Rapport SAE51

Rédigé par :

* Maxime VALLET
* Erwann MADEC
* Gabin PETITCOLAS

**Sommaire**

[I) Contexte 3](#_Toc188716330)

[A) Importance de la supervision des serveurs 3](#_Toc188716331)

[B) Avantages de la supervision 3](#_Toc188716332)

[C) Solution proposée 3](#_Toc188716333)

[D) Solutions similaires 3](#_Toc188716334)

[II) Solution développée 4](#_Toc188716335)

[A) Outils utilisés 4](#_Toc188716336)

[1) Environnement de développement 4](#_Toc188716337)

[2) Interface utilisateur 4](#_Toc188716338)

[3) Serveur 4](#_Toc188716339)

[4) Client 5](#_Toc188716340)

[5) Déploiement 5](#_Toc188716341)

[6) Tests de fonctionnement 5](#_Toc188716342)

[B) Estimation du travail à réaliser 6](#_Toc188716343)

[1) Méthode d’estimation du temps requis 6](#_Toc188716344)

[2) Estimation du temps requis 6](#_Toc188716345)

[C) Fonctionnalités du projet 7](#_Toc188716346)

[1) Fonctionnalités implémentés 7](#_Toc188716347)

[2) Fonctionnalités abandonnées 7](#_Toc188716348)

[III) Organisation 8](#_Toc188716349)

[A) Déroulement du projet 8](#_Toc188716350)

[B) Volume horaire final et respect des délais 8](#_Toc188716351)

[IV) Pistes d’amélioration 8](#_Toc188716352)

I) Contexte

A) Importance de la supervision des serveurs

Les serveurs occupent une place centrale en entreprise car ils hébergent des données et services critiques à son bon fonctionnement. Il est donc nécessaire de les surveiller afin d’assurer la meilleure qualité de service et d’améliorer leur durée de fonctionnement.

En effet, parmi ce qui peut être hébergé on retrouve :

* Active Directory : comprend tous les profils utilisateurs des salariés et peut aussi stocker une partie variable de leurs données.
* Outils utilisés en interne : système de pointage, serveurs VPN notamment pour faire du télétravail ou encore des applications métier
* Site Web de l’entreprise

B) Avantages de la supervision

La supervision permet de surveiller des métriques précises afin de prévenir de futures pannes ou bien d’être informé de problèmes en temps réel.

Par exemple, nous pouvons monitorer la quantité de stockage utilisée afin de pouvoir être prévenu en avance et mettre en place des mesures préventives afin que cela ne se transforme pas en un problème d’une plus grande envergure (impossibilité d’utiliser le serveur sans purger des fichiers manuellement dans ce cas).

Dans le cas énoncé précédemment, cela laisserait assez de temps pour faire le tri dans ce qui est stocké ou alors de rajouter du stockage.

C) Solution proposée

Nous avons proposé un outil avec lequel on interagit par le biais d’une page Web afin de le rendre facilement accessible.

Il se base sur une architecture Client/serveur, c’est-à-dire qu’il y a un serveur qui recueille et traite les données recueillies auprès des clients installés sur les machines surveillées.

Elle inclut un système d’authentification et de gestion des droits d’utilisateurs notamment pour fournir un accès granulaire aux machines qui sont supervisées.

D) Solutions similaires

Parmi les solutions existantes qui réalisent des tâches similaires, on retrouve des outils comme Zabbix ou OpenNMS.

La différence majeure entre l’outil que nous avons développé et ceux préexistant réside dans la façon dont il récupère les informations auprès des machines monitorées. En effet, nous utilisons Java (avec la bibliothèque Oshi) contrairement au protocole SNMP.

II) Solution développée

A) Outils utilisés

1) Environnement de développement

Afin de rendre le développement du projet le plus simple et accessible possible, nous avons opté pour des méthodes de déploiement variées et adaptées pour tous les membres du groupe.

Parmi ces méthodes de déploiement, on retrouve des VM contenant un environnement entièrement configuré, des instructions manuelles mais aussi un script qui installe le projet sans interaction nécessaire.

Voici le tableau contenu dans la documentation, permettant de guider la personne installant le projet vers la méthode d’installation la plus adaptée au contexte.

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

Description générée automatiquement

Figure 1 : Méthodes d'installation

2) Interface utilisateur

L’interface utilisateur a été développée avec les outils suivants : HTML, CSS et JavaScript.

3) Serveur

Le choix qui a été fait côté serveur, a été d’utiliser des outils que nous avions utilisés lors de TP ou projets précédents afin d’optimiser au mieux le temps requis pour réaliser ce projet.

Pour le serveur Web, nous avons utilisé Nginx, un programme très populaire et simple d’utilisation. Sa principale tâche est de distribuer le contenu statique du site Web (HTML, CSS, JS et images).

Afin de pouvoir interagir avec le serveur, nous avons utilisé Tomcat, un serveur Web qui peut faire tourner un projet Java. Son objectif est de faire tourner l’intégralité du serveur (tout est développé en Java) et de permettre de permettre des interactions par le biais du serveur Web grâce à des Servlets.

Afin de stocker les données (utilisateurs, machines…), nous avons utilisé un serveur PostgreSQL.

4) Client

Le client se présente sous la forme d’un projet Java qui récupère les métriques ainsi que la configuration du PC grâce à la librairie nommée Oshi.

Pour les ordinateurs Windows, le programme est lancé par un script PowerShell au démarrage de la machine et pour Linux, il est lancé par Systemd.

5) Déploiement

Pour le déploiement, nous avons créé des conteneurs Docker pour chaque serveur.

Docker est un outil qui permet de créer des conteneurs qui contiennent chacun leur environnement ainsi qu’un unique programme sur lequel il s’exécute. L’objectif de Docker est de créer un environnement léger et déployable n’importe où sans problème de compatibilité (à partir du moment où il est possible d’installer Docker, il est possible de faire tourner le projet).

Un avantage majeur est aussi que les conteneurs Docker peuvent être déployés sur GitHub, rendant le partage du projet très simple.

6) Tests de fonctionnement

Afin de rendre le projet le plus robuste, nous avons décidé de tester la majorité des classes Java du Backend en utilisant un Framework nommé JUnit.

L’intérêt de réaliser ces tests, est qu’ils permettent de trouver des bugs potentiels que ce soit au cours du développement des tests ou encore lors de la modification de classes par la suite.

Sur un projet Java de cette taille, cela s’est révélé très utile, notamment lors de modifications à réaliser sur des dizaines de classes, où l’on peut vite commettre des erreurs et introduire des bugs (mais cela ne se voit généralement pas tout de suite, car il y a trop de cas à tester).

Un autre avantage est aussi que cela permet de rendre le développement Frontend plus simple car le Backend contient moins de bugs. Cela rend la tâche plus simple pour le développeur Frontend car lorsqu’il y a un bug dans le Backend, il doit soit trouver le problème seul dans du code inconnu ou alors attendre que la personne qui a réalisé le code règle le problème, faisant prendre du retard au projet.

B) Estimation du travail à réaliser

1) Méthode d’estimation du temps requis

Ce projet étant un regroupement de notions abordées en cours, lors de projets personnels ou encore réalisés à l’IUT, cette estimation a principalement été basée sur ces précédentes expériences.

Le facteur principal a été la SAE-52 qui a représenté une base pour tout ce qui a touché au Backend et à l’authentification. Cela représentant une portion non négligeable de ce projet, même si nous avons rajouté et modifié une grande partie de ce qui a été réalisé dans cette précédente SAE.

Nous avons de même saisi M. Beysson afin de nous aider à estimer le volume horaire nécessaire à la réalisation de ce projet ainsi que pour discuter du développement de certaines fonctionnalités à l’aide d’un schéma qui décrit la façon dont nous souhaitions initialement réaliser ce projet.

Une image contenant capture d’écran, diagramme, circuit

Description générée automatiquement

Figure 2 : Schéma initial de l'organisation du projet

2) Estimation du temps requis

Nous étions initialement partis sur le plan suivant concernant le volume horaire avec possibilité d’étendre celui-ci en ajoutant des fonctionnalités comme la supervision d’équipements réseau au cas où nous serions un groupe de 4 ou que le projet avance plus vite que prévu.

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

Description générée automatiquement

Figure 3 : Estimation initiale du temps requis

Afin de bien répartir notre temps, nous avons aussi réalisé le diagramme de Gantt ci -dessous.

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

Description générée automatiquement

Figure 4 : Diagramme de Gantt initial

C) Fonctionnalités du projet

1) Fonctionnalités implémentées

a) Client et remontée de données

Le client est connecté en continu à un serveur TCP situé dans le projet Web hébergé par Tomcat. Cela permet de remonter des caractéristiques de la machine ainsi que ses données de suivi. Ces données sont ensuite rendues accessibles à l’interface Web par le biais de Servlets.

b) Système d’authentification

Nous avons mis en place un système d’authentification flexible avec des droits d’utilisateur distincts. Ces droits définissent précisément les ressources accessibles, qu’il s’agisse de postes de travail, de servlets ou de pages web. Toutes ces informations sont stockées dans une base de données, facilitant les modifications de manière simple et claire.

Ce système s’étend également aux postes de travail, offrant aux administrateurs une gestion fine des accès. Pour chaque PC, ils peuvent attribuer ou restreindre l’accès aux utilisateurs via une interface intuitive, permettant des modifications en un seul clic.

c) Contrôle des ordinateurs

Il est possible d’éteindre, de redémarrer et de mettre à jour les PC qui ont un client d’installé et connecté au serveur.

d) Interface Web

Nous avons développé des pages web intégrant les fonctionnalités présentées précédemment. Ces pages ont été conçues de manière modulaire afin de faciliter le développement, améliorer la clarté du code et gérer efficacement les droits utilisateurs. La plupart des fonctionnalités et ressources communes sont regroupées dans des scripts partagés, complétés par des scripts spécifiques à chaque page.

Nous avons également optimisé l’interaction avec les ressources pour garantir une expérience utilisateur fluide. L’interface s’adapte dynamiquement aux permissions de l’utilisateur :

* Une redirection automatique l’empêche d’accéder aux pages non autorisées, sans toutefois exposer de données sensibles, car toute requête vers le serveur exige une authentification appropriée.
* Une barre de navigation dynamique affiche uniquement les pages accessibles à l’utilisateur.
* Certains boutons sont désactivés lorsque l’action est impossible (ex. : éteindre un PC hors ligne ou retirer des droits d’accès à un PC pour un administrateur).

2) Fonctionnalités abandonnées

a) Recherche des clients sur le réseau

Cette fonctionnalité a été remplacée car une approche plus avantageuse a été trouvée. Nous avons décidé que le client serait celui qui initie la connexion vers le serveur, notamment pour éviter des soucis avec un potentiel NAT et car cela est bien plus facile à réaliser.

En effet, lorsqu’une machine est située derrière un NAT, son adresse IP est privée, ce qui fait que nous n’aurions pas pu monitorer de pc en dehors du réseau local. Afin de contrer cela, le client ouvre une communication définitive avec le protocole TCP en direction du serveur. Par conséquent, le serveur peut joindre le client car le NAT redirigera automatiquement les paquets car le client à une connexion ouverte vers le serveur.

III) Organisation

A) Déroulement du projet

Pour l’organisation du projet nous avons utilisées les outils suivants :

* **GitHub** pour le versionnage du code et la collaboration.
* **Discord** pour les communications quotidiennes et les échanges de fichiers.

Ensuite les rôles ont été répartis selon les compétences de chaque membre :

* **Maxime VALLET** : Intégration et déploiement du projet (Nginx, Tomcat, PostgreSQL, création des VM et des conteneurs Docker), développement du Backend et participation sur le Frontend.
* **Erwann MADEC** : Développement du serveur TCP, du client et supervision des métriques via Oshi.
* **Gabin PETITCOLAS** : Conception de l’interface utilisateur (HTML/CSS/JS).

Les étapes clés ont inclus :

* La mise en place de l’infrastructure Docker pour le déploiement du serveur.
* L’intégration et amélioration d’un système rudimentaire d’authentification inspiré de la SAE-52.
* Le contrôle des PC à distance (MAJ, Redémarrer, éteindre).
* La récupération ainsi que l’affichage des métriques

Au cours du projets nous avons rencontré quelques difficultés, notamment la gestion des connexions client-serveur derrière un NAT, résolue en inversant le flux de communication. Ainsi que le traitement des données dynamiques.

B) Volume horaire final et respect des délais

Les écarts qui ont eu lieu s’expliquent par :

* La complexité imprévue de l’intégration de la bibliothèque Oshi pour la collecte de métriques.
* Des ajustements post-tests pour améliorer la stabilité du serveur TCP pour le contrôle à distance.
* L’ajout de fonctionnalités comme des scripts de déploiement, les conteneurs Docker ou encore un système de logs.

Malgré cela, la réutilisation de composants de la SAE-52 ont permis de gagner du temps sur le développement.

**Tableau récapitulatif du volume horaire :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tache | Heures prévues | Maxime | Erwann | Gabin | Total | Écart |
| Développement backend | 50h | 40h | 40h | 0h | 90h | 40h |
| Interface | 35h | 15h | 0h | 45h | 60h | 25h |
| Déploiement et tests | 35h | 50h | 10h | 0h | 60h | 25h |
| Document et débogage | 30h | 20h | 5h | 5 | 30h | 0h |
| Total | 150h | 125h | 55h | 50h | 240h | 90h |

IV) Pistes d’amélioration

* Utiliser une clé SSL certifiée par une vraie autorité de certification reconnue par les navigateurs. Dans notre cas nous utilisons un certificat auto-signé. Ce certificat n’est donc pas reconnu par les navigateurs, ce qui nous oblige à l’ajouter manuellement aux appareils qui se connectent au site (valider pour Nginx et Tomcat). Une solution serait d’utiliser Let’s encrypt mais dans le cadre de cette SAE, cela n’est pas viable à cause du FAI que certains d’entre-nous utilisent le réseau Eduroam à la maison. En effet, pour obtenir un certificat Let’s Encrypt il faut que le serveur soit accessible sur internet et que nous possédions un nom de domaine associé à l’IP de ce serveur (impossible de faire de la redirection de port sur Eduroam).
* Utiliser SNMP pour la surveillance des machines car les utilisateurs pourraient choisir uniquement les métriques qu’ils voudraient surveiller comme la charge CPU. Si nous surveillions beaucoup de machines, le protocole SNMP serait également mieux car le protocole est prévu pour des milliers de machines or nos requêtes https avec java pourraient vite surcharger le réseau.
* Ajouter la fonctionnalité pour surveiller des appareils réseaux pour avoir une solution vraiment complète en termes de surveillance.