Dossier Technique du projet « Accès campus » - Partie gestion des données (PSW)

Table des matières

[I – Présentation 2](#_Toc194649127)

[1.1 - Contexte et objectifs 2](#_Toc194649128)

[1.2 - Description du système 2](#_Toc194649129)

[1.3 - Infrastructure réseau 3](#_Toc194649130)

[1.4 – Modèle conceptuel de données 4](#_Toc194649131)

[II – Incrément 1 5](#_Toc194649132)

[2.1 – Objectif 5](#_Toc194649133)

[2.2 – Planification 6](#_Toc194649134)

[2.3 – Mise en place PSW 6](#_Toc194649135)

[2.4 – Fonctionnement de l’API 7](#_Toc194649136)

# 

# I – Présentation

## 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).



Figure Campus Saint Aubin La Salle

Ma tâche personnelle dans ce projet de grande envergure est de centralisé les données et de permettre l’accès à ces données (*voir Figure 2*).

Une image contenant texte, Police, nombre, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Figure Mission étudiant 5

## 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, permet d’interagir avec la base. Les Poignées Électroniques Autonomes (PEA) communiquent avec le PSW pour vérifier les accès, les Bornes d’Appel Etudiant automatise l’appel dans un cours, tandis que le Poste de Gestion des Salles (PGS) permet d’administrer les réservations et les droits d’accès.

La communication avec l’API se fait grâce à différentes requêtes http soi GET, POST, PUT ou DELETE. L’API, elle, interroge la base de données pour ensuite renvoyer des réponses au format JSON (*voir Figure 3*).

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

Figure Communication entre les différents acteurs

## 1.3 - Infrastructure réseau

Notre infrastructure réseau (*voir figure 4*) est un seul et unique grand réseau d’adresse 192.168.248.0/21, ce que nous offre la possibilité de configurer 2046 hôtes différends. Le réseau est divisé par 3 VLANs :

* Publique (VLAN 10), pour tous les PCs du campus accessible pour tout type d’utilisateur. Un serveur DHCP est dédié pour ce VLAN qui peut attribuer 1583 adresses différentes.
* Badge (VLAN 20), pour toutes les BAEs et PEAs. Le serveur DHCP peut attribuer jusqu’à 666 adresses.
* Administration (VLAN 30), pour tous les serveurs et poste d’administration. Les adresses IP sont fixe pour les différends serveurs ou bien pour le poste de gestion des salles. Les autres adresses sont attribuées par le PSW.

Une image contenant texte, diagramme, capture d’écran, carte

Le contenu généré par l’IA peut être incorrect.

Figure Infrastructure réseau

Bien sur un pare-feu est présent sur le PSW, il bloque tous les ports et n’ouvre que ceux nécessaire. Les suivants donc : 22 (SSH), 53 (DNS), 80 (HTTP), 443 (HTTPS), 3000 (Node.js), 5432 (pgAdmin), 8000 (uvicorn).

Des manuels d’installation sont dispo sur notre GitHub.

## 1.4 – Modèle conceptuel de données

Les données sont divisées en 13 tables (*voir Figure 5*). On a une première série de table qui représente quelque chose de physique : Utilisateur, Badge, Salle, Classe, Salle et Equipement (soi BAE, soi PEA). Ensuite on va avoir les différents emploies du temps (EDTClasse, EDTUtilisateur, EDTSalle), qui ne vont en réalité pas représenter un emploie du temps complet mais plutôt un créneau ou un cours. En raccord avec EDTUtilisateur, on va avoir les retards et absences liés au cours. Enfin on a Autorisation pour les différentes autorisations d’accès qu’un utilisateur peut avoir et Log pour enregistrer tous les mouvements des utilisateurs à chaque fois qu’il badge une BAE ou une PEA.

Une image contenant diagramme, ligne, texte, Plan

Le contenu généré par l’IA peut être incorrect.

Figure Modèle conceptuel de données

Il m’a fallut plusieurs essaie pour atteindre ce modèle. Les autres versions du MCD sont disponibles sur notre GitHub.

# II – Incrément 1

## 2.1 – Objectif

L’objectif de ce premier incrément sera le même qu’au deuxième et au troisième : permettre l’accès aux données. La différence c’est qu’à chaque incrément mes collègues devront répondre à de nouveaux cas d’utilisation.

Pour ce premier incrément je vais devoir répondre à ces cas d’utilisations :

* PEA : Permettre l’accès à l’aide d’un badge RFID et vérifier l’autorisation d’accès.
* BAE : Permettre de signaler l’entrée dans une salle, afficher les informations liées à l’étudiant et mettre à jour les présences et les absences.
* PSW : Permettre de consulter l’historique des absences et des retards.
* PGS : Gérer les badges (Authentifier, créer, supprimer et modifier).

Les diagrammes de cas d’utilisations complets sont disponibles sur notre GitHub.

## 2.2 – Planification

Pour ce premier incrément, une durée de 4 semaine à été fixé pour toute l’équipe afin de réaliser une intégration au bout de cette durée, juste avant la revue 2.

Pour ma partie, je me suis fixé une semaine pour mettre en place le PSW, puis 2 pour mettre en place l’API. La dernière semaine a été désigné pour réaliser les test unitaires (*voir Figure 6*).

Une image contenant capture d’écran, ligne, nombre

Le contenu généré par l’IA peut être incorrect.

Figure Planification Incrément 1

## 2.3 – Mise en place PSW

Pour cette partie je ne suis pas seul à travailler dessus, je travaille avec Thomas Gasche qui lui doit réaliser un site web.

On a choisi de partir sur Debian 12 comme OS, pour sa stabilité. C’est plus adéquat que Ubuntu pour de 24/7. On installe ssh dès le début pour ensuite pouvoir travailler à distance dessus. Ensuite on met en place le pare-feu ufw.

J’installe PostgreSQL puis je créé ma base de données « campus\_db ». J’ai créé deux type ENUM (role, et type) et mes 13 tables (*exemple Figure 7*).

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Figure Exemple de création d'un type ENUM et d'une table en SQL

Le script SQL pour créer toute la base de données est disponible sur notre GitHub.

Ensuite j’ai installé venv pour me créer un environnement virtuel car sinon pip n’est pas disponible sur Debian 12. Je finis par installer toute les bibliothèques pythons dont j’ai besoins et micro, un éditeur de texte.

Pour la production je ne vais pas travailler sur le PSW directement mais plutôt sur une machine virtuelle. Je fais la même installation dessus.

Un guide d’installation et d’utilisation du PSW est disponible sur notre GitHub.

## 2.4 – Fonctionnement de l’API

L’API utilise les bibliothèques spécifiques suivantes (au-delà des basiques) :

* FastAPI, pour le fonctionnement général et la création des différentes routes.
* SQLAlchemy, pour l’enregistrement des données dans la base de données.
* Pydantic, pour la vérification de la réception des bons types de données lors de l’appel d’une requête http.