

Rapport technique

Pour le datacenter d'une entreprise, il est impératif que le parc des machines soit sécurisé. Pour cela, nous allons mettre en oeuvre le principe du "Chaos monkey" qui consiste à simuler des pannes sur les différentes machines.

Les pannes sont stockées en base de données que nous manipulons grâce à la classe `DBManager.java` : classe pour l'utilisation de la BDD.

Nous avons réalisé une console de monitoring. Il s'agit d'une application web qui affiche les pannes en cours, ainsi qu'un certain nombre de statistiques sur le datacenter. Nous avons utilisé les fichiers suivants pour réaliser la console.

`Panne.java` : déclaration des objets de type panne.

`PanneDAO.java` : interface de `Panne`.

`PanneDAOImpl.java` : se connecte à la base de données et retourne la liste des pannes souhaitées en fonction des méthodes appelées (dernière minute, heure, jour ou mois).

`MainServlet.java` : charge la page `Monitoring.jsp` et instancie les attributs correspondant au nombre de pannes au cours de la dernière minute, heure, jour ou mois.

`Monitoring.jsp` : permet de générer la page HTML. Il récupère les attributs de la requête et les affiche au client. Chaque jsp côté client est lié à un servlet et le servlet effectue le traitement côté serveur.

`AjaxServlet.java` : permet de communiquer avec le serveur à l'aide d'une requête ajax qui retourne toutes les pannes dans un format JSON.

`WebSocket.java` : permet de communiquer avec le serveur à l'aide d'un websocket. Quand une insertion est faite dans la BDD, il envoie la nouvelle insertion au client.

Nous avons également réalisé un simulateur de pannes. Il s'agit d'une application web qui permet de générer à la demande des simulations de pannes.

`Simulateur.jsp` : permet à l'utilisateur de simuler la panne de son choix ou de simuler une panne aléatoire simple ou en rafale.

`SimulateurServlet.java` : récupère la requête de l'utilisateur et l'insère dans la BDD. Il recharge ensuite la page `Simulateur.jsp`.

`SimulateurServletRafale.java` : récupère la requête de l'utilisateur et insère autant de pannes que souhaitées dans la BDD. Il recharge ensuite la page `Simulateur.jsp`.

Nous avons été confrontés au problème suivant. Pour la génération d'une unique panne, le traitement se fait côté client en javascript. Ce n'est pas optimal car le traitement devrait se faire côté serveur, comme pour la génération aléatoire en rafale. Pour la génération aléatoire en rafale, à cause du fonctionnement du formulaire HTML et de l'envoi de requêtes en méthode POST au servlet, nous avons dû effectuer le traitement côté serveur en java.