

# Architecture de Données : Application de Suivi de Maintenance

Modélisation Relationnelle SQL et  
Implémentation Backend Python

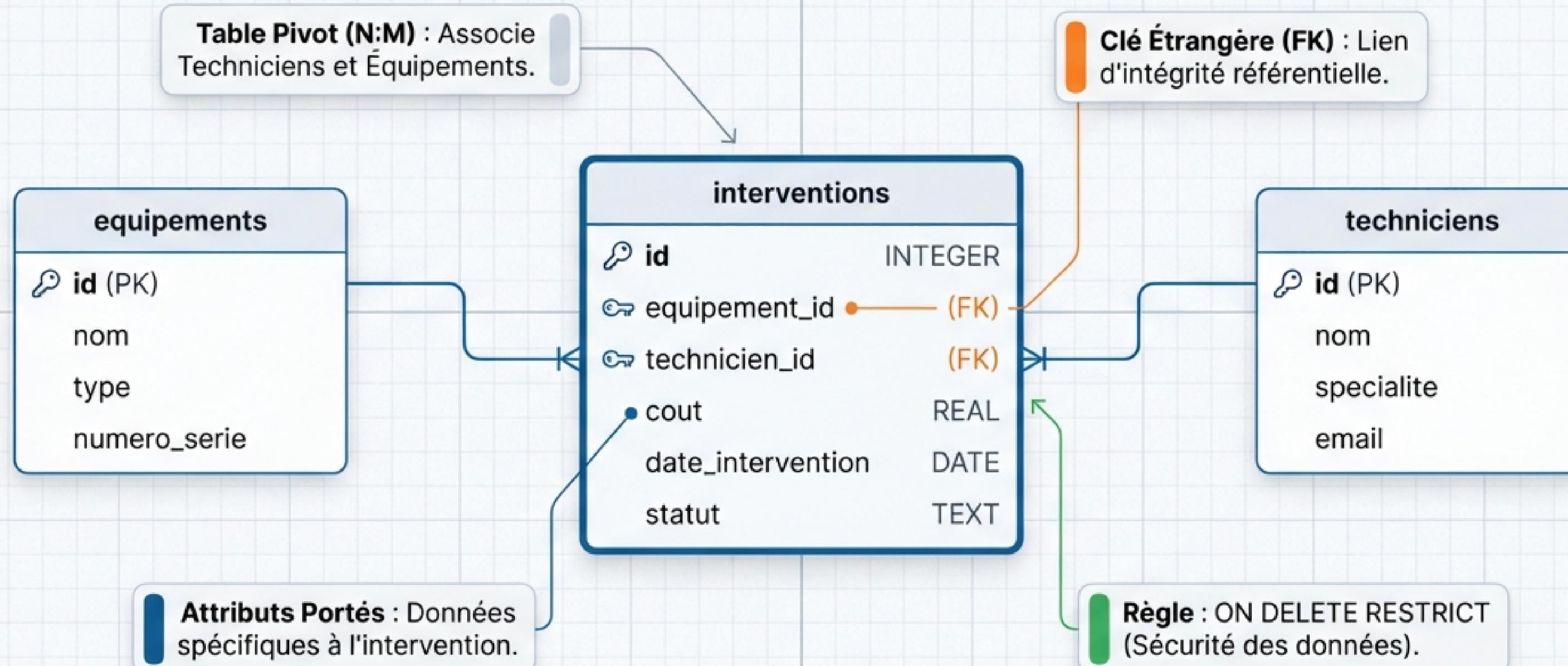
**PROJET :** Gestion de parc matériel et historique d'interventions.

**STACK TECHNIQUE :** Python 3.x (Logique Métier) +  
SQLite 3 (Stockage Relationnel).

**ARCHITECTURE :** Modèle en 3 couches (Présentation,  
Métier, Données).

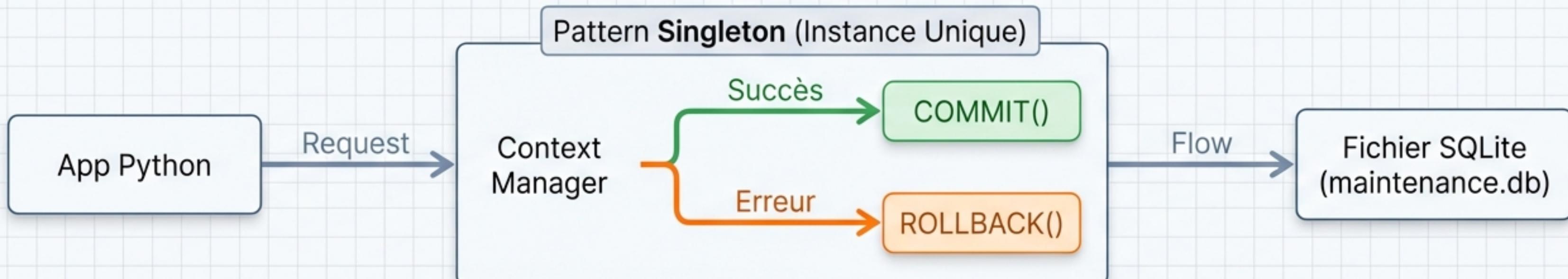


# La Structure SQL : Modélisation Relationnelle



# Mécanisme de Connexion et Sécurité

## Architecture du Driver (`db\_connection.py`)



## Sécurité des Requêtes

**✗ INTERDIT** (Risque Injection SQL)

```
# Interpolation de chaîne dangereuse
id_tech = "1 OR 1=1"
cursor.execute(f"SELECT * FROM techniciens
WHERE id = {id_tech}")
```

**✓ CORRECT** (Requêtes Paramétrées)

```
# Utilisation de tuples et placeholders '?'
id_tech = 1
cursor.execute("SELECT * FROM techniciens
WHERE id = ?", (id_tech,))
```

Sécurité Absolue

# Validation et Test : Analyse des Données Réelles

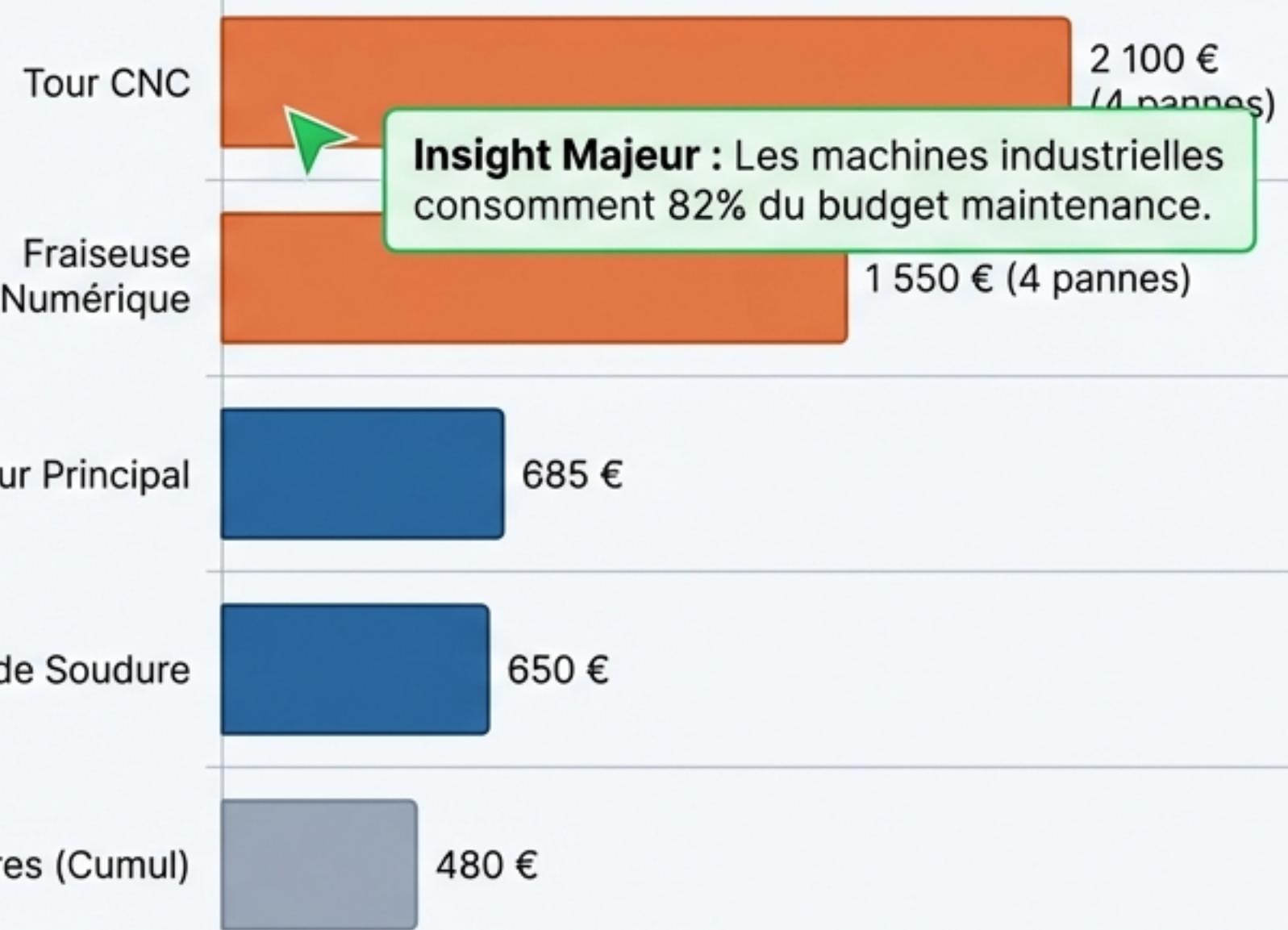
## Jeu de Données de Test (2024)

- 29 Interventions
- 10 Équipements
- 5 Techniciens
- Objectif : Identifier les équipements critiques.

## Requête d'Analyse (SQL)

```
SELECT e.nom, SUM(i.cout) as total_cout
FROM equipements e
JOIN interventions i ON e.id = i.equipement_id
GROUP BY e.nom
ORDER BY total_cout DESC;
```

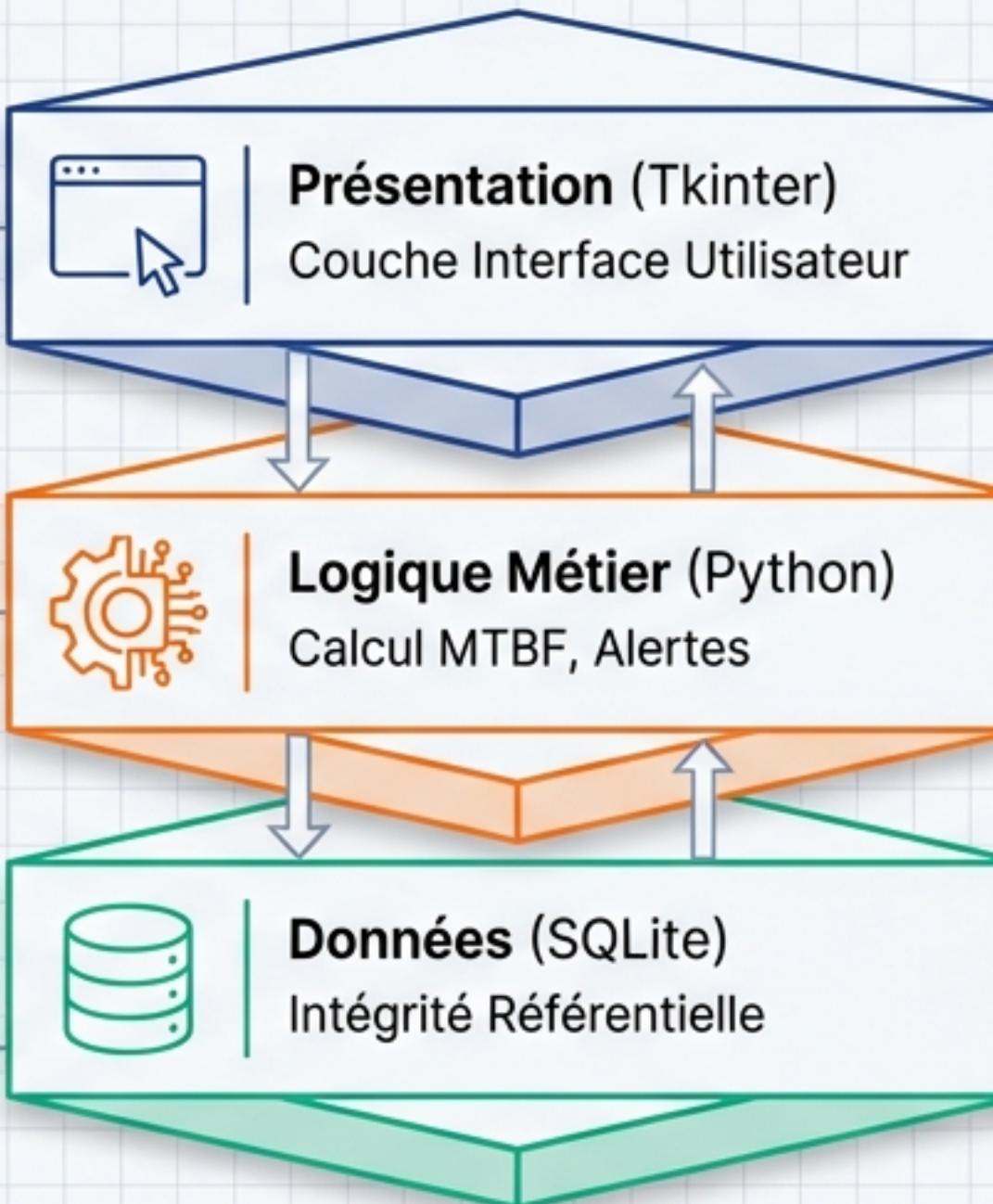
## Coût Maintenance par Équipement



# Synthèse Technique

**Robustesse**

- Intégrité des Données : Garantie par Clés Étrangères (FK). Pas d'orphelins.
- Sécurité : Protection totale contre les injections SQL (Paramètres `?`).



**Performance**

- Architecture : Pattern DAO pour découpler l'accès aux données.
- Optimisation : Indexation des colonnes de dates et types pour recherches rapides.

**CONCLUSION : UNE SOLUTION FIABLE, SÉCURISÉE ET PERFORMANTE.**