Création d’une G.M.A.O répondant au besoin d’une PME

SAÉ 5.02 - Piloter un projet informatique



Réalisé par : Maxime Vallet, Ishac Hamdani, Valentin Millot

Table des matières

[1. Introduction 3](#_heading=h.gjdgxs)

[2. Cahier des charges 5](#_heading=h.30j0zll)

[3. Les différents composants du projet 7](#_heading=h.n45hsi5xexz1)

[Les différents éléments composant le projet : 7](#_heading=h.633phlab8nz7)

[Répartition des tâches : 8](#_heading=h.z1y5uk2e2w1m)

[4. Page Login, mise en place du système de token et création d’un script d’automatisation 9](#_heading=h.gbj1em5qij3j)

[a. Qu'est-ce qu'un Token ? 9](#_heading=h.deh11s6fj5gk)

[b. Types de Tokens 9](#_heading=h.j89ih75w7nuw)

[c. Sécurité des Tokens 9](#_heading=h.oqjep0nk2zye)

[Fonctionnalités du Code gérant les Tokens : 10](#_heading=h.oollya7i6dvd)

[Fonctionnalités des Servlets : 11](#_heading=h.87hxys6kxcdl)

[Voici les différentes fonctionnalités du Script : 11](#_heading=h.f161be2dujnu)

[5. Page de gestion des équipements et gestion des utilisateurs 13](#_heading=h.cnd0q1we6k6l)

[Fonctionnalités du Code gérant les équipements : 13](#_heading=h.w8cal0bicmnq)

[Implémentation des servlets dans NetBeans 14](#_heading=h.pcpr1cssfabf)

[6. Page des tickets 15](#_heading=h.j7j75tvqeuy6)

[7. Difficultés rencontrées 16](#_heading=h.fstvttf20g4w)

[Création de la machine virtuelle (VM) 16](#_heading=h.fkc6x4lssbhm)

[Création d’un système d’authentification (Fullstack) 17](#_heading=h.vxmj1hhjrsc6)

[Difficultés particulières rencontrées (Valentin Millot) 19](#_heading=h.bn6ewm12iu40)

[Résolution des dysfonctionnements et limites du projet 19](#_heading=h.jp60ltyhseni)

[8. Conclusion 20](#_heading=h.2rxg76erexwf)

# Introduction

La Gestion de Maintenance Assistée par Ordinateur (G.M.A.O) est un outil essentiel pour optimiser la gestion des opérations de maintenance au sein d'une entreprise, en particulier pour une PME et ETI / GE. Pour les techniciens réseaux, l'implantation d'une GMAO apporte de nombreux avantages, en facilitant la supervision et l'organisation de la maintenance des infrastructures informatiques et réseau.

Voici la liste des fonctions d’une GMAO conçue pour les techniciens réseau dans une PME :

* Gestion centralisée des équipements : La GMAO permet de recenser et suivre l'état de l'ensemble des équipements réseau (routeurs, switchs, serveurs, etc). Chaque appareil est répertorié avec ses spécifications, son historique de maintenance et ses performances.
* Planification de la maintenance préventive : Grâce à une GMAO, il est possible de programmer des interventions régulières pour éviter les pannes imprévues. Cela permet d'établir un calendrier précis de maintenance préventive, en fonction de la durée de vie des équipements et des recommandations des fabricants.
* Réduction des temps d'arrêt (downtime) : En suivant et en anticipant les besoins de maintenance via la GMAO, les techniciens réseaux peuvent prévenir les pannes, ce qui diminue les interruptions de service. Moins de temps d'arrêt se traduit par une productivité accrue et une meilleure satisfaction client.
* Suivi des interventions : Chaque intervention technique, qu'elle soit corrective ou préventive, est enregistrée dans la GMAO. Cela permet de garder une trace des actions effectuées, facilitant ainsi les audits, et permettant une analyse rétrospective en cas de dysfonctionnement récurrent.
* Gestion des stocks et des pièces détachées : La GMAO permet de suivre les niveaux de stock des pièces de rechange (câbles, composants de réseau, etc). Les techniciens peuvent rapidement savoir ce qui est disponible et éviter les retards dus à une pénurie de matériel.
* Amélioration de la communication et collaboration : Avec un système GMAO, les techniciens et les autres membres de l'équipe (comme les gestionnaires ou les autres départements) peuvent accéder en temps réel aux informations sur l'état des équipements et les interventions en cours. Cela améliore la coordination et permet une meilleure prise de décision.
* Analyse des coûts et optimisation des ressources : La GMAO permet d’analyser les coûts de maintenance et de repérer les sources de gaspillage ou d’inefficacité. Ainsi, les techniciens peuvent mieux allouer les ressources et optimiser l'utilisation des équipements réseau

.

Dans une PME, la GMAO devient un outil stratégique pour les techniciens réseaux, leur permettant de mieux gérer les infrastructures tout en réduisant les coûts et en optimisant les performances. En automatisant et en centralisant la gestion des tâches de maintenance, elle permet aux équipes techniques de se concentrer sur des missions à plus forte valeur ajoutée, tout en garantissant la fiabilité des réseaux et systèmes.

# Cahier des charges

Pour la création de notre GMAO (Gestion de Maintenance Assistée par Ordinateur) à destination des techniciens réseau d'une PME, nous avons élaboré un cahier des charges adapté. Ce dernier répond aux exigences définies lors de l’introduction du projet. L’objectif est de développer un projet réalisable en environ 50 heures de travail.

Objectifs du projet :

1. Mettre en place une GMAO sous forme de site web accessible uniquement en local au sein de l’entreprise.
2. Utiliser des technologies simples, comme HTML/CSS/JS, pour l'interface utilisateur.
3. Sécuriser l'accès au site avec un système d'authentification.
4. Créer une structure adaptée aux besoins des différents profils d'utilisateurs : secrétaire, technicien et administrateur.

**Étapes et contraintes du projet :**

*Accès et sécurité :*

* La GMAO sera un site web local accessible uniquement depuis le réseau interne de l’entreprise.
* Le site inclut un système d'authentification sécurisé, avec des options de connexion pour chaque utilisateur.

*Trois profils d'utilisateur seront mis en place :*

* Secrétaire : Accès limité à la page « Demandes techniques » pour soumettre des requêtes concernant des besoins ou problèmes réseau spécifiques.
* Techniciens : Accès à plusieurs pages, dont :
* Une page dédiée à l'inventaire où ils pourront ajouter et gérer les équipements réseaux (ordinateurs, switchs, routeurs, points d’accès, câbles, etc.).
* Possibilité de réaliser des recherches avancées et de trier les équipements selon divers critères.
* Option d’export des équipements sélectionnés au format Excel, avec la possibilité d’imprimer des listes.
* Une page pour consulter et traiter les demandes des autres départements, concernant généralement des problèmes mineurs ou des pannes.
* Une page avec un planning des maintenances à venir, incluant la planification des réparations et interventions sur les équipements.
* Administrateur : Accès complet avec des droits étendus, lui permettant de modifier ou supprimer des équipements, interventions, demandes, et autres données du système.

*Base de données :*

* Le système devra être connecté à une base de données (PostgreSQL) hébergée localement sur un serveur interne, comme le site lui-même.
* La base de données devra être sécurisée et accessible uniquement par les administrateurs et utilisateurs autorisés.

*Contraintes techniques :*

* Langage de programmation : Le site sera développé en HTML pour l'interface utilisateur. Un langage côté serveur comme PHP, Python ou Java peut être utilisé pour l’interaction avec la base de données.
* Système d'authentification : L'authentification devra être sécurisée (via HTTPS et mots de passe hashés).
* Sécurité : Les accès aux données seront restreints selon les profils utilisateurs, et des sauvegardes régulières devront être mises en place pour assurer la continuité de service en cas de panne.

Ce cahier des charges définit les grandes lignes du projet de GMAO pour les techniciens réseaux d'une PME, en se concentrant sur la sécurité, l'organisation de l'inventaire, et la gestion des demandes techniques pour optimiser la gestion des équipements et interventions.

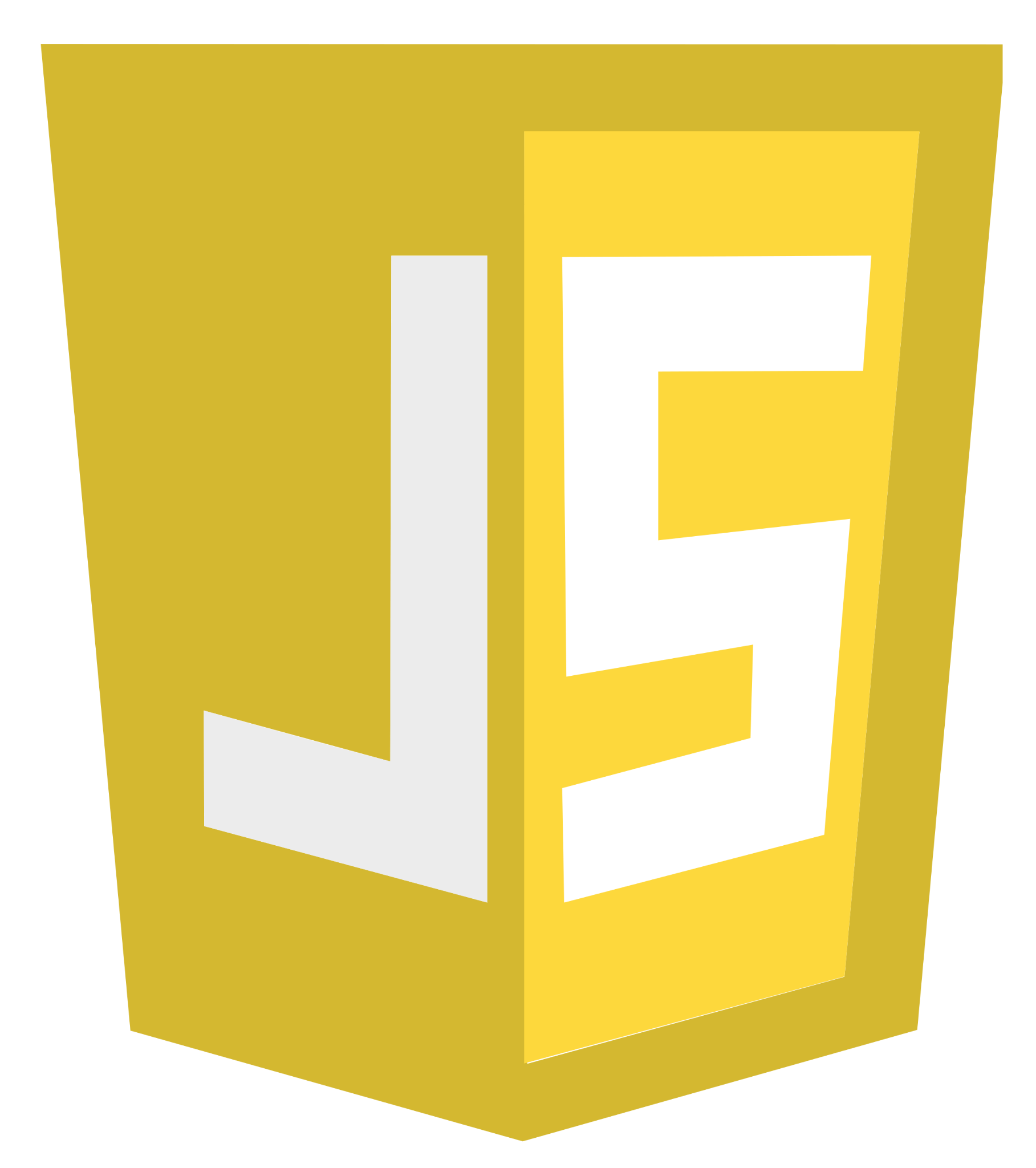
# Les différents composants du projet

### **Les différents éléments composant le projet :**

|  |  |  |
| --- | --- | --- |
| **Outil/Technologie** | **Description** | **Utilisation dans le projet** |
| **PostgreSQL** | Système de gestion de base de données relationnelle et objet, open source sous licence BSD, reconnu pour sa richesse fonctionnelle et sa conformité aux standards SQL. | Stockage des données pour l'application, gestion des attributs des équipements. |
| **Apache Tomcat** | Serveur d’applications web open source conçu pour exécuter des applications Java, développé par la fondation Apache. | Hébergement et déploiement des servlets Java. |
| **NetBeans** | Environnement de développement intégré (IDE) open source, supportant Java et d'autres langages (C, C++, JavaScript, etc.), utilisé principalement pour gérer les servlets Java dans le cadre de la GMAO. | Développement de la partie servlets Java de la gestion de maintenance assistée par ordinateur. |
| **Ubuntu** | Distribution Linux open source basée sur le noyau Linux, gratuite et modifiable, lancée en 2004 par Canonical Ltd, reposant sur Debian. | Hébergement des services Apache Tomcat et de la base de données PostgreSQL. |
| **Visual Studio** | IDE créée par Microsoft, supportant plusieurs langages (C#, VB.NET, C++, etc.), permettant le développement d'applications sur diverses plateformes. | Utilisé avec GitHub pour intégrer les éléments du site, les servlets et la base de données PostgreSQL dans un environnement collaboratif. |
| **GitHub** | Plateforme de gestion de versions et de collaboration pour le développement logiciel, facilitant le suivi des modifications de code et la coopération entre les développeurs. | Assure l'intégration entre Visual Studio, les servlets Java et PostgreSQL, favorisant le travail collaboratif et le partage des connaissances. |

***Les langages utilisées lors de ce projet :***









### **Répartition des tâches :**

* **Monsieur Maxime Vallet** : Responsable de la gestion de la machine virtuelle (VM), du dépôt GitHub, ainsi que du développement des scripts et des servlets.
* **Monsieur Ishac Hamdani** : Chargé du développement HTML et de la contribution aux scripts.
* **Monsieur Valentin Millot** : Impliqué dans le développement des scripts, des pages HTML, et des servlets.

# 4. Page Login, mise en place du système de token et création d’un script d’automatisation

Pour garantir la sécurité de notre Gestion de Maintenance Assistée par Ordinateur (GMAO) et structurer les différents utilisateurs qui testeront notre programme, nous avons décidé de mettre en place un système d'authentification.

Nous avons opté pour une base HTTPS afin de sécuriser les communications et de mettre en œuvre une méthode d'authentification par tokens. Cette approche permettra de protéger les données échangées tout en assurant une identification fiable des utilisateurs.

### **a. Qu'est-ce qu'un Token ?**

Un **token** est une chaîne de caractères générée pour représenter une session ou une information d'authentification. Il est utilisé pour sécuriser les communications entre un client (navigateur) et un serveur.

### **b. Types de Tokens**

Il existe plusieurs types de tokens, dont :

* **Tokens d'authentification** : Utilisés pour vérifier l'identité d'un utilisateur. Par exemple, un token JWT (JSON Web Token) contient des informations sur l'utilisateur et peut être utilisé pour authentifier les requêtes auprès d'un serveur.
* **CSRF Tokens (Cross-Site Request Forgery)** : Utilisés pour protéger les formulaires contre les attaques CSRF. Un token CSRF est généré par le serveur et inclus dans les formulaires HTML. Il doit être soumis avec le formulaire pour prouver que la requête provient d'un utilisateur légitime.

### **c. Sécurité des Tokens**

Il est crucial de protéger les tokens pour éviter les attaques. Les bonnes pratiques incluent :

* **Utilisation de HTTPS** : Pour sécuriser la transmission des tokens.
* **Expiration des tokens** : Pour limiter la durée pendant laquelle un token est valide.
* **Rafraîchissement des tokens** : En utilisant un mécanisme pour générer de nouveaux tokens avant l'expiration des anciens.

Les tokens jouent un rôle crucial dans la sécurisation de la GMAO. Ils facilitent l'authentification des utilisateurs et permettent de suivre les connexions ainsi que les échanges entre les utilisateurs et le serveur, garantissant ainsi une communication sécurisée et fiable.

D’autre part, pour que les tokens prennent effet, il a fallu créer un code dans le script des pages HTML et implémenter de nouvelles méthodes dans NetBeans, en particulier dans la DAO.

### **Fonctionnalités du Code gérant les Tokens :**

1. **Reload sur Déconnexion** :
   * La fonction window.onunload est configurée pour appeler reload() lorsque la page est quittée. Cela permet potentiellement de rafraîchir la page afin de relancer le script JS lorsqu’un utilisateur retourne sur la page précédente, cela est utilisé afin de pouvoir rediriger l’utilisateur.
2. **Récupération de l'Adresse IP du Serveur** :
   * L'adresse IP du serveur est récupérée à partir de l'URL de la page actuelle. Cela est utile pour accéder au serveur Tomcat car l’adresse IP du serveur Nginx est identique à celle du serveur Tomcat.
3. **Gestion des Données de Session** :
   * Les données de session (tokenGMAO, userLogin, et userRole) sont extraites de sessionStorage. Cela permet de conserver l'état de la session de l'utilisateur.
4. **Vérification du Token** :
   * Si le tokenGMAO est défini, une requête POST est envoyée au serveur pour vérifier la validité du token. La réponse est ensuite traitée par la fonction TokenCheck.
   * Si le tokenGMAO n'est pas présent, l'utilisateur est redirigé vers la page de connexion (login.html).
5. **Traitement de la Réponse de Vérification** :
   * La fonction TokenCheck traite la réponse JSON retournée par le serveur. En fonction du rôle de l'utilisateur (Admin, Technicien, Utilisateur), il est redirigé vers la page appropriée.
   * Si le rôle n'est pas reconnu ou si le token est invalide, l'utilisateur est également redirigé vers la page de connexion.

### **Fonctionnalités des Servlets :**

Les servlets permettent d'interagir avec des données au sein d’un serveur HTTP. Nous avons développé nos servlets avec Java (avec Javax) et ils s’exécutent au sein d’un serveur Tomcat.Parmi les fonctionnalités que nous avons implémentées, on retrouve : l’authentification et la gestion des utilisateurs, la gestion du matériel et des tickets (ajout, suppression et listage).

Dans un second temps, nous avons jugé nécessaire de créer un script permettant le déploiement du projet ainsi que son démarrage. Cette démarche vise à optimiser le développement en faisant gagner un temps précieux et à faciliter l'accès à la GMAO.

### **Voici les différentes fonctionnalités du Script :**

1. **Vérification et Démarrage de PostgreSQL** :
   * Le script commence par vérifier l'état du service PostgreSQL. S'il est inactif, il le démarre.
2. **Gestion des Fichiers de l'Application** :
   * Il vide le répertoire de l'application web Apache (sous /var/www/gmao) pour s'assurer qu'aucun ancien fichier n'est présent.
   * Il crée un répertoire pour la documentation Javadoc et copie les fichiers nécessaires depuis un répertoire local vers le répertoire Apache.
3. **Vérification et Démarrage d'Apache** :
   * Il vérifie ensuite si le serveur Apache est en cours d'exécution. Si ce n'est pas le cas, il effectue un démarrage.
4. **Reconstruction de la Base de Données** :
   * Le script propose à l'utilisateur de reconstruire la base de données. Si l'utilisateur accepte, il supprime la base de données existante (sae\_52) et peut éventuellement exécuter un script de configuration pour la reconstruire.
5. **Lancement de NetBeans** :
   * Il vérifie si NetBeans est déjà en cours d'exécution. Si ce n'est pas le cas, il demande à l'utilisateur s'il souhaite le lancer et, en cas d'acceptation, ouvre NetBeans dans un nouvel onglet de terminal.
6. **Rapport d'Exécution** :
   * Enfin, le script affiche l'état des services PostgreSQL et Apache, ainsi que l'état de NetBeans. Si la base de données a été reconstruite, il informe l'utilisateur que l'opération a été réalisée avec succès.
7. **Accès à la Documentation** :
   * Il indique également l'emplacement de la documentation Javadoc générée, accessible via une URL.

# Page de gestion des équipements et gestion des utilisateurs

Cette page est principalement destinée à l'administrateur, qui doit avoir la capacité d'ajouter et de supprimer divers éléments, tels que des ordinateurs, des routeurs, des switches et des câbles, au sein d'une PME.

Pour commencer, nous avons conçu une page HTML qui inclut plusieurs boutons et un tableau demandant les caractéristiques à remplir, ainsi qu'une liste permettant d'afficher les équipements enregistrés. Pour mettre en place ce système, nous avons d'abord créé des tables dans PostgreSQL (pc, switch, router et cable) qui permettront de stocker les différents attributs de chaque équipement, tels que l'adresse MAC, le VLAN, le numéro de série, le nom, etc.

Ensuite, nous avons développé un code HTML qui permet de détecter la méthode à appeler en fonction de l'objet sélectionné. Ainsi, le code interagit de la manière suivante :

### **Fonctionnalités du Code gérant les équipements :**

1. **Écoute de l'Événement de Clic** :
   * Un écouteur d'événements est ajouté au bouton submitButton. Lorsqu'il est cliqué, il empêche le comportement par défaut du formulaire (rechargement de la page) et détermine le type d'équipement sélectionné dans un sélecteur (equipmentTypeSelect).
2. **Redirection selon le Type d'Équipement** :
   * En fonction du type d'équipement sélectionné (PC, routeur, switch, ou câble), la fonction correspondante est appelée (addPC, addRouter, etc.).
3. **Ajout d'un PC (identique pour les switchs, routeurs et câbles)**  :
   * Dans la fonction addPC, les valeurs des champs du formulaire sont récupérées.
   * Une requête fetch est ensuite effectuée pour envoyer les données au serveur via une requête POST à l'URL spécifiée.
4. **Vérification de la Réponse** :
   * La réponse du serveur est traitée. Si le résultat indique que le nom de l'équipement existe déjà, un message d'alerte est affiché. Sinon, vous pouvez ajouter d'autres actions (non spécifiées dans votre code).
5. **Réinitialisation du Formulaire** :
   * Le formulaire est réinitialisé après l'envoi des données, quelle que soit la réponse.

### **Implémentation des servlets dans NetBeans**

Dans le cadre du projet, plusieurs servlets ont été développés dans NetBeans afin de garantir leur compatibilité avec le code existant. Chacun de ces servlets remplit une fonction bien définie :

* **addPC** : Ce servlet permet d’ajouter un nouvel ordinateur dans la base de données.
* **deletePC** : Ce servlet permet de supprimer un ordinateur existant de la base de données.
* **listPC** : Ce servlet permet de récupérer la liste des ordinateurs enregistrés dans la base de données et de la transmettre aux tableaux de la GMAO

L'opération est réitérée pour l'ensemble des équipements, selon un principe qui sera également utilisé pour la gestion et la création des tickets ainsi que des utilisateurs.

Le seul défaut réside dans le fait que nous n’avons pas eu le temps d’implémenter la modification du statut directement depuis le tableau. De plus, les objets ne peuvent pas être stockés en plusieurs exemplaires : chaque PC est unique, même s’il possède des caractéristiques identiques à un autre.

# Page des tickets

Cette partie du projet est dédiée à la gestion des tickets, principalement destinée à l'administrateur, qui doit avoir la capacité de consulter et de supprimer des tickets. L'objectif est de centraliser les demandes de service ou d'assistance au sein d'une entreprise, tout en permettant un suivi efficace.

Pour commencer, nous avons conçu une table dans PostgreSQL appelée **ticket**, qui contient plusieurs colonnes essentielles : id (clé primaire), description, service et status. Cette structure permet de stocker les détails de chaque ticket, tels que sa description, le service concerné, et son état (par exemple : "en attente", "en cours").

Ensuite, nous avons développé un ensemble de méthodes Java pour gérer les tickets via une interaction avec la base de données. Ces méthodes incluent :

* **addTicket** : Permet d'ajouter un ticket avec ses détails (description, service, statut). Cette méthode utilise des PreparedStatement pour garantir la sécurité contre les injections SQL.
* **doTicketExist** : Vérifie si un ticket existe dans la base de données en fonction de son identifiant.
* **getTicket** : Récupère la liste complète des tickets sous format JSON, triés par ordre croissant d’identifiant, afin de les afficher dans une interface utilisateur.
* **updateTicketStatus** : Met à jour le statut d’un ticket existant en fonction de son identifiant.
* **deleteTicket** : Supprime un ticket spécifique de la base de données.
* **getUserTicket** : Permet de récupérer les tickets associés à un utilisateur spécifique, également sous format JSON.

Ces fonctionnalités sont intégrées dans une application web où l’administrateur peut consulter et gérer les tickets via une interface intuitive. L'interaction entre l'interface et la base de données se fait grâce à des servlets développés dans NetBeans, permettant d'assurer la compatibilité avec le système existant.

Le système est fonctionnel, mais il présente quelques limitations. Par exemple, les tickets ne peuvent pas être triés ou modifiés directement depuis l'interface, et nous n'avons pas encore implémenté de fonctionnalité pour autoriser plusieurs utilisateurs à travailler sur un même ticket simultanément. Malgré cela, cette solution répond aux besoins principaux de gestion des tickets et offre une base solide pour des améliorations futures.

# Difficultés rencontrées

### **Création de la machine virtuelle (VM)**

La création de la VM a été l’une des étapes les plus complexes de ce projet. En effet, il était nécessaire de rendre cet environnement accessible à des utilisateurs sans connaissance préalable de ce type de système, tout en garantissant son bon fonctionnement à la fois à l’intérieur et à l’extérieur de la VM. Cette dernière exigence a été ajoutée après la rédaction du cahier des charges.

#### **Difficultés rencontrées**

**1. Installation et configuration de Tomcat**

L’installation de Tomcat s’est avérée particulièrement délicate, nécessitant de nombreuses configurations complexes. Une difficulté majeure réside dans la dispersion des informations : chaque étape doit être recherchée sur des sites différents, souvent sans explication claire des choix de configuration. Cela m’a contraint à composer une documentation personnalisée et à comprendre en profondeur les raisons des réglages requis.

Par exemple, il est conseillé de créer un dossier spécifique avec des droits restreints pour confiner Tomcat. Cependant, cette gestion fine des droits d’accès a posé problème, car certains dossiers devaient être accessibles uniquement par Tomcat, d’autres par l’utilisateur, ou encore par tout le monde. Les messages d’erreur peu explicites compliquaient encore davantage la résolution. De plus, il était nécessaire que le serveur soit accessible à la fois par l’utilisateur connecté et par NetBeans pour le déploiement des fichiers .war.

**2. Équilibre entre accessibilité et complexité**

Un exemple marquant concerne le déploiement des pages Web sur le serveur Apache. Ce déploiement ne s’effectue que via l’exécution d’un script Start.sh. Cette structure a été choisie pour faciliter la gestion des versions, en synchronisant les fichiers du projet avec un dépôt GitHub. Cependant, Apache ne prend pas en charge les liens symboliques en dehors de son répertoire racine, ce qui m’a conduit à centraliser l’ensemble du processus dans un seul script. Bien que cette approche simplifie l’utilisation, elle a suscité des frustrations auprès de mes coéquipiers en raison d’interactions jugées inutiles en dehors des IDE.

**3. Gestion des certificats SSL**

L’intégration de certificats SSL valides a également été un défi. J’ai tenté d’utiliser Let’s Encrypt pour obtenir des clés issues d’une autorité de certification (CA) légitime. Cependant, cela nécessitait que le serveur soit accessible depuis Internet et répertorié dans un DNS public, ce qui était impossible dans le cadre du projet. Résidant en Crous, Ishac et moi (M.Vallet) n’avons pas accès à une redirection de ports, le réseau étant bloqué par un NAT. En conséquence, j’ai dû recourir à des certificats auto-signés générés avec OpenSSL, ce qui oblige à les valider manuellement à chaque connexion sur un nouvel appareil.

**4. Documentation détaillée**

Une part significative de mon temps a été consacrée à la création d’une documentation claire et exhaustive. Elle comprend des instructions pour installer et utiliser le projet manuellement, ainsi qu’une documentation Java (hébergée sur le serveur Apache) permettant d’exploiter les servlets et méthodes sans avoir à consulter le code source. Plusieurs révisions ont été nécessaires pour éliminer toute ambiguïté et erreurs. Cette expérience m’a permis de réaliser l’importance d’une documentation bien structurée, surtout dans un contexte de développement collaboratif sur des environnements complexes.

**5. Réduction de la taille de la VM**

J’ai procédé à plusieurs itérations pour réduire la taille de la VM afin de faciliter son stockage (notamment sur Google Drive) et d’accélérer son téléchargement et son installation. Cela impliquait :

* la suppression de logiciels inutiles, des anciens noyaux Linux, et de *snapcraft* (gourmand en espace) ;
* la compression du disque virtuel (VDI) ;
* la suppression des logs et la réduction de la taille du swap.

Ces optimisations ont permis de passer la taille de la VM de 8,71 Go à 4,48 Go.

### **Création d’un système d’authentification (Fullstack)**

La mise en place d’un système d’authentification complet a également été un défi, nécessitant la prise en charge simultanée de nombreuses technologies (JDBC, API REST) tout en gérant les contraintes de la VM. Cela demandait un arbitrage entre sécurité et rapidité d’implémentation, car la VM avait déjà consommé beaucoup de temps.

#### **Compromis réalisés**

1. **Gestion des mots de passe et des tokens**Les mots de passe sont hachés côté serveur à l’aide de MD5. Bien que ce choix soit moins sécurisé qu’un algorithme moderne comme bcrypt, il a été retenu pour sa simplicité d’implémentation. Les tokens, quant à eux, ne sont pas hachés et ne possèdent pas de date d’expiration, afin de réduire la quantité de code à écrire.
2. **Simplification des redirections**Chaque page inclut une redirection en dur, évitant ainsi le développement d’un servlet spécifique et une complexification du JavaScript. Cette solution permet de gérer l’authentification avec un script unique, mais limite la flexibilité du système.

**Création de la page des tickets**

La mise en place de la page de gestion des tickets a rencontré plusieurs problèmes qui ont affecté son bon fonctionnement. Bien que les utilisateurs puissent soumettre un ticket avec succès, les demandes n’étaient pas transmises à l’administrateur. Après plusieurs investigations, plusieurs causes ont été identifiées.

#### **Problèmes rencontrés**

1. **Mauvaise configuration de la table SQL**

La première cause identifiée concernait une mauvaise configuration des colonnes de la table SQL. Certaines colonnes étaient mal nommées, ce qui entraînait des incohérences dans l’enregistrement des données lors de la soumission des tickets. Concrètement, certaines informations n’étaient pas correctement insérées dans la base de données, rendant les tickets inaccessibles pour l’administrateur.

1. **Échecs d’incrémentation et de liaison des données**

En raison de la mauvaise configuration de la base de données, les opérations d’incrémentation et de liaison entre les tables échouaient. Cela empêchait l’enregistrement correct des tickets et leur transmission au système de gestion de l’administrateur.

#### **Solutions apportées**

Pour résoudre ces dysfonctionnements :

* **Correction des colonnes SQL** : les noms des colonnes ont été corrigés afin d’assurer une cohérence dans l’enregistrement des données.
* **Révision des processus d’insertion des données** : les mécanismes d’ajout des tickets ont été ajustés pour garantir que les informations soient correctement liées et enregistrées dans la base de données.

Ces modifications ont permis de restaurer le bon fonctionnement du système, assurant ainsi la réception des tickets par l’administrateur et une gestion optimale des demandes.

### **Difficultés particulières rencontrées (Valentin Millot)**

N’étant pas issu de cette formation à l'origine, l’utilisation d’outils tels que NetBeans, Apache et les servlets m’était totalement inconnue. Ce projet m’a donc offert une opportunité d’apprendre et de développer de nombreuses nouvelles compétences, bien que M. Vallet ait joué un rôle important dans le déploiement global du projet.

La recherche des différents fonctionnements de ces outils s’est avérée très enrichissante, et la création des servlets a représenté un véritable défi pour moi.

En ce qui concerne le HTML et le CSS, bien que ce n’était pas ma principale responsabilité, je me suis replongé dans ces langages. Cela m’a permis de me remettre à niveau, car je n’y avais pas touché depuis plusieurs années. Malgré tout, j’ai pu m’adapter et participer un maximum sur ce projet.

### **Résolution des dysfonctionnements et limites du projet**

Malgré les progrès rapides réalisés dans le projet, en grande partie grâce à M. Vallet, qui a développé la VM dès les premières étapes, permettant ainsi de poser rapidement les bases du travail, certaines fonctionnalités prévues dans le cahier des charges initial n’ont pas pu être mises en œuvre par manque de temps.

Ces fonctionnalités incluent :

* **L’ajout d’interventions via un planning** : une option permettant de planifier des interventions directement dans l’application.
* **Le changement de statut des équipements dans la liste** : une fonctionnalité pour modifier facilement l’état des équipements suivis.
* **L’export des listes au format Excel** : une fonctionnalité permettant de générer des fichiers Excel à partir des données de la base de données.

# Conclusion

Au cours de ce projet, nous avons pu mobiliser et appliquer de nombreuses compétences acquises durant notre apprentissage en BUT, notamment dans les domaines couverts par les ressources R1.07, R1.08, R1.15, R3.08, R3.10, R4.05 et R5.04.

Bien que nous ayons rencontré de nombreuses difficultés, en particulier lors de la création de la VM, de la mise en place des services, de la préparation des scripts et de l’utilisation de GitHub, nous avons finalement réussi à développer une GMAO fonctionnelle répondant à une grande partie des critères définis dans le cahier des charges.

Ce projet nous a permis de mieux comprendre l’étendue de nos compétences et de mesurer l’importance de la collaboration en équipe dans le cadre de projets de développement. Bien que l’utilisation de GitHub ait été difficile à appréhender au départ, nous avons rapidement réussi à en tirer parti pour travailler efficacement ensemble et faire progresser le projet..

# 