RAPPORT SAE 301

Contexte:

La SAE consiste au développement d'une application de gestion de musique et publicité sur des lecteurs. Nous avons le service marketing qui peut contrôler le planning des musiques , insérer et supprimer des fichiers puis quand le planning est enregistré, il est stocké dans une base de données du côté du serveur, ce serveur synchronise automatiquement tous les lecteurs avec la playlist récente avec un fichier m3u.

Le service commercial a la capacité de pouvoir mettre en pause la playlist et diffuser une urgence qui pourrait être une publicité ou une annonce quelconque. Quand une urgence est diffusée, le serveur enregistre l'état de la musique et le moment exact où elle est interrompue puis quand l'urgence prend fin, le serveur reprend la musique où elle s'est arrêtée.

Structure de la Base de données :

Pour la base de données, on a établi un Modèle-Entité-Association (MEA) regroupant fichier audio, playlist, utilisateurs et leur groupe, planning et lecteur avec leur localisation. On est parti de la table "Utilisateur" où ces derniers auront un prénom, nom, email, mot de passe et rôle.

Pour le rôle, on a décidé de créer une autre table "Groupe_Role" et de la joindre à "Utilisateur" pour ne pas avoir les rôles tapés en dur mais avec un identifiant, évitant la redondance.

Ensuite, on a créé la table "Ajoute" pour retracer l'ajout d'un fichier audio de la table "Fichier_audio" par un utilisateur avec sa date d'ajout. Pour différencier les fichiers audio entre les pubs et les musiques, on a créé une table "Type_contenu" reliée à "Fichier audio" puis chaque fichier audio à une playlist par la table "Playlist".

Pour savoir quel playlist joue dans quel lecteur on a relié "Playlist" à "Lecteur" par "joue_dans" puis on a relié "Lecteur" à "Localisation" pour une position précise du lecteur avec longitude, latitude et ville et à "Log" pour savoir qui s'est connecté au lecteur". Enfin, les plannings étant composé de playlist, on a relié "Playlist" à "Planning"

Modele MVC :

Pour contrôler les différents services données aux clients , nous avons choisi d'utiliser un modèle MVC (Model Vue Controller) en Python Flask, afin de séparer la logique métier de l'interface utilisateur, ce qui nous permettra de pouvoir accéder de manière sécurisée à la base de données, permettant aux différents services de ne pas recouvrir les données entre eux.

Le Service marketing a pour mission de contrôler le planning des musiques, pour cela, nous avons mis en place une architecture logicielle et une API optimisée. Du côté de la sécurité des fichiers, nous analysons d'abord le fichier, neutralisant tout tentative d'attaque par saut de répertoire principale, ce contrôle a une complexité linéaire O(n) négligeable et aussi renforcé en acceptant seulement les fichiers mp3 de moins de 50 Mo pour ne pas déborder la base de données.

Le Service Commercial a pour mission de diffuser des annonces à certains moment de la journée , pour répondre à ceci nous avons implémenté un algorithme basé sur une interruption, ce module repose sur un événement déclenché par la diffusion d'une urgence.

Ce choix de priorité absolue sur la musique se justifie sur le besoin d'une visibilité immédiate des pubs, l'algorithme se permet de mettre en pause la musique pour diffuser la publicité en évitant la superposition avec la musique. Cela repose sur une méthode POST, pour un envoi sécurisé des fichiers où chaque diffusion est écrite dans une historique ce qui permet de tracer chaque annonce faite.

Pour le Service Administrateur qui est connecté à la partie réseau, il a pour mission de gérer les utilisateurs, ainsi que la synchronisation des lecteurs.

Pour la partie MVC de ce service , comme ce projet est plutôt une application locale à l'entreprise, nous avons laissé le contrôle de créations et suppressions d'utilisateur aux administrateurs. Nous avons mis une certaine restriction aux administrateurs qui les empêchent de supprimer ou d'ajouter d'autres administrateurs , seulement un superAdmin aura l'autorisation de faire cela.

Partie Réseau :

Pour la Partie Réseau , le service Administrateur a accès au serveur hébergé en utilisant TailScale, qui permet de faciliter la mise en place d'un réseau privé sécurisé entre les machines, ainsi que SSH permettant d'avoir un accès direct aux machines (lecteurs). Le Serveur récupère la playlist de la journée sous format m3u a partir de la base de données, synchronisant automatiquement tous les lecteurs grâce à Rsync, si certains lecteurs ne sont pas synchronisés, l'administrateur peut le faire manuellement.

Afin de jouer de la musique à distance, nous utilisons MPD pour assurer la lecture audio sur les machines avec l'aide de MPC comme client de contrôle permettant le contrôle de la lecture automatique sans avoir besoin d'intervention manuelle de l'administrateur.

Quand une urgence est lancée par le service commercial avant de mettre en pause la musique , elle est conservée dans son état et nous vérifions localement dans le

serveur le moment où la musique est interrompue afin de pouvoir reprendre la lecture où elle a été interrompue.

Pour vérifier l'état des lecteurs, le serveur envoie avec une certaine intervalle de temps des pings et attendent une réponse en retour, si les machines ne répondent pas, le serveur les déclare KO et le serveur les forcera à se réactiver. Leur état s'actualise sur la page Admin qui peut ensuite synchroniser les lecteurs ou arrêter la playlist.