

2025

# TRAFIGANDME : Documentation technique



Abdoul-Waric Konaté, Abdoul Fadel Blacou,  
Hamza Belyahiaoui, Jean-Philippe Delon  
Trafine

## SOMMAIRE

Présentation du projet .....	2
Stack technique .....	5
•     Frontend : React (Web) & React Native (Mobile) .....	6
•     Backend : Spring Boot .....	7
•     Base de données : PostgreSQL.....	7
•     Conteneurisation : Docker .....	8
•     Conclusion générale .....	8
Architecture de l'application.....	9
<b>Modèle relationnel – Base de données Traficandme.....</b>	<b>10</b>
<b>1. Users.....</b>	<b>10</b>
<b>2. Role.....</b>	<b>11</b>
<b>3. Signalement.....</b>	<b>11</b>
<b>4. Vote.....</b>	<b>12</b>
<b>5. Itinerary.....</b>	<b>12</b>
<b>6. Traffic.....</b>	<b>13</b>
<b>7. calendar_stats .....</b>	<b>14</b>
<b>Schéma des relations (récapitulatif) .....</b>	<b>14</b>
Description des composants .....	16
API REST .....	18
Déploiement .....	19
Sécurité .....	22
Authentification .....	25
Perspective d'évolution .....	28
Conclusion .....	29

# Présentation du projet

Traficandme est une application web et mobile de **navigation intelligente et participative**, conçue pour **améliorer l'expérience de conduite** en zones urbaines et périurbaines.

Grâce à son interface moderne et communautaire, l'application permet de :

- Visualiser le trafic en temps réel
- Signaler les incidents (accidents, bouchons, dangers)
- Calculer des itinéraires optimisés
- Estimer le coût des trajets
- Recevoir des alertes en fonction de la position

Traficandme repose sur une architecture **microservices** moderne, combinant :

- **React + Vite** pour le frontend web
- **React Native** pour le mobile
- **Spring Boot** pour le backend
- **PostgreSQL** pour la base de données
- **JWT** pour l'authentification
- **Docker** pour le déploiement
- **TomTom API** pour la cartographie et la navigation
- **GitHub** pour la gestion du code source et la collaboration

Compatible avec des intégrations tierces via une API REST, **Traficandme** vise aussi bien les particuliers que les collectivités ou entreprises souhaitant enrichir leurs services avec une solution de navigation collaborative.

Traficandme est le fruit d'un travail collaboratif entre plusieurs membres d'une équipe passionnée par l'innovation et la technologie. Chaque membre a apporté son expertise dans différentes parties du projet, de la conception à la mise en production.

- **Hamza Belyahiaoui** - Développeur Full Stack / Chef de Projet
- **Abdoul-waris Konate** - Développeur Back-End / Architecte d'application
- **Fadel Biaou** - Développeur Front-End / UX Designer / Rédacteur technique
- **Jean-Philippe Delon** - Développeur Mobile / Spécialiste en Intelligence Artificielle

Ce travail d'équipe a permis de créer une solution moderne et évolutive, répondant aux besoins des utilisateurs tout en assurant une sécurité et une performance de haut niveau.



# Stack technique

Voici ci-dessous un tableau récapitulatif des technologies utilisées pour développer notre SaaS :

Composant	Technologie utilisée	Description
Frontend Web	React + Vite	Pour une interface web moderne, rapide et réactive grâce à la compilation Vite.
Frontend Mobile	React Native	Développement multiplateforme natif pour Android et iOS avec un seul codebase.
Backend	Spring Boot (Java)	Framework robuste pour construire des API REST performantes et maintenables.
Base de données	PostgreSQL	SGBD relationnel fiable, puissant, open-source, adapté aux données structurées.
Authentification	JWT	Sécurisation des accès utilisateurs
Déploiement	Docker	Conteneurisation pour faciliter le déploiement, la scalabilité et la portabilité.
Cartographie/ GPS	TomTom SDK/API	Navigation GPS, affichage des cartes et gestion du trafic
Outil de gestion	Github	Hébergement du code source et collaboration en équipe

## Pourquoi ce choix de stack ?

Notre objectif était de construire une application **moderne, évolutive, performante et maintenable**.

- Le **couplage React / React Native** nous permet de mutualiser une partie du code tout en ciblant plusieurs plateformes (web et mobile).
  - **Spring Boot** a été choisi pour sa stabilité, sa scalabilité, et sa large adoption en entreprise.
  - **JWT** permet une authentification simple à mettre en œuvre dans un environnement décentralisé comme une architecture microservices.
  - **PostgreSQL** offre une grande fiabilité pour les données critiques (utilisateurs, signalements).
  - Enfin, **Docker** simplifie considérablement le déploiement local et en production, rendant notre solution facilement portable.
  - Notre objectif était de concevoir une solution moderne, scalable, maintenable et multi-plateforme. Le choix de stack s'est fait après avoir comparé plusieurs technologies, en tenant compte de leurs avantages respectifs, de leur communauté, et de leur adéquation avec les exigences du projet Traficandme.
- 
- **Frontend : React (Web) & React Native (Mobile)**

Critère	React / React Native	Alternatives (Angular, Flutter)	Justification
Réutilisation du code	Haute (logique et composants similaires)	Moyenne (Flutter), Faible (Angular)	Mutualisation partielle du code entre web et mobile.
Courbe d'apprentissage	Modérée	Élevée (Flutter + Dart), Modérée (Angular + TypeScript)	La majorité de l'équipe connaît déjà JavaScript.
Écosystème et communauté	Très large	Large aussi	Grande disponibilité de bibliothèques tierces et d'outils.
Performances mobiles	Natives (React Native)	Très bonnes (Flutter), Moyennes (Angular via Ionic)	React Native offre de vraies performances mobiles avec rendu natif.

Conclusion : Le couple React / React Native offre un bon compromis entre efficacité de développement, portabilité du code et performance.

- **Backend : Spring Boot**

Critère	Spring Boot	Alternatives (Node.js, Django, Express)	Justification
Robustesse & scalabilité	Très élevée	Moyenne (Django, Express)	Conçu pour des architectures microservices et des environnements critiques.
Support entreprise	Excellent	Limité (sauf Django en environnement Python)	Spring est très utilisé dans l'industrie, gage de longévité.
Sécurité intégrée	Native avec Spring Security	Nécessite des ajouts (Express, Node)	Facilite la mise en place d'authentification et d'autorisations.
Performances	Excellentes	Bonnes	Optimisé pour la haute performance dans des applications REST.

Conclusion : Spring Boot est un choix mature et fiable, adapté à notre besoin de construire une API robuste et sécurisée.

- **Base de données : PostgreSQL**

Critère	PostgreSQL	Alternatives (MySQL, MongoDB)	Justification
Fiabilité et intégrité	Très élevée	Élevée (MySQL), Faible (MongoDB)	Transactions ACID parfaites pour des données critiques.
Support des données géographiques	Oui (PostGIS)	Limité (MySQL), Natif (MongoDB)	Utile pour les fonctionnalités basées sur la géolocalisation.
Performances	Excellentes sur requêtes complexes	Bonnes (MySQL), Variables (MongoDB)	Meilleur support des jointures complexes.

Conclusion : PostgreSQL offre un excellent compromis entre performance, sécurité et puissance analytique, particulièrement pertinent pour notre gestion de signalements, d'itinéraires et de statistiques.

- **Conteneurisation : Docker**

Critère	Docker	Alternatives (VM, Podman, Vagrant)	Justification
Portabilité	Très élevée	Moyenne	Reproductibilité des environnements entre dev, staging et prod.
Rapidité de déploiement	Élevée	Faible (VM), Moyenne (Podman)	Lancement rapide des conteneurs.
Communauté et support	Très large	Moins répandue	Large documentation et compatibilité CI/CD.

Conclusion : Docker permet un déploiement simplifié, une meilleure gestion des dépendances et une mise en production rapide.

- **Conclusion générale**

- La stack React / React Native – Spring Boot – PostgreSQL – JWT – Docker s'est imposée comme la solution la plus adaptée à notre besoin de développement multi-plateforme, performant et maintenable. Elle nous permet de répondre aux exigences fonctionnelles et techniques du projet Traficandme, tout en assurant une évolutivité future.

# Architecture de l'application

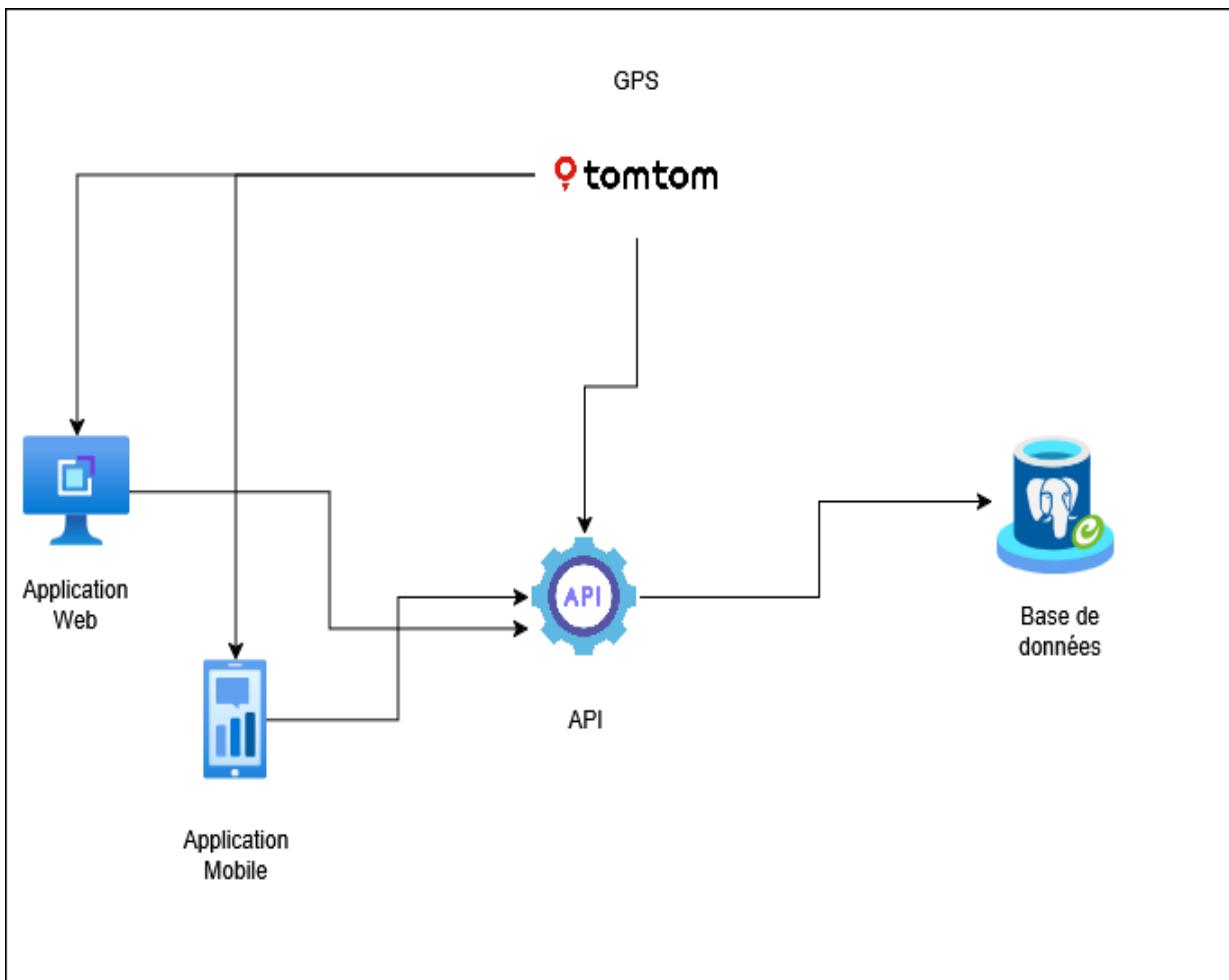
La structure de notre projet :

traficandme/

```
|—— api-traficandme/  
|—— web-traficandme/ # Application web pour utilisateurs et administrateurs  
|—— mobile-traficandme/ # Application mobile pour utilisateurs
```

- api-traficandme contient tous les services backend (authentification, signalements, itinéraires...)
- web-traficandme représente l'application web pour utilisateurs et l'interface administrateur
- mobile-traficandme est l'application mobile pour utilisateurs

Voici ci-dessous la structure en schéma de Traficandme :



## Modèle relationnel – Base de données Traficandme

Ce modèle de données constitue l'ossature de l'application Traficandme. Il repose sur une structure relationnelle conçue pour assurer la cohérence, la scalabilité et la traçabilité des données utilisateurs, signalements de trafic, itinéraires, statistiques et rôles d'accès.

### 1. Users

Rôle : Représente les comptes utilisateurs de l'application (visiteurs, modérateurs, administrateurs, etc.).

Champs principaux :

- id : Int — Identifiant unique.
- firstName : String — Prénom.
- lastName : String — Nom.
- Email : String — Adresse e-mail (unique).
- Password : String — Mot de passe (crypté).
- ProviderID : Int — Référence pour l'authentification OAuth2.
- status : Enum — Statut du compte (actif, désactivé, supprimé, en attente).
- create\_date : DateTime — Date de création du compte.
- update\_date : DateTime — Dernière modification du compte.

Relations :

- Un utilisateur appartient à un rôle (Role) → relation 1 - \*.
- Un utilisateur peut signaler des incidents (Signalement) → relation 1 - \*.
- Un utilisateur peut créer des itinéraires (Itinerary) → relation 1 - \*.

## 2. Role

Rôle : Décrit le rôle attribué à un utilisateur pour contrôler les droits d'accès.

Champs principaux :

- Code\_Role : String — Code interne (ex. “ADMIN”, “USER”).
- Role\_Name : String — Libellé lisible (ex. “Administrateur”).

Relations :

- Un rôle peut être attribué à plusieurs utilisateurs → relation 1 - \*.

## 3. Signalement

Rôle : Contient les informations liées aux événements signalés par les utilisateurs (ex. bouchons, accidents, dangers).

Champs principaux :

- id : Int — Identifiant unique du signalement.
- type : String — Type d'événement (accident, bouchon, etc.).

- longitude / latitude : String — Coordonnées géographiques.
- address : String — Adresse du lieu signalé.
- date\_crea : DateTime — Date de création du signalement.
- date\_upd : DateTime — Dernière mise à jour.
- status : Enum — Statut (nouveau, validé, rejeté).
- date\_signalement : Date — Date effective de l'événement.
- username : String — Nom d'utilisateur ayant effectué le signalement.
- update\_date : DateTime — Dernière modification.

Relations :

- Créé par un utilisateur → 1 - \* avec Users.
- Peut recevoir des votes → 1 - \* avec Vote.
- Associé à une date de calendrier → \* - 1 avec calendar\_stats.

## 4. Vote

Rôle : Permet aux utilisateurs d'interagir avec les signalements via des “likes” ou “dislikes”.

Champs principaux :

- id : Int — Identifiant.
- type : Enum — Type de vote (like/dislike).
- like\_count : Int(10) — Nombre de likes.
- dislike\_count : Int(10) — Nombre de dislikes.

Relations :

- Chaque vote est associé à un signalement → \* - 1 avec Signalement.
- Chaque vote est enregistré dans un contexte temporel → \* - 1 avec calendar\_stats.

## 5. Itinerary

Rôle : Gère les itinéraires générés ou enregistrés par les utilisateurs.

Champs principaux :

- id : Int — Identifiant unique.
- adress\_start / adress\_end : String — Adresses de départ et d'arrivée.
- start\_latitude / start\_longitude — Coordonnées de départ.
- end\_latitude / end\_longitude — Coordonnées d'arrivée.
- mode : Enum — Mode de transport (voiture, piéton, vélo...).
- status : Enum — État de l'itinéraire (en cours, terminé, historique...).
- date\_itinerary : Date — Date de création.
- create\_date / upd\_date : DateTime — Dates de suivi.
- username : String — Utilisateur associé.

Relations :

- Relié à un utilisateur → \* - 1 avec Users.
- Relié à la table de trafic pour associer les conditions de circulation → \* - 1 avec Traffic.
- Associé à une entrée temporelle → \* - 1 avec calendar\_stats.

## 6. Traffic

Rôle : Contient les données de circulation en temps réel issues d'une API externe (ex. TomTom).

Champs principaux :

- id : Int
- latitude / longitude : String — Position géographique.
- currentSpeed : Int(4) — Vitesse mesurée.
- freeFlowSpeed : Int(4) — Vitesse théorique en conditions normales.
- congested : String — Booléen (trafic fluide ou congestionné).
- source : String — Fournisseur de données (ex. TomTom).
- fetched\_at : DateTime — Date de collecte.

Relations :

- Lié à un ou plusieurs itinéraires → 1 - \* avec Itinerary.

## **7. calendar\_stats**

Rôle : Table de dimension temporelle utilisée pour les analyses statistiques.

Champs principaux :

- calendar\_date : Date — Date de référence.
- day\_name, day\_number
- week\_start\_date, week\_end\_date, week\_number, week\_name
- month\_start\_date, month\_end\_date, month\_number, month\_name
- quarter\_start\_date, quarter\_end\_date, quarter\_number, quarter\_name

Utilité :

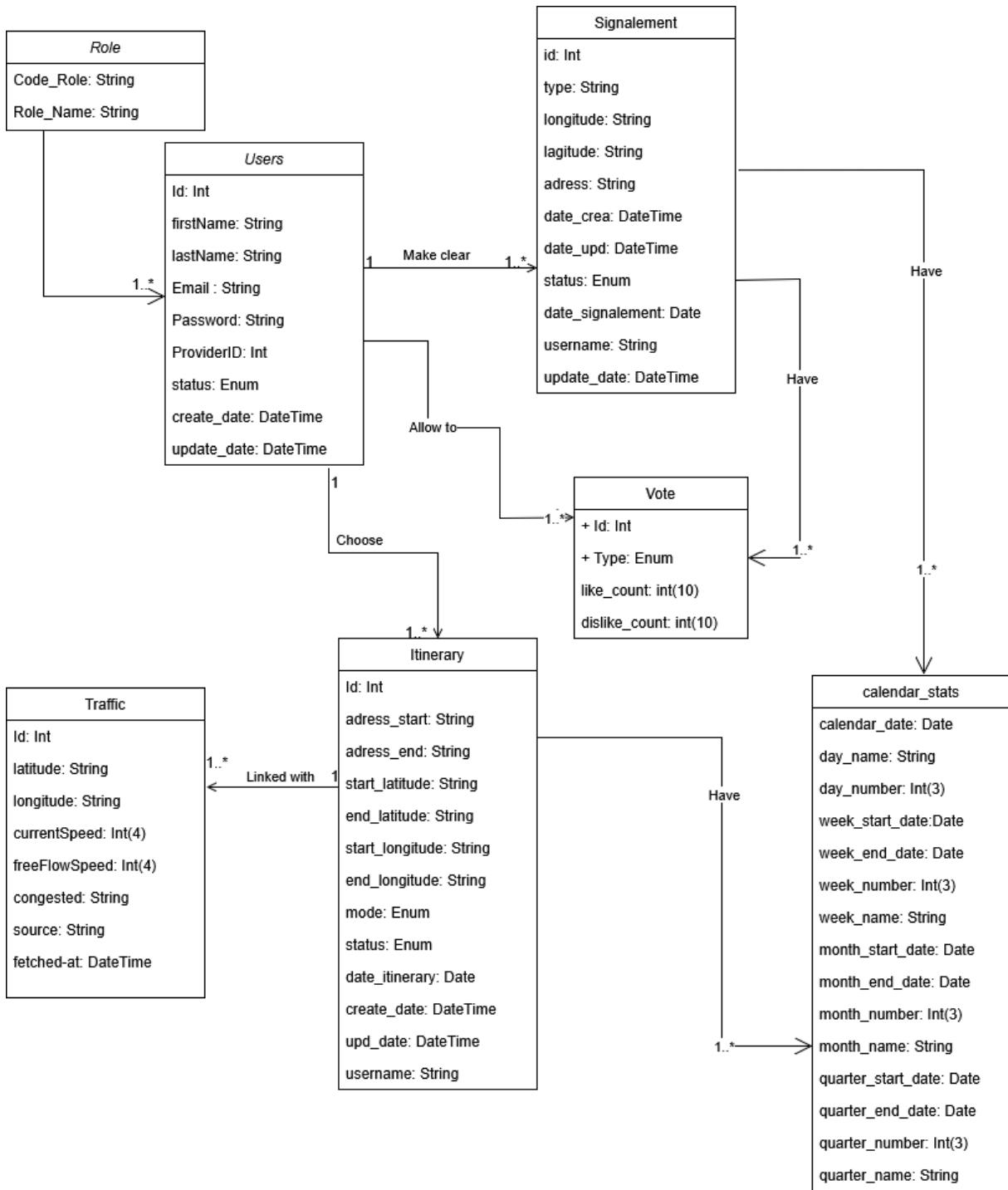
- Sert de référence temporelle commune pour les signalements, votes et itinéraires.
- Permet de réaliser des statistiques temporelles (ex. signalements par semaine, trajets par mois).

Relations :

- Utilisée dans :
  - Signalement
  - Vote
  - Itinerary

## **Schéma des relations (récapitulatif)**

- Users (1) — (N) Signalement, Itinerary
- Users (N) — (1) Role
- Signalement (1) — (N) Vote
- Signalement, Vote, Itinerary (N) — (1) calendar\_stats
- Itinerary (N) — (1) Traffic



# Description des composants

L'architecture microservice de **Traficandme** repose sur plusieurs composants clés, chacun jouant un rôle spécifique dans l'écosystème global. Cette modularité favorise l'évolutivité, la maintenabilité et la robustesse de la plateforme.

## ➤ Frontend Web

- **Technologie :** React + Vite
- **Rôle :** Fournir une interface web responsive pour les utilisateurs et les administrateurs.
- **Fonctionnalités principales :**
  - Connexion et inscription sécurisées
  - Consultation du trafic en temps réel
  - Signalement d'incidents
  - Visualisation des itinéraires optimisés
  - Tableau de bord administrateur (gestion des utilisateurs et des signalements)

## ➤ Frontend Mobile

- **Technologie :** React Native
- **Rôle :** Permettre une expérience mobile fluide et native sur Android et iOS.
- **Fonctionnalités principales :**
  - Navigation GPS avec affichage en temps réel
  - Signalement rapide d'événements via géolocalisation
  - Notifications push contextuelles
  - Sauvegarde des trajets favoris

## ➤ Backend

- **Technologie :** Spring Boot (Java)
- **Rôle :** Fournir l'ensemble des services métiers via des endpoints REST.
- **Services gérés :**
  - Authentification et gestion des rôles
  - Gestion des utilisateurs

- Traitement des signalements
- Calcul d'itinéraires
- Envoi de notifications
- Intégration avec les services tiers (TomTom, JWT)

➤ **Base de données**

- **Technologie :** PostgreSQL
- **Rôle :** Stocker les données de manière fiable et structurée.
- **Données générées :**
  - Informations utilisateurs
  - Historique de signalements
  - Logs d'événements
  - Statistiques de trafic

➤ **Service TomTom**

- **Technologies : Routing APIs, Traffic APIs, Map Display API**
- **Rôle :** Fournir les services de cartographie, de géolocalisation et de calcul d'itinéraires.
- **Fonctionnalités intégrées :**
  - Affichage de la carte
  - Suivi GPS en temps réel
  - Calcul dynamique de trajets
  - Données de trafic actualisées

➤ **Système d'Authentification**

- **Technologie :** JWT
- **Rôle :** Sécuriser l'accès aux services et aux données utilisateur.
- **Fonctionnement :**
  - JWT (JSON Web Token) est utilisé pour maintenir la session utilisateur de manière stateless
  - Intégration avec Spring Security pour la gestion des autorisations

➤ **Conteneurisation**

- **Technologie :** Docker
- **Rôle :** Assurer le déploiement, l'isolation et la portabilité de chaque service.
- **Avantages :**
  - Environnement unifié pour les développeurs
  - Déploiement simplifié sur n'importe quelle machine
  - Meilleure scalabilité et gestion des versions

## API REST

L'API REST de **Traficandme** constitue le cœur de la communication entre les différents composants (frontend web, mobile et services tiers). Développée avec **Spring Boot**, elle suit les standards RESTful pour garantir une structure claire, maintenable et facilement extensible.

Pour avoir plus de détails sur l'API, suivez les consignes de la rubrique déploiement puis consultez ce [lien](#).

# Déploiement

- Prérequis

Avant de démarrer l'installation de Traficandme, assurez-vous que les outils suivants sont installés sur votre machine :

- **Node.js** (v22 ou supérieur)
- **npm** (v10 ou supérieur)
- **Java** (version 17)
- **Docker & Docker Compose**
- **TomTom API Key** (nécessaire pour les services de cartographie)
- **MongoDB** (doit être installé et lancé localement)

- **Installation**

## 1. Cloner le dépôt

```
git clone https://github.com/trafine/traficandme.git
```

```
cd traficandme
```

## 2. Installer les Dépendances

Installer les dépendances pour chaque service :

- Front-End

```
cd web-traficandme
```

```
./mvnw spring-boot:run
```

- Mobile

```
cd ../mobile-traficandme
```

```
npm install
```

- API

```
cd ..../api-traficandme
```

```
npm install
```

### 3. Initialiser la Base de Données

Initialiser la base de données avec les données initiales :

Processus automatique après le lancement du projet

### 4. Démarrer les Services

Ouvrir un terminal pour chaque service et exécuter les commandes suivantes :

- API

```
cd api-traficandme
```

```
./mvnw spring-boot:run
```

- Front-End

```
cd web-traficandme
```

```
npm run dev
```

- Mobile (développement)

```
cd mobile-traficandme
```

```
npx expo start
```

### 5. Déploiement avec Docker

Pour lancer tous les services en mode conteneurisé :

```
./run.sh
```

## Utilisation de l'Application

Pour utiliser l'application, suivez ces étapes :

1. Assurez-vous que MongoDB est en cours d'exécution sur votre système
2. Ouvrez un terminal et naviguez vers le répertoire racine du projet
3. Démarrez tous les services dans des terminaux séparés en utilisant les commandes ci-dessus
4. Accédez à l'application :

Une fois tous les services démarrés, les interfaces sont accessibles via les adresses suivantes :

- **Frontend Web** : <http://localhost:5173>
- **Documentation API (Swagger)** : <http://localhost:8080/swagger-ui/index.html>
- **API Backend** : <http://localhost:8080/api>

# Sécurité

La sécurité est au cœur de l'architecture de Traficandme. Pour garantir la confidentialité, l'intégrité et l'authenticité des échanges, nous avons mis en place plusieurs mécanismes solides de protection des utilisateurs et des services.

## Authentification via JWT

L'authentification dans Traficandme repose actuellement sur l'utilisation de **JSON Web Tokens (JWT)**. Cette méthode permet de sécuriser les échanges entre le client et l'API, tout en maintenant une architecture sans état (stateless).

### Fonctionnement :

1. **Connexion de l'utilisateur** : L'utilisateur saisit ses identifiants (email et mot de passe) via l'interface de connexion.
2. **Vérification par le backend** : Le backend valide les identifiants à l'aide des données stockées en base.
3. **Génération d'un JWT** : En cas de succès, un token JWT est généré, signé et renvoyé au client.
4. **Utilisation du token** : Ce token est stocké côté client et transmis dans le header Authorization : Bearer <token> pour chaque requête vers une route protégée.
5. **Contrôle du token** : Le backend vérifie la signature et les droits associés au JWT avant d'autoriser l'accès.

### Avantages :

- Authentification rapide et sécurisée
- Architecture sans session côté serveur
- Facilité d'intégration sur mobile et web
- Réduction des échanges inutiles avec la base de données

## Gestion des rôles et des autorisations

Les **rôles** des utilisateurs (utilisateur standard, administrateur...) sont codés dans les **claims** du token JWT.

Le backend Spring Boot utilise ces informations pour **restreindre l'accès** à certaines routes avec des annotations comme @PreAuthorize.

## Sécurité de l'API

- **JWT obligatoire** sur toutes les routes privées
- **Contrôle d'accès par rôle**
- **Validation des entrées utilisateur**
- **CORS configuré finement** pour éviter les accès non autorisés
- **Swagger (documentation API)** accessible uniquement en mode développement

## Sécurité des données

La sécurité des utilisateurs et de leurs informations personnelles est au cœur de l'architecture de Traficandme. Plusieurs mécanismes sont mis en place pour assurer la protection des données tout au long de leur cycle de vie :

- Chiffrement des mots de passe : Tous les mots de passe sont hashés de manière sécurisée avant d'être stockés en base de données.
- Communication chiffrée : Toutes les communications entre le client et le serveur sont sécurisées via HTTPS (protocole TLS), garantissant la confidentialité et l'intégrité des échanges.
- Réponses épurées : Aucune donnée sensible (comme les mots de passe, tokens, ou informations critiques) n'est transmise dans les réponses JSON.
- Logs sécurisés : Les journaux applicatifs sont configurés pour éviter toute fuite d'informations personnelles ou d'identifiants sensibles.

## Sécurité Docker

- Services isolés par **conteneurs indépendants**
- **Secrets et variables d'environnement** stockés dans des fichiers .env (jamais versionnés)
- **Ports restreints** et exposés uniquement si nécessaire

## Bonnes pratiques

- Rotation des tokens et gestion de l'expiration
- Mise à jour régulière des bibliothèques et dépendances
- Analyse de sécurité statique et dynamique
- Prévention des attaques XSS, CSRF, Injection, etc. via Spring Security et bonnes pratiques frontend

# Authentification

L'authentification des utilisateurs est gérée par un système sécurisé reposant sur des tokens JWT (JSON Web Token), permettant de garantir des échanges fiables et stateless. Cependant, afin de faciliter les tests et la gestion de l'application, plusieurs comptes utilisateurs préconfigurés sont fournis avec le système.

## Comptes utilisateurs préconfigurés

Les utilisateurs préconfigurés permettent de tester les fonctionnalités de l'application selon différents rôles. Voici les comptes fournis :

### Administrateur

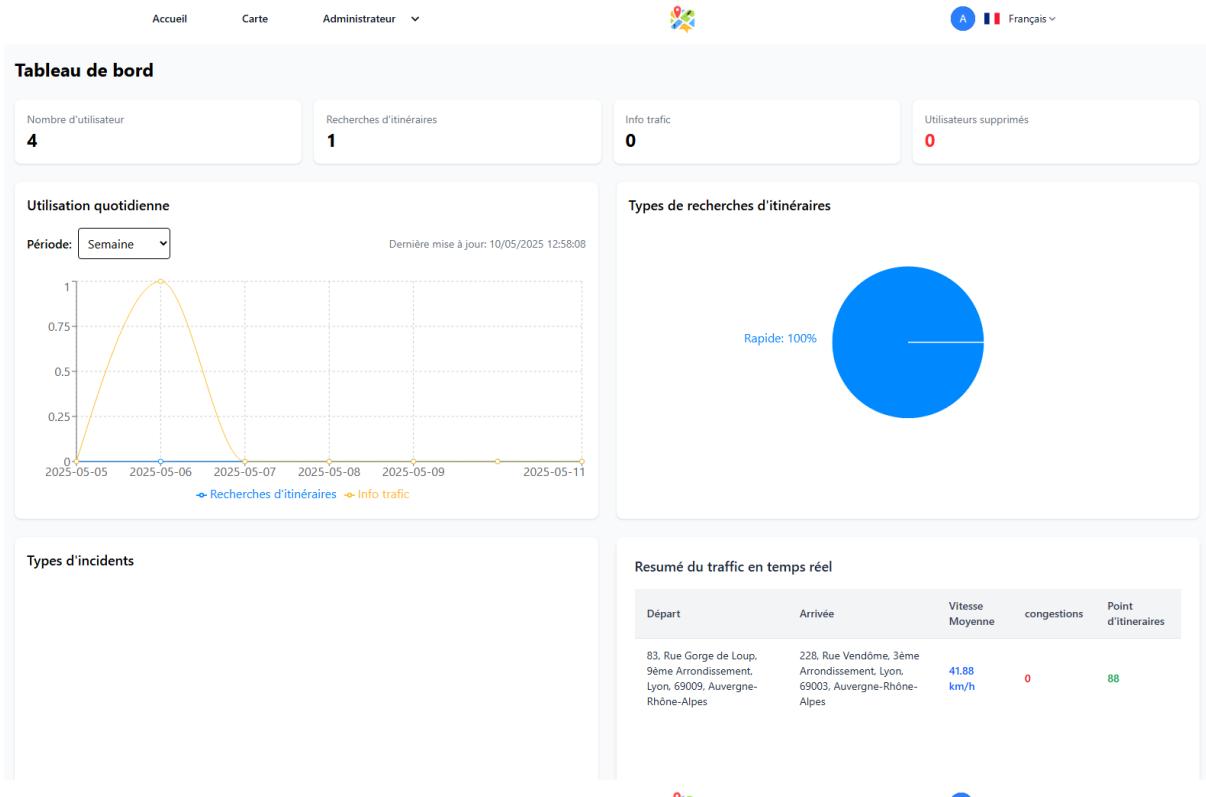
- **Email** : admin@traficandme.com
- **Mot de passe** : \*\*\*\*\* (voir sur le ReadMe)
- **Rôle** : ADMIN
- **Accès** : Gestion complète de l'application, administration des utilisateurs, et visualisation des statistiques détaillées.

### Modérateur

- **Email** : moderator@traficandme.com
- **Mot de passe** : \*\*\*\*\* (voir sur le ReadMe)
- **Rôle** : MODERATOR
- **Accès** : Validation et suppression des signalements d'utilisateurs, gestion des alertes de trafic.

 **Remarque** : Ces comptes sont fournis à des fins de démonstration et de test. Il est recommandé de les désactiver ou de les sécuriser davantage avant le déploiement en production.

Voici des images illustrant les fonctionnalités de l'administrateur et du modérateur :



Gestion des utilisateurs							Ajouter un utilisateur
ID	Username	Email	Rôle	Statut	Date de création		
1	Admin User	admin@traficandme.com	Protégé	ADMIN	ACTIVE	2025-05-05T15:07:39.186+00:00	<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Supprimer définitivement</a>
2	Admin User	admin2@traficandme.com		ADMIN	ACTIVE	2025-05-05T15:07:39.512+00:00	<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Supprimer définitivement</a>
3	Hamza Bely	hamza.bely@traficandme.com		USER	ACTIVE	2025-05-05T15:07:39.783+00:00	<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Supprimer définitivement</a>
4	Moderator User	moderator@traficandme.com		MODERATOR	ACTIVE	2025-05-05T15:07:40.035+00:00	<a href="#">Modifier</a> <a href="#">Supprimer</a> <a href="#">Supprimer définitivement</a>

## Authentification et gestion des rôles

L'application permet d'attribuer des rôles différents à chaque utilisateur (Administrateur, Modérateur, Utilisateur), afin de garantir des niveaux d'accès et des priviléges appropriés à chaque type de profil. Les droits d'accès sont vérifiés à chaque requête sécurisée, et l'API s'assure que l'utilisateur dispose des permissions nécessaires avant de lui accorder l'accès aux ressources protégées.

**Note :** Le système de gestion des rôles peut évoluer pour intégrer de nouvelles fonctionnalités, comme la gestion dynamique des permissions ou l'intégration d'autres mécanismes d'authentification via des fournisseurs externes (Google, Apple).

# Perspective d'évolution

Le projet **Traficandme** est conçu pour évoluer et s'adapter aux besoins des utilisateurs. L'équipe prévoit d'ajouter plusieurs correctifs et nouvelles fonctionnalités afin d'améliorer l'expérience globale :

## Améliorations prévues :

- Intégration de nouvelles langues (espagnol, italien, portugais) pour élargir l'accessibilité à un public international.
- Amélioration de l'interface graphique afin d'offrir une navigation plus intuitive et agréable.
- Déploiement sur le Cloud

## Fonctionnalités à venir :

- Authentification via des comptes Google (OAuth2) et Apple pour faciliter la connexion.
- Intégration de la reconnaissance faciale pour renforcer la sécurité et offrir une méthode de connexion moderne.
- Affichage d'historiques de trajets personnalisés.
- Système de gamification (points, badges, classements) pour encourager l'utilisation participative de l'application.
- Partage d'itinéraires entre utilisateurs.
- Intégration d'un assistant vocal pour une interaction sans les mains.
- Ajout d'un système de covoiturage

Ces évolutions ont pour objectif de rendre l'application plus complète, conviviale et compétitive face aux solutions de navigation existantes.

# Conclusion

En conclusion, **Traficandme** est un projet d'équipe ambitieux et prometteur, porté par une volonté commune de proposer une solution innovante et utile à la mobilité urbaine. Grâce à une collaboration efficace entre les membres du groupe, nous avons su mettre en œuvre une architecture technique moderne et performante, combinant des technologies récentes telles que **React**, **Spring Boot**, **PostgreSQL**, **Docker** et **TomTom**.

L'ensemble du système repose sur une approche **microservices**, garantissant une meilleure évolutivité, une maintenance simplifiée et une robustesse accrue. L'authentification sécurisée **JWT** illustre notre attention portée à la sécurité des utilisateurs. Le projet a également été pensé pour être accessible sur le web et sur mobile, afin de répondre aux besoins d'un large public.

Notre équipe a su relever les défis techniques et organisationnels en mettant en place une gestion rigoureuse du code avec **GitHub**, un déploiement conteneurisé, et des interfaces conviviales. Nous avons aussi conçu notre application pour être facilement étendue grâce à une API REST bien structurée.

Avec des perspectives d'évolution claires — ajout de langues, nouvelles méthodes de connexion, améliorations UI/UX — **Traficandme** est une base solide pour un produit pérenne. Ce projet a renforcé notre esprit d'équipe, notre sens de l'initiative, et notre capacité à concevoir une solution complète répondant à des problématiques concrètes du quotidien.