# Architecture de la Base de Données - MyConcierge

## 1. Modélisation de la Base de Données

### Table ReferenceTypes (Gère les types d’unités et d’utilisateurs)

| Id | Nom | Categorie |  
|----|-------------|----------------|  
| 1 | Immeuble | Unite |  
| 2 | Appartement | Unite |  
| 3 | Chambre | Unite |  
| 4 | Parking | Unite |  
| 5 | Locataire | Utilisateur |  
| 6 | Locateur | Utilisateur |  
| 7 | Concierge | Utilisateur |

Cette table permet d'ajouter de nouveaux types sans modifier la structure de la base.

### Table Utilisateurs (Propriétaires, Locataires, Concierges)

| Id | Nom | Email | Téléphone | ReferenceTypeId (FK) |  
|----|------|---------------|--------------|----------------------|  
| 1 | Alice | alice@email.com | 438-555-9876 | 5 (Locataire) |  
| 2 | John | john@email.com | 514-555-1234 | 6 (Locateur) |

### Table Unites (Immeubles, Appartements, Chambres, Parkings)

| Id | Nom | ReferenceTypeId (FK) | ParentUniteId (FK) | ProprietaireId (FK) | Prix | EstLouee |  
|----|----------------|----------------------|--------------------|----------------------|---------|----------|  
| 1 | Immeuble A | 1 (Immeuble) | NULL | 2 (John) | NULL | false |  
| 2 | Appartement 101 | 2 (Appartement) | 1 | 2 (John) | 1500.00 | false |  
| 3 | Chambre 1 | 3 (Chambre) | 2 | 2 (John) | 600.00 | false |  
| 4 | Parking 1 | 4 (Parking) | 1 | 2 (John) | 200.00 | false |

### Table ContratsLocation (Contrats entre Locataire et Unité)

| Id | LocataireId (FK) | UniteId (FK) | DateDebut | DateFin | Montant |  
|----|----------------|-------------|------------|------------|---------|  
| 1 | 1 (Alice) | 3 (Chambre 1) | 2024-03-01 | 2025-03-01 | 600.00 |

## 2. Relations entre les Tables

Utilisateur (Locateur, Locataire, Concierge)  
 ├──> ReferenceType (Définit les types d'utilisateurs)  
 ├──> Propriétaire possède plusieurs Unites  
 ├──> Unite (Immeuble, Appartement, Chambre, Parking)  
 ├──> ReferenceType (Définit les types d'unités)  
 ├──> ParentUniteId (Permet de lier une Chambre à un Appartement)  
 ├──> ProprietaireId (L'utilisateur qui possède cette unité)  
 ├──> Locataire loue une Unite via ContratLocation

## 3. Avantages de cette Architecture

✅ Évolutivité → Ajout facile de nouveaux types (Bureau, Magasin, etc.).  
✅ Centralisation des données → Toutes les unités sont dans une seule table `Unites`.  
✅ Hiérarchie flexible → Un Appartement peut contenir des Chambres, et un Immeuble peut contenir plusieurs unités.  
✅ Gestion simplifiée des utilisateurs → Tous les utilisateurs (Locataires, Locateurs, Concierges) sont dans une seule table `Utilisateurs` avec un type défini via `ReferenceTypeId`.

## 4. Prochaine Étape : Implémentation de l'API REST

📌 Controllers à implémenter :  
1️⃣ `ReferenceTypeController` → Ajouter et gérer les types d’unités et d’utilisateurs.  
2️⃣ `UtilisateurController` → Ajouter/modifier des utilisateurs (Locateur, Locataire, Concierge).  
3️⃣ `UniteController` → Ajouter/modifier/supprimer des unités (Appartement, Chambre, Parking).  
4️⃣ `ContratController` → Assigner un locataire à une unité et gérer les contrats de location.

## Conclusion

🔥 Ton architecture est solide et prête à être implémentée en API REST.  
🎯 Toutes les entités sont bien organisées pour une gestion flexible et évolutive.