SOLANA-DUPLAA Léo

Stage de deuxième année en entreprise

Tuteur : Guillaume MATHEN

EFFSEIT
SYSTEM & CYBERSECURITY

Développement Font-end

Du 8/01/2024 au 23/02/2024

Remerciements

Premièrement, je tiens à remercier M. Mathen, responsable ingénierie et formation de l'entreprise EFFSEIT, pour avoir accepté d'être mon tuteur tout au long de mon stage de deuxième année. Ses conseils et son soutien ont été très importants pour moi et je suis très reconnaissant de l'opportunité qui m'a été offerte de participer au développement de l'application Web GMAO d'EFFSEIT.

M. Mathen m'a intégré au groupe, ce qui m'a permis de découvrir le travail en équipe dans ce milieu spécifique.

Je tiens vraiment à dire un grand merci à Monsieur Cerutti. Son accompagnement pendant mon stage a vraiment fait toute la différence. Sa façon d'être toujours là pour moi et de me soutenir à chaque étape m'a aidé dans mon expérience ici. Je suis reconnaissant d'avoir pu compter sur lui.

Je tiens également à exprimer ma gratitude envers Axel, Arthur et Romain du côté réseaux, et Enzo et Parceval du côté développement. Ils ont su créer une ambiance d'équipe très agréable.

Enfin, je veux remercier les intervenants de l'E.P.S.I., pour m'avoir accompagné avant, pendant et après mon stage. Leurs recommandations ont été précieuses dans la préparation de ce stage professionnelle de deuxième année. Je suis satisfait de pouvoir suivre une formation aussi complète et qualitative que celle de l'E.P.S.I.

SOMMAIRE

Table des matières

Introduction

Présentation de l'entreprise

EFFSEIT, une entreprise spécialisée en cybersécurité, a vu le jour à Montpellier en 2019. Son objectif est de fournir des solutions de sécurité informatique personnalisées pour protéger les entreprises contre les menaces numériques. Les services proposés par EFFSEIT englobent la protection informatique, l'accompagnement numérique, la sensibilisation et la formation, ainsi que l'audit de gestion de la sécurité. En tant qu'entreprise à taille humaine, EFFSEIT s'engage à répondre aux besoins spécifiques de chaque client.

Domaines d'expertise d'EFFSEIT

Protection informatique :

EFFSEIT conçoit et met en place des solutions de sécurité informatique personnalisées.
L'approche vise à sécuriser les systèmes d'exploitation, les serveurs, les postes de travail et les applications critiques, afin de prévenir les attaques potentielles telles que les virus, les logiciels malveillants et les intrusions.

Accompagnement numérique :

Accompagnement des clients dans leur transition numérique en intégrant des solutions de sécurité dès la conception de leurs projets informatiques.
Cette approche proactive garantit une sécurité intégrée à chaque étape du développement numérique de l'entreprise cliente.

Sensibilisation et formation :

EFFSEIT propose des programmes de sensibilisation et de formation pour renforcer la culture de la cybersécurité parmi les employés des entreprises.
Ces programmes visent à réduire les risques liés aux erreurs humaines et à accroître la vigilance face aux menaces en constante évolution.

Audit de gestion de la sécurité :

L'entreprise réalise des audits approfondis pour évaluer et améliorer la sécurité des systèmes d'information.

Elle aide également les entreprises à mettre en place, optimiser et sécuriser leur réseau informatique, ainsi qu'à concevoir leur système d'information et applications.

Organigramme

Romain CERUTTI
Président

EFFSEIT

Guillaume MATHEN
Vice-Président

Les missions données par l'entreprise

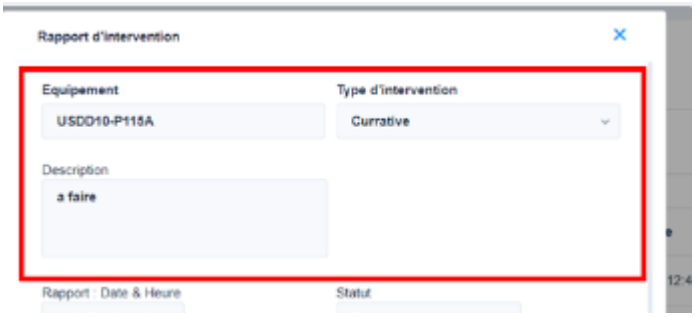
Les missions principales :

L'équipe m'a donné un bon nombre de missions principales. Les missions principales sont toutes les missions qui tournent autour du front-end. Nous étions trois développeurs dont deux en front-end et un en back-end. Nous avons donc dû nous partager les missions front-end à deux le but étant d'avancer le plus possible le site E-GMAO. C'est un site qui a pour but d'être déployé dans des industries ou autres afin faciliter la gestion de parc de machine, mais également la gestion des pièces, l'entretien des machines ainsi que les coûts. Voici donc toutes les missions principales qu'ils ont pu me donner durant mon stage chez EFFSEIT :

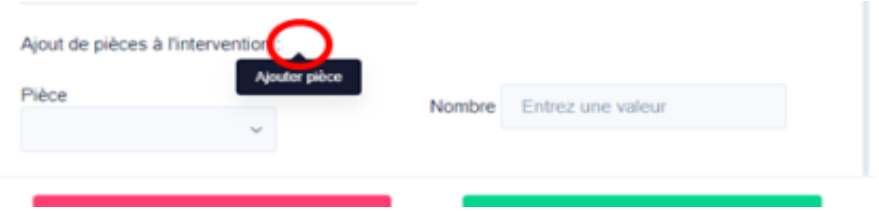
- Formation Angular (tour-of-heroes).
- Inspection du code et des problèmes rencontrés sur l'application Web
- Correction du tooltip d'édition des utilisateurs (169).

```
<nb-icon icon="edit-outline" pack="eva"></nb-icon>
<span class="action-text">Éditer</span>
<span class="action-text">Informations</span>
</button>
```

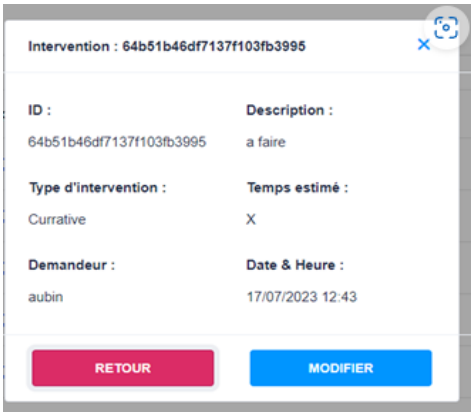
- Corriger la description, l'équipement et le type d'intervention dans le formulaire (152).



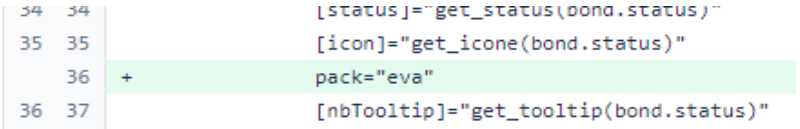
- Correction de l'ajout de plusieurs pièces au formulaire de rapport (153).



- Correction de l'affichage des informations d'une intervention possédant un rapport (154).



- Correction de l'affichage des icon-boutons des commandes (externes) (170).



- Correction de l'affichage du bouton de rapport pour une intervention close (146).
- Création d'une fonction fermer-intervention.
- Création d'une fonction ouvrir-intervention.

- Changement de bouton "Historique" par "Clôturées" (176)
- Modification du rapport (176)

Les missions secondaires :

- Apprentissage des fonctions en back-end afin de mieux comprendre comment faire mes fonctions en front-end.
- Installation des bureaux.
- Création d’articles
- Déploiement de l’application web

Les outils utilisés

Visual Studio Code :

Visual Studio Code a été utilisé comme environnement de développement pour travailler sur le projet Angular. Visual Studio Code est un éditeur de code source léger et puissant, offrant des fonctionnalités avancées telles que la coloration syntaxique, la complétion automatique et le débogage intégré pour TypeScript, le langage utilisé dans Angular.

Visual Studio Code a été intégré avec Jira et Bitbucket. En utilisant des extensions et des fonctionnalités intégrées, j'ai pu lier mon environnement de développement à nos outils de suivi de projet et de gestion de code source. Cela m'a permis de suivre les tâches assignées dans Jira et de synchroniser mes commits avec les tickets correspondants. En outre, j'ai utilisé les fonctionnalités de gestion de Git intégrées dans Visual Studio Code pour créer et basculer entre les branches, fusionner les modifications et gérer les conflits de manière transparente, facilitant ainsi le processus de développement et de déploiement dans notre workflow.

Angular :

Angular est un framework open-source développé par Google, utilisé pour créer des applications web côté client.

Jira :

Jira a été utilisé comme outil de suivi de projet pour gérer les tâches et les tickets associés aux missions du projet. Jira est une plateforme de gestion de projet agile qui permet de créer, suivre et gérer les tickets, en fournissant une vue d'ensemble claire des tâches assignées et de leur statut.

J'ai principalement utilisé Jira pour créer et suivre les tickets liés aux missions assignées, en détaillant les objectifs, les exigences et les étapes nécessaires à leur réalisation. J'ai pu collaborer avec les membres de l'équipe en commentant et en mettant à jour les tickets au fur et à mesure de l'avancement du projet.

Bitbucket :

Bitbucket a été utilisé comme plateforme de gestion de code source pour le projet front-end. Bitbucket est un service de contrôle de version distribué qui facilite la collaboration au sein de l'équipe de développement en permettant de stocker, gérer et partager le code source du projet.

Dans le cadre de ce projet, Bitbucket a principalement été utilisé pour récupérer le code source du front-end. J'ai pu cloner le référentiel Git contenant le code front-end depuis Bitbucket vers mon environnement de développement local à l'aide de la commande git clone.

Teams :

Microsoft Teams a été notre plateforme principale de communication et de collaboration tout au long du projet. Teams est une plateforme de communication unifiée qui offre des fonctionnalités telles que la messagerie instantanée, les appels vidéo, les réunions en ligne, le partage de fichiers et l'intégration avec d'autres outils de productivité.

ChatGPT :

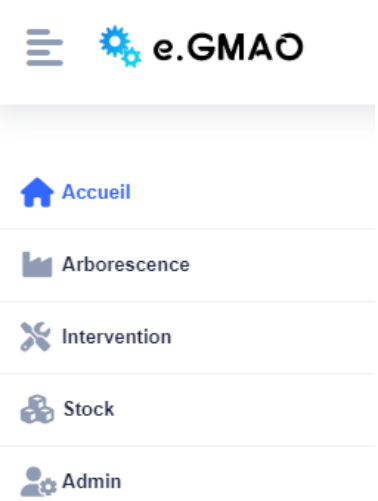
ChatGPT est un modèle de langage développé par OpenAI, basé sur l'architecture GPT (Generative Pre-trained Transformer). Il est conçu pour générer du texte de manière naturelle et répondre à une grande variété de requêtes et de questions posées par les utilisateurs.

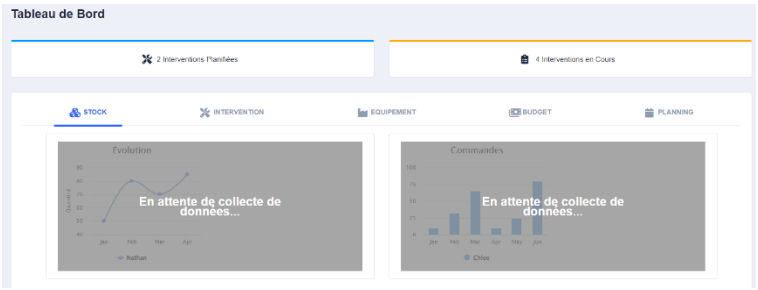
Mes missions réalisées

Missions principales

L'application Web sur laquelle nous avons travaillé avait déjà été intégré dans une entreprise afin de la tester.

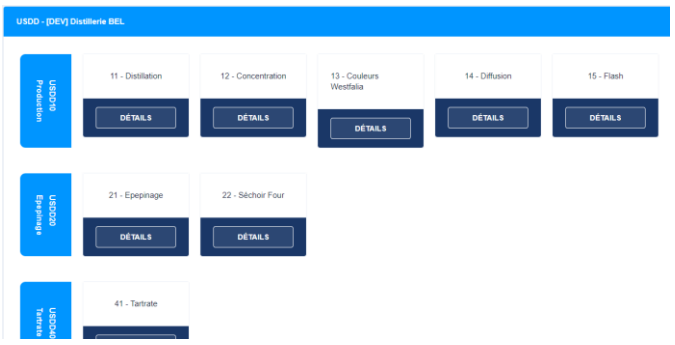
Commençons en présentant la composition de l'application Web GMAO. Pour vous donner une idée du travail global à faire. Premièrement, Il y a un menu qui sépare le site en 5 parties (Accueil, Arborescence, Intervention, Stock et Admin)





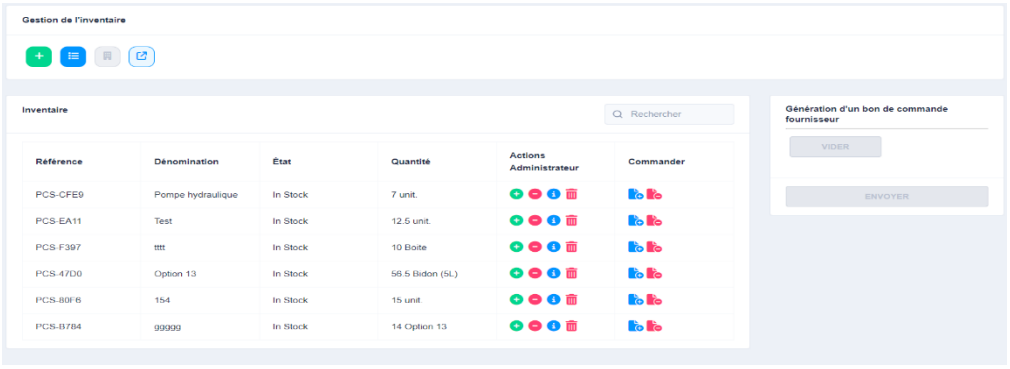
Il y a la page d'accueil qui regroupe un certain nombre d'information sur les stocks, les interventions, les équipements, les budgets et les plannings.

Ensuite il y a la page arborescence qui regroupe toutes les machines. En cliquant sur Détails, nous arrivons sur la page de la machine en question.

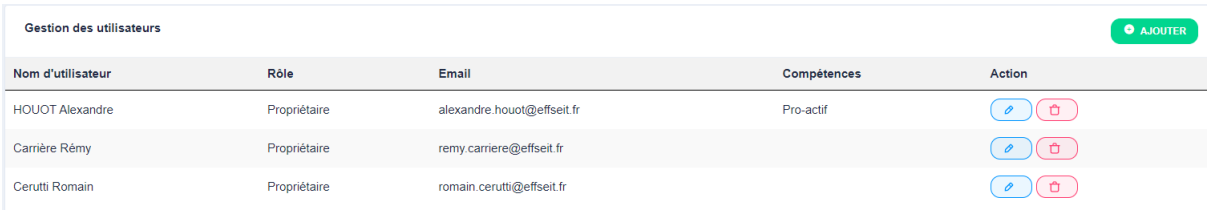


Il y a également une page intervention qui sert à voir toutes les interventions planifiées ou terminées. Nous pouvons créer des interventions, les supprimer, faire un rapport et terminer l'intervention.

Voici la page Stock qui regroupe toutes les pièces. Sur cette page nous pouvons ajouter une pièce, la supprimer ou la commander. De cette page, nous pouvons accéder à d'autres pages comme la liste des fournisseurs ou encore les demandes externes.

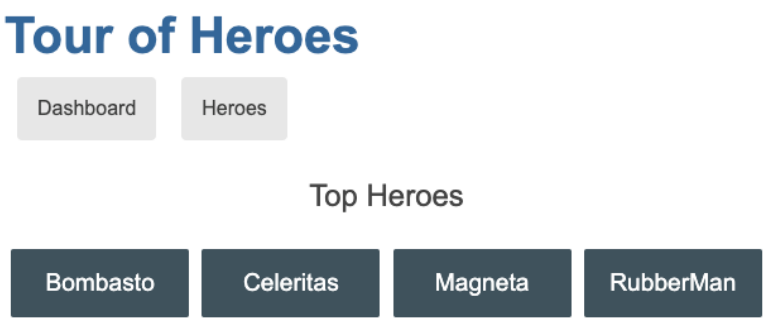


La page Admin est la page où se regroupe tous les utilisateurs. Nous pouvons en créer, les modifier ou encore les supprimer.

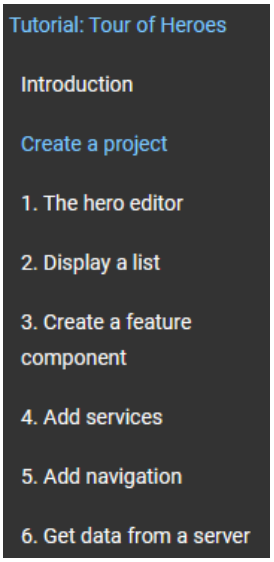


Pour finir, il y a une page Paramètres afin d'accéder aux informations de l'utilisateur et de modifier le mot de passe de celui-ci.

Pour ma première mission, j'ai découvert pour la première fois Angular. « **Tour of Heroes** » est un tutoriel très populaire fourni par l'équipe de développement d'Angular. C'est un guide pas à pas qui permet aux développeurs d'apprendre les bases d'Angular en créant une application de type "Tour des héros". L'application met en œuvre des concepts clés d'Angular tels que les composants, les services, le routage, les formulaires et la gestion des données.



Le Tour des héros est une application simple où les utilisateurs peuvent afficher une liste de héros, en sélectionner un pour afficher ses détails, et même modifier les détails du héros sélectionné. Ce tutoriel est souvent utilisé comme point de départ pour les nouveaux développeurs Angular afin de leur donner une compréhension pratique de la façon dont Angular fonctionne et de comment construire des applications avec ce Framework.



Nous commençons directement par l'introduction du tutoriel, il nous explique ce qu'on va devoir faire et ce qu'il est attendu par ce tutoriel.

Ensuite nous passons à la création du projet, nous pouvons créer un projet directement sur notre projet ordinateur ou alors commencer le projet sur le site web Stackblitz qui nous permet de commencer un projet Angular avec quelques bases et avoir des aides supplémentaires. J'ai donc décidé de commencer le projet Tour of Heroes sur Stackblitz afin de comprendre plus facilement ce que je faisais.

Après cette création de projet, nous sommes guidés sur comment créer un nouveau composant pour afficher les informations sur les héros et le placer dans sur le site web du projet.

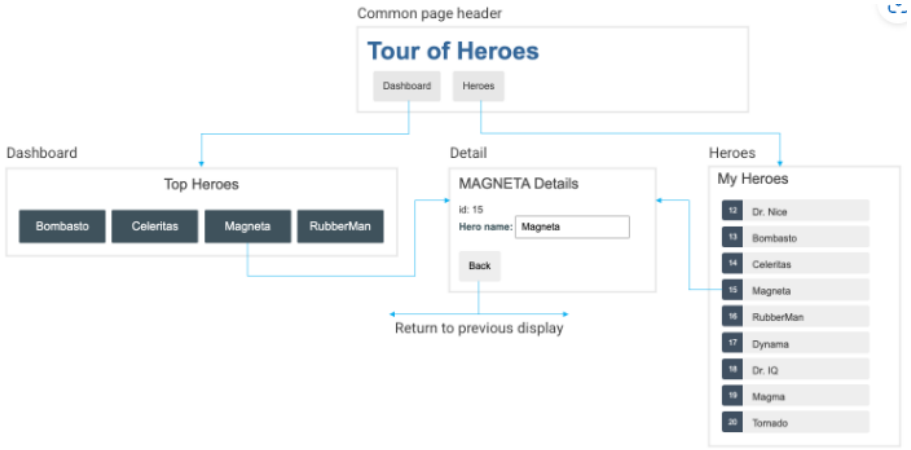
Nous passons donc à la troisième étape du projet, elle nous montre comment développer l'application Tour of Heroes afin afficher une liste de héros et elle permet aux utilisateurs de sélectionner un héros et d'afficher les détails du héros.

À l'heure actuelle, le site web affiche à la fois la liste des héros et les détails du héros sélectionné. Il n'est pas possible de conserver toutes les fonctionnalités dans un seul composant au fur et à mesure que l'application se développe. Ce didacticiel divise les composants volumineux en sous-composants plus petits, chacun se concentrant sur une tâche ou un flux de travail spécifique. La première étape consiste à déplacer les détails du héros dans un fichier séparé, réutilisable et à obtenir le composant « **HeroDetailComponent** »

Juste après ceci, il fallait traiter la partie des récupérations des données.

Pour l'avant dernière étape, il fallait ajouter une vue de tableau de bord, ajouter la possibilité de naviguer entre les vues **Héros** et **Tableau de bord**, lorsque les utilisateurs cliquent sur le nom d'un héros dans l'une ou l'autre vue, ils accèdent à une vue détaillée du héros

sélectionné, lorsque les utilisateurs cliquent sur un lien dans un e-mail, ça ouvre la vue détaillée d'un héros.



Enfin pour finir, ce tutoriel, le but de la dernière étape était d'ajouter les fonctionnalités de persistance des données suivantes à l'aide de l'application **HttpClient**. Obtenir des données des héros avec des requêtes **HTTPHeroService**. Les utilisateurs peuvent ajouter, modifier et supprimer des héros et enregistrer ces modifications via http. Les utilisateurs peuvent rechercher des héros par leur nom.

Après avoir terminé le tutoriel Tour of Heroes, il m'a été demandé de me pencher sur le code de GMAO. Ils m'ont donc donné un mail @effseit.fr afin de pouvoir me connecter à Jira et Bitbucket pour récupérer le code de l'application Web afin de le lancer en local. Une fois la connexion faite, l'inspection, la modification, la suppression du code étaient possibles sans affecter l'appli Web car il faut faire une demande afin que le code passe sur l'application « en réel ». Une bonne journée a été nécessaire pour apprendre et comprendre des méthodes ou autres que je n'avais pas vu dans le tutoriel Tour of Heroes. J'ai beaucoup appris sur Angular et ça m'a permis de voir que ce Framework est très complet.

Je suis donc aller rechercher les bugs ou erreurs que je pouvais trouver sur GMAO afin de faire une liste de toute les petites choses à revoir. Enfin, je suis allé voir les tickets présents sur Jira afin de voir réellement toutes les missions à faire. Ensuite, nous nous sommes attribué des tickets avec Parceval qui est stagiaire en front-end comme moi pour savoir sur quelle mission nous allions travailler.

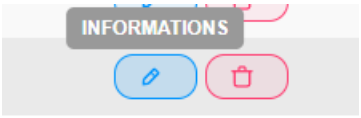
Mission 1

A partir de là, le codage pour l'application Web a réellement commencé: Tout d'abord une mission très simple (le ticket 169), il suffisait de changer le texte qui apparait quand nous passons la souris sur le bouton informations de la page Admin. Il a fallu remplacer EDITER par INFORMATIONS.

Le plus compliqué dans cette mission a été de trouver le bon fichier et la bonne ligne à modifier, car, encore non-habitué à voir un agencement de fichiers tel que celui d'Angular.

Informations dans le fichier ./src/app/pages/admin/admin.component.html

```
1  <span class="action-text">Informations</span>
```



Cette première mission de développement front-end pour l'application Web GMAO a été réussie.

Il suffisait donc de « Commit and Push » le code, c'est-à-dire d'envoyer mon code sur une nouvelle branche de Jira afin que mon code soit accepté par Romain Cerutti.

Mission 2

Correction de la description, l'équipement et le type d'intervention dans le formulaire (le ticket 152). Le but était de fixer les paramètres équipement, type d'intervention et description c'est-à-dire qu'une fois la création de l'intervention, quand nous faisons le rapport de celle-ci, ça nous indique les informations de l'intervention sans pouvoir les modifier. Ces informations apparaissent sous forme de Pop-up qui renvoie toutes les informations de l'interventions et aussi, les nouvelles cases où nous pouvons remplir le rapport, indiquer le temps passé sur cette intervention, la date du rapport ainsi que les pièces et le nombre de pièces utilisées pour l'intervention.

Tout d'abord, j'ai changé la forme des informations.

J'ai remplacé un champ où nous pouvons choisir par une case où nous pouvons seulement voir les informations

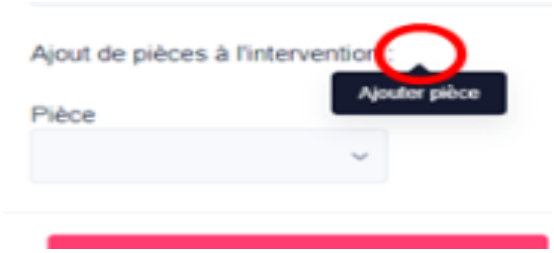
```
<nb-select placeholder="Type d'intervention" formControlName="type" [formControl]="selectedItemFormControl">
    <nb-option value="Currative">Currative</nb-option>
    <nb-option value="Préventive">Préventive</nb-option>
</nb-select>
iv>
<input nbInput formControlName="type" readonly>
```

Avec le « readonly », j'ai pu fixer toutes les informations où il était nécessaire qu'elles restent inchangeables. Deuxième mission pour GMAO terminée.

Un « Commit and Push » pour transmettre mes modifications à mon maître de stage.

Mission 3

Le ticket 153 qui est la correction de l'ajout de plusieurs pièces au formulaire de rapport. Le problème : Les icones pour l'ajout des pièces lors des rapports d'intervention n'apparaissent pas du tout.



La seule chose qui apparait était le texte « **Ajouter pièce** », c'est-à-dire que le bouton fonctionnait normalement mais il ne voulait pas s'afficher.

Une fois la bonne ligne trouvée, une icône était bien définie pour le bouton :

```
icon="plus-circle-outline"
```

J'ai donc cherché s'il n'y avait pas du CSS qui venait cacher cette icône ou encore le « display: none; » qui est une propriété CSS qui permet de cacher un élément HTML sur une page web. Lorsqu'un élément est rendu avec cette propriété, il est complètement retiré du flux de la mise en page, ce qui signifie qu'il n'occupe pas d'espace sur la page et n'est pas visible pour les utilisateurs. Puis, je me suis rappeler que je pouvais cliquer sur le bouton donc il ne pouvait pas être en « display :none ; »

Des recherches sur l'utilisation de ce pack « eva-icons » m'ont indiqué qu'il fallait mettre une ligne supplémentaire dans le HTML afin d'importer toutes les icônes de ce pack pour ensuite les utiliser.

C'est la ligne suivante qu'il fallait ajouter dans le balise du bouton

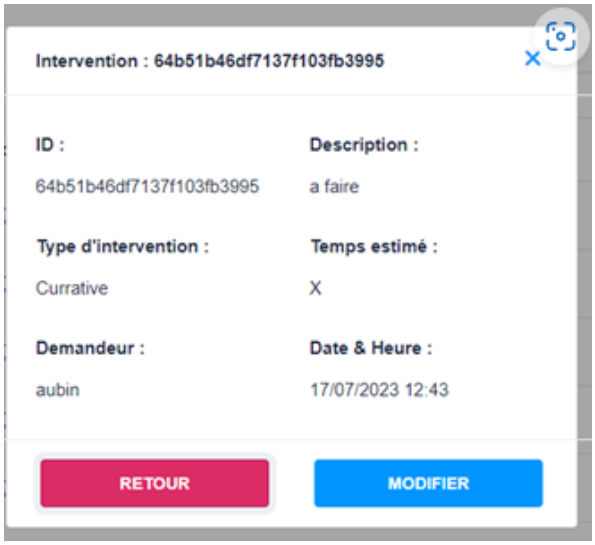
```
pack="eva"
```

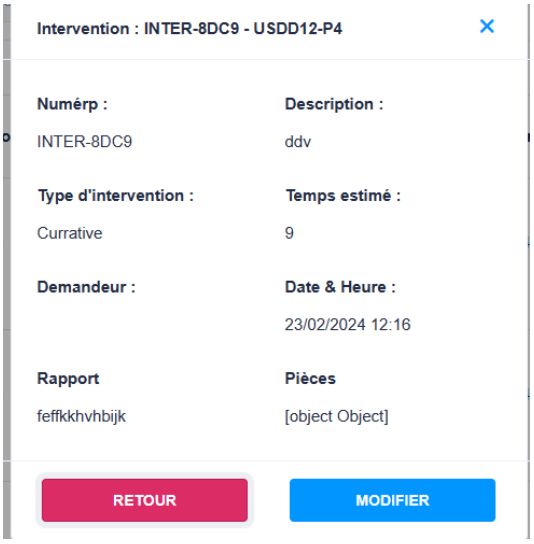
Après l'ajout de cette ligne, le bouton apparaissait parfaitement bien, je venais donc de finir cette mission.

« Commit and push » par la suite, mission terminée.

Mission 4

Le ticket 154, c'était la correction de l'affichage des informations d'une intervention possédant un rapport. Pour que ça soit plus clair, il fallait juste ajouter les informations « rapport » et « pièces » dans la pop-up d'informations de l'intervention.





- Observation de la constitution du code afin de comprendre comment importer les données nécessaires à la correction.
- Visite du fichier ou se trouvaient toutes les informations sur les interventions.

```
<div class="col-6">
  <p><b>Rapport</b></p>
  <p>{{interv.rapport }}</p>
</div>
<div class="col-6">
  <p><b>Pièces</b></p>
  <p>{{interv.piece }}</p>
</div>
```

Test de mes modifications, le résultat n'était pas réellement celui attendu. C'est-à-dire que pour le rapport, ça marchait mais pour les pièces, ça n'affichait pas les pièces :

Pièces

[object Object]

Romain Cerutti m'a dit qu'au final je pouvais arrêter ce que je faisais car il fallait refaire totalement les pièces : arrêt de la mission.

Mission 5

Correction de l'affichage des icon-boutons des commandes (externes) (170). Pour cette mission, c'est le même problème que pour le ticket 153, c'est-à-dire que c'est un problème d'importation de pack d'icônes. Connaissant l'erreur, il n'a fallu que 2 minutes afin de trouver le bon fichier où il fallait rajouter l'appelle du pack d'icônes en question. C'est donc le pack eva-icons qu'il fallait appeler :

```
pack="eva"
```

Mission 6

Le ticket 146. Le but de cette mission était la modification de beaucoup de fonctionnalités et l'ajout de boutons ainsi que des fonctions afin de pouvoir rendre la page intervention opérationnelle.

Premièrement, faire fonctionner le bouton historique qui doit normalement montrer toutes les interventions qui sont en statut terminer.

Recherche dans TypeScript afin de comprendre, puis découverte du fichier page.services.ts qui regroupe toutes les fonctions en front-end qui peuvent être appelées.

La fonction qui affichait soit les interventions terminées soit les interventions ouvertes avaient déjà été faite.

Pour que cette fonction marche, il fallait rajouter un attribut dans la définition des interventions.

```
export interface INTERVENTION {
  piece: any[];
  date_debut: Date;
  date_fin: Date;
  _id: BSON.ObjectID;
  demandeur?: string;
  equipement: string;
  type: string;
  urgent: boolean;
  description: string;
  valide_par: string;
  date_heure: Date;
  temps_estime: string;
  temps_passe:string;
  statut: string;
  rapport: string;
  close: boolean;
  etat_apres:string;
  etat_avant:string;
}
```

Ajout de cet attribut dans la création de l'objet intervention puis création d'une fonction qui vérifie si le close de chaque intervention était vrai ou faux puis qui affiche les bonnes interventions au bon moment.

J'ai demandé à Enzo de rajouter directement dans la base de données des valeurs « True » donc « vrai » afin de voir si les interventions s'affichaient bien au bon endroit.

Dès l'ajout de quelques données, la fonction a pu être opérationnelle.

Ajout d'une petite case à cocher dans le formulaire de rapport de l'intervention qui change la valeur de close de l'intervention. La réouverture et la fermeture d'une intervention est devenu possible.

Finalement, Romain Cerutti nous a précisé qu'il voulait un tout autre type de fonctionnement. Il voulait un bouton à côté du bouton « Rapport » pour terminer une intervention. Puis lorsque l'intervention est dans l'historique un bouton retour « ouvrir » afin de la renvoyer avec toutes les autres interventions.

J'ai donc demandé à Enzo de faire deux fonction en back-end afin de terminer et d'ouvrir une intervention pour que je fasse de mon côté deux fonctions en front-end en m'appuyant sur ses fonctions :

```
public fermer_intervention(id_interv): Promise<any[]> {
  return new Promise((resolve, reject) => {
    this.realmService
      .realm_function('fermer_intervention', id_interv)
      .then((doc: any[]) => {
        resolve(doc);
      })
      .catch((err) => {
        reject(err);
      });
  });
}
```

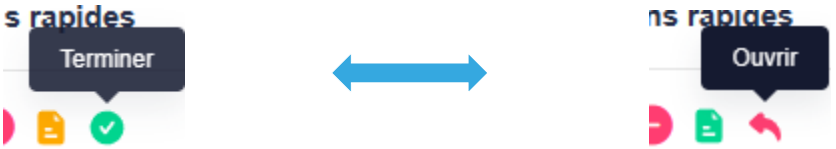
```
public ouvrir_intervention(id_interv): Promise<any[]> {
  return new Promise((resolve, reject) => {
    this.realmService
      .realm_function('ouvrir_intervention', id_interv)
      .then((doc: any[]) => {
        resolve(doc);
      })
      .catch((err) => {
        reject(err);
      });
  });
}
```

Test avec succès des fonctions front-end et back-end en faisant une fonction dans le composant qui teste la fonction **fermer_intervention**.

Ensuite, réalisation avec succès d'une fonction qui appelle **fermer_intervention** ou **ouvrir_intervention** en fonction de la valeur du close de l'intervention.

Présentation des modifications à Monsieur Cerutti afin qu'il me donne plus de détails sur ce qu'il voulait au niveau de la présentation des boutons ou autres avant de « Commit and Push ».

Il m'a répondu qu'il fallait différencier le bouton Ouvrir et Fermer ainsi qu'écrire Terminer et Ouvrir quand on passe la souris sur les boutons.



Obtention du résultat ci-dessus, partage avec mes camarades ainsi que mes supérieurs et comme ils ont validé le résultat, envoi de mon code à Romain Cerutti

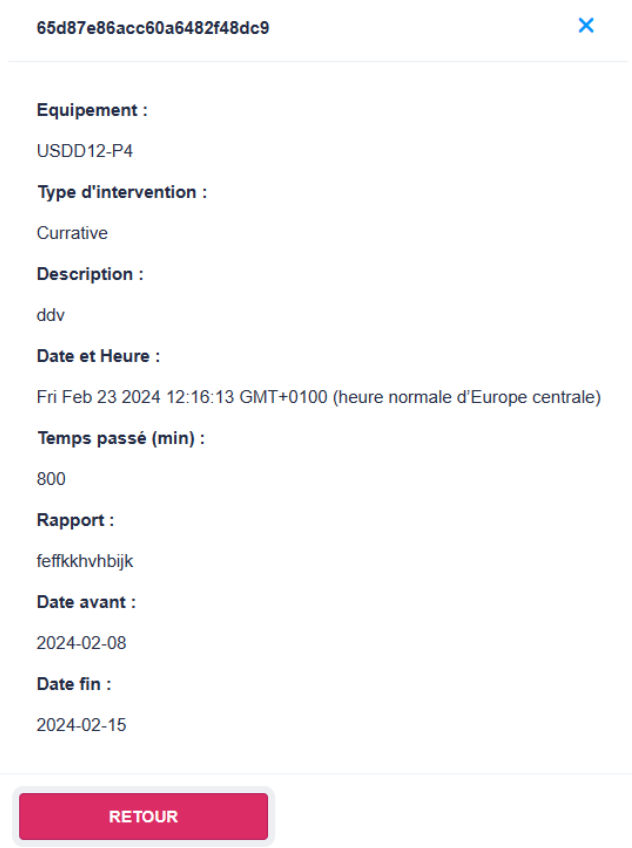
Mission 7

Le ticket 176 : faire le changement de "Historique" par "Clôturées". Quelques secondes ont suffi pour de trouver le bon fichier à modifier car je commençais à très bien connaître la composition du code et l'emplacement des fichiers. Modification dans le fichier « liste-intervention.component.html ».

Mission 8

Le rapport de l'intervention, il fallait le réorganiser, créer un rapport non modifiable et faire fonctionner l'ajout de pièce dans le rapport. C'est la suite du ticket 176.

Tout d'abord, création un rapport non modifiable c'est-à-dire une vue du rapport avec quelques informations primordiales sur l'intervention que l'on ne peut pas changer. A partir d'un travail réalisé précédemment par un précédent développeur.



La Modification est un succès, mais il m'a été demandé de corriger la disposition des éléments avec BootStrap.

Pour la refonte du rapport d'intervention, Enzo est venue m'aider car à ce moment-là il n'avait aucune autre mission. Enzo avait déjà fait un peu de BootStrap donc nous avons pu établir le schéma du rapport sur un tableau avant de le faire réellement sur la GMAO.

Nous avons présenté notre schéma à Romain afin qu'il nous valide le schéma. Il nous a fait changer quelques petits détails sur la position des éléments puis nous avons commencé la modification sur l'application Web.

Premièrement, nous avons enlevé tout le contenu du rapport afin d'y voir plus clair et de refaire toute la structure du rapport. En jouant uniquement avec les classes « row » et « col-12 » ou « col-6 » qui servent à créer des colonnes ou des lignes avec une certaine taille donnée, nous avons réussi à tout placer directement .

Nous avons partagé le résultat du rapport d'intervention avec Romain Cerutti qui nous a tout de suite dit qu'il était très content du résultat par rapport à l'ancien rapport d'intervention. Je vous laisse voir l'annexe 1 et 2 pour comparer.

Pour finir avec ma dernière mission, il me restait plus qu'à faire fonctionner le bouton d'ajout de plusieurs pièces. J'ai donc réorganisé le placement de la case de sélection des pièces et la case et ça a directement fonctionné.

C'était le dernier jour et la dernière heure de mon stage.

Missions secondaires

Je vais donc maintenant passer aux missions secondaires que j'ai pu faire durant mon stage.

Premièrement, afin de pouvoir faire mes fonctions en front-end, j'ai dû étudier les fonctions en back-end pour bien envisager les fonction et aussi pour bien associer le front-end et le back-end et bien communiquer avec la base de données.

Ensuite, pour ma deuxième mission secondaire, nous avons installé les bureaux dans les locaux du bas. Nous avons installé les écrans, les pieds d'écrans, les chaises…

Grâce à ça, Enzo, Parceval et moi avons pu travailler en double écrans ce qui est indispensable aujourd'hui pour des développeurs.

Durant le stage, toute l'équipe devait faire des articles pour le référencement de GMAO. Pour faire ces articles, nous avons suivi une formation sur teams afin de savoir comment faire. Ils nous ont donné un document avec toutes les informations nécessaires à la création des articles. Il suffisait de trouver des titres avec un certain plan à suivre. Ensuite, avec l'aide de chat GPT, il fallait créer des articles de 200 à 300 mots. Nous devions évidemment changer les phrases et vérifier tout le contenu.

Et pour finir, nous avons assisté au déploiement de l'application Web GMAO, Romain nous a montré comment la mettre en production et j'ai beaucoup appris ce jour-là.

Description de code

Rapport intervention

Pour cette partie, il suffisait juste de comprendre comment organiser les lignes (**row**) et les colonnes (**col**). Ici, on commence par créer une div **row** qui contient une col-12 c'est-à-dire une colonne qui prend toute la longueur de la ligne, puis à nouveau une ligne qui contient deux colonnes côte à côte. Avec BootStrap, la taille de la colonne se coupe en 12 parties, donc col-12 c'est toute la longueur et par exemple col-6 c'est la moitié de la longueur.

Nous avons bien deux colonnes qui prennent la moitié de de la ligne chacune et qui se place côte à côte.

```
<nb-card-body>
    <div class="row">
        <div class="col-12">
            <div class="row">
                <div class="col-6">
                    <div class="row">
                        <p>Équipement concerné</p>
                        <input nbInput formControlName="equipement" readonly
                        class="col-12">
                    </div>
                    <div class="row">
                        <p>Type d'intervention</p>
                        <input nbInput formControlName="type"  class="col-12" readonly>
                    </div>
                    <div class="row">
                        <p>État avant intervention</p>
                        <input nbInput formControlName="etat_avant"  class="col-12" readonly>
                    </div>
                    <div class="row">
                        <p>Date et heure de l'intervention</p>
                        <input nbInput formControlName="date_heure"  class="col-12" readonly>
                    </div>
                    <div class="row">
                        <p>Temps estimé de l'intervention</p>
                        <input nbInput formControlName="temps_estime"  class="col-12" readonly>
                    </div>
                    <div class="row">
                        <p>Description de l'intervention</p>
                        <textarea nbInput placeholder="" rows="3" formControlName="description" readonly
                        class="col-12"></textarea>
                    </div>
                </div>
                <div class="col-6">
                    <div class="row">
                        <p>Date et heure début</p>
                        <input type="date" #autoInput nbInput class="col-12" formControlName="date_debut"
                        (input)="onChange()" placeholder="Entrez une valeur" min="0" />
                    </div>
                    <div class="row">
                        <p>Date et heure fin</p>
                        <input type="date" #autoInput nbInput class="col-12" formControlName="date_fin"
                        (input)="onChange()" placeholder="Entrez une valeur" min="0" />
                    </div>
                    <div class="row">
                        <p>État après intervention</p>
                        <div class="row col-12">
                            <nb-select placeholder="Etat de la machine apres" formControlName="etat_apres">
                                <nb-option value="En-service">En service</nb-option>
```

A l'intérieur de chaque col-6, nous y mettons des lignes **row** contenant toutes les informations, les inputs, les **textareas** et les nb-selects qui servent à récolter des données rentrées par l'utilisateur. Grâce à ça, chaque ligne **row** que nous rajoutons dans le col se mettent les unes sous les autres. Ensuite il suffit de mettre les informations à l'intérieur de la **row** en col-12 afin que chaque élément prenne toute la longueur de la col-6.

BootStrap met automatiquement des **margin** c'est-à-dire des petits espaces entre les éléments afin que toutes les lignes **row** ne soient pas collées mais un peu espacées.

Pour finir, pour la première col-6, j'ai rajouté l'argument **readonly** ce qui permet de bloquer la modification d'un élément. Romain voulait que toutes les informations dans la première col-6 et donc dans la colonne de gauche soient non-modifiables. Par conséquent, les autres informations se trouvant à droite dans la deuxième col-6 peuvent être modifiées dans le rapport.

Moyens de communication

Nous avons utilisé plusieurs moyens de communication, que ça soit pour des réunions, des visioconférences, pour échanger sur tout ou même pour nous donner des missions à réaliser durant le stage chez EFFSEIT.

Premièrement, Romain Cerutti nous a ajouté dans un groupe Teams dans lequel nous avions accès à « Général » dans lequel il y avait beaucoup de documents rangés dans des dossiers auxquels nous avions également accès. Il y avait aussi des informations sur le groupe Teams. Par exemple, pour la création des articles, nous devions ranger des images dans des dossiers pour les retrouver facilement, en y mettant le titre de l'article comme nom de dossier. Nous pouvions également remplir les articles directement dans l'une des sous-parties et aussi les modifier dans une autre sous-partie .

Ils nous ont aussi donné l'accès à une partie « Stage SN2 » séparé en trois sous-parties. Premièrement, il y avait une sous-partie « Publications » où nous pouvions remplir un compte rendu sur ce que nous avions fait dans la journée, ce qui nous avaient bloqué ou ralenti dans la journée ainsi que ce que nous envisagions de faire le lendemain.

Date : 14-02-2024

Ce Que j'ai Fait :

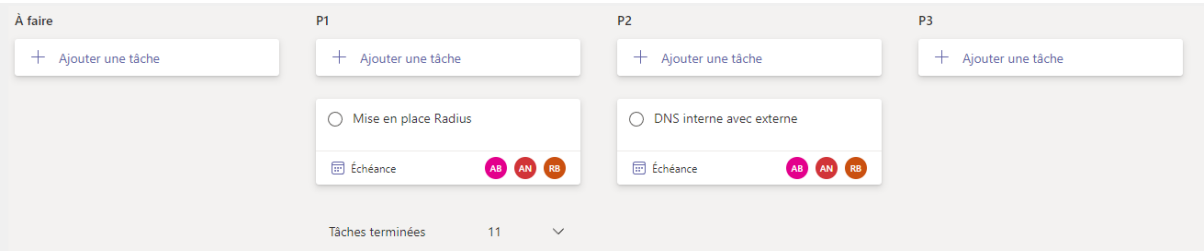
Ce qui me bloque :

Ce que je vais faire demain :

Grâce à ça, je pouvais garder une trace de tout ce que je faisais durant le stage, ce qu'y m'a vraiment aidé à garder le fil.

Dans cette partie, il y avait aussi une sous-partie « fichiers » dans laquelle nous retrouvions par exemple des documents pour faire des articles, nos titres d'articles ou d'autres documents concernant les réseaux. Il y avait donc la possibilité de rajouter nos documents afin d'en faire profiter à toute l'équipe.

Pour finir avec la composition de cette partie, il restait la sous-partie Stage – Tasks qui regroupait toutes les missions globales à faire qui n'avaient pas de lien avec le code de GMAO comme par exemple le tutoriel Tour of Heroes ou encore des missions que les réseaux avaient à faire.



Ça nous arrivait de faire des petites formations par exemple pour apprendre à rédiger des articles ou même des petites réunions afin de savoir où est-ce que nous en étions.

Pour finir, le dernier moyen de communication que nous avons utilisé est Jira, grâce à ce site Web, nous pouvions directement nous attribuer des tickets, créer des tickets ou seulement regarder des tickets afin de savoir ce qu'il fallait faire en lien direct avec le code de l'application Web.

Conclusion

Mon immersion chez EFFSEIT, une entreprise à la pointe de la cybersécurité, a été marquée par l'utilisation de la méthodologie Agile, un pilier dans le monde du développement logiciel. Cette expérience a révélé l'importance de l'agilité dans le succès de mon parcours de stagiaire, en particulier dans le développement de l'application Web GMAO.

Le tutoriel "Tour of Heroes" a été ma porte d'entrée dans cet univers. Cette méthode d'apprentissage direct m'a permis de saisir les subtilités d'Angular, me dotant des outils nécessaires pour affronter des défis de taille.

L'outil de gestion de projet Jira s'est révélé indispensable pour visualiser les différentes missions qui m'étaient assignées. La répartition des tâches via des tickets et la collaboration avec l'équipe ont optimisé mon organisation sur les objectifs du projet.

La collaboration entre Visual Studio Code, Jira et Bitbucket a transformé notre manière de travailler. Cette intégration a fluidifié le développement et le déploiement, renforçant la cohésion de l'équipe et accélérant la livraison des fonctionnalités.

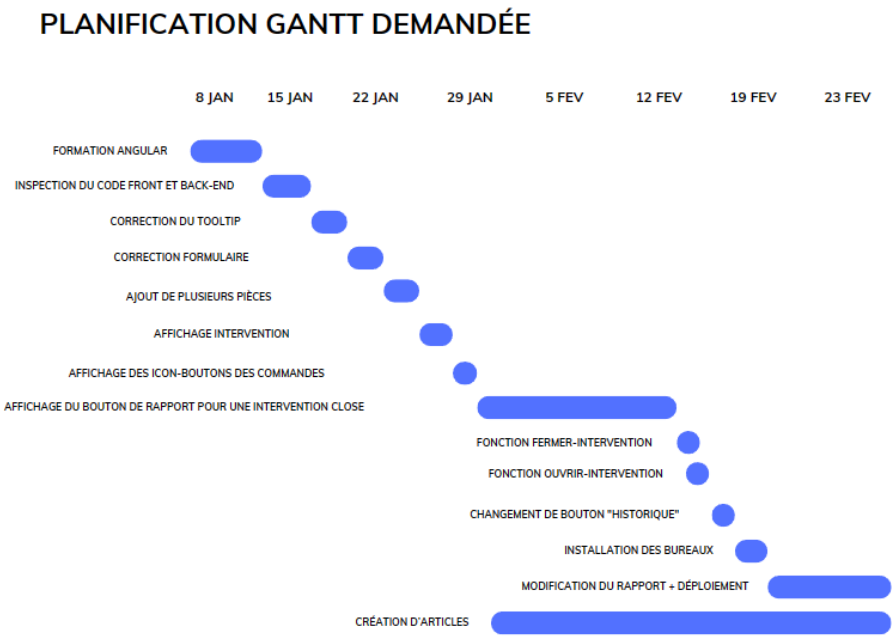
La communication a été un autre pilier de notre réussite. Grâce à Microsoft Teams et Jira, les échanges d'idées, la résolution de problèmes et la coordination des efforts se sont faits sans accroc, même à distance. Les réunions et visioconférences régulières ont consolidé notre vision commune, a ssurant une avancée cohérente du projet.

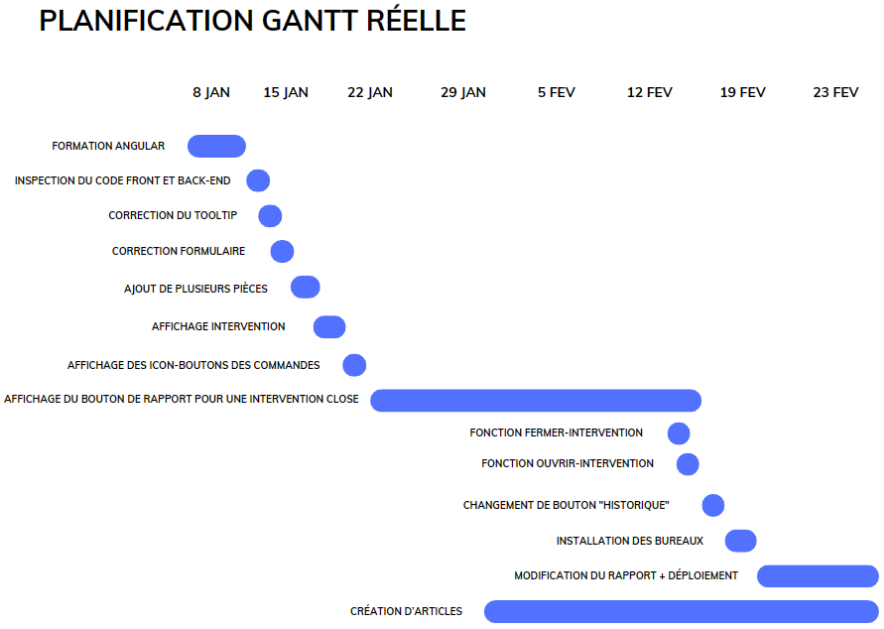
L'agilité nous a également appris à être réactifs et adaptables face aux imprévus. Cette capacité à évoluer et à ajuster le tir en fonction des besoins du projet a été déterminante pour la qualité et la p ertinence de nos livrables.

En résumé, mon passage chez EFFSEIT a illustré la valeur de l'approche Agile. Ce stage a mis en lumière le rôle joué par la collaboration, la flexibilité et la clarté dans mon adaptation rapide à l'équipe et dans l'atteinte des buts fixés, le tout en maintenant
un niveau de qualité et d'efficacité remarquables.

Planification Gantt

Globalement, j'ai été plus rapide que le temps voulu pour mes missions sauf pour une mission où on s'était mal compris avec Romain Cerutti ce qui a donc par conséquent demandé plus de temps que prévu.





Glossaire

- **Front-end** : c'est la partie du code qui est reçue par le navigateur Web. Il s'agit finalement des éléments du site web que l'on aperçoit à l'écran et avec lesquels on pourra interagir. Ces éléments sont composés de 3 langages : HTML, CSS et Javascript.

- **Back-end** : Le back end désigne les parties du code d'une application ou d'un logiciel permettant son fonctionnement et inaccessibles à l'utilisateur. On le désigne aussi sous le nom de couche d'accès aux données d'un logiciel ou d'une machine. Il inclut toutes les fonctionnalités nécessitant un accès et une navigation par des moyens numériques.

- **:hover** : c'est grâce à cette écriture qu'on met derrière le ".nom-de-classe" en CSS qu'on peut modifier une classe lorsqu'on passe la souris dessus.

- **Pop-up** : La pop-up est un terme marketing qui désigne une fenêtre qui s'affiche lors d'une navigation au sein d'un site web. Elle s'ouvre automatiquement et incite l'internaute à l'action. De plus petite taille que la fenêtre principale, elle chevauche le contenu de la page web qui est consultée.

- **En local** : c'est le fait de travailler directement sur sa machine pour éviter de modifier les fichiers du serveur. C'est une forme de sécurité.

- **Commit and Push** : Cette action est utilisée dans les systèmes de contrôle de version comme Git. "Commit" fait référence à l'action de valider les modifications apportées à un ensemble de fichiers dans un référentiel local. Cela crée un instantané des fichiers à un moment donné avec un message descriptif pour expliquer les modifications. Ensuite, "Push" envoie ces modifications du référentiel local vers un référentiel distant, tel qu'un serveur Git central, pour les rendre disponibles pour d'autres collaborateurs ou pour sauvegarder les modifications sur un serveur distant.

- **Framework** : Un framework est un ensemble de bibliothèques, d'outils et de conventions qui permettent aux développeurs de créer plus rapidement et efficacement des applications en fournissant une structure de base sur laquelle construire.

- **HTML** (HyperText Markup Language) : HTML est le langage de balisage standard utilisé pour créer et structurer le contenu des pages Web.

- **CSS** (Cascading Style Sheets) : CSS est un langage de feuilles de style utilisé pour définir la présentation visuelle et la mise en page des éléments HTML d'une page Web.

- **Javascript** : Javascript est un langage de programmation de scripts principalement utilisé pour rendre les pages Web interactives et dynamiques.

- **TypeScript** : TypeScript est un langage de programmation open-source développé par Microsoft. Il s'agit d'une extension de JavaScript qui ajoute des fonctionnalités de typage statique au langage. TypeScript permet aux développeurs de spécifier explicitement les types de données des variables, des paramètres de fonction et des valeurs de retour, ce qui facilite la détection précoce des erreurs et améliore la maintenance du code. Il est souvent utilisé pour le développement d'applications Web et est largement adopté dans l'écosystème de développement JavaScript.

Balise (HTML) : Élément de syntaxe utilisé pour marquer le début et la fin d'un élément dans une page HTML. Les balises sont généralement constituées de chevrons angulaires (< et >) et sont souvent utilisées par paires, avec une balise d'ouverture et une balise de fermeture.

Bibliographie

Durant mon stage, j'ai eu recours à divers sites internet. Non seulement pour me former sur le Framework Angular mais aussi afin de récupérer des informations, de communiquer avec l'équipe, de prendre connaissance des missions à faire, de récupérer le code de l'application Web et pour finir l'application Web en question pour vérifier les changements réalisés.

Tout d'abord, le site GMAO

URL : https://gmao.effseit-lab.fr

Ensuite, pour en apprendre davantage sur le framework Angular, j'ai suivis un petit tutoriel Tour of Heroes

URL : https://angular.io/tutorial/tour-of-heroes

Le site angular.io nous conseillait d'utiliser le site stackblitz afin de coder directement sur le site et avoir des aides supplémentaires.

URL : https://stackblitz.com

J'ai utilisé le site Teams pour communiquer avec mes camarades et échanger nos problèmes, nos avancés et autres.

URL : https://teams.microsoft.com

Pour coder directement en local sur mon ordinateur, il m'a fallu aller sur le site BitBucket. Sur ce site-là, j'ai pu récupérer le code du front-end de l'application Web et commencer mes missions de développement sur GMAO.

URL : https://bitbucket.org

Justement, ayant le code de la GMAO, il me fallait aussi les missions à réaliser. C'est pour cela que je me rendais sur le site Atlassian afin d'accéder à Jira. Grâce à ça je pouvais voir toutes les missions et me les attribuer.

URL : https://atlassian.net

Je suis également allé sur le site Akveo pour me renseigner sur les noms icones à importer dans les bonnes situations.

URL : https://akveo.github.io/eva-icons/#/

Description : Algorithme de recommandation de Google utilisé pour trouver des sujets tendances.

URL : https://google.fr

Pour accéder à tous les sites ou autres, j'ai eu recourt à Outlook afin de passer les vérifications de connexions.

URL : https://outlook.office.com/mail/

Également, je me suis dirigé vers le site Developer Mozilla afin d'avoir des renseignement en HTML ou encore en CSS.

URL : https://developer.mozilla.org

Afin d'en savoir plus sur le HTML ou seulement pour quelques renseignements, je me suis rendu sur une documentation HTML sur w3docs.

URL : https://fr.w3docs.com/apprendre-html.html

De plus, j'ai aussi fait un tour sur la documentation CSS pour avoir quelques précisions sur le site CSS référence.

URL : https://cssreference.io/

Pour but de rédiger des articles, j'ai utilisé Chat GPT qui m'a beaucoup aidé à trouver par exemple des idées de titres d'articles ou encore le forme des articles. Je suis donc allé sur le site OpenAI.

URL : https://chat.openai.com/

Dans le même but, je suis allé sur l'intelligence artificielle de Bing afin de générer des images ou des textes pour les articles.

URL : https://copilot.microsoft.com/

Bitbucket Documentation

URL : https://bitbucket.org/product/fr/guides

Jira Documentation

URL : https://confluence.atlassian.com/jira

Angular Documentation

URL : https://angular.io/docs

TypeScript Documentation

URl : https://www.typescriptlang.org/docs/

BootStrap Documentation

URL : https://getbootstrap.com/docs/4.1/getting-started/introduction/

Annexes

Image 1 : Rapport d'intervention avant

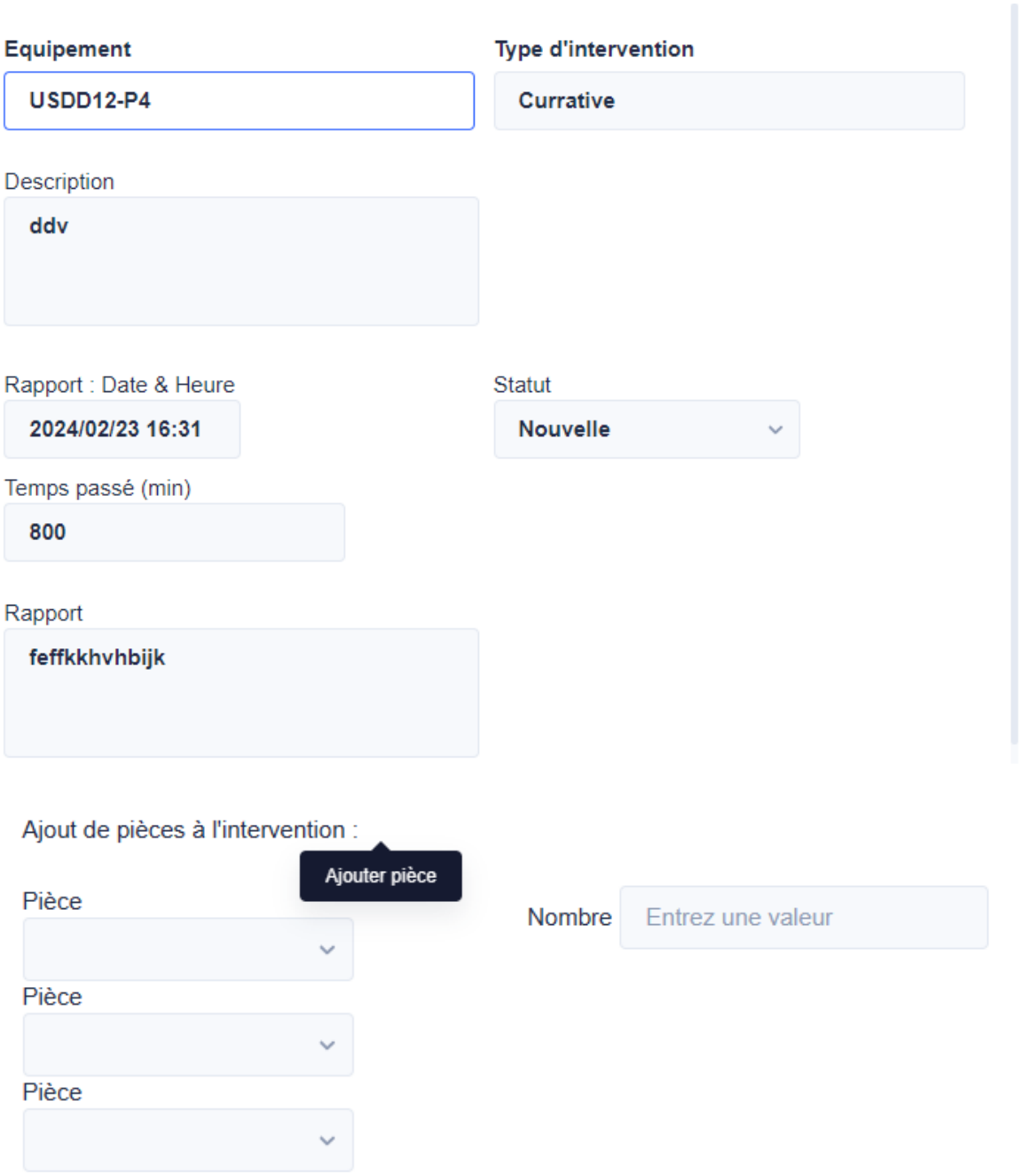


Image 2 : Rapport d'intervention après

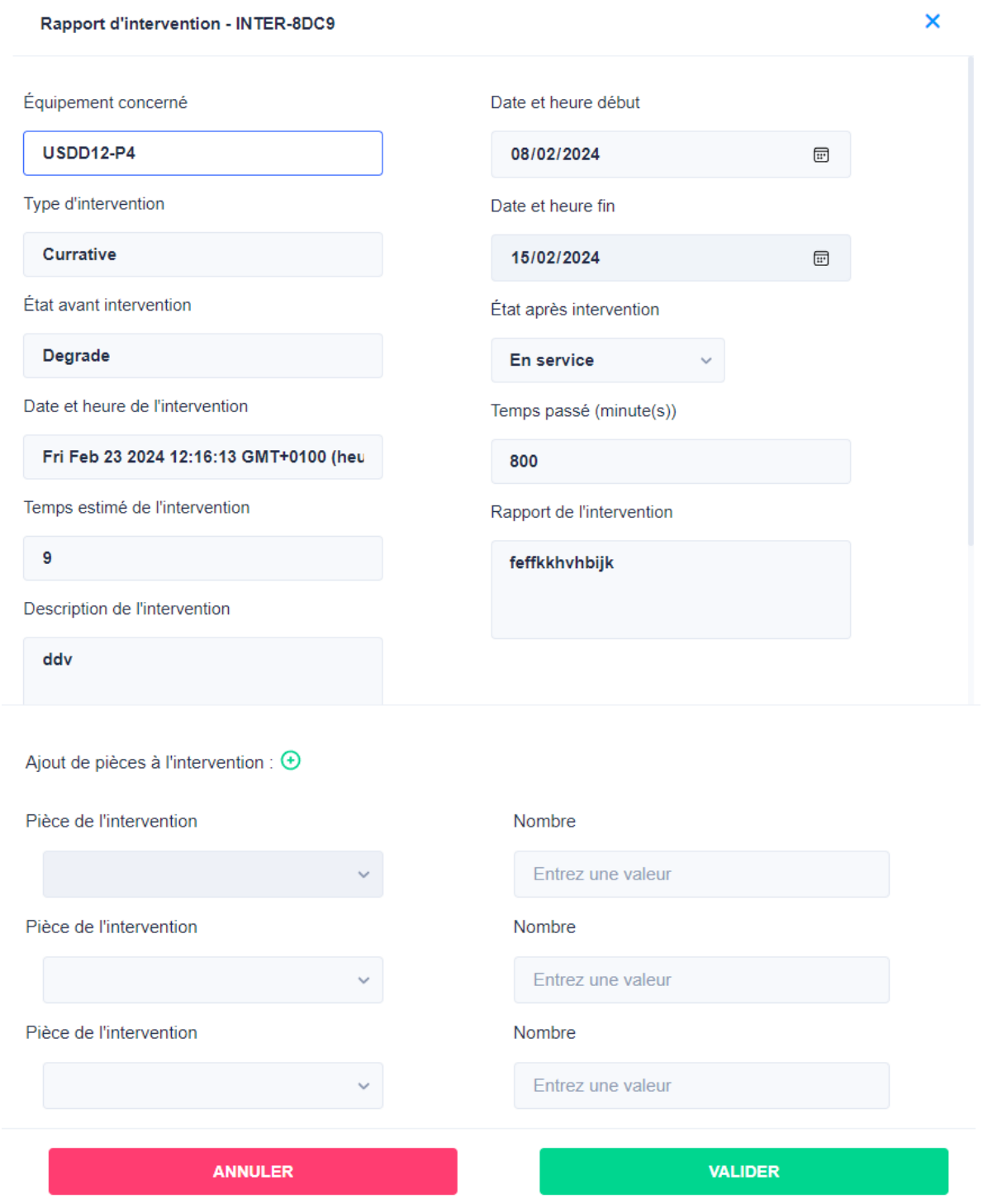


Image 3 : Formulaire de l'intervention

```
// Initialisation du formulaire
formulaire_intervention = this.fb.group({
  _id: [null],
  equipement: [''],
  type: [''],
  date_heure: [new Date, Validators.required],
  date_debut: [new Date, Validators.required],
  date_fin: [new Date, Validators.required],
  urgent: [this.checked],
  description: [''],
  valide_par: ['X'],
  date_intervention: ['X'],
  temps_estime: ['X'],
  temps_passe: ['X'],
  statut: ['En Attente'],
  rapport: [''',Validators.required],
  //piece: [],
  piece: this.fb.array([this.creer_element()]),
  close: false,
  etat_apres:['X',Validators.required],
  etat_avant:[''',Validators.required],
  //nombre_piece : []
})
```

Image 4 : Initialisation de l'intervention

```
export interface INTERVENTION {
  piece: any[];
  date_debut: Date;
  date_fin: Date;
  _id: BSON.ObjectID;
  demandeur?: string;
  equipement: string;
  type: string;
  urgent: boolean;
  description: string;
  valide_par: string;
  date_heure: Date;
  temps_estime: string;
  temps_passe:string;
  statut: string;
  rapport: string;
  close: boolean;
  etat_apres:string;
  etat_avant:string;
}
```

Image 5 : Fonction du bouton fermer/ouvrir une intervention

```
terminer_ouvrir_intervention(_interv: INTERVENTION){
  if (_interv.close === false){
    this.pagesService.fermer_intervention(_interv).then(f=>{this.m_a_j_intervention()});
  }
  else{
    this.pagesService.ouvrir_intervention(_interv).then(f=>{this.m_a_j_intervention()});
  }
}
```

Image 6 : Fonction ouvrir_intervention

```
public ouvrir_intervention(id_interv): Promise<any[]> {
  return new Promise((resolve, reject) => {
    this.realmService
      .realm_function('ouvrir_intervention', id_interv)
      .then((doc: any[]) => {
        resolve(doc);
      })
      .catch((err) => {
        reject(err);
      });
  });
}
```

Image 7 : Fonction ouvrir_intervention

```
public fermer_intervention(id_interv): Promise<any[]> {
  return new Promise((resolve, reject) => {
    this.realmService
      .realm_function('fermer_intervention', id_interv)
      .then((doc: any[]) => {
        resolve(doc);
      })
      .catch((err) => {
        reject(err);
      });
  });
}
```

Image 8 : HTML du formulaire de rapport d'intervention

```
<nb-card-body>
    <div class="row">
        <div class="col-12">
            <div class="row">
                <div class="col-6">
                    <div class="row">
                        <p>Équipement concerné</p>
                        <input nbInput formControlName="equipement" readonly
                        class="col-12">
                    </div>
                    <div class="row">
                        <p>Type d'intervention</p>
                        <input nbInput formControlName="type"  class="col-12" readonly>
                    </div>
                    <div class="row">
                        <p>État avant intervention</p>
                        <input nbInput formControlName="etat_avant"  class="col-12" readonly>
                    </div>
                    <div class="row">
                        <p>Date et heure de l'intervention</p>
                        <input nbInput formControlName="date_heure"  class="col-12" readonly>
                    </div>
                    <div class="row">
                        <p>Temps estimé de l'intervention</p>
                        <input nbInput formControlName="temps_estime"  class="col-12" readonly>
                    </div>
                    <div class="row">
                        <p>Description de l'intervention</p>
                        <textarea nbInput placeholder="" rows="3" formControlName="description" readonly
                        class="col-12"></textarea>
                    </div>
                </div>
                <div class="col-6">
                    <div class="row">
                        <p>Date et heure début</p>
                        <input type="date" #autoInput nbInput class="col-12" formControlName="date_debut"
                        (input)="onChange()" placeholder="Entrez une valeur" min="0" />
                    </div>
                    <div class="row">
                        <p>Date et heure fin</p>
                        <input type="date" #autoInput nbInput class="col-12" formControlName="date_fin"
                        (input)="onChange()" placeholder="Entrez une valeur" min="0" />
                    </div>
                    <div class="row">
                        <p>État après intervention</p>
                        <div class="row col-12">
                            <nb-select placeholder="Etat de la machine apres" formControlName="etat_apres">
                                <nb-option value="En-service">En service</nb-option>
```