Documentation Technique - Securelim

1. Introduction

Ce document présente le projet Securelim de La Poste dans son aspect technique.

2. Architecture Générale

Le projet est composé de deux parties principales :

Une API développée avec FastAPI pour gérer les utilisateurs, employés et accès.
Un frontend en Vue.js pour l'interface utilisateur, la gestion des accès et des utilisateurs de l'application web.

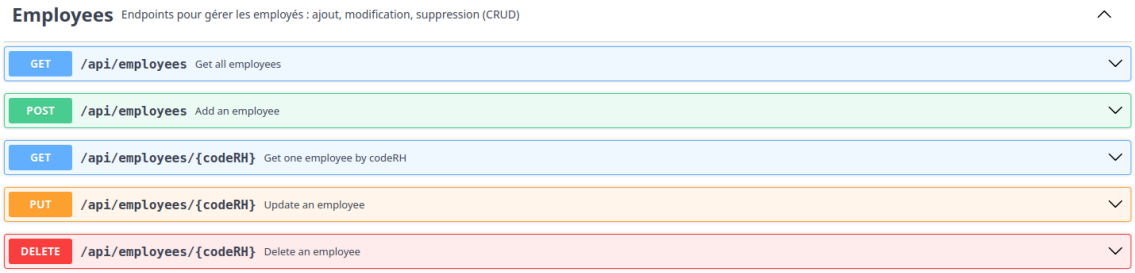
3. API FastAPI

L'API est construite avec FastAPI et utilise MariaDB comme base de données. Elle gère l'authentification JWT, la gestion des utilisateurs, employés et accès.

3.1 Endpoints principaux

Voici les principaux endpoints disponibles dans l'API :
POST /api/auth : Authentification des utilisateurs avec JWT.
GET /api/employees : Récupérer la liste des employés.
PUT /api/employees/{codeRH} : Modifier un employé.
DELETE /api/employees/{codeRH} : Supprimer un employé.

On a aussi la même chose pour les utilisateurs de l'application, les rôles etc..



4. Base de données

La base de données est une MariaDB contenant plusieurs tables :

USERS : Stocke les informations des utilisateurs.
EMPLOYEES : Contient les informations des employés et leurs accès.
ROLES : Gère les rôles des utilisateurs.

Il y a aussi des liaisons entre ces différentes tables

5. Frontend Vue.js

L'interface utilisateur est construite avec Vue.js et communique avec l'API via Axios. Elle permet d'afficher, créer, modifier, supprimer les employés et leurs accès.

La vue `Access.vue` permet de gérer les accès des employés avec les fonctionnalités suivantes :
Affichage de la liste des employés.
Modification des informations d'un employé.
Suppression d'un employé.

D’autres vues et composants sont aussi présentes pour gérer les utilisateurs ou même la page de login, aussi pour gérer la page d’erreur. De plus les requêtes sont directement appelé avec un lien dans le .env qui est directement modifiable pour les utilisateurs au cas où l’adresse IP est changé.

4. Problèmes rencontrés et solutions

Plusieurs problèmes ont été rencontrés et résolus au cours du développement :
Problème d'authentification JWT en production : Utilisation de `jwt` au lieu de `PyJWT` en production.
Erreur CORS en production : Ajout du middleware CORS dans `main.py`.
Problème de mise à jour des employés : Modification de la requête SQL pour éviter les problèmes de mise à jour.

6. Améliorations possibles

Voici quelques pistes d'amélioration :
Implémentation d'une gestion des erreurs plus robuste côté frontend et backend.

Sécurisation accrue des données, notamment en évitant d'exposer des informations sensibles.
Optimisation des requêtes SQL pour améliorer les performances.

7. Mise en place de l’application

Prérequis :

Environnement développement : Machines virtuelles et VS Code
Compétences requises : Javascript, Vue.js, Python, Sql

Installation de la Base de Données :

Choix de la BDD : MariaDB
Instruction d’installation : Voir Création de BDD Securelim

Installation de l’API :

Dépendances : L’installer via les requirements du fichier
Instructions d’installation : Voir Documentation Création sur Gitea
Test de l’API : Utiliser le Swagger de Fast API à {ipAPI}/api/docs

Installation du Projet :

Clonage du dépôt : Cloner le dépôt depuis le Gitea

Lancement :

Démarrage du serveur : npm install et npm run dev
Accès à l’application : localhost

8. Conclusion

Le projet Securelim est une application fonctionnelle permettant de gérer les accès d'employés via une API FastAPI et un frontend Vue.js. Divers problèmes ont été résolus, et des améliorations restent possibles pour renforcer la stabilité et la sécurité.