COMMUNAUTE FRANCAISE
HAUTE ECOLE ROBERT SCHUMAN
DEPARTEMENT ECONOMIQUE
6800 LIBRAMONT

Projet intégré

UIT Manager - Rapport final

Projet, Environnements de développement de logiciels
[INEC0002-2]-[INEC0002-C-a]
Benoit Burlion, Laurent Schalkwijk

GERARD Alex
GUÉRISSE Pauline
HESBOIS Dorian
RIGAUX Germain
SPRONCK Mathis
WUIDAR Florentin

Numéros de contact
Gerard Alex: 0479 70 29 32
Rigaux Germain: 0472 81 50 27

BAC3 INFORMATIQUE - DEVELOPPEMENT D'APPLICATIONS
ANNEE 2024-2025

Table des matières

1 Introduction

La gestion d'un parc informatique est un défi majeur pour les entreprises disposant de nombreuses machines. Sans outils centralisés, le suivi des interventions et la maintenance préventive deviennent complexes, entraînant des pertes de temps et un risque accru d'erreurs.

Pour répondre à ces enjeux, notre application propose une solution complète : un inventaire automatisé des machines et un portail centralisé pour le suivi des informations. Elle améliore la visibilité sur l'état du parc informatique, facilite la prise de décision et optimise la gestion, réduisant ainsi les pertes d'informations.

Développée avec la méthodologie agile Scrum, l'application bénéficie de sprints réguliers pour garantir l'atteinte des objectifs et améliorer la qualité finale.

2 Partie groupe

Le groupe est composé de 6 membres.

Alex a mis en place le CI/CD dans GitLab, permettant d'identifier les problèmes majeurs via les tests automatisés. Il a initié l'équipe à Figma pour harmoniser les mockups et a travaillé sur la WebApp ainsi qu'une partie de l'agent, tout en contribuant à la conception des informations machine. Il a été le Scrum Master du *sprint 4* et le Product Owner du *sprint 1*

Pauline a structuré les rapports, pris des notes lors des Sprint Reviews, et harmonisé les écrits. Elle a créé la page d'inventaire des machines, les fonctionnalités comme le breadcrumb, et réalisé les diagrammes de séquence. Git Maintainer pendant la majorité du projet, elle a été le Scrum Master du *sprint 0* et le Product Owner du *sprint 5*

Dorian a principalement développé la WebApp, en créant l'affichage des informations machine et les pages pour les groupes de critères d'alarme. Il a été le Scrum Master du *sprint 3* et le Product Owner du *sprint 2*

Germain a contribué à la structure du projet WebApp, de la base de données, et des rapports. Il s'est concentré sur les alarmes dans la WebApp (pages « Raised Alarms » et « Alarm Details ») et l'analyse des informations de l'agent. Son dernier sprint s'est focalisé sur les livrables finaux. Il a été le Scrum Master du *sprint 5* et le Product Owner du *sprint 0*

Mathis a créé la page de détail machine et les modèles pour la WebApp. Il a principalement travaillé sur l'agent pour structurer et envoyer les données à l'API, et a contribué, avec Florentin, aux projets API et AlarmManager, où il a implémenté l'envoi d'e-mails d'alarme. Git Maintainer lors du dernier

sprint, il a été le Scrum Master du *sprint 2* et le Product Owner du *sprint 3*

Florentin a travaillé sur l'affichage des groupes de critères d'alarme et la gestion des utilisateurs. Il a aidé à la cohérence des données dans la base et conçu, avec Mathis, les projets API et AlarmManager pour le déclenchement des alarmes. Il a été le Scrum Master du *sprint 1* et le Product Owner du *sprint 4*

2.1 Réflexion globale du fonctionnement

Dès le début du projet, nous avons établi des bases solides pour assurer une organisation claire et un travail collaboratif efficace. Nous avons adopté les conventions des langages utilisés pour garantir un code propre et maintenable, et choisi Discord comme outil principal avec des canaux dédiés pour centraliser les échanges et éviter les malentendus.

Pour la gestion de projet, nous avons appliqué la méthodologie Agile avec Scrum. Nos premières itérations étaient maladroites, les cérémoniaux (comme les Sprint Retrospectives et Planifications) étant trop chronophages. De plus, l'attribution de dates fixes aux issues a entraîné des blocages inutiles et une pression excessive.

Progressivement, nous avons ajusté notre approche : réduction du nombre et de la durée des réunions, objectifs plus clairs et gestion des tâches plus flexible. Cette amélioration a permis une meilleure répartition du travail, une anticipation des imprévus, et une coordination renforcée au sein de l'équipe.

Avec cette évolution, nous avons constaté une nette amélioration dans notre organisation, notre efficacité, et notre capacité à livrer des résultats de qualité dans les délais impartis.

2.2 Difficultés rencontrées

Cette section du rapport met en lumière les principales difficultés rencontrées lors du développement du projet, celles qui ont nécessité le plus de temps et d'efforts pour être surmontées.

2.2.1 Identity Framework

Nous avons rencontré des difficultés lors de l'implémentation du framework Identity dans le projet de la WebApp qui a paralysé l'avancement du projet durant quelques jours. En effet, nous n'avions pas prévu de l'implémenter à ce moment là du projet mais nous nous sommes rendu compte que nous avions besoin d'utiliser les utilisateurs. Nous avons décidé de créer notre classe ApplicationUser qui hérite de celle de Identity et d'utiliser les pages de connexions générées par Identity mais lors de la création de ces pages, nous avions demander d'utiliser notre classe ApplicationUser. Cependant les pages générées utilisaient toujours IdentityUser, il a fallu modifier dans les différentes pages pour utiliser notre classe User.

2.2.2 Refactorisation du projet Agent

Lors de l'implémentation de l'API, nous nous sommes rendu compte que l'envoi des données depuis l'Agent allait rendre le déclenchement des alarmes plus compliqué. En effet, nous n'avions pas encore exactement pensé à comment structurer les données liés aux déclenchements d'alarmes lors de la rédaction du code du projet de l'agent. Nous avons dû modifier ce que nous avions fait au début.

2.3 Indicateurs globaux

Au cours du projet, nous avons utilisé divers outils pour suivre et organiser notre travail. Chaque section présentera les statistiques associées à chacun de ces outils.

2.3.1 Clockify

Nous avons utilisé Clockify pour suivre notre temps de travail, en exploitant notamment des tags pour catégoriser les différentes tâches effectuées.

2.3.2 Répartition du temps sur le projet

Clockify nous permet de connaître le temps total du temps passé à travailler sur le projet.

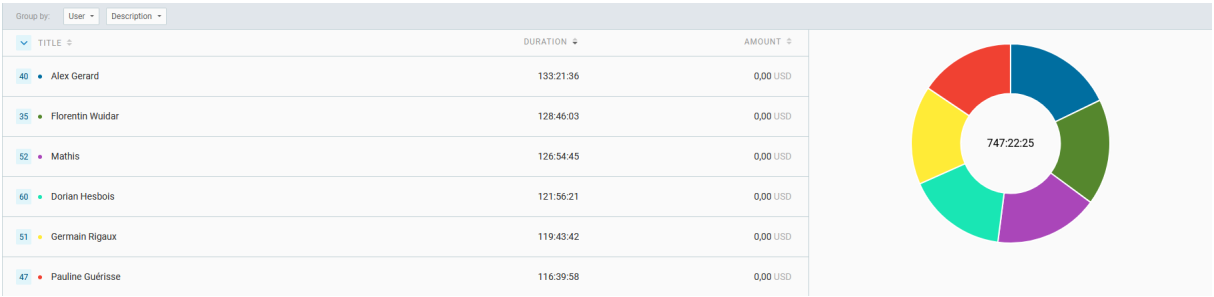

Fig. 1. – Temps total du temps passé sur le projet

2.3.3 Répartition du temps par tags

Nous avons utilisé différents tags pour catégoriser notre temps de travail. Nous en avons utilisé 12 dans ce projet.

8	Report	14:44:48	0,00 USD
4	Pair Programming	12:29:12	0,00 USD
2	Development, Security	11:40:57	0,00 USD
6	Development, Pair Programming	10:25:19	0,00 USD
6	Setup	07:57:37	0,00 USD
3	Security	05:12:15	0,00 USD
2	Sprint Planning, Retrospective Sprint	03:49:08	0,00 USD
1	Development, Documentation	03:15:52	0,00 USD
1	Development, Report, Documentation	00:57:07	0,00 USD
1	Development, Analysis	00:38:29	0,00 USD
1	Report, Retrospective Sprint	00:21:53	0,00 USD

Fig. 3. – Répartition du temps total par rapport aux différents tags

2.3.4 Gitlab

Nous avons utilisé GitLab pour identifier et gérer les issues les plus importantes à réaliser.

2.4 Répartition des issues par contributeurs

Cette section présente les issues réalisées par chaque contributeur.

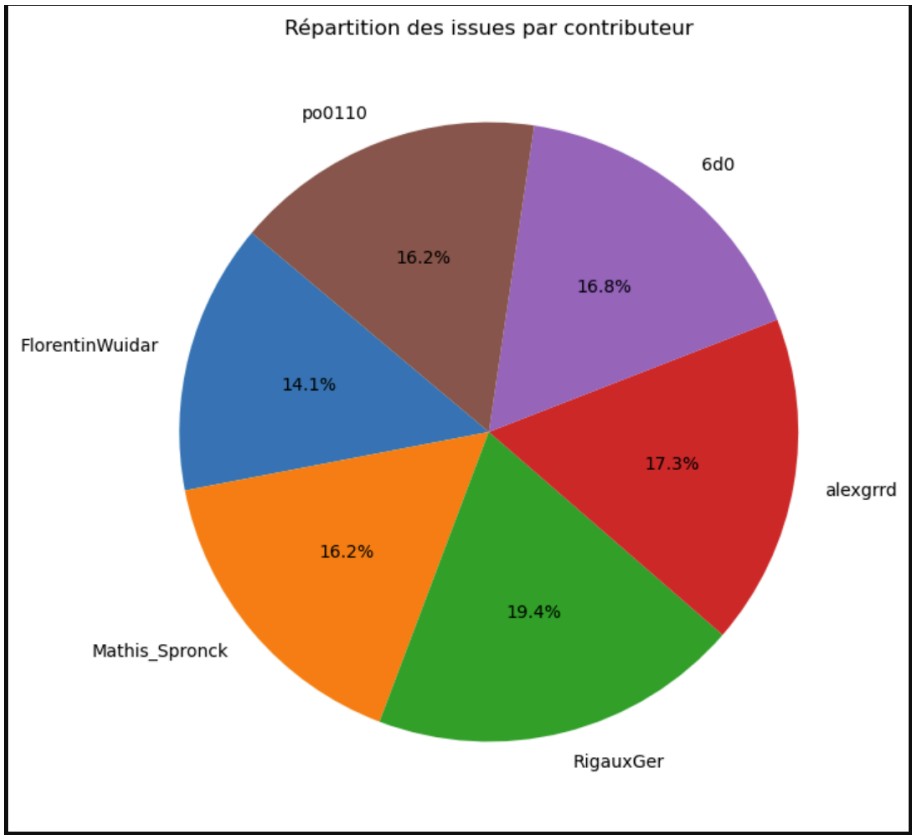

Fig. 4. – Issues Gitlab réalisées par contributeur

2.5 Statistiques générales par contributeurs

Cette section détaille le nombre de lignes de code produites par chaque contributeur.

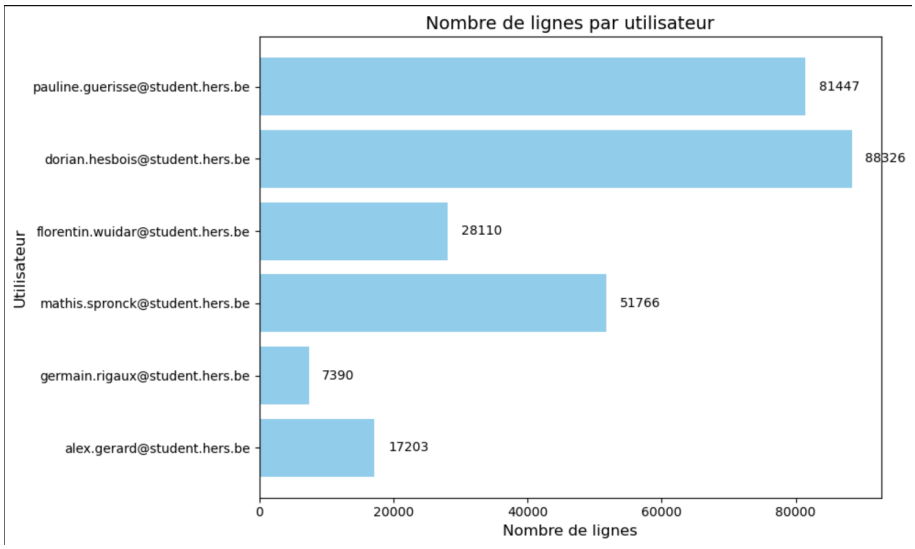

Fig. 5. – Nombre de lignes codées par contributeur

3 Partie projet

Cette partie est consacrée à une analyse globale du projet, incluant son architecture et son fonctionnement.

3.1 Analyse

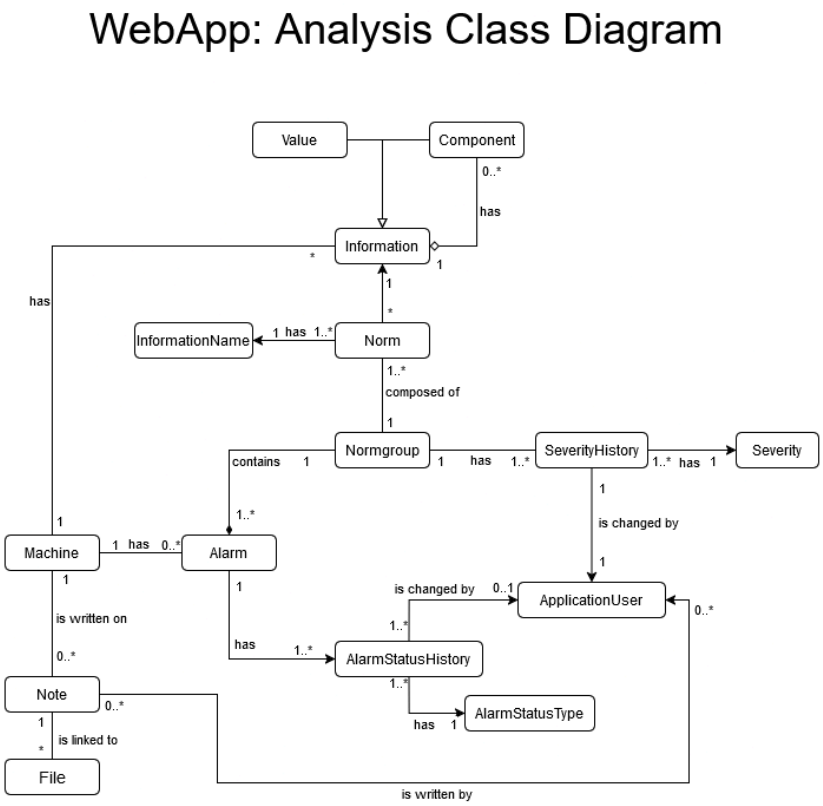

Fig. 6. – Diagramme de classes d'analyse du projet

Le diagramme ci-dessus présente les différents modèles utilisés dans la partie web du logiciel. On y retrouve notamment les liens entre les machines, les alarmes qui possèdent un groupe de critère, qui a lui une liste des critères. On y retrouve le pseudo Composite Pattern entre « Information », « Value » et « Component ».

3.2 Choix d'implémentation

3.2.1 ChartJS

ChartJS, une bibliothèque JavaScript simple, flexible et légère, s'est imposée comme la solution idéale pour afficher les graphiques sur la page d'accueil. Elle garantit une compatibilité étendue avec les navigateurs et couvre efficacement tous les types de diagrammes dont nous avons besoin.

3.2.2 Utilisation de SMTP

Nous avons opté pour le Simple Mail Transfer Protocol (SMTP) afin de rester dans le cadre pédagogique de notre projet, qui impose l'utilisation exclusive des bibliothèques internes à .NET.

3.2.3 Pseudo Composite Pattern

Lors de la création des modèles pour la page de détail machine, nous avons initialement décidé de nous inspirer du composite pattern pour mieux structurer les informations d'une machine. Cependant, à mesure que l'implémentation avançait, nous nous sommes éloignés du modèle initial. Nous avons opté pour une classe abstraite plutôt qu'une interface, et nous avons également choisi de retirer les éléments uniques du composant ainsi que ceux de la feuille (valeurs dans notre code).

3.2.4 Le projet AlarmManager et l'Observeur Pattern

Nous avons décidé de créer un projet du nom de AlarmManager qui a pour but de déclencher les alarmes lors de la remontée des informations des agents et lors de la modification et de la création des groupes de critères (NormGroups). Pour récupérer ces informations, nous avons utilisé SignalR une bibliothèque inclus dans ASP.NET qui observe les deux projets (API et WebApp) et qui va récupérer les données quand les méthodes observées seront appelées.

3.2.5 La base de données Docker (Postgres)

L'utilisation de PostgreSQL dans un conteneur Docker pour un projet .NET offre une isolation et une portabilité garantissant un environnement cohérent entre le développement, les tests et la production. Cela simplifie le déploiement, la gestion des versions et la configuration, tout en assurant la reproductibilité des environnements. PostgreSQL, avec ses fonctionnalités avancées (JSONB, types personnalisés, etc.) et sa compatibilité avec Entity Framework, s'intègre parfaitement aux applications .NET. Docker, de son côté, renforce la sécurité, facilite l'administration et permet une montée en charge efficace, rendant l'ensemble plus flexible et mieux adapté aux besoins modernes.

3.3 Architecture

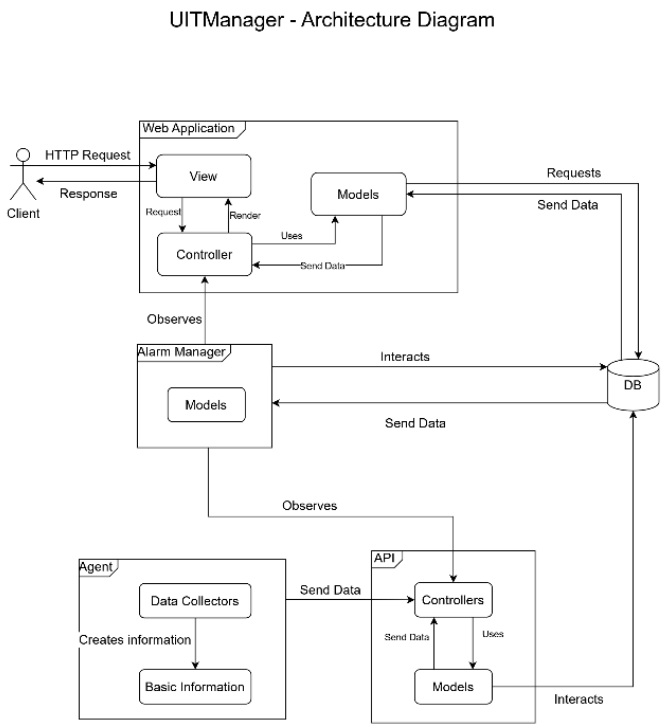

Fig. 7. – Diagramme d'architecture du projet

L'utilisateur final ne pourra communiquer qu'avec l'application web. L'application web va communiquer avec la base de données de sorte à remplir la base de données et afficher les informations présentes dans celle-ci. La partie « Agent » va envoyer les données récoltées sur les machines à l'API. Quant à l'API, elle va s'occuper d'envoyer ces mêmes informations à la base de données. Alarm Manager s'occupe d'observer les autres projets de la solution afin de garantir une réactivité lors du déclenchement de nouvelles alarmes. Il interagit également avec la base de données.

4 Partie méthode agile/scrum

Dans cette section, nous détaillons la méthode Agile que nous avons utilisée lors du développement de ce projet.

4.1 Vue d'ensemble des sprint

Scrum Master	Product Owner	Git Maintainer	Date début	Date fin	N° Sprint
Pauline Guérisse	Germain Rigaux	Non défini	09/10/2024	23/10/2024	*0*
Florentin Wuidar	Alex Gerard	Pauline Guérisse	23/10/2024	06/11/2024	*1*
Mathis Spronck	Dorian Hesbois	Pauline Guérisse	06/11/2024	20/11/2024	*2*
Dorian Hesbois	Mathis Spronck	Pauline Guérisse	20/11/2024	04/12/2024	*3*
Alex Gerard	Florentin Wuidar	Pauline Guérisse	04/12/2024	17/12/2024	*4*
Germain Rigaux	Pauline Guérisse	Mathis Spronck	18/12/2024	27/12/2024	*5*

4.2 Product Backlog

PRODUCT BACKLOG

THEME		ROLE	JE VEUX/JE DOIS	AFIN DE	# POIDS	# BUSINESS VALUE
Dashboa...	1	Technicien	consulter les alarmes	pouvoir identifier les problèmes à régler rapidement	9	100
Agent	2	Agent	récupérer les informations (ram,cpu,...)	de les envoyer	9	81
Agent	3	Agent	envoyer les informations concernant la machine au démarrage de la machine	de les faire remonter aux techniciens	8	79
Agent	4	Agent	envoyer les informations concernant la machine à une fréquence donnée	de les faire remonter aux techniciens si la machine ne redémarre pas	4	78
Détail	5	Technicien	gérer une alarme	remontrer une éventuelle évolution de résolution de cette dernière	7	87
Détail	6	Technicien	Consulter l'inventaire des machines	avoir un listing de toute les machines du parc	7	82
Détail	7	Technicien	consulter informations relatives à une machine (ram,cpu,...)	pouvoir anticiper/relativiser	7	80
Gestion	8	Responsable...	modifier la priorité d'une alarme	qu'elle soient traitées différemment par les techniciens	6	85
Gestion	9	Responsable...	gérer le groupe de critères d'une alarme	qu'elle puisse s'activer suivant un besoin différent	6	75
Détail	10	Technicien	gèrer les notes des machine	pouvoir les consulter	6	65
User	11	Technicien	se (dé)connecter	d'accéder à/sortir de/ la plateforme	5	53
Détail	12	Technicien	consulter les alarmes résolues d'une machine (historique)	identifier des problèmes récurrents pour une même machine	5	40
User	13	DSI	Gestion des utilisateurs	afin gerer les informations des employés	4	30

Fig. 8. – Product backlog

4.3 Rétrospective des sprint review

Globalement, les retours issus de nos Sprint Reviews se concentraient principalement sur deux aspects : la gestion de nos temps individuels via Clockify et la répartition initiale de la charge de travail, qui s'est révélée inefficace. Dès le début du projet, des malentendus ont émergé, notamment sur les éléments à enregistrer dans Clockify, ce qui a conduit à des incohérences dans le suivi des contributions. De plus, une mauvaise distribution des tâches a engendré des déséquilibres, certains membres étant surchargés tandis que d'autres manquaient de tâches clairement définies.

Conscients de ces problèmes, nous avons progressivement ajusté notre organisation en nous efforçant d'appliquer au mieux les principes de la méthode Agile. Nous avons optimisé la répartition des tâches en augmentant le nombre d'issues à accomplir par Sprint, afin que chaque membre puisse avancer sur une tâche s'il termine plus rapidement, et nous avons encouragé le pair programming.

Par ailleurs, bien que nos premières itérations aient été marquées par des retards dans la production

de pages ou de fonctionnalités visibles pour le client, nous avons su rattraper le temps perdu. En concentrant nos efforts sur la qualité, nous avons produit des éléments cohérents et conformes aux attentes. L'un des points forts du projet réside dans la cohérence visuelle et fonctionnelle des pages livrées, reflet de notre attention aux détails et de notre capacité à collaborer efficacement malgré les difficultés initiales.

En résumé, bien que nous ayons rencontré des problèmes d'organisation et de collaboration en début de projet, les ajustements réalisés tout au long du processus ont permis d'améliorer la répartition de la charge de travail, de renforcer la cohésion de l'équipe et de livrer un produit final de qualité.

5 Partie Sécurité

Dans cette section, nous abordons les différentes failles de sécurité identifiées par l'outil OWASP ZAP dans notre WebApp. Cependant, il est important de noter que l'absence de détection de failles par OWASP ZAP ne signifie pas qu'il n'en existe pas.

5.1 Evaluation de la sécurité

Nous avons effectué une première analyse de sécurité avec OWASP ZAP sur notre WebApp dans son état brut, sans configuration préalable (`sitemap.xml`). À l'issue de cette première analyse, nous avons identifié plusieurs vulnérabilités et nous les avons corrigées, renforçant ainsi la sécurité globale de notre application.

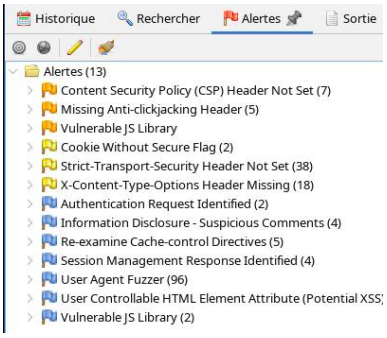

Fig. 9. – Résultat de OWASP ZAP

5.1.1 Content Security Policy (CSP) Header Not Set

Cette faille survient parce que la WebApp n'est pas protégée contre les attaques XSS (Cross-Site Scripting). Ces attaques exploitent la confiance des navigateurs envers le contenu reçu du serveur, permettant l'exécution de scripts malveillants.

5.1.1.1 Risques associés

- Vol de comptes;
- Phishing;

- Spreading Malware;
- Perte de données;
- Création de backdoors.

5.1.1.2 Résolution de la faille

Pour résoudre cette faille, il est nécessaire de mettre en place un en-tête CSP (`Content Security Policy`). Ce dernier doit spécifier les domaines autorisés à exécuter des scripts dans l'application, ainsi que les éléments (CSS, polices, images) qu'il est permis d'utiliser. Il est recommandé de n'utiliser que des éléments internes au serveur. Dans le cadre de notre projet, nous avons fait en sorte de sécuriser du mieux que nous pouvions en ajoutant notre CSP. Dans ce CSP, nous avons rajouté les nonces que nous avons placés sur nos balises script.

5.1.2 Missing Anti-clickjacking Header

Cette faille existe parce que la WebApp n'est pas protégée contre les attaques de type « clickjacking ». Dans ce type d'attaque, un utilisateur peut être détourné de la page qu'il visite et incité à cliquer sur un bouton ou un lien dissimulé sous une superposition d'iframe, CSS ou champs de texte, redirigeant ainsi l'action vers un autre objectif.

5.1.2.1 Risques associés

- Vol de données sensibles;
- Attaque de CSRF (Cross-Site Request Forgery);
- Exploitation des privilèges de l'utilisateur;
- Attaques de phishing;
- Attaques Man-in-the-Middle (MitM).

5.1.2.2 Résolution de la faille

Pour corriger cette faille, il faut définir les en-têtes HTTP appropriés en ajoutant `frame-ancestor` et `X-Frame-Options` (pour les navigateurs dépréciés). Il est également recommandé de modifier l'attribut `SameSite` pour limiter l'accès aux cookies uniquement sur le même site et empêcher les scripts JavaScript d'y accéder. Enfin, il est important de vérifier que l'application s'affiche dans la fenêtre principale et non dans une iframe.

5.1.3 Vulnerable JS Library

Cette faille existe parce que la version utilisée de la bibliothèque `jQuery Validation` (1.19.5) contient des vulnérabilités connues. L'une des principales est une faille XSS (Cross-Site Scripting), qui permet à un attaquant d'injecter et d'exécuter des scripts malveillants dans le navigateur de l'utilisateur via la méthode `showLabel`. Une autre vulnérabilité notable est liée à un ReDoS (Regular Expression Denial of Service), qui permet de bloquer le serveur en exploitant des expressions régulières mal optimisées.

5.1.3.1 Risques associés

- Exécution de scripts malveillants;
- Contournement des validations côté client;
- Manipulation de données utilisateur;
- Compromission des sessions utilisateur;
- Exploitation de failles pour des attaques secondaires (XSS, CSRF).

5.1.3.2 Résolution de la faille

Pour corriger cette faille, la bibliothèque `jquery-validation` a été mise à jour via NuGet en installant la dernière version stable la version 1.21.0.

5.1.4 Cookie Without Secure Flag

Cette faille est liée à l'absence du flag `Secure` sur certains cookies. Cela permet aux cookies d'être transmis en clair sur des connexions non sécurisées, ce qui représente un risque élevé en cas de compromission du réseau (attaque Man-in-the-Middle).

5.1.4.1 Risques associés

- Exfiltration de données sensibles : Un attaquant peut intercepter le trafic et voler les cookies d'authentification ou de session.
- Prise de contrôle de session : Un attaquant peut usurper l'identité de l'utilisateur en capturant un cookie d'authentification non sécurisé.
- Violation de la politique de sécurité : L'absence du flag `Secure` ne respecte pas les bonnes pratiques de sécurité web.

5.1.4.2 Résolution de la faille

Pour résoudre cette faille, les cookies doivent être configurés pour inclure le flag `Secure`, garantissant qu'ils ne sont envoyés que sur des connexions HTTPS. Cela concerne notamment les cookies d'authentification, anti-CSRF et de session.

5.1.5 Strict-Transport-Security Header Not Set

La faille « HSTS » (HTTP Strict Transport Security) survient lorsqu'un site web ne définit pas l'en-tête `Strict-Transport-Security`. Cet en-tête est crucial pour forcer l'utilisation de HTTPS et éviter des attaques de type Man-in-the-Middle (MITM).

5.1.5.1 Risques associés

- Attaques MITM: Un attaquant pourrait intercepter et altérer les requêtes entre l'utilisateur et le serveur, compromettant ainsi les données sensibles comme les mots de passe;

- Attaques de downgrade: Si un utilisateur tente de se connecter en HTTP, un attaquant pourrait forcer une connexion non sécurisée, exposant ainsi l'utilisateur aux risques d'attaque.

5.1.5.2 Résolution de la faille

Pour corriger ce problème, il est nécessaire de configurer le serveur web pour ajouter l'en-tête `Strict-Transport-Security`, qui forcera l'utilisation de HTTPS sur tous les sites web.

5.1.6 X-Content-Type-Options Header Missing

Cette faille est liée à l'absence de l'en-tête HTTP `X-Content-Type-Options`, qui empêche les navigateurs de tenter de deviner ou d'interpréter un type MIME autre que celui spécifié par le serveur.

5.1.6.1 Risques associés

- Attaques XSS : Des fichiers tels que `.txt` ou `.json` pourraient être interprétés comme du JavaScript malveillant, permettant l'exécution de scripts dans le navigateur;
- Téléchargements malveillants : Des fichiers apparemment inoffensifs (images, vidéos) peuvent contenir du code exécutable, ouvrant la porte à des attaques;
- Exploitation des API : Des réponses mal interprétées (par exemple, des fichiers JSON) pourraient permettre des injections malveillantes.

5.1.6.2 Résolution de la faille

Pour résoudre ce problème, il convient de configurer le serveur web ou l'application pour inclure l'en-tête `X-Content-Type-Options` avec la valeur `nosniff` dans toutes les réponses HTTP.

6 Conclusion

Nous avons implémenté les principales fonctionnalités demandées : la consultation des alarmes et de l'inventaire pour les techniciens, la configuration des alarmes pour les responsables, et l'amélioration du suivi grâce à l'ajout de notes sur les machines. La gestion des alarmes a aussi simplifié la migration des machines vers une nouvelle version de leur système d'exploitation.

Cette première expérience avec la méthode agile Scrum a renforcé nos compétences en collaboration et gestion de projet. Bien que certains aspects aient dû être approfondis, les derniers sprints se sont concentrés sur les fonctionnalités clés et la correction des vulnérabilités, notamment via OWASP ZAP.

Le projet a été un défi enrichissant, et nous sommes fiers de livrer une solution conforme aux attentes des clients, avec des résultats très satisfaisants.

7 Annexes

7.1 Vocabulaire

- **Alarme [FR]/ Alarm [EN]** : Signal déclenché lorsque les critères définis dans un groupe de critères actifs ne sont pas respectés;
- **Inventaire [FR]/ Inventory [EN]** : Centralisation de toutes les machines enregistrées, accessible par le directeur du service informatique, les responsables de maintenance et les techniciens;
- **Machine [FR]/ Machine [EN]** : Représentation d'une machine physique dans le parc informatique de l'UIT, avec des notes de suivi, des spécifications techniques et les alarmes qu'elle a déclenchées;
- **Critère [FR]/ Norm [EN]** : Élément déclencheur d'une alarme lorsque une condition n'est pas remplie. Un critère inclut une valeur (par exemple, 80 %) et une condition de satisfaction (par exemple, > 80 %). Si cette condition est remplie, que le critère appartient à un groupe actif et que le groupe actif de critères a toutes les conditions remplies, une alarme est déclenchée.
- **Groupe de critères [FR]/ NormGroup [EN]** : Ensemble défini de critères. Si tous les critères d'un groupe sont satisfaits sur une machine, l'alarme associée à ce groupe de critères est déclenchée;
- **Déclenchement d'une alarme** : Action permettant d'informer les techniciens d'un dysfonctionnement ou du non-respect des critères définis, entraînant ainsi le déclenchement de l'alarme
- **Priorité d'une alarme [FR] / Alarm's priority [EN]** : Chaque alarme a un niveau de priorité. Le système met en avant les alarmes prioritaires, définies par le responsable de maintenance en fonction de leur urgence;
- **Sévérité d'un groupe de critère [FR] / Norm group's severity [EN]** : Niveau de sévérité attribué à un groupe de critères par une personne compétente, en fonction de l'impact potentiel du non-respect de ces critères.