# Dossier Technique du projet « Accès campus » - Partie PSW

## Table des matières



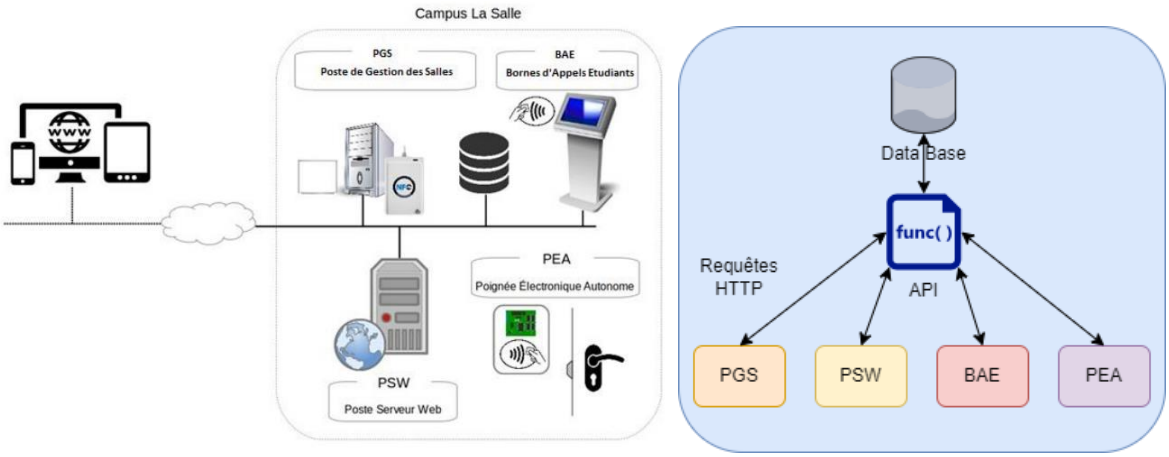
# I – Présentation du projet

## 1.1 - Contexte et objectifs

Le projet Campus Accès vise à améliorer la gestion et la sécurité des accès aux salles du Campus Saint Aubin La Salle. Grâce à un système automatisé basé sur des badges RFID, les étudiants et le personnel peuvent accéder aux salles de manière sécurisée tout en enregistrant leurs entrées et sorties. L'objectif est d'assurer un contrôle centralisé des accès, optimisé grâce au Poste Serveur Web (PSW), qui héberge la base de données PostgreSQL et l'API FastAPI pour traiter les demandes en temps réel.

## 1.2 - Description du système

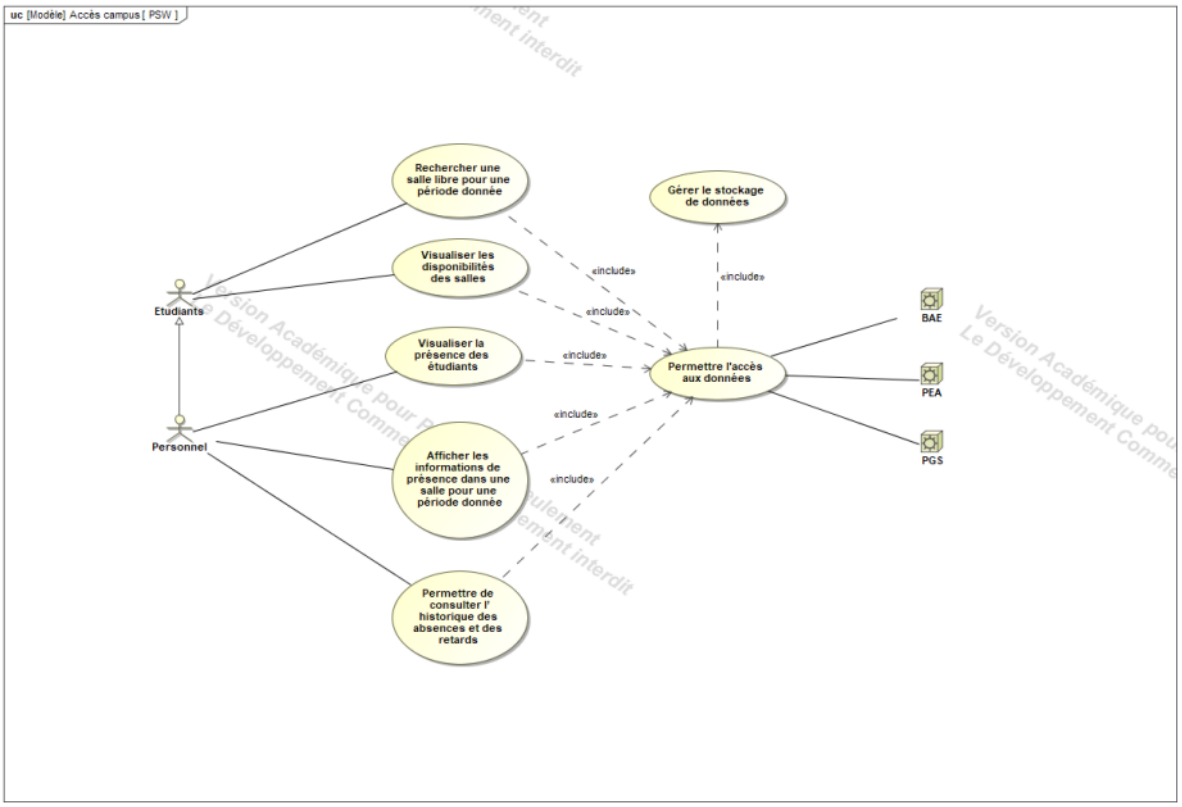
Le Poste Serveur Web (PSW) est au cœur du système, assurant la gestion centralisée des accès. Il héberge la base de données PostgreSQL, qui stocke les informations des utilisateurs, des badges et des historiques d'accès. L'API FastAPI, développée en Python, permet d'interagir avec la base et de fournir une interface accessible aux autres composants du projet. Les Poignées Électroniques Autonomes (PEA) communiquent avec le PSW pour vérifier les accès, tandis que le Poste de Gestion des Salles (PGS) permet d'administrer les réservations et les droits d'accès. L'ensemble fonctionne sur un serveur Debian 11, garantissant une exécution stable et sécurisée.




## 1.3 – Fonctionnement

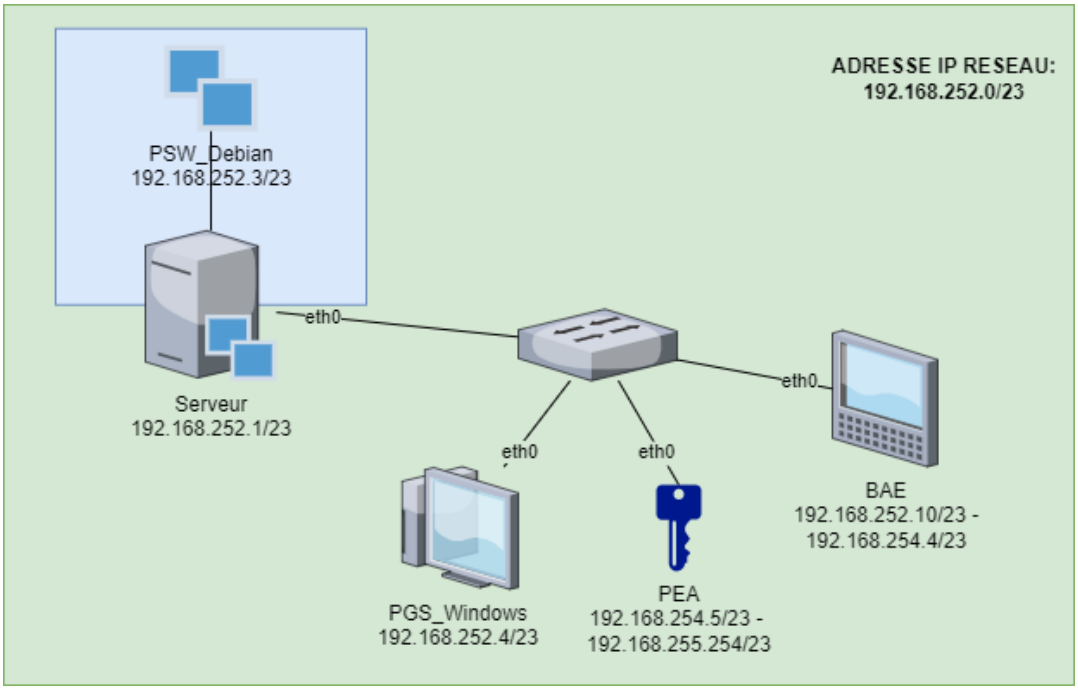
Lorsqu'un étudiant scanne son badge sur une PEA, l'API FastAPI envoie une requête à la base de données PostgreSQL, qui vérifie si l'utilisateur est autorisé. Si l'accès est validé, la porte s'ouvre et l'entrée est enregistrée. En cas de refus, une alerte est envoyée à l'administrateur. Ce dernier utilise le PGS pour modifier les permissions et gérer les réservations des salles. Toutes les interactions sont centralisées sur le PSW, qui stocke et analyse les données d'accès en temps

réel.




## 1.4 - Architecture du système

Le PSW repose sur une architecture en plusieurs couches, garantissant un traitement efficace des requêtes. L'interface utilisateur, accessible depuis un navigateur web, interagit avec l'API FastAPI, qui exécute les requêtes et communique avec la base de données PostgreSQL. Le serveur Debian 11, hébergeant l'ensemble, assure la disponibilité du service et la gestion des accès en continu. Ce modèle permet une réponse rapide et sécurisée aux demandes des PEA et du PGS.

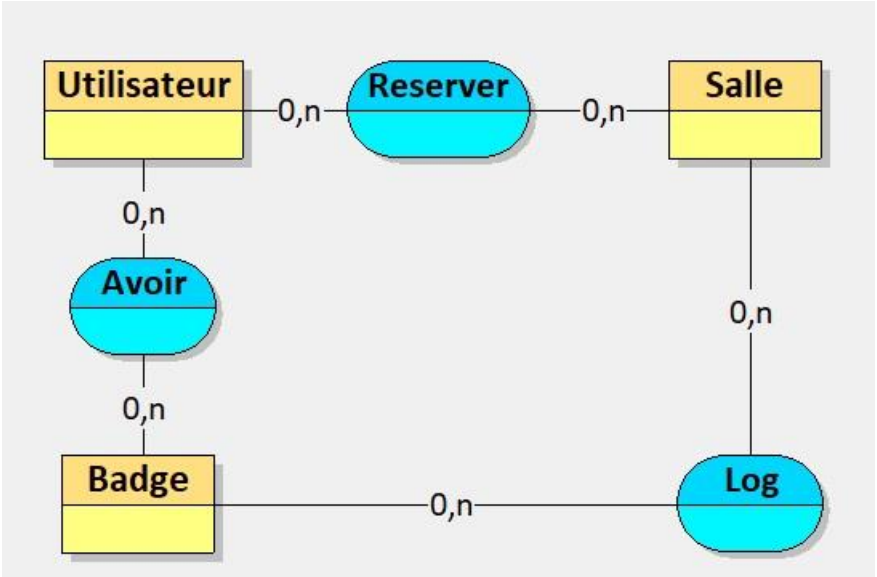


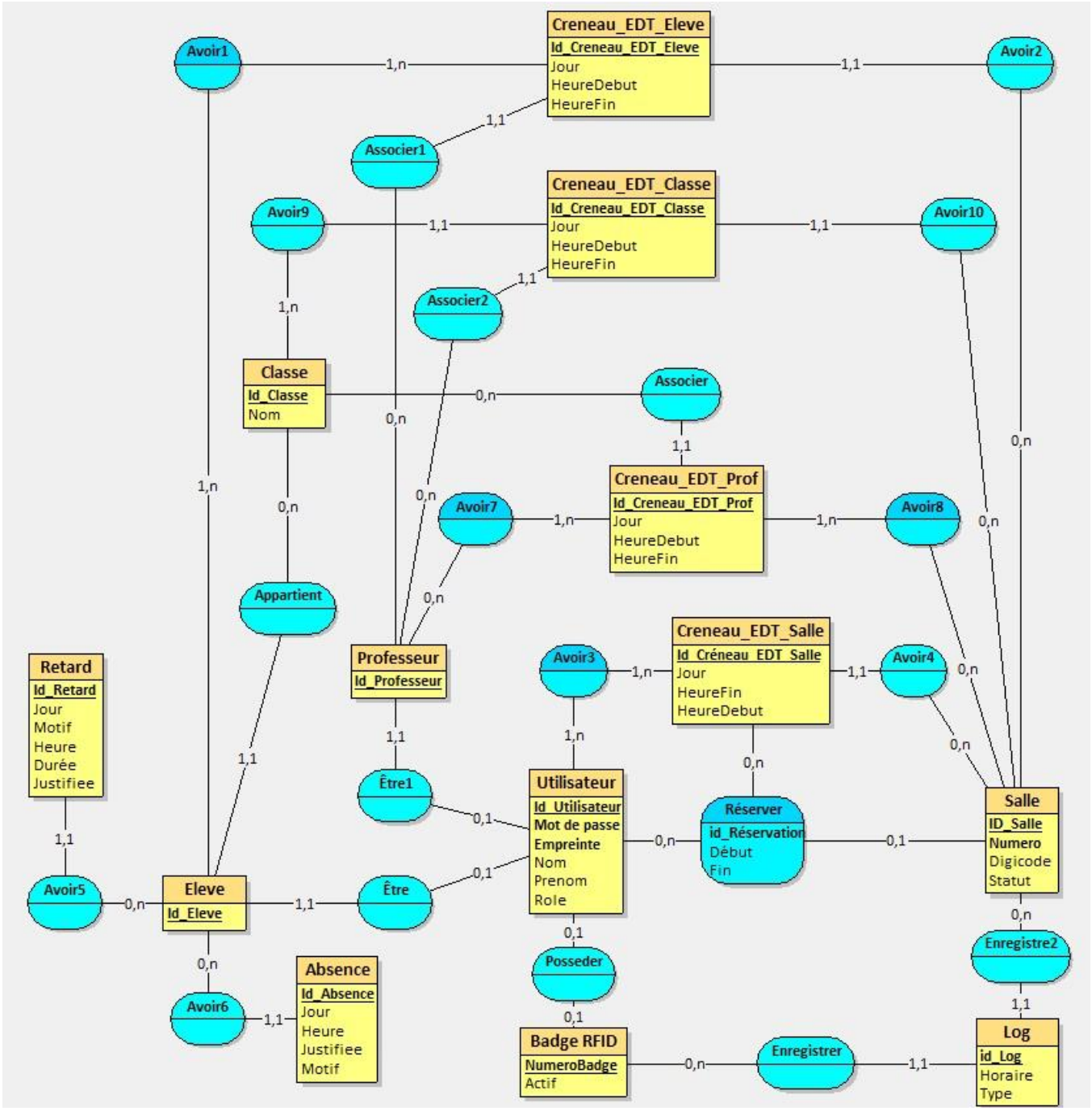

# II – Spécification techniques

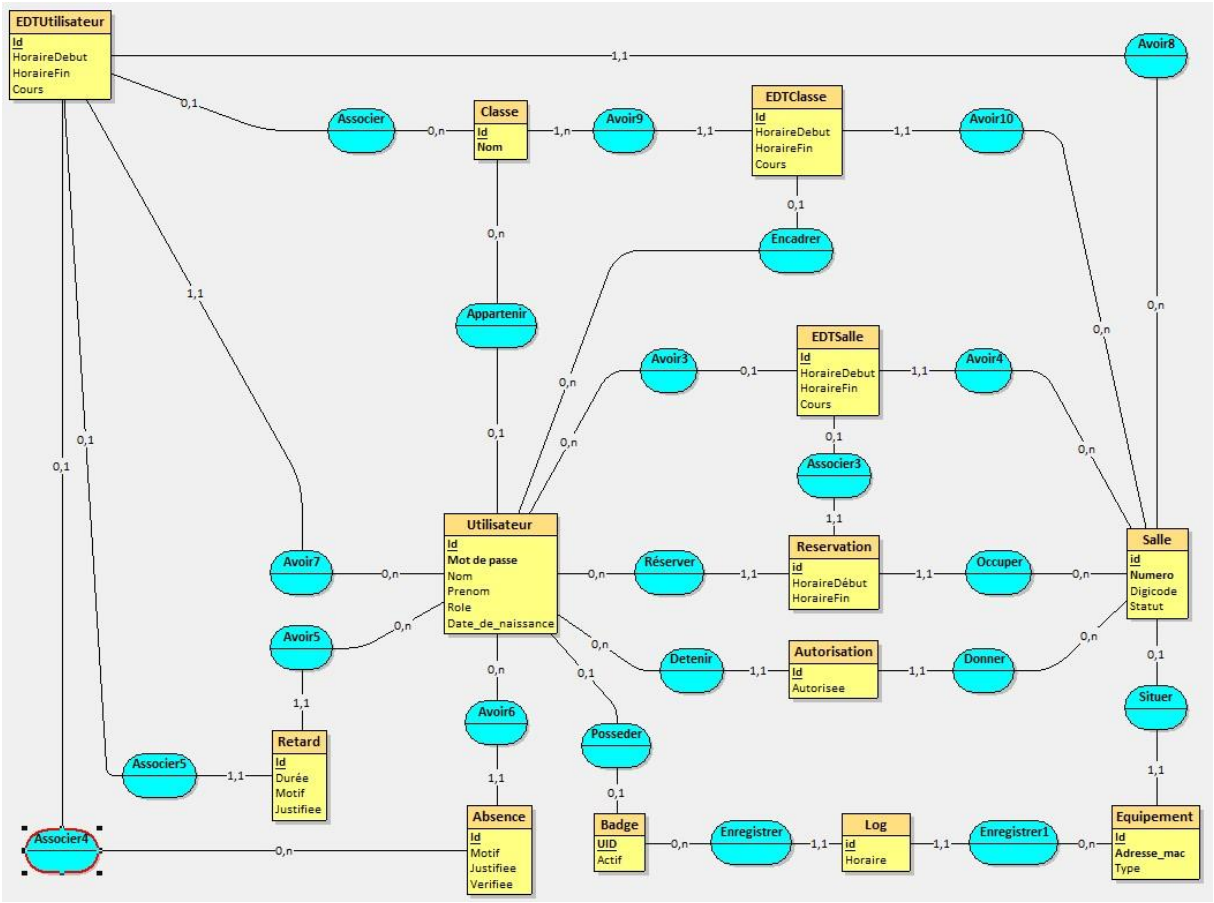
## 2.1 – Base de données

Le MCD définit les différentes entités et relations qui composent la base de données du PSW. Chaque utilisateur est relié au maximum à un badge, il peut sois être un élève, un enseignant, un personnel, un invité, ou alors un administrateur. L'accès aux salles est tracé grâce à la table Log qui enregistre les passages. Cette table sert aussi à notifier les absences et les retards en comparant le lieu et l'horaire avec la table Creneau_EDT_Elève. Ils sont ensuite enregistrés dans des tables à leur nom. Les différentes tables « Creneau_EDT » servent à l'affichage des emplois du temps sur le PSW. La base de données est donc relationnelle donc il est tout logique de partir sur du langage SQL.

Voici une évolution de la représentation du MCD de la base de données :




Creneau_EDT_Eleve
Id_Creneau_EDT_Eleve
Jour
HeureDebut
HeureFin
Avoir1
Avoir2
Associer1
Creneau_EDT_Classe
Id_Creneau_EDT_Classe
Jour
HeureDebut
HeureFin
Avoir9
Avoir10
Associer2
Classe
Id_Classe
Nom
Associer
Creneau_EDT_Prof
Id_Creneau_EDT_Prof
Jour
HeureDebut
HeureFin
Avoir7
Avoir8
Appartient
Professeur
Id_Professeur
Creneau_EDT_Salle
Id_Créneau_EDT_Salle
Jour
HeureFin
HeureDebut
Avoir3
Avoir4
Retard
Id_Retard
Jour
Motif
Heure
Durée
Justifiee
Être1
Utilisateur
Id_Utilisateur
Mot de passe
Empreinte
Nom
Prenom
Role
Réserver
id_Réservation
Début
Fin
Salle
ID_Salle
Numero
Digicode
Statut
Avoir5
Eleve
Id_Eleve
Être
Enregistre2
Avoir6
Absence
Id_Absence
Jour
Heure
Justifiee
Motif
Posseder
Badge RFID
NumeroBadge
Actif
Enregistrer
Log
id_Log
Horaire
Type




## 2.2 Contraintes et exigences

La base de données doit être capable de traiter plus de 2000 badges et 300 salles en garantissant une disponibilité 24/7. La base de données PostgreSQL doit être optimisée pour supporter un grand nombre de requêtes simultanées, avec un temps de réponse inférieur à 500 ms. La sécurité des données est une priorité, avec une gestion rigoureuse des permissions et un chiffrement des accès. L'ensemble du système doit fonctionner de manière fiable, sans interruption, pour assurer une gestion fluide et sécurisée des accès au campus.

## 2.3 Technologies utilisées

Le PSW est basé sur des technologies robustes et performantes. L'API backend est développée en Python avec FastAPI, garantissant une gestion rapide et asynchrone des requêtes. La base de données PostgreSQL, optimisée pour gérer un grand nombre d'entrées, assure un stockage sécurisé des informations d'accès. L'ensemble fonctionne sur Debian 11, un système d'exploitation reconnu pour sa stabilité. Grâce à l'intégration des badges RFID, la gestion des accès est fluide et automatisée, réduisant ainsi la charge administrative.

# III- Production

## 1.1 Premier incrément

Pour mon premier incrément je mettre en place ma base de données et permettre l'accès aux données pour les premiers incréments de mes collègues.

Je dois donc ajouter les tables EDTUtilisateur, Equipement, Utilisateur, Badge, Salle, Classe, Absence, Retard et Autorisation.

Voici donc le script SQL permettant l'ajout de ces tables dans PostgreSQL :

```
CREATE TYPE role AS ENUM ('Invite', 'Personnel', 'Eleve', 'Prof', 'Admin');

CREATE TYPE type AS ENUM ('BAE', 'PEA');

CREATE TABLE Salle(
   Id SMALLSERIAL,
   Numero CHAR(4) NOT NULL,
   Digicode CHAR(4),
   Statut BOOLEAN NOT NULL,
   PRIMARY KEY(Id),
   UNIQUE(Numero)
);

CREATE TABLE Classe(
   Id SMALLSERIAL,
   Nom VARCHAR(20) NOT NULL,
   PRIMARY KEY(id),
   UNIQUE(Nom)
);

CREATE TABLE Equipement(
   Id SMALLSERIAL,
   Adresse_mac CHAR(17) NOT NULL,
   Type type NOT NULL,
   Id_Salle SMALLINT NOT NULL,
   PRIMARY KEY(id),
   UNIQUE(Adresse_mac),
   FOREIGN KEY(Id_Salle) REFERENCES Salle(Id)
);

CREATE TABLE Utilisateur(
   Id SERIAL,
   Mot_de_passe CHAR(8),
   Nom VARCHAR(30),
   Prenom VARCHAR(30) NOT NULL,
   Role role NOT NULL,
   Date_de_naissance DATE,
   Id_Classe SMALLINT,
   PRIMARY KEY(id),
   UNIQUE(Mot_de_passe),
   FOREIGN KEY(Id_Classe) REFERENCES Classe(Id)
);
```

```
CREATE TABLE Badge(
   UID CHAR(8),
   Actif BOOLEAN NOT NULL,
   Id_Utilisateur INT,
   PRIMARY KEY(UID),
   UNIQUE(Id_Utilisateur),
   FOREIGN KEY(Id_Utilisateur) REFERENCES Utilisateur(Id)
);
```

```
CREATE TABLE EDTUtilisateur(
   Id SERIAL,
   HoraireDebut TIMESTAMP NOT NULL,
   HoraireFin TIMESTAMP NOT NULL,
   Cours VARCHAR(20),
   Id_Salle SMALLINT NOT NULL,
   Id_Classe SMALLINT,
   Id_Utilisateur INT NOT NULL,
   PRIMARY KEY(Id),
   FOREIGN KEY(Id_Salle) REFERENCES Salle(Id),
   FOREIGN KEY(Id_Classe) REFERENCES Classe(Id),
   FOREIGN KEY(Id_Utilisateur) REFERENCES Utilisateur(Id)
);

CREATE TABLE Absence(
   Id SERIAL,
   Motif VARCHAR(50),
   Justifiee BOOLEAN NOT NULL,
   Verifiee BOOLEAN NOT NULL,
   Id_Utilisateur INT NOT NULL,
   Id_EDTUtilisateur INT NOT NULL,
   PRIMARY KEY(Id),
   FOREIGN KEY(Id_Utilisateur) REFERENCES Utilisateur(Id),
   FOREIGN KEY(Id_EDTUtilisateur) REFERENCES EDTUtilisateur(Id)
);

CREATE TABLE Retard(
   Id SERIAL,
   Duree INT NOT NULL,
   Motif VARCHAR(50),
   Justifiee BOOLEAN NOT NULL,
   Id_Utilisateur INT NOT NULL,
   Id_EDTUtilisateur INT NOT NULL,
   PRIMARY KEY(Id),
   FOREIGN KEY(Id_Utilisateur) REFERENCES Utilisateur(Id),
   FOREIGN KEY(Id_EDTUtilisateur) REFERENCES EDTUtilisateur(Id)
);
```

```
CREATE TABLE Autorisation(
   Id SERIAL,
   Autorisee BOOLEAN NOT NULL,
   Id_Utilisateur INT NOT NULL,
   Id_Salle SMALLINT NOT NULL,
   PRIMARY KEY(Id),
   FOREIGN KEY(Id_Utilisateur) REFERENCES Utilisateur(Id),
   FOREIGN KEY(Id_Salle) REFERENCES Salle(Id)
);
```

Ensuite je code l'API avec FastAPI avec l'organisation suivante :

/API

- main.py
- schemas.py
- models.py
- database.py

    /routes

    - salle.py
    - classe.py
    - etc...