Conception et Déploiement d'une Plateforme Géolocalisée de Gestion des Points d'Intérêt (PoI) Multicanal et Temps Réel

Class: Yowyob Inc. Ltd
Projet code: Lock Navi or YowPoint
Document draft by : Prof Thomas Djotio Ndié,
Document Revised by:, Ing Jean Yves Etougue, Software Chief Architect
Applied Security : Confidential
Students Team: Negoum Arthur, Nguiffo Varnel, Audain Meli and Nanfah Elsa and, Ossombe et al From 3 GI

Date: 17.05.2025, revised version: 12.06.2025
fichiers joints: poi.sql, poi.puml,

Résumé du Projet :

Ce projet a pour objectif de concevoir et de développer une plateforme web et mobile de nouvelle génération permettant de gérer, consulter et rechercher des **Points d'Intérêt (PoI)** (restaurants, hôpitaux, lieux touristiques, etc.) en s'appuyant sur la **géolocalisation**, l'**architecture microservices**, et des **interfaces réactives**. En plus des fonctionnalités classiques (ajout, visualisation sur carte, recherche par type, etc.), le système intégrera :

- L'ajout, la modification et la suppression des POI avec leurs coordonnées géographiques.
- La consultation des POI sur une **carte interactive** (Leaflet + OpenStreetMap).
- Des recherches par proximité, type, ou mot-clé.
- Un affichage temps réel des mises à jour via **WebSockets**.
- Une expérience utilisateur fluide sur **web (Next.js)** et **mobile (React Native)**.
- L'intégration d'un backend sécurisé avec **Spring Boot** et **PostgreSQL/PostGIS**.
- Des **mécanismes temps réel** (notifications de nouveaux POI, mises à jour en direct),
- Et des **algorithmes de programmation dynamique** pour l'**optimisation de parcours** ou la **recommandation personnalisée**.

Objectifs Pédagogiques :

- Apprendre à construire une **architecture microservices moderne**.
- Manipuler des **données spatiales** et les exploiter via PostGIS.
- Mettre en œuvre de la **programmation reactive** dans un contexte pratique.
- Mettre en œuvre de la **programmation dynamique** dans un contexte pratique.
- Intégrer des **systèmes temps réel** dans une application distribuée.
- Développer une application **full-stack** réactive et géolocalisée.

Modules Fonctionnels :

1. **Back-end (Spring Boot)**
    - Gestion des utilisateurs, rôles et authentification (JWT / OAuth2) basé sur notre API OneID.
    - API RESTful pour la gestion des POIs.
    - Programmation reactive avec Spring boot Webflux
    - Service de géolocalisation avec PostGIS.
    - WebSocket pour la diffusion en temps réel, mises à jour en direct (ajout, modification, fermeture d'un POI).

- Microservices conteneurisés (Docker).
- Gestion CRUD des PoIs (coordonnées, type, description...).
- **Service d'optimisation** :
  → Itinéraires optimaux entre plusieurs POIs (programmation dynamique : problème du voyageur TSP, plus court chemin de Dijkstra modifié).
  → Recommandations de POIs selon préférences et contraintes horaires.

2. **Front-end Web App(Next.js + Leaflet)**
   - Interface publique avec carte interactive et SEO.
   - Tableau de bord d'administration sécurisé.
   - Consommation des APIs POI + notifications en temps réel.
   - Carte interactive basée sur OpenStreetMap.
   - Recherche filtrée, navigation intuitive.
   - Visualisation d'itinéraires optimaux (résultats des algorithmes dynamiques).
   - Tableau de bord admin.

3. **Application Mobile (React Native)**
   - Affichage des POI à proximité de l'utilisateur.
   - Géolocalisation et navigation.
   - Mode hors-ligne avec synchronisation (SQLite).
   - **Planification d'un parcours optimal** de visite.
   - Notifications en temps réel (ajout/fermeture de POIs, recommandations).
   -
4. **Infrastructure & Déploiement**
   - Base de données PostgreSQL avec extension PostGIS.
   - Docker pour le déploiement conteneurisé.
   - Observabilité basique (logs, surveillance, état des services).
   - Redis pour cache géospatial rapide.
   - Elasticsearch pour recherche textuelle floue.
   - WebSocket Server (Spring) pour temps réel.

Compétences Mobilisées :

- Modélisation de bases de données spatiales (PostGIS).
- Développement backend microservices (Spring Boot, Spring Webflux, REST, WebSocket).
- Développement frontend moderne (React, Next.js, React Native).
- DevOps de base (Docker, CI/CD, PostgreSQL).
- Gestion de projet logiciel agile (Scrum, Kanban).

Livrables Attendus :

- Cahier des charges fonctionnel et technique.
- Architecture logicielle et base de données.
- Code source documenté (Git).
- Vidéo de démonstration.
- Rapport final de projet.
- Présentation orale devant un jury.

Point_of_interest Data model

TECHNOLOGICAL STACK BY LAYER

Backend Microservices (Spring Boot)

Service	Description
Auth Service	Yowyob Auth (OAuth2 + JWT)
POI Service	CRUD, spatial queries, fuzzy search, clustering, proximity
Media Service	Upload/download of POI logos/images (MinIO/S3 + CDN)
Notification Service	WebSocket, Email (SMTP/Mailgun), SMS (Twilio), Push (Firebase)
Analytics Service	Materialized views + usage stats (optional)
Search Service	Elasticsearch + geospatial indexing + autocomplete/fuzzy

Database & Geospatial Stack

Component	Role
PostgreSQL + PostGIS	Main relational and geospatial database
Redis	Cache for frequent geospatial queries (e.g., hot POIs)
Elasticsearch	Full-text and geospatial search (e.g., POI name, type, keyword)

Frontend - Web (Next.js)

Feature	Tech/Libs
SSR & SEO	getServerSideProps, next-sitemap, next-seo
Maps	Leaflet + OpenStreetMap + Cluster markers
Data Fetching	React Query / SWR / GraphQL
Admin Panel	RBAC secured dashboard (custom or react-admin)
Real-Time Updates	WebSocket (SockJS + Stomp.js) or Socket.IO client

Mobile App (React Native)

Feature	Tech/Libs
Mapping	react-native-maps (with Leaflet integration if needed)
Offline Access	SQLite or MMKV

Notifications	Firebase Cloud Messaging (FCM)
Realtime Updates	WebSocket/SSE with reconnects
Location Awareness	Expo Location / react-native-geolocation

Real-time Features

Use Case	Tech Suggestion
Live POI status/updates	WebSocket via Spring (STOMP + SockJS) or Socket.IO
Live reviews, new POIs nearby	Redis Pub/Sub + WebSocket broadcasting
Push Notifications	Firebase (FCM)

DevOps & CI/CD Stack

Area	Tooling
Containerization	Docker + Docker Compose
Orchestration	Kubernetes (K8s) - GKE, EKS or DigitalOcean
CI/CD	GitHub Actions, GitLab CI/CD
API Gateway	Spring Cloud Gateway / Kong Gateway
Secrets/Configs	Vault / Kubernetes Secrets / Doppler
Monitoring	Prometheus + Grafana
Logging	Loki or ELK Stack (Elasticsearch, Logstash, Kibana)

Optional Enhancements

Feature	Stack
AI-Based Recommendations	Python ML microservice (Flask/FastAPI + Scikit-learn or TensorFlow)
Geocoding/Reverse Geocoding	Nominatim (self-hosted OSM geocoder) or Google Geocoding API
CDN for POI Images	Cloudflare, AWS CloudFront, or BunnyCDN
Event-Driven Architecture	Apache Kafka or RabbitMQ for POI updates/events

Bonus: API Gateway Features

- Path-based routing per microservice
- Token-based authentication (JWT, OAuth2)
- Rate limiting / Throttling

- Request tracing and logging
- WebSocket proxying

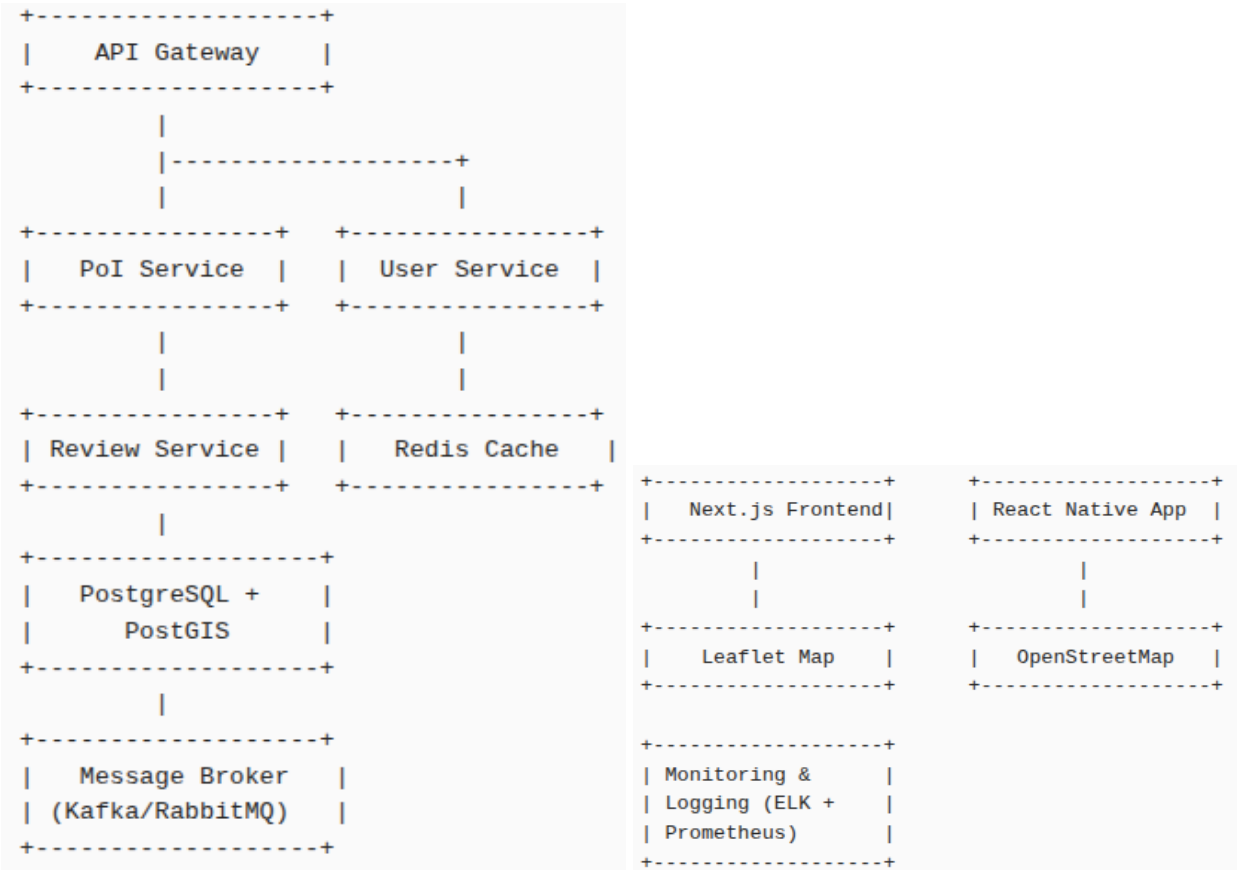
Benefits of This Architecture

- **Scalability**: Stateless services + k8s autoscaling
- **Real-time**: Live POI updates and feedback
- **SEO + UX**: Fully indexable web front, mobile support
- **Modularity**: Easy to extend (add chat, reviews, ranking, etc.)
- **Observability**: Full stack traceability and performance monitoring

Optimal architecture for your Point of Interest (PoI) microservice platform

- Spring Boot for backend microservices
- PostgreSQL + PostGIS for spatial data
- Leaflet + OpenStreetMap for mapping
- Next.js with SEO for web frontend
- React Native for mobile apps
- Realtime capabilities
- And a complementary tech stack for scalability, performance, and developer experience

Diagram Representation



Cas d'usage de la Programmation Dynamique :

Cas d'usage	Description	Technique
Optimisation d'itinéraires	Trouver le meilleur ordre pour visiter n POIs	TSP avec mémoïsation
Planification intelligente	Choisir les POIs selon durée, type, disponibilité	Algorithmes gloutons + DP
Recommandations contextuelles	En fonction du profil, de l'historique et de l'heure	Système hybride (score + contraintes dynamiques)

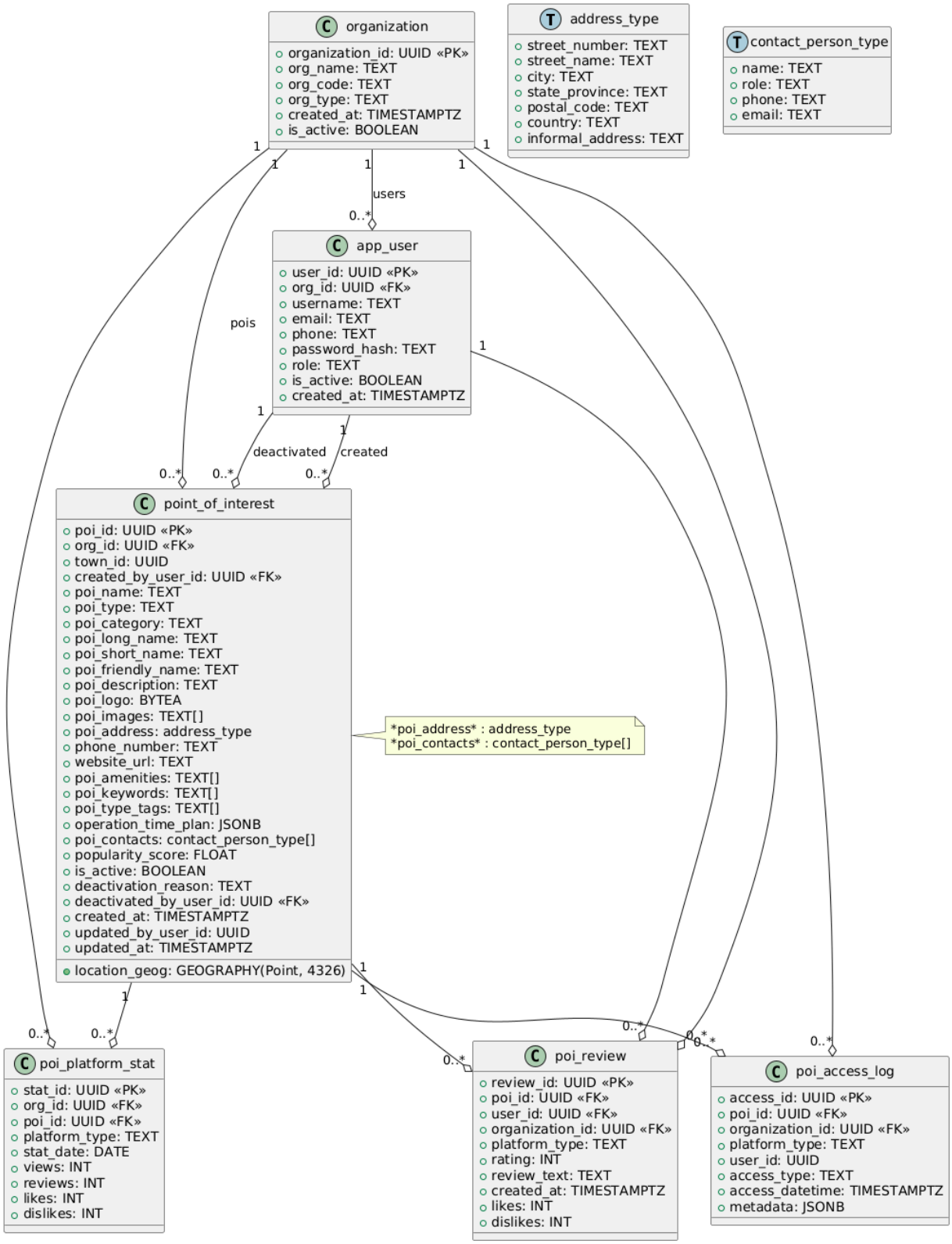
Livrables attendus :

- Cahier des charges + documentation technique.
- Maquettes web/mobile.
- Code source avec dépôt Git.
- Rapport scientifique expliquant l'utilisation des algorithmes de programmation dynamique.
- Présentation et démonstration devant un jury.

Répartition possible par groupe :

Sous-groupe	Tâches principales
Backend	Microservices, sécurité, algorithmes dynamiques
Frontend Web	Interface admin + carte web (Leaflet + Next.js)
Mobile	Application React Native + géolocalisation
DevOps & Infra	Docker, CI/CD, base PostGIS, WebSocket, déploiement
Data & Algo	Programmation dynamique, planification, recommandations

Architecture UML


organization
organization_id: UUID «PK»
org_name: TEXT
org_code: TEXT
org_type: TEXT
created_at: TIMESTAMPTZ
is_active: BOOLEAN
address_type
street_number: TEXT
street_name: TEXT
city: TEXT
state_province: TEXT
postal_code: TEXT
country: TEXT
informal_address: TEXT
contact_person_type
name: TEXT
role: TEXT
phone: TEXT
email: TEXT
users
app_user
user_id: UUID «PK»
org_id: UUID «FK»
username: TEXT
email: TEXT
phone: TEXT
password_hash: TEXT
role: TEXT
is_active: BOOLEAN
created_at: TIMESTAMPTZ
pois
deactivated
created
point_of_interest
poi_id: UUID «PK»
org_id: UUID «FK»
town_id: UUID
created_by_user_id: UUID «FK»
poi_name: TEXT
poi_type: TEXT
poi_category: TEXT
poi_long_name: TEXT
poi_short_name: TEXT
poi_friendly_name: TEXT
poi_description: TEXT
poi_logo: BYTEA
poi_images: TEXT[]
poi_address: address_type
phone_number: TEXT
website_url: TEXT
poi_amenities: TEXT[]
poi_keywords: TEXT[]
poi_type_tags: TEXT[]
operation_time_plan: JSONB
poi_contacts: contact_person_type[]
popularity_score: FLOAT
is_active: BOOLEAN
deactivation_reason: TEXT
deactivated_by_user_id: UUID «FK»
created_at: TIMESTAMPTZ
updated_by_user_id: UUID
updated_at: TIMESTAMPTZ
location_geog: GEOGRAPHY(Point, 4326)
poi_address : address_type
poi_contacts : contact_person_type[]
poi_platform_stat
stat_id: UUID «PK»
org_id: UUID «FK»
poi_id: UUID «FK»
platform_type: TEXT
stat_date: DATE
views: INT
reviews: INT
likes: INT
dislikes: INT
poi_review
review_id: UUID «PK»
poi_id: UUID «FK»
user_id: UUID «FK»
organization_id: UUID «FK»
platform_type: TEXT
rating: INT
review_text: TEXT
created_at: TIMESTAMPTZ
likes: INT
dislikes: INT
poi_access_log
access_id: UUID «PK»
poi_id: UUID «FK»
organization_id: UUID «FK»
platform_type: TEXT
user_id: UUID
access_type: TEXT
access_datetime: TIMESTAMPTZ
metadata: JSONB

PostgreSQL + PostGIS Equivalent Schema

```
-- 1. Extensions nécessaires
CREATE EXTENSION IF NOT EXISTS postgis;
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- 2. Types composites pour adresse et contact
DO $$ BEGIN
    CREATE TYPE address_type AS (
        street_number TEXT,
        street_name TEXT,
        city TEXT,
        state_province TEXT,
        postal_code TEXT,
        country TEXT,
        informal_address TEXT -- e.g. Situé à 200m de Total Mokolo Melen
    );
EXCEPTION WHEN duplicate_object THEN NULL; END $$;

DO $$ BEGIN
    CREATE TYPE contact_person_type AS (
        name TEXT,
        role TEXT,
        phone TEXT,
        email TEXT
    );
EXCEPTION WHEN duplicate_object THEN NULL; END $$;

-- 3. Table principale : Points d'Intérêt
CREATE TABLE IF NOT EXISTS point_of_interest (
    poi_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    town_id UUID,                       -- FK vers table de villes
    created_by_user_id UUID,            -- FK vers table des utilisateurs

    poi_name TEXT NOT NULL,
    poi_type TEXT NOT NULL,             -- Ex: 'RESTAURANT', 'HOTEL'
    poi_category TEXT NOT NULL,         -- Ex: 'Food & Drink', 'Transport'
    poi_long_name TEXT,
    poi_short_name TEXT,
    poi_friendly_name TEXT,
    poi_description TEXT,

    poi_logo BYTEA,                     -- Pour petites images/logo
    poi_images TEXT[],                  -- Tableau d'URLs

    location_geog GEOGRAPHY(Point, 4326) NOT NULL, -- Latitude/Longitude

    poi_address address_type,           -- Adresse structurée

    phone_number TEXT,
    website_url TEXT,

    poi_amenities TEXT[],               -- Ex: ['Wi-Fi', 'Parking', "Jardin"]
```

```
    poi_keywords TEXT[],               -- Ex: ["BUS STOP", "HOPITAL", "CENTRE
VILLE"]
    poi_type_tags TEXT[],              -- Tags complémentaires

    operation_time_plan JSONB,          -- Exemple: { "Monday": "8:00-18:00",
... }

    poi_contacts contact_person_type[],-- Plusieurs contacts possibles

    popularity_score FLOAT DEFAULT 0,  -- Score calculé

    is_active BOOLEAN DEFAULT TRUE, -- Modération par le super admin
    deactivation_reason TEXT,        -- Motif de désactivation
     deactivated_by_user_id UUID REFERENCES app_user(user_id), -- User qui a
désactivé, souvent le super admin

    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_by_user_id UUID,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 4. Table pour organisations/tenants (si non existante)
-- Permet d'associer toute action/statistique à l'organisation cliente.
CREATE TABLE IF NOT EXISTS organization (
    organization_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    org_name TEXT NOT NULL,
    org_code TEXT UNIQUE,
    org_type TEXT, -- (optionnel) : Entreprise, collectivité, etc.
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
    is_active BOOLEAN DEFAULT TRUE

);

-- 4. Table des utilisateurs (rattachés à une organisation)
CREATE TABLE IF NOT EXISTS app_user (
    user_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    org_id UUID NOT NULL REFERENCES organization(org_id) ON DELETE CASCADE,
    username TEXT NOT NULL,
    email TEXT,
    phone TEXT,
    password_hash TEXT,
    role TEXT NOT NULL, -- e.g. 'USER', 'ADMIN', 'SUPER_ADMIN'
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);


-- 5. Table pour les accès/consultations/statistiques par plateforme et
organisation
CREATE TABLE IF NOT EXISTS poi_access_log (
    access_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
      poi_id UUID NOT NULL REFERENCES point_of_interest(poi_id) ON DELETE
CASCADE,
```

```
    organization_id UUID NOT NULL REFERENCES organization(organization_id) ON
DELETE CASCADE,
      platform_type TEXT NOT NULL,      -- 'Android', 'iOS', 'Linux', 'Web',
'Windows', etc.
    user_id UUID,                    -- Utilisateur effectif (optionnel, selon
anonymat)
    access_type TEXT,                -- 'view', 'click', 'review', etc.
    access_datetime TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
     metadata JSONB                     -- Détail additionnel (IP, User-Agent,
version app, etc.)
);

-- 6. Table des reviews (avis/commentaires) avec rattachement à l'organisation
et à la plateforme
CREATE TABLE IF NOT EXISTS poi_review (
    review_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
       poi_id UUID NOT NULL REFERENCES point_of_interest(poi_id) ON DELETE
CASCADE,
    user_id UUID NOT NULL, -- FK vers utilisateurs
    organization_id UUID NOT NULL REFERENCES organization(organization_id) ON
DELETE CASCADE,
    platform_type TEXT NOT NULL,    -- Plateforme source du review
    rating INT CHECK (rating >= 1 AND rating <= 5),
    review_text TEXT,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    likes INT DEFAULT 0,
    dislikes INT DEFAULT 0
);

-- 7. Table de statistiques d'usage par organisation et plateforme (usage
analytique)
CREATE TABLE IF NOT EXISTS poi_platform_stat (
    stat_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    org_id UUID NOT NULL REFERENCES organization(org_id) ON DELETE CASCADE,
    poi_id UUID REFERENCES point_of_interest(poi_id) ON DELETE CASCADE,
    platform_type TEXT NOT NULL, -- e.g. 'IOS', 'ANDROID', 'LINUX', etc.
    stat_date DATE NOT NULL,
    views INT DEFAULT 0,
    reviews INT DEFAULT 0,
    likes INT DEFAULT 0,
    dislikes INT DEFAULT 0
    -- Ajoute ici d'autres KPIs si besoin (ex: favoris, partages)
);


-- 7. Indexes pour performance (tables principales et logs)
CREATE INDEX IF NOT EXISTS idx_poi_org_id             ON point_of_interest
(org_id);
CREATE INDEX IF NOT EXISTS idx_poi_type               ON point_of_interest
(poi_type);
CREATE INDEX IF NOT EXISTS idx_poi_category           ON point_of_interest
(poi_category);
CREATE INDEX IF NOT EXISTS idx_poi_name               ON point_of_interest
(poi_name);
```

```
CREATE INDEX IF NOT EXISTS idx_poi_phone_number ON point_of_interest
(phone_number);

CREATE INDEX IF NOT EXISTS idx_poi_is_active ON point_of_interest
(is_active);

CREATE INDEX IF NOT EXISTS idx_poi_keywords ON point_of_interest USING
GIN (poi_keywords);
CREATE INDEX IF NOT EXISTS idx_poi_amenities ON point_of_interest USING
GIN (poi_amenities);
CREATE INDEX IF NOT EXISTS idx_poi_type_tags ON point_of_interest USING
GIN (poi_type_tags);

CREATE INDEX IF NOT EXISTS idx_poi_location_geog ON point_of_interest USING
GIST (location_geog);

CREATE INDEX IF NOT EXISTS idx_access_log_poi ON poi_access_log (poi_id);
CREATE INDEX IF NOT EXISTS idx_access_log_org ON poi_access_log
(organization_id);
CREATE INDEX IF NOT EXISTS idx_access_log_platform ON poi_access_log
(platform_type);
CREATE INDEX IF NOT EXISTS idx_access_log_date ON poi_access_log
(access_datetime);

CREATE INDEX IF NOT EXISTS idx_poi_review_poi_id ON poi_review (poi_id);
CREATE INDEX IF NOT EXISTS idx_poi_review_user_id ON poi_review (user_id);
CREATE INDEX IF NOT EXISTS idx_poi_review_org ON poi_review
(organization_id);
CREATE INDEX IF NOT EXISTS idx_poi_review_platform ON poi_review
(platform_type);
CREATE INDEX IF NOT EXISTS idx_stat_org_platform ON poi_platform_stat
(org_id, platform_type, stat_date);

-- 8. Vues matérialisées pour agrégats rapides multi-tenants /
multi-plateformes

-- Statistiques d'accès par type de plateforme et organisation
CREATE MATERIALIZED VIEW IF NOT EXISTS mv_stats_poi_access_by_platform_org AS
SELECT
    organization_id,
    platform_type,
    poi_id,
    COUNT(*) AS access_count,
    MIN(access_datetime) AS first_access,
    MAX(access_datetime) AS last_access
FROM
    poi_access_log
GROUP BY organization_id, platform_type, poi_id;

-- Statistiques reviews par plateforme et organisation
CREATE MATERIALIZED VIEW IF NOT EXISTS mv_stats_poi_reviews_by_platform_org AS
SELECT
    organization_id,
    platform_type,
```

```
    poi_id,
    COUNT(*) AS review_count,
    AVG(rating) AS avg_rating,
    SUM(likes) AS total_likes,
    SUM(dislikes) AS total_dislikes
FROM
    poi_review
GROUP BY organization_id, platform_type, poi_id;

-- Statistiques globales classiques
CREATE MATERIALIZED VIEW IF NOT EXISTS mv_poi_by_type AS
SELECT org_id, poi_type, COUNT(*) AS poi_count
FROM point_of_interest
WHERE is_active
GROUP BY org_id, poi_type;
WHERE is_active
GROUP BY org_id, poi_type;

CREATE MATERIALIZED VIEW IF NOT EXISTS mv_poi_by_category AS
SELECT org_id, poi_category, COUNT(*) AS poi_count
FROM point_of_interest
WHERE is_active
GROUP BY org_id, poi_category;

CREATE MATERIALIZED VIEW IF NOT EXISTS mv_poi_by_popularity AS
SELECT
    org_id,
    poi_id,
    poi_name,
    poi_type,
    poi_category,
    location_geog,
    popularity_score
FROM point_of_interest
WHERE is_active
ORDER BY org_id, popularity_score DESC;

-- 9. Exemple de recherche spatiale : POI dans un rayon donné (ici 5 km)
-- Remplacer :lon et :lat par les valeurs réelles
/*
SELECT
    poi_id,
    poi_name,
    poi_type,
    poi_category,
    ST_AsText(location_geog) AS location_wkt,
           ST_Distance(location_geog,   ST_SetSRID(ST_MakePoint(:lon,   :lat),
4326)::geography) AS distance_meters
FROM
    point_of_interest
WHERE
    ST_DWithin(
        location_geog,
        ST_SetSRID(ST_MakePoint(:lon, :lat), 4326)::geography,
```

```
        5000
    )
ORDER BY
    distance_meters;
*/

-- 10. Conseils de maintenance
-- REFRESH MATERIALIZED VIEW mv_stats_poi_access_by_platform_org;
-- REFRESH MATERIALIZED VIEW mv_stats_poi_reviews_by_platform_org;
-- REFRESH MATERIALIZED VIEW mv_poi_by_type;
-- REFRESH MATERIALIZED VIEW mv_poi_by_category;
-- REFRESH MATERIALIZED VIEW mv_poi_by_popularity;
-- (Automatisable via pg_cron, pgAgent, ou tâches programmées côté backend)

-- 10. Exemple de recherche spatiale (multi-tenant & activation)
-- Paramètres : :org_id, :lon, :lat
/*
SELECT
    poi_id,
    poi_name,
    poi_type,
    poi_category,
    ST_AsText(location_geog) AS location_wkt,
           ST_Distance(location_geog,   ST_SetSRID(ST_MakePoint(:lon,   :lat),
4326)::geography) AS distance_meters
FROM
    point_of_interest
WHERE
    org_id = :org_id
    AND is_active
    AND ST_DWithin(
        location_geog,
        ST_SetSRID(ST_MakePoint(:lon, :lat), 4326)::geography,
        5000
    )
ORDER BY
    distance_meters;
*/

-- 11. Conseils de maintenance
-- REFRESH MATERIALIZED VIEW mv_poi_by_type;
-- REFRESH MATERIALIZED VIEW mv_poi_by_category;
-- REFRESH MATERIALIZED VIEW mv_poi_by_popularity;
-- (Automatisable via pg_cron, pgAgent, ou tâches programmées côté backend)

-- 10. Requête de recherche spatiale (exemple d'usage)
-- Chercher les POIs à moins de 5km d'une position GPS (lon, lat)
SELECT *
FROM point_of_interest
WHERE ST_DWithin(
    location_geog,
    ST_MakePoint(:lon, :lat)::geography,
    5000 -- 5km en mètres
);
```