CAHIER DES CHARGES

# Objectif global

Créer une solution qui permet :

1. **Cartographier les télés** dans les bâtiments/salles.
2. **Réserver une télé** (planning, disponibilité).
3. **Suivre l’état de fonctionnement** de chaque télé.
4. **Mettre à jour les données** facilement.
5. **Localiser les télés perdues/oubliées** via Air Tags ou autre technologie.

# Les fonctionnalités à intégrer

**1. Cartographie des salles et télés**

* Interface visuelle (plan des bâtiments ou carte stylisée).
* Points cliquables représentant les télés.
* Détail affiché sur clic : salle, état, modèle, dispo.

**2. Réservation des télés**

* Planning (type calendrier ou Google Agenda).
* Affectation à une salle ou à un événement.
* Notification ou alerte si plusieurs réservations ou conflit.

**3. Suivi de l’état du matériel**

* Champs : Fonctionnelle / En panne / En réparation / Inconnue.
* Historique des incidents (optionnel).
* Possibilité de signaler une panne via l’interface.

**4. Mise à jour des informations**

* Interface admin (formulaire CRUD : Create, Read, Update, Delete).
* Export/Import possible en CSV ou autre.

**5. Localisation en temps réel / alerte perte**

* **Option 1 : Air Tags →**. Fonctionne bien mais dépend du réseau.
* **Option 2 : Puces GPS / BLE (Bluetooth Low Energy) →** Plus complet mais demande plus de budget et de technique.
* **Option 3 : Rapprochement manuel →** On enregistre juste dans quelle salle la télé devrait être et on vérifie physiquement.

# Stack techno possible

**Backend**

* **Python + Flask** ou **Node.js + Express** pour l’API
* **Base de données** : SQLite (léger) ou PostgreSQL (robuste)

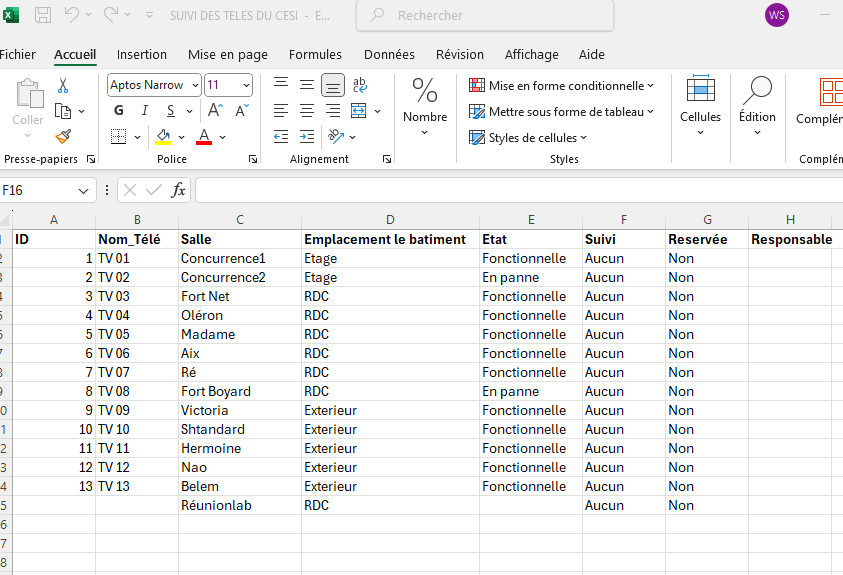
**Frontend**

* **Tkinter si local** ou **React/HTML/CSS** si interface web
* **Leaflet.js** ou **Google Maps API** pour la cartographie interactive

**Autres**

* **QR code sur chaque télé** pour les scanner et mettre à jour leur état facilement
* **Excel/CSV import/export** pour les administrateurs

# Exemple de base de données

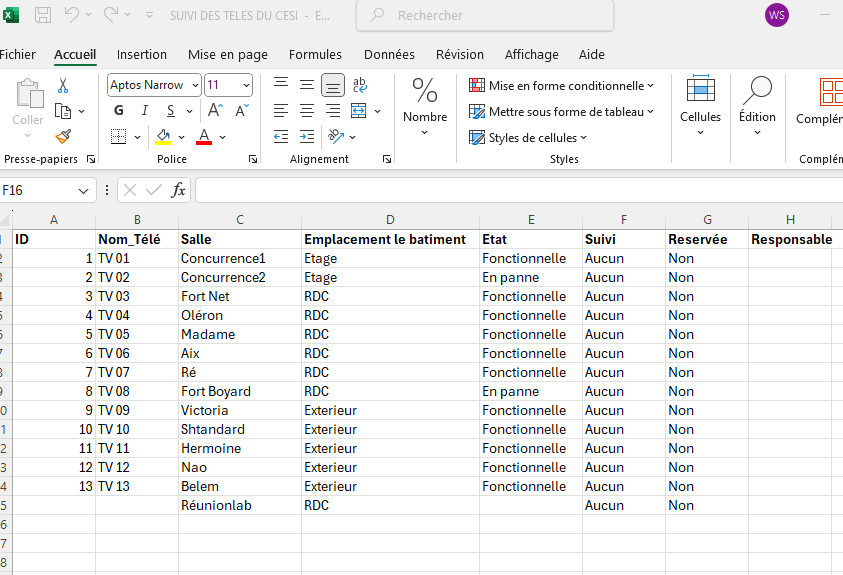


# Étapes du projet

1. Faire l'inventaire des télés existantes (Excel ou CSV)
2. Valider les moyens de localisation (Air Tags, GPS, etc.)
3. Créer une maquette (Figma ou même papier) de l’interface
4. Développer le module carto + planning
5. Ajouter la gestion d’état et l’édition
6. Test en condition réelle (1 bâtiment pour commencer)
7. Déploiement progressif

REALISATION

# Faire l'inventaire des télés existantes (Excel ou CSV)



**Option A — Interface locale avec Python (Tkinter + PIL)**

Avantages : simple à faire, pas besoin d’Internet  
Limites : pas super précis niveau position

On importe un plan d’étage au format image (.png, .jpg)  
On places manuellement les télés avec leurs coordonnées X, Y (par pixel)  
Les points sont cliquables, et un pop-up affiche les détails

Idéal pour un prototype **rapide** et si on n’a pas de plan 3D

**Option B — Interface web avec Leaflet.js ou Google Maps**

Avantages : carte dynamique, zoom, gestion facile de points  
Limites : plus long à mettre en place, nécessite navigateur et un peu de JavaScript

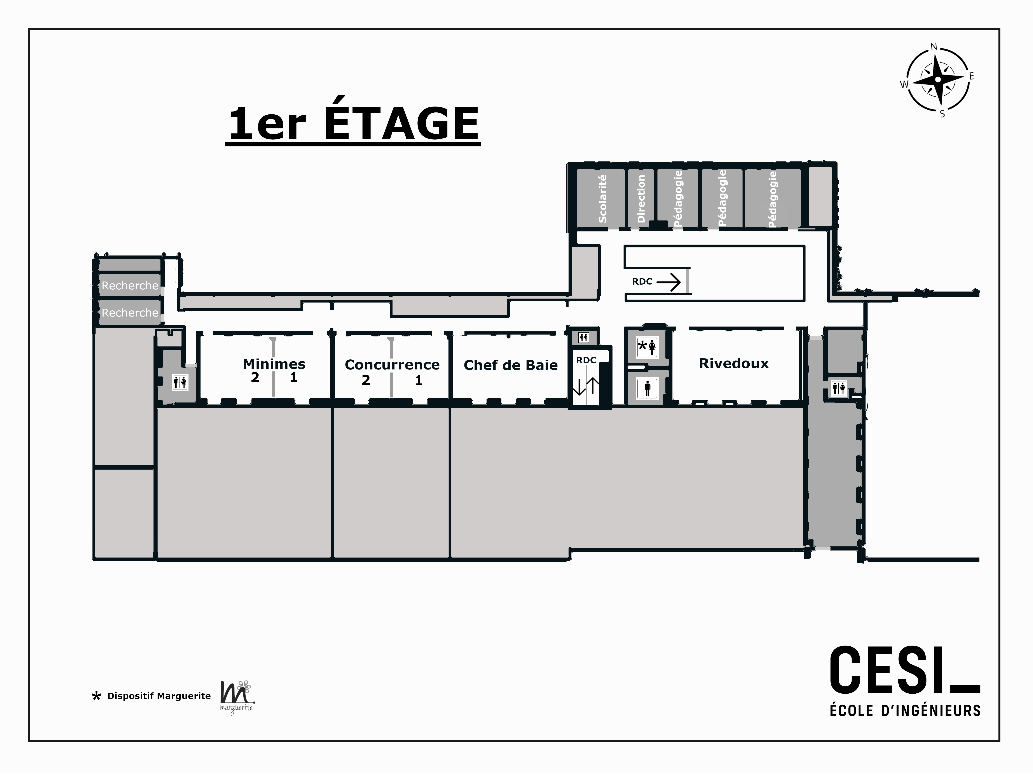
Les télés sont placées par coordonnées GPS (si disponibles)  
On peut créer une carte de bâtiment avec des couches personnalisées  
On peut intégrer un planning, filtres, etc.

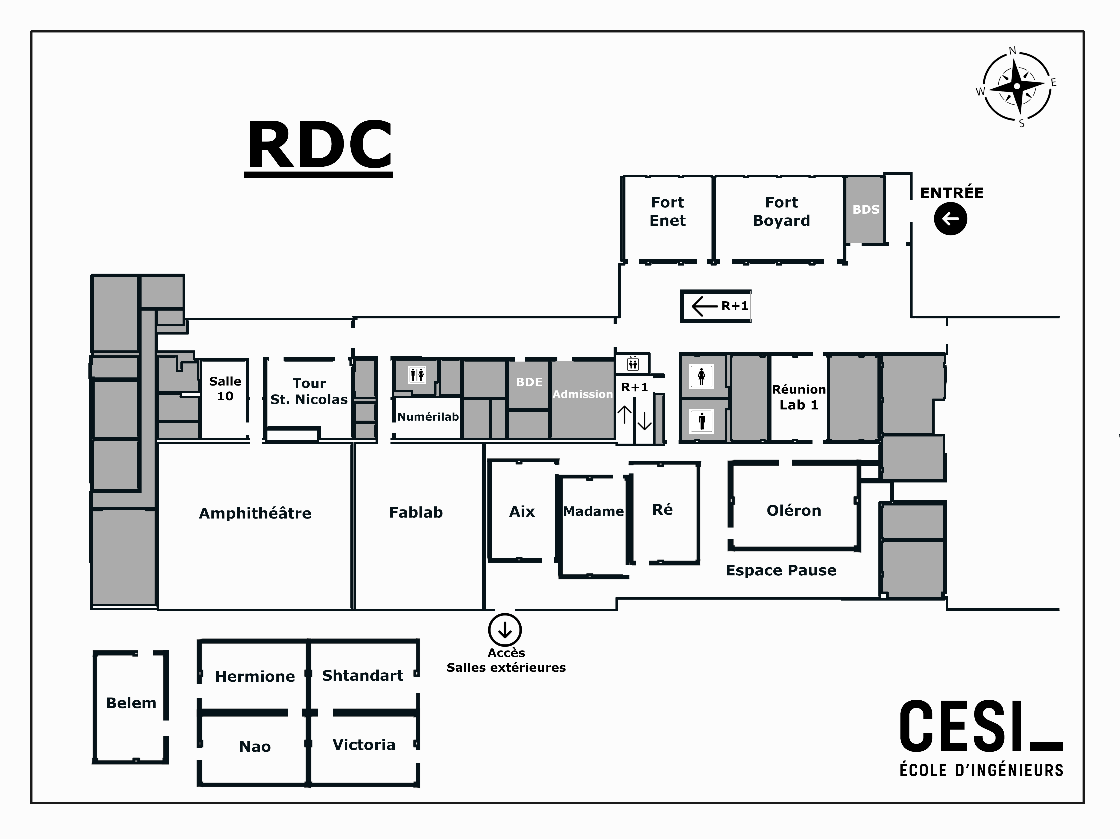
Idéal pour un outil **interne à long terme** ou si on a beaucoup de matériel réparti

**Option A — Interface locale avec Python (Tkinter + PIL)**

On va :

1. Afficher les deux plans comme fond





1. Placer des **points** représentant les télés
2. Cliquer sur un point pour **voir les infos de la télé**
3. Permettre ensuite de **modifier leur position**
4. Ajouter une télé par clic droit
5. Modifier l’état d’une télé

* Ajout d’une fenêtre pop-up (ou clic droit sur le point) pour modifier l’état (ex : Fonctionnelle*,* En panne*,* Réparée).
* Ces infos seront stockées dans une base (CSV ou SQLite selon l’option choisie).

1. Relier ça à un fichier une base SQLite(**SQLite** → plus solide, permet les mises à jour, les suppressions et les recherches complexes.)

Proposition :

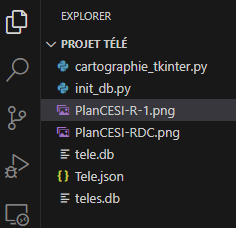
* tele.db(nom de la base de données) avec une table télés(id, nom, x, y, état, étage, bâtiment)
* Tu peux filtrer par étage/bâtiment, et faire des requêtes pour les stats ou l’état général.

1. Gérer plusieurs étages ou bâtiments
2. Pouvoir supprimer une télé

* Clic droit ou sélection → bouton Supprimer.
* Confirmation via messagebox.askyesno().
* Suppression dans la base de données puis rafraîchissement de l’interface.

1. Générer un QR code et lien pour donner accès à des multiples utilisateurs

Pour mettre tout ça en place on n’utilisera des script python qui mettra en place une interface tkinter avec toutes les fonctionnalités disponible. Il a une utilisation manuel ou les utilisateurs devrons eux même entrer les données concernant les tétés.



Avec :

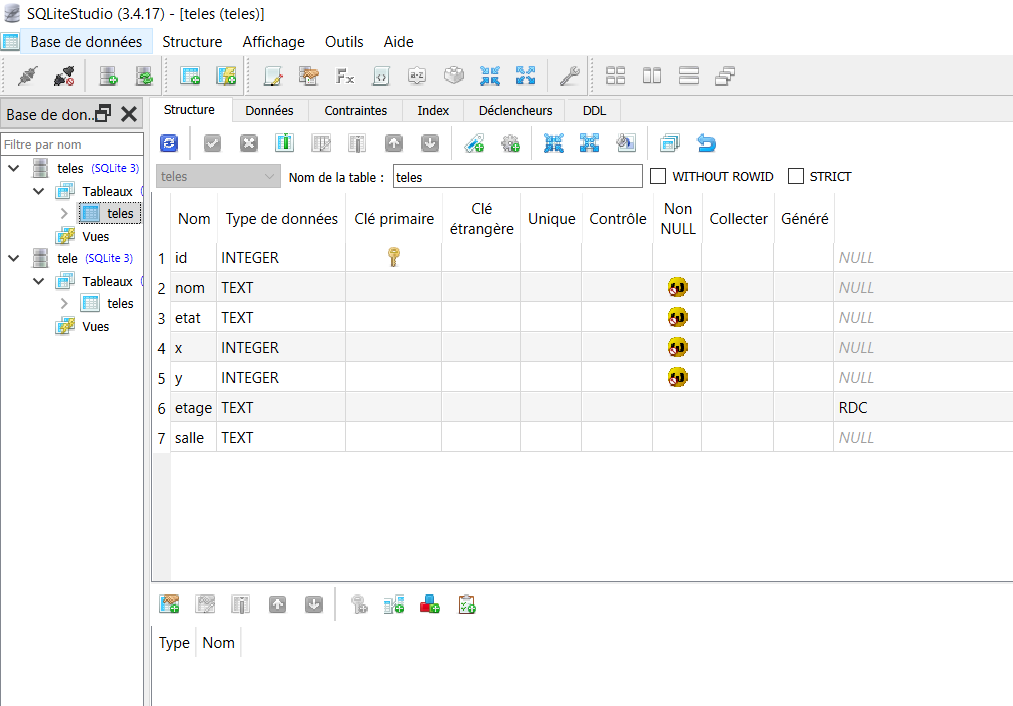
# Cartographie\_tkinker.py

L’interface en script python montrant les plans du bâtiment CESI

* Il affiche dynamiquement les télés sur un plan selon l’étage,
* gère l’ajout, la suppression, le déplacement et la modification d’état,
* sauvegarde les données dans une base SQLite,
* utilise un Tool tip propre,
* et gère deux étages avec un menu déroulant.

# Init\_db.py

La base de données en question



import sqlite3

conn = sqlite3.connect('teles.db')

cur = conn.cursor()

def creer\_table\_si\_absente(self):

    self.conn.execute("""

        CREATE TABLE IF NOT EXISTS teles (

            id INTEGER PRIMARY KEY AUTOINCREMENT,

            nom TEXT NOT NULL UNIQUE,

            etat TEXT NOT NULL,

            x INTEGER NOT NULL,

            y INTEGER NOT NULL,

            etage TEXT DEFAULT 'RDC',

            salle TEXT

        )

    """)

    self.conn.execute("""

        CREATE TABLE IF NOT EXISTS reservations (

            id INTEGER PRIMARY KEY AUTOINCREMENT,

            nom\_tele TEXT,

            date TEXT,

            action TEXT,

            FOREIGN KEY(nom\_tele) REFERENCES teles(nom)

        )

    """)

    self.conn.commit()

conn.commit()

conn.close()