2025

TRAFICANDME : Documentation technique

Abdoul-Warie Konaté, Abdoul Fadel Bienou, Hamza Belyahiaoui, Jean-philippe Delon
Traffino

SOMMAIRE

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
- **OAuth2 + 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.

Stack technique

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

Composant	Technologie utilisée	Description
Frontend Web	React + Vite	Interface web rapide, moderne et responsive
Frontend Mobile	React Native	Application mobile native multiplateforme
Backend	Spring Boot (Java)	API REST performante et sécurisée
Base de données	PostgreSQL	Stockage relationnel des données utilisateurs et signalements
Authentification	OAuth2+JWT	Sécurisation des accès utilisateurs
Déploiement	Docker	Conteneurisation de tous les services
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

Architecture de l’application

La structure de notre projet :

traficandme/

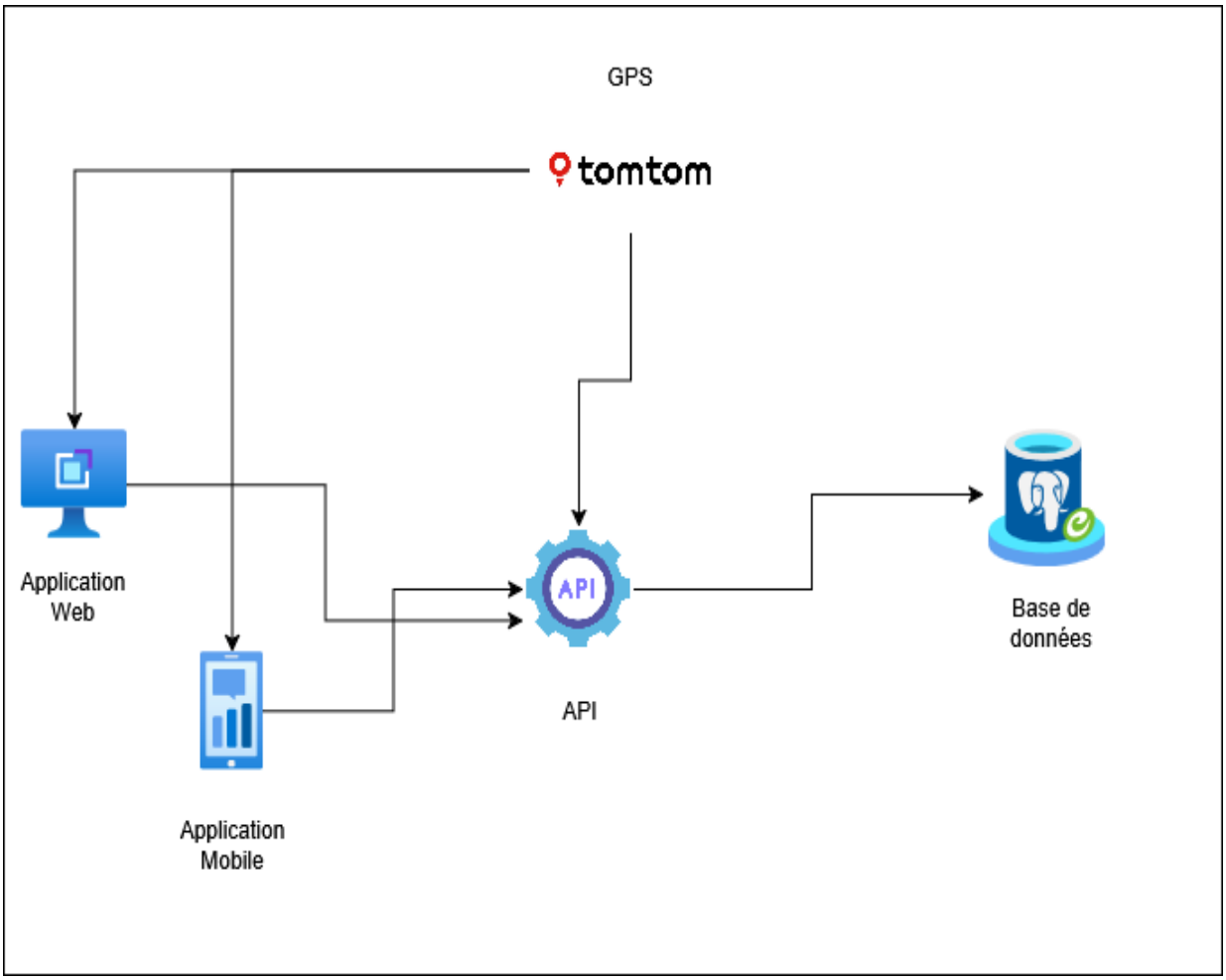
├— api-traficandme/

├— 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 :


GPS
tomtom
Application
Web
API
API
Base de
données
Application
Mobile

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, OAuth, JWT)

➢ **Base de données**

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

➢ **Service TomTom**

- **Technologie :** SDK/API TomTom
- **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

- **Technologies :** OAuth2 + JWT
- **Rôle :** Sécuriser l'accès aux services et aux données utilisateur.
- **Fonctionnement :**
  - OAuth2 assure la délégation d'identité avec des fournisseurs tiers
  - 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, 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 :

Construire les images Docker

docker-compose build

Démarrer les services

docker-compose up -d

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 Google OAuth2 + JWT

L'application utilise l'**OAuth2 de Google** pour l'authentification des utilisateurs. Après authentification, un **JSON Web Token (JWT)** est généré pour permettre une communication sécurisée avec les services backend.

Fonctionnement :

1. **Redirection vers Google OAuth2** : Lorsqu'un utilisateur tente de se connecter, il est redirigé vers la page d'authentification Google.
2. **Validation par Google** : Si l'identifiant est valide, Google redirige l'utilisateur vers notre backend avec un authorization code.
3. **Échange de code contre un token JWT** : Le backend vérifie le code et génère un JWT signé pour le client.
4. **Utilisation du JWT** : Le token JWT est envoyé dans le header Authorization: Bearer <token> pour chaque requête API sécurisée.
5. **Vérification du token** : L'API vérifie la validité, la signature et les permissions intégrées dans le JWT.

Avantages :

- Connexion rapide avec compte Google
- Moins de gestion d'identifiants côté app
- Communication sécurisée et stateless

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

- **Chiffrement des mots de passe** (s’ils sont utilisés pour autre chose que Google OAuth2)
- **Communication via HTTPS (TLS)** pour chiffrer toutes les données en transit
- **Aucune donnée sensible dans les réponses JSON**
- **Logs sécurisés** sans fuite d’informations personnelles

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

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 :

Correctifs prévus :

- 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.

Fonctionnalités à venir :

- Authentification via des comptes Google 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 via **OAuth2** (avec Google) et **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.