│   ├── logs/                      # Logs serveur
```

```
│
├── frontend/                  # Partie React Frontend
│   ├── package.json
│   ├── vite.config.js           # Si Vite est utilisé pour React (très rapide)
│   ├── public/                 # Public static files
│   ├── src/                   # Code source React
│   │   ├── index.js
│   │   ├── App.js
│   │   ├── api/                # Appels API vers backend DRF
│   │   ├── assets/              # Images, logos
│   │   ├── components/           # Composants React réutilisables (Button, Card, Modal,
Chat, etc.)
│   │   ├── context/             # Context global (ex : UserContext, ThemeContext)
│   │   ├── hooks/               # Custom React hooks (ex: useTracking, useWebSocketChat)
│   │   ├── pages/               # Pages principales (Accueil, Connexion, Profil, Recherche
Covoiturage, etc.)
│   │   │   ├── Home.jsx
│   │   │   ├── Login.jsx
│   │   │   ├── Register.jsx
│   │   │   ├── Profile.jsx
│   │   │   ├── KYC.jsx
│   │   │   ├── PublishOffer.jsx
│   │   │   ├── PublishRequest.jsx
│   │   │   ├── SearchRides.jsx
│   │   │   ├── RideDetail.jsx
│   │   │   ├── Chat.jsx
│   │   │   ├── Tracking.jsx
│   │   │   ├── Billing.jsx
│   │   │   ├── Support.jsx
│   │   │   ├── Notifications.jsx
│   │   │   ├── Settings.jsx
```

```
│   │   │   ├── FAQ.jsx
│   │   ├── router/              # Gestion du routing (ex : React Router)
│   │   ├── services/            # Services externes (Push Notifications, Chatbot API)
│   │   ├── styles/              # Styles globaux (ex : Tailwind config)
│
├── docs/                        # Documentation projet
│   ├── architecture.md
│   ├── api-spec.md
│   ├── readme.md
│
├── .gitignore                   # Fichiers ignorés par Git
└── README.md                    # Readme général du projet
```

II. Outils recommandés et/ou à installer…

- **Backend**
  - Django
  - Django REST Framework
  - Django Channels
  - Redis
  - Celery (optionnel pour background tasks type génération factures)
- **Frontend**
  - React.js
  - React Router DOM
  - Tailwind CSS
  - Axios (pour les appels API)
  - MapLibre GL JS (carte + globe)
  - Firebase FCM client lib
  - React Context (gestion global state)

III. Fonctionnalités

N°	Fonctionnalité	Description	Modules/Django/Technos recommandés
1	Inscription utilisateur	Inscription avec nom, prénom, email, téléphone, mot de passe + choix rôle conducteur/passager	Django User Model personnalisé
2	Connexion utilisateur	Connexion email/téléphone + mot de passe / réinitialisation mot de passe	Django auth
3	Profil utilisateur complet	Nom, prénom, photo, point de départ et d'arrivée habituel (avec affichage de son itinéraire sur carte après clic sur cette information), horaires de départ et d'arrivée habituels , mode de déplacement(si conducteur(si véhicule avoir les informations : marque, modèle, nombre de places disponibles))+ bouton discuter ou demander covoiturage Badge montrant si l'utilisateur a passé ou non la vérification KYC	Django Model, React form

N°	Fonctionnalité	Description	Modules/Django/Technos recommandés
4	Modification profil	L'utilisateur peut changer son rôle, sa photo et les autres informations sur le profil à tout moment, changer son nom et prénom tous les 30jrs..	Paramètres du profil
5	Page conducteurs du jour / passagers du jour	Liste filtrable : mode, départ, arrivée, heure, prix → affichage sur carte interactive	React + Django + Mapbox GL
6	Publication d'offre conducteur	Publication trajet avec départ, arrivée, heure, places, description libre, prix/fixe/gratuit	Django Model + React form
7	Publication de demande passager	Demande trajet avec départ, arrivée, heure, prix/fixe/gratuit	Django Model + React form
8	Double recherche (conducteur ↔ passager)	Conducteurs et passagers peuvent rechercher selon le même principe(mode, départ, arrivée, heure, prix)	Workflow mutualisé Django + React
9	Filtrage avancé des offres/demandes	Filtrage par mode, départ, arrivée, heure, prix	React + Django Filters
10	Algorithme de matching	Matching sur proximité(points de départ et d'arrivée) + horaire + préférences + filtres avancés	Django service, Algorithme optimisé
11	Affichage matching	Résultats clairs en liste + carte avec pastilles interactives (globe animée au chargement)	Mapbox GL, React, Animation UX
12	Clic pastille → Profil annonce/demande	Au clic sur pastille si annonce : vue détaillée de l'annonce + bouton discuter ou demander covoiturage + possibilité d'accéder au profil de base Au clic sur pastille si demande : vue détaillée sur le profil usager	React modal + page profil
13	Messagerie instantanée	Chat en temps réel conducteur ↔ passager avec notifications	Django Channels + WebSocket + React
14	Historique conversations	Liste des conversations passées + accès à l'historique	Django DB + React
15	Gestion des notifications push / UI	Badge sur app, push mobile, alertes interactives	Firebase FCM + React + Web API
16	Notification en temps réel	Demandes, réponses, nouveaux messages → push notification	Django Channels + Firebase FCM

N°	Fonctionnalité	Description	Modules/Django/Technos recommandés
17	Validation de la demande de covoiturage	Chaque partie peut valider ou non sans ou après chat	Workflow logique Django + React
18	Tracking GPS en temps réel	Après acceptation des 2 : tracking visible sur app avec consentement par partie	Django Channels + Mapbox + React
19	Gestion consentement tracking	L'utilisateur choisit de partager ou non sa position	React UX + Consent flow + DB flag
20	Tracking visible en temps réel sur carte	Tracking natif sur app / carte / point par point	Django Channels + React + Mapbox
21	Bouton d'urgence	Bouton accessible durant tracking actif → appel ou notif d'urgence et via le menu principal lorsque le covoiturage est en "mode actif"	UI + React + Web API
22	Points de rendez-vous intelligents	Suggestion points de RDV optimisés selon profils et trafic	Algorithme suggestion + Mapbox API
23	Estimation dynamique du prix	Conducteur choisit : prix fixe, estimation dynamique, gratuit	Form logique + Django Service
24	Facture automatisée après trajet	Génération automatique de facture envoyée aux deux parties via un bouton de demande si le tracking est arrivée au point d'arrivée. Une page historique des factures disponibles via les paramètres.	Django background task + PDF lib
25	Vérification identité KYC	Upload pièce identité + validation admin. La vérification se fera depuis les paramètres, des qu'elle est approuvée l'utilisateur le saura par le badge sur son profil et la confirmation présent dans les paramètres.	Django Model + React upload + backoffice admin
26	Tests de fiabilité conducteur	Badge fiabilité basé sur historique, feedback, nombre trajets à faire apparaitre sur le profil	Django Service + front UX
27	Thème sombre / clair	Option thème clair/sombre dans paramètres	React + CSS vars + Tailwind
28	Déconnexion	L'utilisateur peut se déconnecter facilement via les paramètres	Django auth + React menu
29	Support client intégré (Chatbot)	Chatbot intégré avec base FAQ + IA / bouton discuter support	Botpress / Rasa / GPT wrapper

IV. Backlog Projet IFRI_Comotorage

Un backlog est un document établi pour avoir :

- une vision claire des tâches à faire
- une priorité logique pour le dev
- une liste utilisable pour le suivi agile (scrum/kanban)

Phase 1 : Authentification & gestion comptes (Sprint 1)

Objectif : permettre à un utilisateur de s'inscrire, se connecter et gérer son compte.

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
1.1	Initialiser projet Django + React + MySQL	Setup de l'architecture complète du projet.	-	Django REST + React	Dev Backend / Frontend
1.2	Créer modèle User personnalisé	Ajouter rôle, KYC status, consentement tracking dans User model.	-	Django	Dev Backend
1.3	Inscription utilisateur (rôle + KYC optionnel)	Formulaire d'inscription + API avec validation téléphone/email unique.	Page Inscription	React + Django REST	Dev Backend / Frontend
1.4	Connexion utilisateur	Formulaire de login + JWT / Session sécurisée.	Page Connexion	React + Django REST	Dev Backend / Frontend
1.5	Réinitialisation mot de passe	Flow complet de reset via email ou SMS.	Page Réinitialisation MDP	React + Django REST	Dev Backend / Frontend
1.6	Déconnexion utilisateur	Bouton logout + invalidation du token.	Menu Profil	React + Django REST	Dev Frontend

Phase 2 : Profil utilisateur (Sprint 1-2)

Objectif : permettre à l'utilisateur de configurer son profil et ses préférences.

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
2.1	Page Profil utilisateur complete	CRUD complet du profil : infos perso, photo, rôle, badge, notation, bouton message et demander covoiturage…	Page Mon Profil	React + Django REST	Dev Backend / Frontend
2.2	Modification profil utilisateur	Bouton d'édition du profil : Ajout/modification photo de profil + nom modifiable 30j + heure + départ-arrivée	Page Paramètres	React + Django REST	Dev Backend / Frontend
2.3	Changement de rôle (conducteur/passager)	Switch dynamique avec règles de validation.	Page Paramètres	React + Django REST	Dev Backend / Frontend
2.4	Paramétrage point de départ habituel	Form avec auto-completion / map.	Page Mon Profil (Edition)	React + Django REST	Dev Backend / Frontend
2.5	Paramétrage horaires habituels	Sélecteur intelligent d'horaires récurrents.	Page Mon Profil (Edition)	React + Django REST	Dev Backend / Frontend
2.6	Gestion infos véhicule (conducteur)	CRUD infos véhicule.	Page Mon Profil (Edition)	React + Django REST	Dev Backend / Frontend
2.7	Consentement tracking par défaut	Stockage consentement + toggle dans profil.	Page Paramètres	React + Django REST	Dev Backend / Frontend

Phase 3 : Offre / Demande de covoiturage (Sprint 2)

Objectif : permettre de publier et rechercher des trajets.

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
3.1	Publication offre conducteur	CRUD offre avec description personnalisée.	Page Publier Offre	React + Django REST	Dev Backend / Frontend

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
3.2	Publication demande passager	CRUD demande passager.	Page Publier Offre	React + Django REST	Dev Backend / Frontend
3.3	Filtrage avancé	Moteur de filtrage multi-critères.	Page Recherche Covoiturage	React + Django REST	Dev Backend / Frontend
3.4	Page Conducteurs/Passagers du jour + Carte	Affichage globe annimé + pastilles.	Page Recherche Covoiturage + carte	React + Mapbox GL JS	Dev Frontend + Designer
3.5	Clic pastille → Profil détaillé	Vue détaillée avec bouton "Discuter" / "Demander covoiturage".	Page Recherche Covoiturage + Carte	React + Mapbox GL JS	Dev Frontend

Phase 4 : Matching & Validation Covoiturage (Sprint 2-3)

Objectif : gérer tout le workflow de validation d'un trajet.

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
4.1	Algorithme matching intelligent (lieu + horaires + préférences)	Proximité géographique + horaire + filtres + scoring.	Backend service	Django Service	Dev Backend
4.2	Résultats matching (liste + carte)	UI responsive + UX.	Page Recherche Covoiturage	React + Mapbox GL	Dev Frontend
4.3	Clic pastille → Profil / Annonce détaillée	Vue détaillée avec bouton "Discuter" / "Demander covoiturage".	Page Profil Conducteur / Passager	React	De Frontend
4.4	Validation demande covoiturage	Process bidirectionnel avec notifications.	Page Profil Conducteur / Passager	React + Django REST	Dev Backend / Frontend
4.5	Gestion statut demande (En attente, Acceptée, Refusée)	Modèle état + UI correspondante.	Page Mes demandes & Trajets	React + Django REST	Dev Backend / Frontend

Phase 5 : Messagerie instantanée (Sprint 3)

Objectif : permettre la communication temps réel.

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
5.1	Messagerie instantanée avec WebSocket	Implémentation avec Django Channels + React.	Page Chat	Django Channels + React	Dev Backend / Frontend
5.2	Historique conversation	Stockage en base, affichage dans l'UI.	Page Chat	Django REST + React	Dev Backend / Frontend
5.3	Notifications temps réel nouveaux messages	Intégration FCM / Websocket client.	Global App	Firebase FCM + WebSocket	Dev Frontend

Phase 6 : Tracking GPS en temps réel (Sprint 3-4)

Objectif : permettre un suivi GPS natif et respectueux du consentement.

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
6.1	Gestion consentement tracking	UI et flow consentement (unilatéral possible).	Page Confirmation Covoiturage	React + Django REST	Dev Backend / Frontend
6.2	Tracking temps réel	Implémentation WebSocket + Mapbox GL.	Page Suivi Trajet	Django Channels + Mapbox GL	Dev Backend / Frontend
6.3	Affichage tracking unilatéral / bilatéral	UI dynamique selon consentement.	Page Suivi Trajet	React + Mapbox GL	Dev Frontend

Phase 7 : Notifications & Sécurité (Sprint 3-4)

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
7.1	Notifications push globales : demandes, réponses, validation	Demandes, acceptation, refus, messages, tracking.	Global App	Firebase FCM	Dev Backend / Frontend
7.2	Bouton d'urgence pendant le trajet	Intégration UI + API système.	Page Suivi Trajet + Menu Global	React + API système	Dev Frontend

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
7.3	Vérification identité (KYC)	Upload doc + admin validation.	Page Vérification Identité	React + Django REST + Django Admin	Dev Backend / Frontend

Phase 8 : Factures & Fiabilité (Sprint 4)

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
8.1	Génération facture PDF	Backend auto PDF génération.	Backend service	Django REST + PDF Lib	Dev Backend
8.2	Page Historique factures	CRUD factures, consultation utilisateur.	Page Historique Factures	React + Django REST	Dev Frontend
8.3	Estimation prix dynamique/gratuit/fixe	Service Django + UI correspondante.	Page Publier Offre	Django Service + React	Dev Backend / Frontend
8.4	Badge de fiabilité conducteur (basé sur stats / avis / KYC)	Calcul automatique + UI correspondante.	Profil Conducteur	React + Django REST	Dev Backend / Frontend

Phase 9 : UX/UI & Support (Sprint 4)

N°	Tâche	Description détaillée	Page associée	Outils/Frameworks	Responsable
9.1	Chatbot support client	Intégration Botpress / Rasa / GPT wrapper.	Page Aide & Support	Botpress / Rasa / GPT Wrapper	Dev Backend / Frontend
9.2	Thème sombre / clair	Toggle dynamique + persistance.	Page Paramètres	React + Tailwind CSS	Dev Frontend

V. Outils et technologies par phase du backlog

Phase 1 : Authentification & Gestion de comptes

Côté	Outils / Technos recommandés
Backend	Django (framework Python)
API	Django REST Framework (DRF)
DB	MySQL
Authentification	Django Auth + JWT (djangorestframework-simplejwt)
Email (reset MDP)	SMTP gratuit (Gmail SMTP en test, Sendinblue/Sendgrid en prod limité gratuit)

Phase 2 : Profil utilisateur

Côté	Outils / Technos recommandés
Backend	Django + DRF
Frontend	React.js
CSS / UI	Tailwind CSS
Upload image	Django FileField + django-cleanup
Géolocalisation saisie adresse	OpenStreetMap + Nominatim API (gratuit)

Phase 3 : Offre / Demande de covoiturage

Côté	Outils / Technos recommandés
Backend	Django + DRF
Base géographique	PostGIS (extension spatiale de PostgreSQL) ou MySQL avec spatial support
Frontend	React + React Hooks + Tailwind CSS
Carte interactive + globe	Mapbox GL JS (gratuit jusqu'à 50k views/mois) ou MapLibre GL JS (fork 100 % open source de Mapbox GL) → je recommande MapLibre pour rester 100 % open source

Côté	Outils / Technos recommandés
Animation globe	MapLibre GL JS globe + CSS Anim
Filtrage avancé	React Select / React Hook Form

Phase 4 : Matching & Validation Covoiturage

Côté	Outils / Technos recommandés
Algorithme matching	Service Python custom dans Django (pas besoin de lib externe)
Backend API	Django + DRF
Notifications validation	WebSocket (via Django Channels) + Firebase FCM pour push

Phase 5 : Messagerie

Côté	Outils / Technos recommandés
WebSocket server	Django Channels + Redis (gratuit en local)
Frontend	React + WebSocket native API / Socket.io client
Stockage messages	Django Model
Notifications	Firebase FCM

Phase 6 : Tracking GPS

Côté	Outils / Technos recommandés
Géolocalisation utilisateur	JS API native navigator.geolocation
Backend temps réel	Django Channels + Redis
Carte suivi	MapLibre GL JS (Mapbox GL open source fork)
Tracking bilateral / unilatéral	Logique custom Django + Front React

Phase 7 : Notifications & Sécurité

Côté	**Outils / Technos recommandés**
Push Notifications	Firebase Cloud Messaging (FCM)
Bouton d'urgence	Intégration native (call tel:// URI + email / SMS API)
KYC	Django FileField upload + backoffice Django Admin pour validation

Phase 8 : Factures & Fiabilité

Côté	**Outils / Technos recommandés**
Génération PDF	WeasyPrint (Python lib open source), ou ReportLab
Stockage factures	Django Model + Media files
Badge fiabilité	Algorithme custom Django + calcul basé sur historique DB

Phase 9 : UX/UI & Support

Côté	Outils / Technos recommandés
Chatbot	Botpress CE (Community Edition open source) ou Rasa X CE
Thème sombre/clair	React + Tailwind CSS darkMode class

Résumé visuel des phases

Sprint	Objectifs clés
Sprint 1 (Jours 1-2)	Authentification + Profil
Sprint 2 (Jours 2-3)	Offre / Demande + Matching
Sprint 3 (Jours 4-5)	Messagerie + Tracking + Notifications

Sprint	Objectifs clés
Sprint 4 (Jours 6-7)	Facture + Fiabilité + Chatbot + Finitions UX/UI

VI. Mapping complet - Pages et fonctionnalités associées

Pages liées au compte utilisateur

Page	Fonctionnalités associées
Page Inscription	Inscription + Choix du rôle conducteur/passager + KYC initial
Page Connexion	Connexion utilisateur
Page Réinitialisation Mot de Passe	Flow complet de reset mot de passe
Page Profil utilisateur	Affichage du profil complet
Page Profil utilisateur (Edition)	Edition des infos personnelles, photo de profil, pseudo, point de départ habituel, horaires, infos véhicule (accès via la Page Paramètres)
Page Paramètres	Changement de rôle, Consentement tracking, Thème sombre/clair, Paramètres généraux…
Page Vérification Identité	Upload KYC, suivi du statut KYC (via Page Paramètres)

Pages liées au covoiturage

Page	Fonctionnalités associées
Page Publier Offre	Création offre conducteur + Description + Choix de type de paiement
Page Publier Demande	Création demande passager

Page	Fonctionnalités associées
Page Recherche Covoiturage	Recherche avancée + Filtrage multi-critères + Animation Globe + Carte interactive
Page Carte Interactive	Carte Mapbox GL interactive + Clic pastille profil/annonce
Page Profil Conducteur	Affichage annonce conducteur complet (infos véhicule, description, type de paiement) + bouton "Discuter" + "Demander covoiturage"
Page Profil Passager	Affichage demande passager + bouton "Discuter" + "Proposer covoiturage"
Page Mes Demandes & Trajets	Suivi des demandes envoyées, reçues, état des demandes (en attente, acceptée, refusée), historique des trajets passés

Pages liées à la messagerie

Page	Fonctionnalités associées
Page Messagerie (Liste conversations)	Historique des conversations passées et en cours
Page Chat Conversation	Conversation en temps réel WebSocket + Notifications

Pages liées au suivi GPS

Page	Fonctionnalités associées
Page Suivi Trajet (Tracking GPS en temps réel)	Tracking GPS natif (Mapbox GL), suivi trajet, bouton d’urgence, infos trajets en cours

Pages liées à la facturation et fiabilité

Page	Fonctionnalités associées
Page Historique Factures	Liste factures générées après trajets passés + consultation PDF
Page Badge Fiabilité (visible sur profil conducteur)	Visualisation badge fiabilité (basé sur KYC + nombre de trajets + feedback) → intégré au Profil Conducteur

Pages de support

Page	Fonctionnalités associées
Page Aide & Support	Chatbot intégré (Botpress / Rasa / GPT wrapper), FAQ, contact support humain possible

Pages globales et système

Page	Fonctionnalités associées
Page d'accueil / Dashboard	Dashboard avec résumé de l'activité, propositions rapides (Publier offre, rechercher trajet, factures récentes...)
Page À propos / Légal	Mentions légales, CGU, politique de confidentialité
Page Notifications	Centre de notifications (demandes reçues, acceptation/refus, nouveaux messages, factures générées, tracking démarré, etc.)

Total pages principales : ~22 pages

VII. Modèle de base de données complet

Table : users (personnalisation du User)

But : stocker les informations de base des utilisateurs.

Champ	**Type**	**Description**
id	PK (auto)	Clé primaire
email	varchar(255), unique	Email utilisateur
phone_number	varchar(20), unique	Téléphone utilisateur
password	hashed	Mot de passe hashé
first_name	varchar(100)	Prénom
last_name	varchar(100)	Nom

Champ	Type	Description
username	varchar(150), unique	Pseudo (modifiable 1x tous les 60j)
photo_profile	file path	Photo de profil
role	ENUM (PASSENGER, DRIVER)	Rôle actif de l'utilisateur
default_start_point	varchar(255)	Point de départ habituel
default_end_point	varchar(255)	Point d'arrivée habituel
default_start_time	time	Heure de départ habituelle
default_end_time	time	Heure d'arrivée habituelle
consent_tracking_default	boolean	Consentement par défaut au tracking
is_kyc_validated	boolean	KYC validé ou pas
is_active	boolean	Actif / Désactivé
date_joined	datetime	Date d'inscription
last_modified_username	datetime	Dernière modif pseudo

Table : kyc_documents

But : stocker les documents de KYC.

Champ	Type	Description
id	PK	
user_id	FK → users	Lien utilisateur
document_file	file path	Chemin du document
status	ENUM (PENDING, APPROVED, REJECTED)	Statut de vérification
created_at	datetime	Date upload

Champ	Type	Description
validated_at	datetime	Date validation admin

Table : vehicles (infos véhicule conducteur)

But : stocker les infos des véhicules des conducteurs.

Champ	Type	Description
id	PK	
user_id	FK → users	Conducteur concerné
brand	varchar(100)	Marque
model	varchar(100)	Modèle
license_plate	varchar(50)	Plaque immatriculation
seats_available	int	Nb de places dispo
created_at	datetime	

Table : ride_offers (offres des conducteurs)

But : stocker les offres de covoiturage.

Champ	Type	Description
id	PK	
driver_id	FK → users	Conducteur proposant l'offre
start_point	varchar(255)	Lieu de départ
end_point	varchar(255)	Lieu d'arrivée
start_time	datetime	Date + heure de départ
seats_available	int	Places dispo
description	text	Description personnalisée
price_type	ENUM (FREE, FIXED, ESTIMATED)	Type de paiement

Champ	Type	Description
fixed_price	decimal(10,2), nullable	Prix fixe le cas échéant
created_at	datetime	
status	ENUM (ACTIVE, CANCELLED, COMPLETED)	Statut de l'offre

Table : ride_requests (demandes des passagers)

But : stocker les demandes de covoiturage.

Champ	Type	Description
id	PK	
passenger_id	FK → users	Passager demandeur
start_point	varchar(255)	Lieu de départ
end_point	varchar(255)	Lieu d'arrivée
start_time	datetime	Date + heure souhaitée
price_preference	ENUM (FREE, FIXED, ESTIMATED)	Préférence paiement
max_price	decimal(10,2), nullable	Budget max (optionnel)
created_at	datetime	
status	ENUM (ACTIVE, CANCELLED, COMPLETED)	Statut de la demande

Table : ride_matches (validation de covoiturage)

But : stocker les correspondances validées.

Champ	Type	Description
id	PK	
ride_offer_id	FK → ride_offers	L'offre concernée
ride_request_id	FK → ride_requests	La demande concernée
driver_accepted	boolean	Conducteur a accepté

Champ	Type	Description
passenger_accepted	boolean	Passager a accepté
tracking_driver_consent	boolean	Conducteur consent au tracking
tracking_passenger_consent	boolean	Passager consent au tracking
status	ENUM (PENDING, CONFIRMED, REFUSED, CANCELLED, COMPLETED)	Statut global
created_at	datetime	

Table : tracking_positions

But : stocker les positions GPS des utilisateurs en tracking.

Champ	Type	Description
id	PK	
match_id	FK → ride_matches	Le covoiturage en cours
user_id	FK → users	Utilisateur (conducteur ou passager)
latitude	float	Latitude
longitude	float	Longitude
timestamp	datetime	Date de la position

Table : chat_messages

But : stocker les messages de la messagerie.

Champ	Type	Description
id	PK	
sender_id	FK → users	Auteur du message
receiver_id	FK → users	Destinataire
match_id	FK → ride_matches, nullable	Lien vers covoiturage (optionnel)

Champ	Type	Description
content	text	Message texte
created_at	datetime	Date d’envoi

Table : notifications

But : stocker l’historique des notifications push.

Champ	Type	Description
id	PK	
user_id	FK → users	Utilisateur destinataire
type	ENUM (MATCH_REQUEST, MATCH_CONFIRMED, CHAT_MESSAGE, TRACKING_STARTED, RIDE_COMPLETED, GENERAL)	Type de notif
title	varchar(255)	Titre notif
body	text	Message notif
created_at	datetime	Date envoi

Table : invoices (factures)

But : stocker les factures générées.

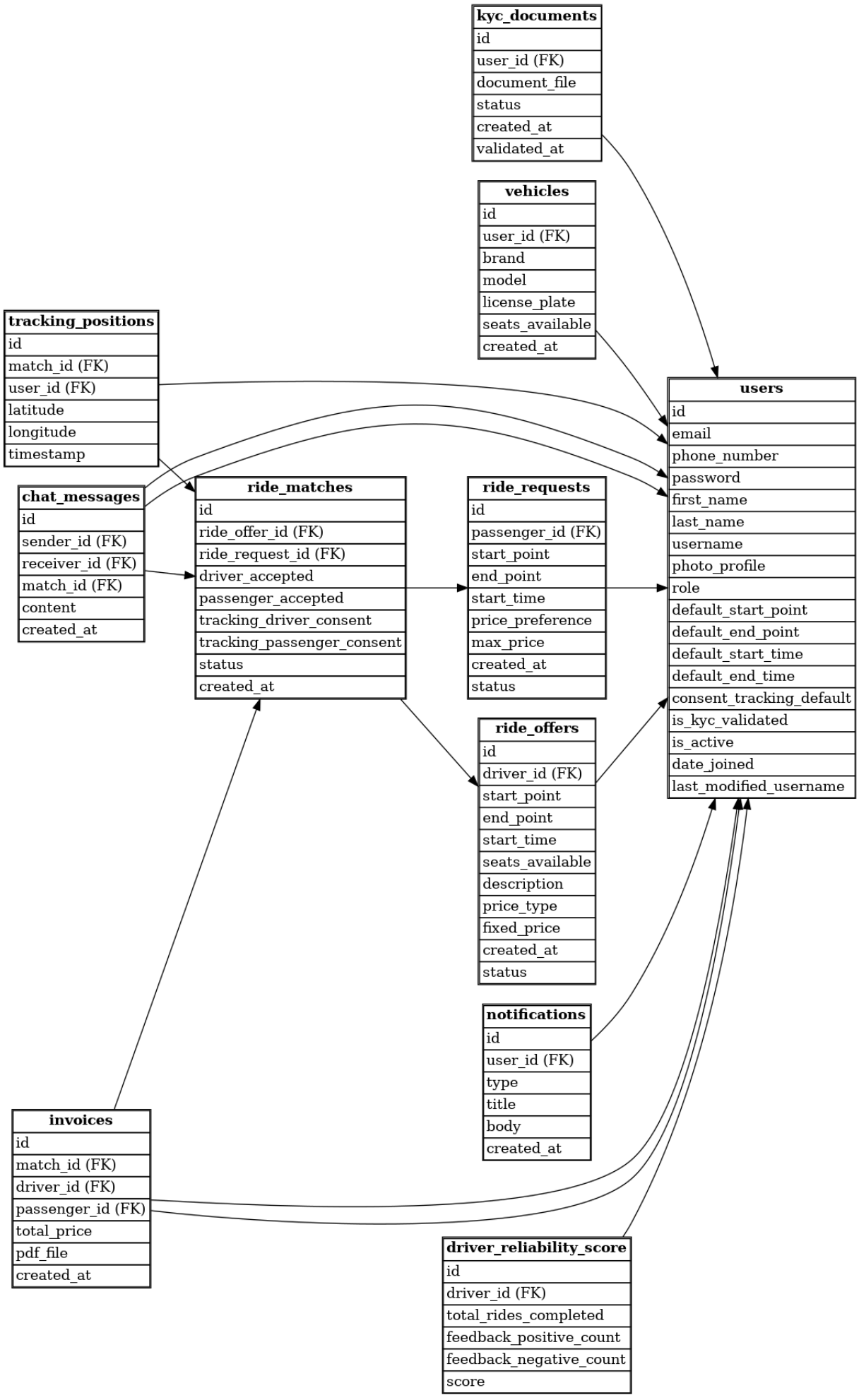
Champ	Type	Description
id	PK	
match_id	FK → ride_matches	Trajet concerné
driver_id	FK → users	Conducteur
passenger_id	FK → users	Passager
total_price	decimal(10,2)	Prix payé (si applicable)
pdf_file	file path	Fichier PDF facture
created_at	datetime	Date génération

Table : driver_reliability_score

But : stocker les scores de fiabilité des conducteurs.

Champ	Type	Description
id	PK	
driver_id	FK → users	Conducteur
total_rides_completed	int	Nb trajets complétés
feedback_positive_count	int	Nb retours positifs
feedback_negative_count	int	Nb retours négatifs
score	float	Score de fiabilité calculé

VIII. Diagramme ERD (Entity-Relationship Diagram)


kyc_documents
id
user_id (FK)
document_file
status
created_at
validated_at
vehicles
id
user_id (FK)
brand
model
license_plate
seats_available
created_at
tracking_positions
id
match_id (FK)
user_id (FK)
latitude
longitude
timestamp
users
id
email
phone_number
password
first_name
last_name
username
photo_profile
role
default_start_point
default_end_point
default_start_time
default_end_time
consent_tracking_default
is_kyc_validated
is_active
date_joined
last_modified_username
chat_messages
id
sender_id (FK)
receiver_id (FK)
match_id (FK)
content
created_at
ride_matches
id
ride_offer_id (FK)
ride_request_id (FK)
driver_accepted
passenger_accepted
tracking_driver_consent
tracking_passenger_consent
status
created_at
ride_requests
id
passenger_id (FK)
start_point
end_point
start_time
price_preference
max_price
created_at
status
ride_offers
id
driver_id (FK)
start_point
end_point
start_time
seats_available
description
price_type
fixed_price
created_at
status
notifications
id
user_id (FK)
type
title
body
created_at
invoices
id
match_id (FK)
driver_id (FK)
passenger_id (FK)
total_price
pdf_file
created_at
driver_reliability_score
id
driver_id (FK)
total_rides_completed
feedback_positive_count
feedback_negative_count
score