**Rapport Intermédiaire du développement Logiciel SNMP**

Notre logiciel de supervision se divise en 2 serveur Python et une base de donnée construite avec des dossiers et des fichiers json, un serveur qui en fonction des fichiers json présent dans la base de donnée va exécuter des requêtes SNMP et écrire les résultats des requête dans certains fichier json, donc un serveur de requête SNMP, ce server utilise un séquenceur qui va lancer la création de Thread pour effectuer chaque requête. Et un deuxième serveur Python Flask (serveur Web avec des pages HTML), qui va afficher à l’aide d’un site web les résultats présents dans les fichiers de la BDD, et permettant une administration de la supervision des équipements.

Cf (schéma de fonctionnement)

Les fonctionnalités demandées pour la réalisation de notre projet :

Partie Serveur de requête SNMP :

|  |  |
| --- | --- |
| Récupération de données d’équipement | Fonctionnel + intégré |
| Récupération de données des procédures | Fonctionnel + intégré |
| Fonctionnement procédure test-ping | Fonctionnel + intégré |
| Fonctionnement procédure SNMP | Non fonctionnel |
| Répétition procédure avec fréquence paramétrable | Fonctionnel + intégré |
| Inscription résultat procédures Ping | Fonctionnel + intégré |
| SNMP | non-fonctionnel |
| Fonctionnement procédure en multithreading | Fonctionnel + intégré |
| Fonction d’élimination des processus fils | Fonctionnel + intégré |
| Mise en place fonction d’enregistrement des évènements avec analyse | Partiellement fonctionnel |
| Ajout d’un seuil d’évènement | Possible évolution |
| Optimisation fonctionnement | En cours |

Partie Serveur Python Flask :

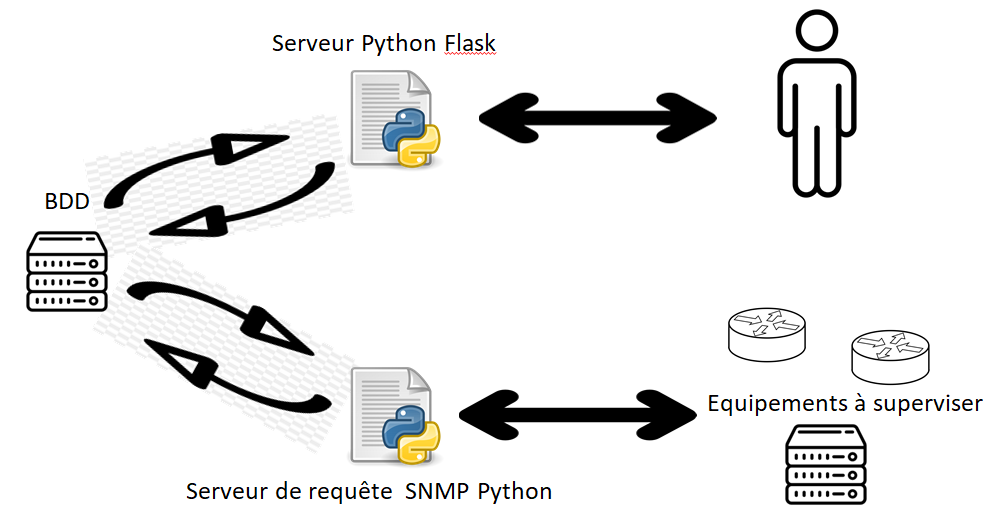
|  |  |
| --- | --- |
| Page d’ajout d’équipement a supervisé | Faite |
| Création des dossiers et fichiers pour ajouter un équipement a supervisé | Fait |
| Page de suppression d’équipement a supervisé | Fait |
| Suppression des dossiers et fichiers pour ajouter un équipement a supervisé | Fait |
| Création d’une page principale qui affiche les infos de base et qui rafraichi les donnée automatiquement | Fait |
| Page d’infos détaillé d’un équipement souhaité | Créer et en cours |
| Page d’ajout, de suppression, de modification d’un test via SNMP | Créer et en Cours |
| Page de log | Pas encore faite |

Note sur les difficultés et travail à fournir prochainement :

Pour la partie Serveur de requête, nous estimons que 60 % du programme est complet. En effet le plus compliqué qui est de faire fonctionner les procédures SNMP va nécessiter plus de temps que pour la procédure test-ping mais cela m’a permis de créer toute la base du fonctionnement. Ensuite viendra une étape d’optimisation du fonctionnement pour rendre le code plus évolutif et compréhensible.

Pour la partie Serveur Python Flask, je pense que 50% du travail est effectué car la partie affichage de graphe risque d’être compliquer. Les difficultés que j’ai rencontrées pour la première partie de la réalisation du serveur Web a été la recherche de la meilleure solution pour mettre à jour les résultats présents dans différents fichiers de résultat sur la page web sans la rafraichir, j’ai adopté de l’ajax et il est mis en place pour la page principale du logiciel.

Schéma de fonctionnement :



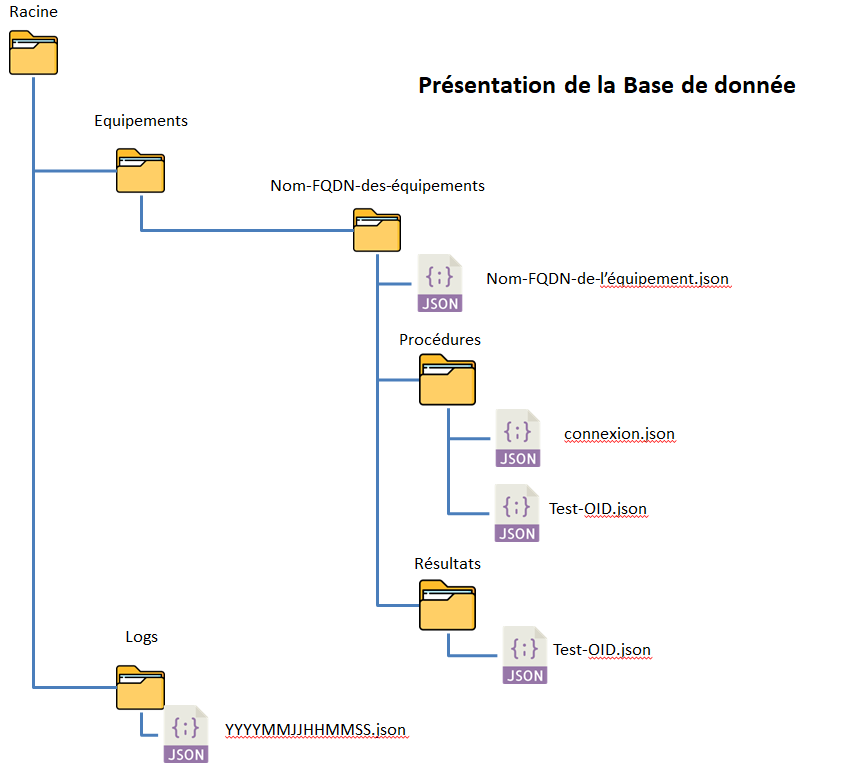
Interface Homme-Machine

Lecture écriture des fichiers json

Exécution de requête SNMP

Lecture écriture des fichiers json

BDD :



Exemple d’arborescence de base de données rempli pour les tests :

\---racine

+---equipements

| +---cisco.usmb.asa

| | | cisco.usmb.asa.json

| | |

| | +---procedures

| | | connexion.json

| | | test-ping.json

| | |

| | \---resultats

| | test-ping.json

| | test-ram.json

| |

| +---cisco.usmb.backup

| | | cisco.usmb.backup.json

| | |

| | +---procedures

| | | connexion.json

| | | test-ping.json

| | |

| | \---resultats

| | test-cpu.json

| | test-ping.json

| |

|

\---logs

20220930140330.json