

# 📦 Projet IoT - Smart Home

---

Application de gestion d'objets connectés pour la maison intelligente avec interface web et microservices backend.

---

## 🏗 Architecture du projet

Ce projet utilise une **architecture microservices** composée de :

- **Frontend** : ViteJs + React + TypeScript (interface utilisateur)
  - **Backend** : Architecture microservices avec NestJS
    - **Gateway** : API principale et point d'entrée unique
    - **Service Thermostat** : Gestion du thermostat connecté
    - **Service Lamp** : Contrôle des lampes intelligentes
    - **Service Motion** : Détection de mouvement
  - **Base de données** : SQLite avec Prisma ORM
- 

## 📋 Prérequis

Avant de commencer, assurez-vous d'avoir installé les outils suivants sur votre machine :

- **Node.js** (v18 ou supérieur) - [Télécharger Node.js](#)
- **npm** (inclus avec Node.js) ou **yarn** - Gestionnaire de paquets
- **Git** - Pour cloner le dépôt

Pour vérifier vos versions installées :

```
node --version  
npm --version  
git --version
```

## 🚀 Installation et lancement du projet

### 1 Cloner le projet

```
git clone https://github.com/Mathischn/iot-project.git  
cd iot-project
```

### 2 Configuration de la base de données (Prisma + SQLite)

**Important** : Cette étape doit être effectuée en premier !

```
# Aller dans le dossier du backend  
cd backend  
  
# Aller dans le dossier Prisma  
cd prisma  
  
# Installer les dépendances Prisma  
npm install  
  
# Générer le client Prisma  
npx prisma generate  
  
# Créer et initialiser la base de données  
npx prisma migrate dev --name init
```

**Note :** Cette étape crée la base de données SQLite (`dev.db`) et applique le schéma défini dans `prisma/schema.prisma`. Le fichier `dev.db` sera créé automatiquement dans le dossier `backend`.

**Optionnel** : Visualiser la base de données avec Prisma Studio

```
npx prisma studio
```

### ③ Lancement du Frontend

```
# Depuis la racine du projet, ouvrir un nouveau terminal  
cd iot-front  
  
# Installer les dépendances  
npm install  
  
# Lancer le serveur de développement  
npm run dev
```

Le frontend sera accessible sur : **http://localhost:5173**

**Astuce** : Le serveur frontend se recharge automatiquement à chaque modification du code (Hot Reload).

### ④ Lancement des services Backend

**⚠ Important** : Les 4 services backend doivent être lancés **dans 4 terminaux séparés** et doivent **tous être actifs** en même temps pour que l'application fonctionne correctement.

- ◊ **Terminal 1 : Gateway (Port 3000)**

## Point d'entrée principal de l'API - **À lancer en premier**

```
# Depuis la racine du projet
cd backend/gateway

# Installer les dépendances
npm install

# Lancer la gateway en mode production
npm run start:prod
```

Gateway accessible sur : **http://localhost:3000**

---

### ◊ Terminal 2 : Service Thermostat (Port 3002)

Gestion du thermostat connecté

```
# Ouvrir un nouveau terminal depuis la racine
cd backend/thermostat-back

# Installer les dépendances
npm install

# Lancer le service
npm run start:prod
```

Service thermostat accessible sur : **http://localhost:3002**

---

### ◊ Terminal 3 : Service Lamp (Port 3001)

Contrôle des lampes intelligentes

```
# Ouvrir un nouveau terminal depuis la racine
cd backend/lamp-back

# Installer les dépendances
npm install

# Lancer le service
npm run start:prod
```

Service lamp accessible sur : **http://localhost:3001**

---

## ❖ Terminal 4 : Service Motion (Port 3003)

Détection de mouvement

```
# Ouvrir un nouveau terminal depuis la racine
cd backend/motion-back

# Installer les dépendances
npm install

# Lancer le service
npm run start:prod
```

Service motion accessible sur : **http://localhost:3003**

## ⌚ Récapitulatif des URLs

Service	URL	Description
<b>Frontend</b>	http://localhost:5173	Interface utilisateur web
<b>Gateway</b>	http://localhost:3000	API principale (point d'entrée)
<b>Lamp</b>	http://localhost:3001	Microservice lampes
<b>Thermostat</b>	http://localhost:3002	Microservice thermostat
<b>Motion</b>	http://localhost:3003	Microservice détection mouvement
<b>Base de données</b>	backend/dev.db	Fichier SQLite

**Total : 5 terminaux actifs requis** (1 frontend + 4 backend services)

## ⚠ Important - Ordre de démarrage

**ATTENTION :** Il est **crucial** d'attendre que la **Gateway soit complètement démarrée** avant de lancer les autres services (Lamp, Thermostat, Motion).

Les services doivent s'enregistrer auprès de la Gateway au démarrage. Si vous lancez les services avant que la Gateway ne soit prête, ils ne pourront pas se connecter et l'application ne fonctionnera pas correctement.

**Ordre recommandé :**

1.  Lancer la **Gateway** → Attendre le message de démarrage complet
2.  Lancer le **Service Lamp**
3.  Lancer le **Service Thermostat**
4.  Lancer le **Service Motion**
5.  Lancer le **Frontend**