

# COVOITURAGE

---

MARAM BEN RHOUMA  
MARAM GHOUA  
REFKA MAHJOUR  
SAMAR BEN HOUDI  
EYA BEN AMEUR

# SOMMAIRE

---

**1. DESCRIPTION DU PRJET**

**2. NOTRE HISTOIRE**

**3. MEMBRES ET CONTRIBUTIONS**

**4. CHOIX DES TECHNOLOGIES**

**5. REPO GITHUB**

**6. VIDEO**

# DESCRIPTION DU PROJET

— Notre plateforme est une application web de mise en relation entre utilisateurs où chaque utilisateur peut à la fois jouer le rôle de conducteur (driver) ou passager (rider). Sans rôle fixe à l'inscription, un utilisateur peut s'inscrire, se connecter et :

- Consulter son profil avec l'historique de ses rides/drives, les avis reçus et donnés.
- Rechercher d'autres utilisateurs et les signaler en cas d'abus.
- Explorer les postes disponibles, ajouter des commentaires et envoyer une demande de participation.
- Accepter les demandes d'autres utilisateurs sur ses propres trajets.
- Clôturer un ride une fois terminé.
- Discuter en temps réel avec d'autres membres via un système de chat WebSocket.
- Recevoir des notifications instantanées (SSE) lors de demandes de participation, d'acceptations ou des messages .

# DESCRIPTION DU PROJET

---

**Un utilisateur avec le rôle "admin" a accès à un tableau de bord (dashboard) lui permettant de :**

- **Consulter les statistiques globales de la plateforme (utilisateurs, rides, etc.).**
  - **Accéder à la liste des reports (signalements) effectués sur des utilisateurs afin de modérer l'activité.**
- 

# MEMBRES ET CONTRIBUTIONS

## MARAM BEN RHOUMA

### Authentification, Rôles et Profils

Responsable de la gestion des utilisateurs

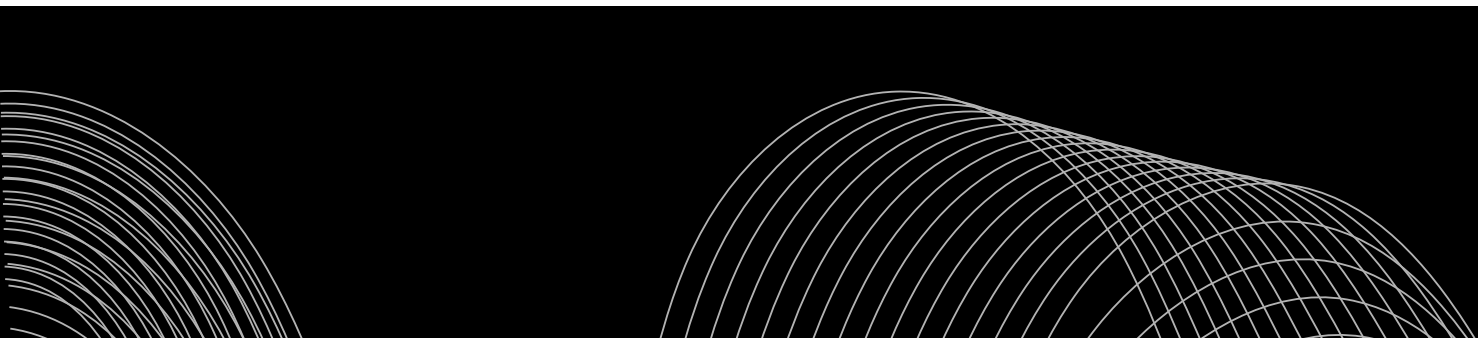
- Mise en place des APIs d'inscription, connexion et déconnexion
- Gestion des tokens JWT
- Accès basé sur les rôles (admin/utilisateur)
- CRUD sur les profils utilisateurs
- Middleware pour les vérifications de permissions

## REFKA MAHJOUB

### Messagerie et Notifications

Responsable de la logique temps réel

- Implémentation du système de chat (WebSocket)
- Stockage et récupération de l'historique des messages
- Système de notifications : messages, notif de demande, notif d'acceptation de demande, et signales
- Logique de livraison des notifications via SSE



# MEMBRES ET CONTRIBUTIONS

## EYA BEN AMEUR

### Signalements et Modération Admin

Responsable de la dashboard admin et la logique des signalements

- Endpoint de signalement : signaler un utilisateur avec description, fichier ou ID de ride.
- Modélisation : chaque signalement est lié à un utilisateur , avec preuves optionnelles et suivi de statut (“pending” ,”approved”, “declined”).
- Recherche intégrée : filtre par status et recherche possible sur la raison , le sujet ,...
- Dashboard Admin : différentes statistiques sur les données de notre site, un diagramme contenant les trajet par mois, les trajets récents et les utilisateurs récents

## MARAM GHOUMA

### Publications, Rides et Historique

Responsable de la logique des trajets

- Publications, Rides et Historique
- Implémentation de la logique des trajets
- Endpoint de création de posts
- Gestion de la logique de participation aux rides
- Système de commentaires sur les posts
- Fonctionnalités de join ride et end ride
- Suivi de l'historique des trajets :
  - Trajets pris ou créés
  - Rôle joué dans chaque trajet (conducteur/passager)
  -

# MEMBRES ET CONTRIBUTIONS

---

**SAMAR BEN HOUIDI**

**Avis, Calendrier et Recherche**

## **Avis**

- Soumission et consultation d'avis.
- Ajouter/supprimer avec vérification d'auteur.
- Séparer en : écrits / reçus.

Recherche d'avis par mot clé .

## **Calendrier**

- Endpoints pour trajets/événements.
- Vue calendrier pour les visualiser.

## **Recherche**

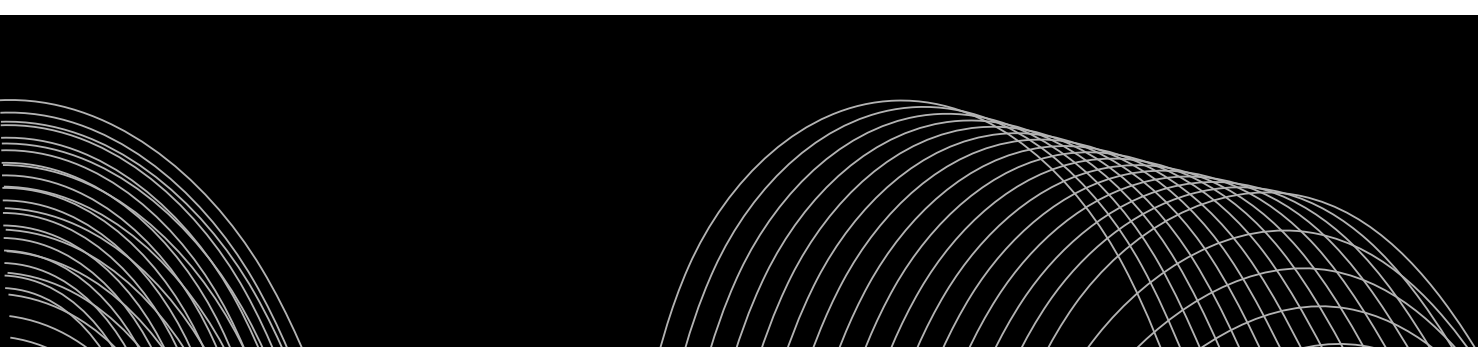
- Moteur de recherche : utilisateurs, posts, trajets, avis.

## **Utilisateurs**

- Page avec liste et liens vers profils + signalement.

## **Autres**

- Home page simple.
- Pagination pour les listes.



# CHOIX DES TECHNOLOGIES

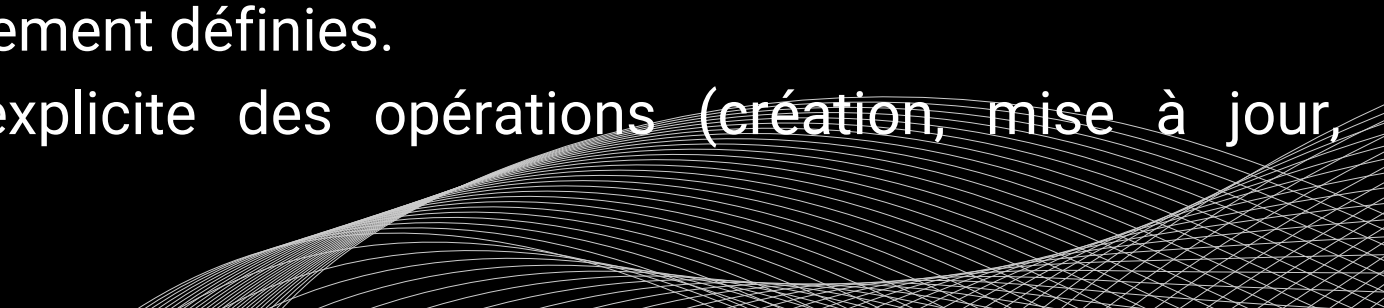
---

## REST

Utilisé pour :

- Création (ajout) d'un signalement
- Mise à jour (modification) d'un signalement
- Suppression d'un signalement

Gestion des Avis – Choix Technique

- Opérations CRUD via REST API :
    - Création des avis (endpoint POST /reviews)
    - Modification des avis (endpoint PUT/PATCH /reviews/{id})
    - Suppression des avis (endpoint DELETE /reviews/{id})
  - Justification :
    - Approche REST simple et efficace pour des actions unitaires et clairement définies.
    - Utilisation des méthodes HTTP standards pour une gestion explicite des opérations (création, mise à jour, suppression).
    - Implémentation plus directe
- 



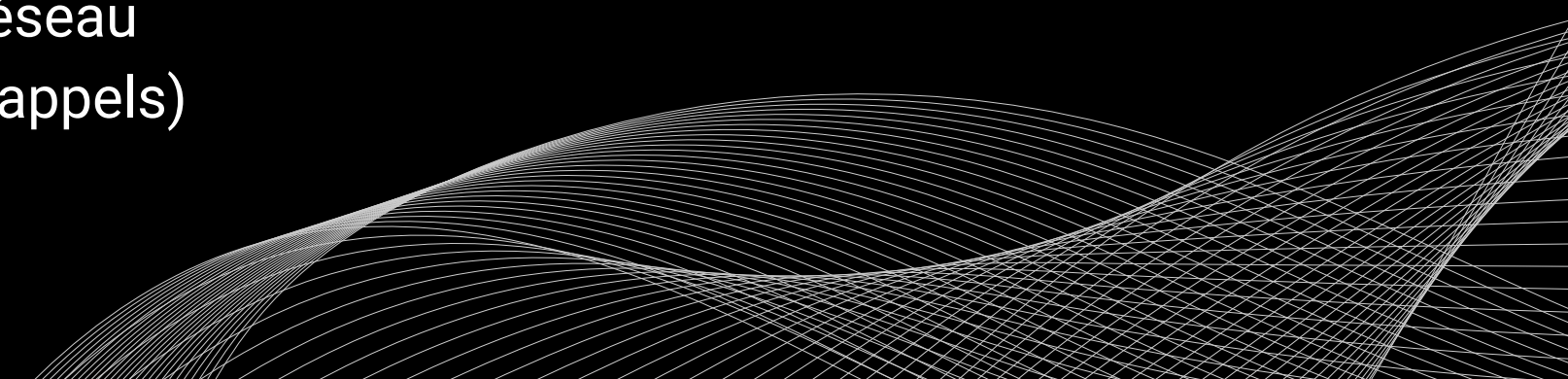
# CHOIX DES TECHNOLOGIES

---

## GRAPHQL

- Utilisé pour : profils utilisateurs, reviews, posts et rides
- Nous avons choisi GraphQL pour les parties du projet qui nécessitent la récupération de données riches, imbriquées et fortement liées, comme :
  - Le profil utilisateur (contenant ses avis, posts, trajets en tant que conducteur et passager, etc.)
  - Les détails d'un ride avec les participants et d'un post avec le créateur, les commentaires...
  - Les reviews, avec l'auteur, la cible, la note, le commentaire...
  - les données relatives à la dashboard admin
  - les reports (signalements) d'un user ou tous (pour l'admin)
  - la recherche et la pagination

### Avantages :

- Permet d'obtenir exactement les données nécessaires sans surcharge réseau
  - Réduit le nombre de requêtes (vs REST où il faut souvent faire plusieurs appels)
  - Idéal pour des interfaces dynamiques
- 

# CHOIX DES TECHNOLOGIES

— EVENT DRIVEN API

## Implémentation WebSocket

### 1. Utilisation de la Passerelle (Gateway)

Objectif : Gère la communication bidirectionnelle en temps réel entre conducteurs et passagers.

- Caractéristiques Clés :
  - Gère les connexions et déconnexions WebSocket
  - Organise des salons de discussion par trajet (chat\_{rideId})
  - Diffuse instantanément les messages à tous les participants d'un trajet

### 2. Émission et Abonnement aux Événements

- Émission :
  - Les messages sont envoyés à des salons spécifiques lorsqu'un utilisateur envoie un message
  - Les événements incluent le contenu du message, l'identifiant de l'expéditeur et des horodatages
- Abonnement :
  - Les clients (conducteurs/passagers) s'abonnent au salon de leur trajet lors de la connexion
  - Les mises à jour en temps réel sont envoyées sans interrogation (polling)

# Implémentation SSE (Server-Sent Events)

## 1. Utilisation de la Passerelle (Gateway)

**Objectif :** Livre des notifications en temps réel (mises à jour de trajet, messages, alertes).

**Caractéristiques Clés :**

- Utilise le streaming HTTP pour des mises à jour unidirectionnelles serveur-vers-client
- Se reconnecte automatiquement si le client perd le réseau

## 2. Émission et Abonnement aux Événements

**Émission :**

- Les notifications (ex: "Trajet accepté", "Nouveau message") sont émises comme événements déclenchés par le serveur
- Les événements sont structurés avec type, contenu et utilisateur cible

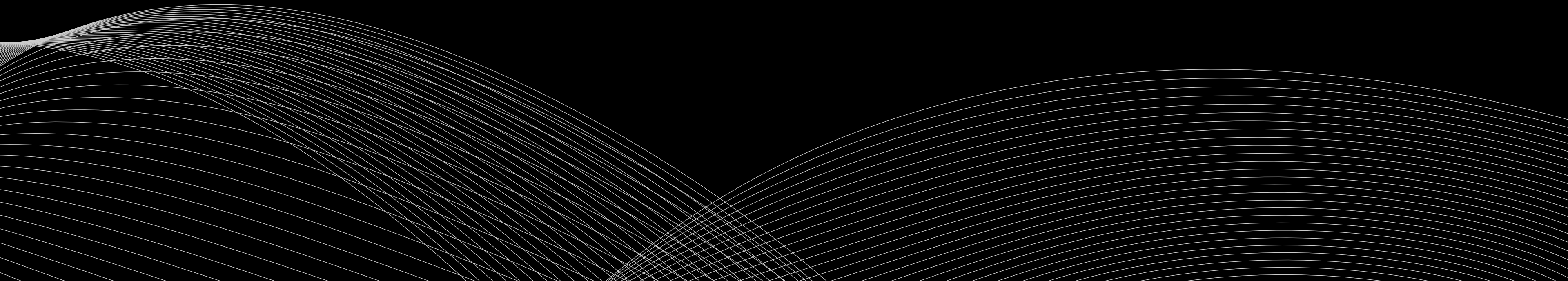
**Abonnement :**

- Les clients s'abonnent à leur flux de notifications personnel (/notifications/{userId})
- Aucun besoin d'interrogation - les événements sont envoyés dès qu'ils se produisent

# REPO GITHUB

---

[HTTPS://GITHUB.COM/MAHJOUBR/COVOITURAGE-PLATFORM](https://github.com/MAHJOUBR/COVOITURAGE-PLATFORM)



# VIDEO

---

**LIEN VERS LE DRIVE : PROFILE-RIDES AND POSTS-REVIEWS AND REPORTS**

**LIEN VERS LE DRIVE : NOTIFICATIONS AND CHAT**

**LIEN VERS LE DRIVE : DASHBOARD AND ALL REPORTS**



**MERCI**