Université de Sousse

École Nationale d'Ingénieurs de Sousse


# Rapport de Projet Module
# Sujet : Smart City Analytics Platform

**Réalisé par :**
Oussema Ben Ameur
Yacin Ben Kacem

**Encadré par :**
M. Bel Hadj Tahar Jamel

Décembre 2025

# Table des matières

# Introduction Générale

Dans un contexte mondial marqué par une urbanisation croissante et des défis environnementaux majeurs, le concept de "Smart City" ou ville intelligente s'impose comme une solution incontournable pour optimiser la gestion des ressources urbaines et améliorer la qualité de vie des citoyens. La métropole "NeoSousse 2030" s'inscrit pleinement dans cette dynamique de transformation digitale.

Les bases de données jouent un rôle central dans cette transformation. Elles permettent de collecter, stocker, structurer et analyser les volumes massifs de données générés par les capteurs IoT, les interactions citoyennes et les systèmes de transport intelligents. Sans une base de données robuste et bien conçue, il serait impossible d'exploiter ces informations pour prendre des décisions éclairées en temps réel.

Ce projet, intitulé "Smart City Analytics Platform", a pour objectif de concevoir et de développer une plateforme complète de gestion des données urbaines. Il s'articule autour de la modélisation d'une base de données relationnelle capable de gérer les capteurs intelligents, les interventions de maintenance, l'engagement citoyen et la flotte de véhicules autonomes municipaux.

Le présent rapport détaille les étapes de réalisation de ce projet, structuré en deux parties principales. La première partie, théorique, se concentre sur la modélisation conceptuelle et relationnelle, ainsi que sur la normalisation de la base de données. La seconde partie, pratique, présente l'implémentation technique de la solution, incluant la création de la base de données PostgreSQL, le développement du backend avec Django et la réalisation d'un tableau de bord interactif avec Streamlit.

# Chapitre 1

# Étude théorique

## 1.1 Introduction

L'étude théorique est une étape fondamentale dans la conception de tout système d'information. Elle permet de traduire les besoins fonctionnels exprimés dans le cahier des charges en un modèle de données formel et structuré. Ce chapitre présente la démarche de modélisation adoptée, allant du modèle conceptuel (Entité-Association) au modèle relationnel normalisé, en passant par l'analyse des dépendances fonctionnelles.

## 1.2 Modélisation conceptuelle : Diagramme E/A, Cardinalités et Contraintes

### 1.2.1 Identification des Entités

Sur la base de l'analyse des besoins, nous avons identifié les entités principales suivantes :

- **Capteur** : Représente les dispositifs IoT installés dans la ville (qualité de l'air, trafic, etc.).
- **Propriétaire** : Entité possédant les capteurs (Municipalité ou Partenaire Privé).
- **Intervention** : Actions de maintenance effectuées sur les capteurs.
- **Technicien** : Personnel qualifié effectuant ou validant les interventions.
- **Citoyen** : Habitants participant à la vie de la cité et aux consultations.
- **Consultation** : Initiatives citoyennes ou sondages lancés par la ville.
- **Véhicule** : Véhicules autonomes municipaux.
- **Trajet** : Déplacements effectués par les véhicules.

### 1.2.2 Diagramme Entité-Association

Le diagramme Entité-Association (E/A) modélise les relations entre ces entités.

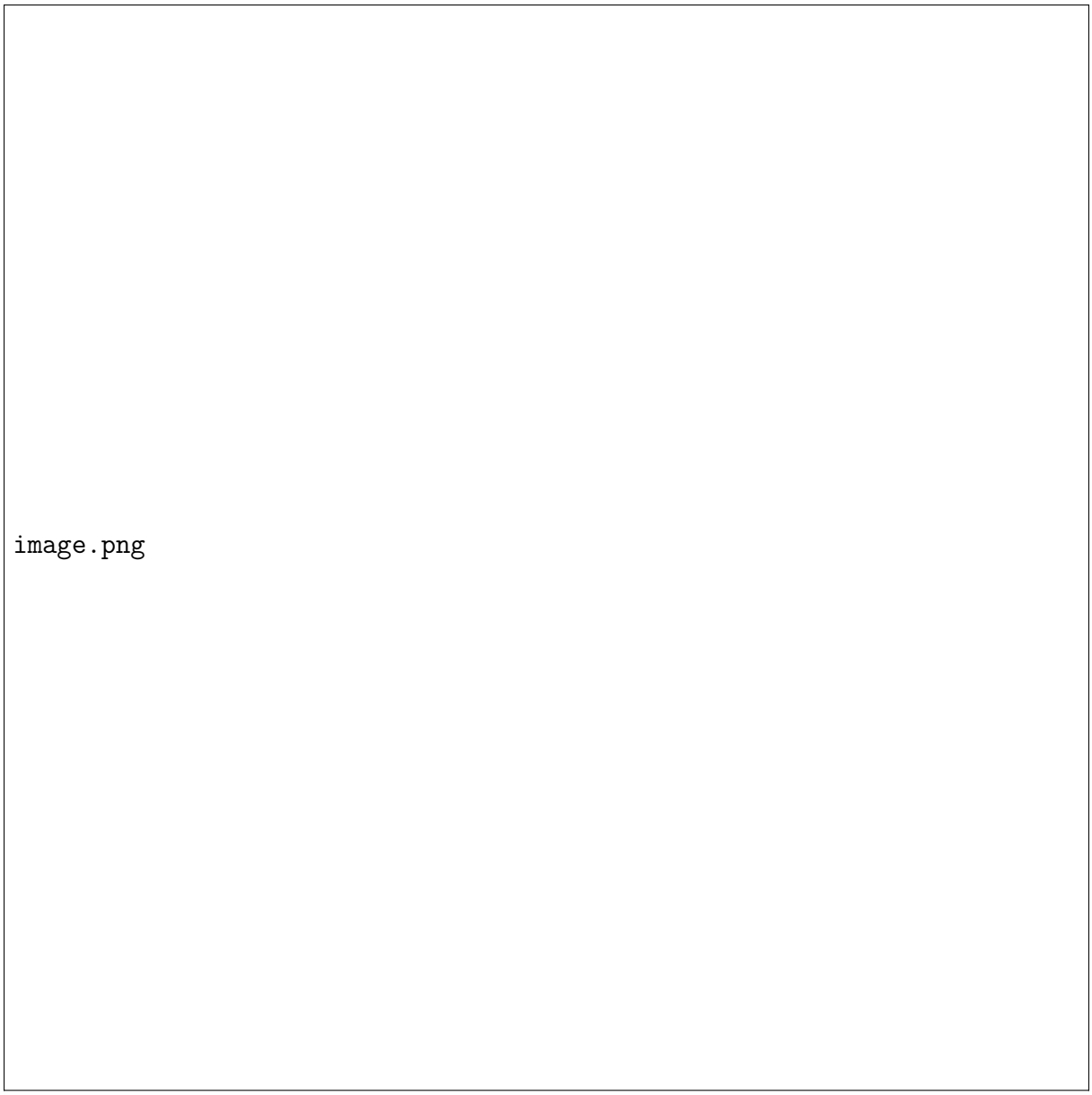


FIGURE 1.1 – Diagramme Entité-Association Global

— Un **Propriétaire** possède plusieurs **Capteurs** (1,n). Un **Capteur** appartient à un seul **Propriétaire** (1,1).
— Un **Capteur** peut subir plusieurs **Interventions** (0,n). Une **Intervention** concerne un seul **Capteur** (1,1).
— Une **Intervention** est réalisée par un **Technicien** (Intervenant) et validée par un autre **Technicien** (Validateur).
— Un **Citoyen** peut participer à plusieurs **Consultations** (0,n). Une **Consultation** peut avoir plusieurs participants (0,n).
— Un **Véhicule** effectue plusieurs **Trajets** (0,n). Un **Trajet** est effectué par un seul **Véhicule** (1,1).

## 1.3 Modèle Relationnel

La traduction du modèle E/A vers le modèle relationnel nous donne le schéma suivant :

- **Proprietaire** (id_proprietaire, nom, type_proprietaire, adresse, telephone, email)
- **Capteur** (id_capteur, type_capteur, latitude, longitude, statut, quartier, date_installation, #id_proprietaire)
- **Technicien** (id_technicien, nom, certification)
- **Intervention** (id_intervention, #id_capteur, date_heure, type_intervention, duree, cout, impact_co2)
- **InterventionTechnicien** (#id_intervention, #id_technicien, role)
- **Citoyen** (id_citoyen, nom, adresse, telephone, email, score_ecologique, preferences_mobilite)
- **Consultation** (id_consultation, titre, description, date_debut, date_fin, statut)
- **Participation** (#id_citoyen, #id_consultation, date_participation)
- **VehiculeAutonome** (id_vehicule, plaque_immatriculation, type_vehicule, energie_utilisee)
- **Trajet** (id_trajet, #id_vehicule, origine, destination, duree, economie_co2)

## 1.4 Analyse des Redondances et Normalisation

### 1.4.1 Formes Normales

Nous avons veillé à ce que toutes les relations respectent au moins la 3ème Forme Normale (3FN) pour éviter les redondances et les anomalies de mise à jour.

- **1FN** : Tous les attributs sont atomiques (pas de groupes répétitifs). Par exemple, les coordonnées du propriétaire sont séparées en attributs distincts.
- **2FN** : Toutes les tables ont une clé primaire simple (UUID) ou, dans le cas des tables d'association, tous les attributs non-clés dépendent de la totalité de la clé primaire composite.
- **3FN** : Il n'y a pas de dépendances transitives entre les attributs non-clés. Par exemple, l'adresse du propriétaire dépend uniquement de l'ID du propriétaire, pas du capteur.

### 1.4.2 Dépendances Fonctionnelles

Pour chaque table, nous avons identifié les dépendances fonctionnelles (DF). Par exemple pour la table **Capteur** :

$$id\_capteur \rightarrow \{type, latitude, longitude, statut, id\_proprietaire\}$$

Cette DF confirme que *id_capteur* est bien la clé primaire et qu'il n'y a pas de redondance interne.

## 1.5 Conclusion

La phase de modélisation théorique a permis d'établir un schéma de base de données solide, respectant les règles de normalisation. Ce schéma servira de fondation pour l'implémentation pratique dans le chapitre suivant.

# Chapitre 2

# Étude pratique

## 2.1 Introduction

Ce chapitre décrit la mise en œuvre technique de la plateforme "Smart City Analytics Platform". Nous détaillerons les choix technologiques, l'implémentation de la base de données, ainsi que le développement de l'interface utilisateur.

## 2.2 Technologies Utilisées

L'architecture de la solution repose sur une séparation claire entre le backend (gestion des données et API) et le frontend (visualisation). Voici le détail des technologies employées :

### 2.2.1 Backend : Django et PostgreSQL

- **PostgreSQL** : Choisi pour sa robustesse et sa gestion native des types de données complexes. Nous utilisons le pilote `psycopg2-binary` pour l'interface avec Python.
- **Django Framework** : Le cœur du système. Il assure la structure MVC (Modèle-Vue-Contrôleur) et la sécurité.
- **Django REST Framework (DRF)** : Utilisé pour exposer les données de la base sous forme d'API RESTful (JSON). Cela permet de découpler totalement le frontend du backend.
- **Django CORS Headers** : Middleware indispensable pour permettre au frontend (Streamlit) d'effectuer des requêtes vers l'API (Django) alors qu'ils tournent sur des ports différents.
- **Faker** : Bibliothèque utilisée dans nos scripts (`generate_data.py`) pour peupler la base de données avec des milliers d'enregistrements réalistes (noms, adresses, coordonnées GPS) afin de simuler une ville active.

### 2.2.2 Frontend : Streamlit et Visualisation

- **Streamlit** : Framework Python permettant de transformer des scripts de données en applications web interactives rapidement. Il consomme l'API Django via la bibliothèque `requests`.

— **Pandas** : Utilisé intensivement pour charger les données JSON reçues de l'API dans des DataFrames, faciliter les calculs statistiques (moyennes, groupements par quartier) et la préparation des données pour les graphiques.
— **Folium & Streamlit-Folium** : Permet l'intégration de cartes interactives (Leaflet.js). Nous l'utilisons pour géolocaliser les capteurs et tracer les trajets des véhicules autonomes sur la carte de Sousse.
— **Plotly Express** : Bibliothèque de visualisation graphique avancée. Elle est utilisée pour générer les histogrammes (pollution par zone), les diagrammes en barres (disponibilité) et les courbes d'évolution (coûts de maintenance) avec une interactivité riche (zoom, survol).

## 2.3 Implémentation de la Base de Données

La base de données a été implémentée en utilisant le langage SQL (pour la définition du schéma) et l'ORM de Django. Voici un extrait du script SQL de création des tables principales :

```sql
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE IF NOT EXISTS capteurs (
    id_capteur UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    type_capteur VARCHAR(50) NOT NULL,
    latitude DECIMAL(9,6) NOT NULL,
    longitude DECIMAL(9,6) NOT NULL,
    statut VARCHAR(20) NOT NULL,
    quartier VARCHAR(100),
    proprietaire_type VARCHAR(20),
    date_installation DATE NOT NULL
);
```

Listing 2.1 – Création de la table Capteurs

Nous avons également défini les modèles Django correspondants dans le fichier `models.py` pour faciliter l'interaction avec l'application.

## 2.4 Interface et Visualisation

Le tableau de bord développé avec Streamlit offre une vue d'ensemble en temps réel de la ville intelligente. Il est composé de plusieurs sections :

### 2.4.1 Métriques Clés

En haut du tableau de bord, des indicateurs clés de performance (KPI) sont affichés :
— Nombre de capteurs actifs par rapport au total.
— Coût annuel de la maintenance.
— Score écologique moyen des citoyens.
— Quantité de CO2 économisée grâce aux trajets optimisés.

### 2.4.2 Carte Interactive

Une carte interactive (basée sur Folium) affiche la position géographique de tous les capteurs et des véhicules autonomes en temps réel. Les marqueurs sont colorés selon le statut des capteurs (Vert : Actif, Rouge : Hors service).

### 2.4.3 Analyses Détaillées

Des onglets permettent d'accéder à des analyses spécifiques :
— **Pollution** : Graphique des zones les plus polluées (basé sur l'indice AQI).
— **Disponibilité** : Taux de disponibilité des capteurs par quartier.
— **Citoyens** : Classement des citoyens les plus engagés.
— **Interventions** : Suivi des coûts et des types d'interventions (prédictive vs corrective).
— **Trajets** : Visualisation des trajets les plus écologiques sur la carte.

## 2.5 Validation des Questions Métiers

La plateforme a été conçue pour répondre précisément aux questions stratégiques posées dans le cahier des charges. Voici comment chaque interrogation est traitée par notre système, avec les requêtes SQL correspondantes et leur visualisation.

### 2.5.1 Zones les plus polluées (24h)

**Question :** Quelles sont les zones les plus polluées de la ville sur les dernières 24 heures ?

**Solution :** Nous agrégeons les mesures des capteurs de type 'qualité_air' par quartier sur la dernière journée.

```
SELECT c.quartier, AVG(m.valeur) as aqi_moyen
FROM mesures m
JOIN capteurs c ON m.id_capteur = c.id_capteur
WHERE c.type_capteur = 'qualit _air'
  AND m.date_mesure >= NOW() - INTERVAL '24 hours'
GROUP BY c.quartier
ORDER BY aqi_moyen DESC;
```

Listing 2.2 – Requête SQL Pollution par Zone

**Visualisation :** Un graphique en barres dans l'onglet "Pollution" du tableau de bord affiche l'indice AQI moyen par quartier, avec un code couleur (Vert à Rouge) pour identifier rapidement les zones critiques.

### 2.5.2 Disponibilité des Capteurs

**Question :** Quel est le taux de disponibilité des capteurs par arrondissement ?

**Solution :** Nous calculons le ratio entre les capteurs 'actifs' et le nombre total de capteurs pour chaque zone.

```
SELECT quartier,
       COUNT(*) FILTER (WHERE statut = 'actif') * 100.0 / COUNT(*) as
   taux_disponibilite
FROM capteurs
```

```
GROUP BY quartier;
```

Listing 2.3 – Requête SQL Disponibilité

**Visualisation :** Un diagramme en barres montre le pourcentage de disponibilité par zone, permettant aux équipes de maintenance de cibler les quartiers prioritaires.

### 2.5.3 Engagement des Citoyens

**Question :** Quels sont les citoyens les plus engagés dans la réduction de leur empreinte carbone ?

**Solution :** Nous sélectionnons les citoyens ayant le score écologique le plus élevé.

```
SELECT nom, email, score_ecologique
FROM citoyens
ORDER BY score_ecologique DESC
LIMIT 10;
```

Listing 2.4 – Requête SQL Top Citoyens

**Visualisation :** Un classement (Leaderboard) affiche le Top 10 des citoyens, encourageant la gamification et l'engagement civique.

### 2.5.4 Interventions Prédictives

**Question :** Combien d'interventions prédictives ont été réalisées ce mois-ci, et quelle économie ont-elles générée ?

**Solution :** Nous filtrons les interventions de type 'prédictive' sur le mois courant et sommons les économies réalisées (calculées par rapport au coût d'une panne majeure évitée).

```
SELECT COUNT(*) as nb_interventions,
       SUM(cout_evite - cout_reel) as economie_totale
FROM interventions
WHERE type_intervention = 'pr dictive'
  AND date_heure >= DATE_TRUNC('month', CURRENT_DATE);
```

Listing 2.5 – Requête SQL Interventions Prédictives

**Visualisation :** Des indicateurs numériques (KPI) affichent le nombre d'interventions et le montant économisé en temps réel.

### 2.5.5 Trajets Écologiques

**Question :** Quels trajets de véhicules autonomes ont permis la plus grande réduction de CO ?

**Solution :** Nous identifions les trajets ayant le meilleur indicateur d'économie de CO2.

```
SELECT origine, destination, economie_co2
FROM trajets
ORDER BY economie_co2 DESC
LIMIT 5;
```

Listing 2.6 – Requête SQL Top Trajets CO2

**Visualisation :** Une carte interactive trace les lignes des trajets les plus vertueux, permettant d'analyser les flux de mobilité les plus efficients.

## 2.6 Conclusion

L'implémentation pratique a permis de concrétiser le modèle théorique. L'utilisation combinée de PostgreSQL, Django et Streamlit a permis de créer une plateforme performante, évolutive et dotée d'une interface utilisateur intuitive pour la gestion de la Smart City.

# Conclusion Générale

Le projet "Smart City Analytics Platform" a permis de mettre en pratique les concepts fondamentaux des bases de données relationnelles dans un contexte moderne et pertinent. De la modélisation conceptuelle à l'implémentation technique, chaque étape a contribué à la construction d'un système d'information cohérent.

La base de données conçue permet de répondre efficacement aux besoins de gestion des ressources urbaines, en assurant l'intégrité et la disponibilité des données. Le tableau de bord interactif offre aux décideurs une vision claire de l'état de la ville, facilitant ainsi la prise de décision pour une gestion plus durable et efficace.

Ce projet ouvre la voie à de futures améliorations, telles que l'intégration de données en temps réel via des protocoles IoT (MQTT), l'utilisation de l'intelligence artificielle pour la maintenance prédictive avancée, ou encore l'extension de la plateforme à d'autres domaines de la gestion urbaine.

# Table des figures