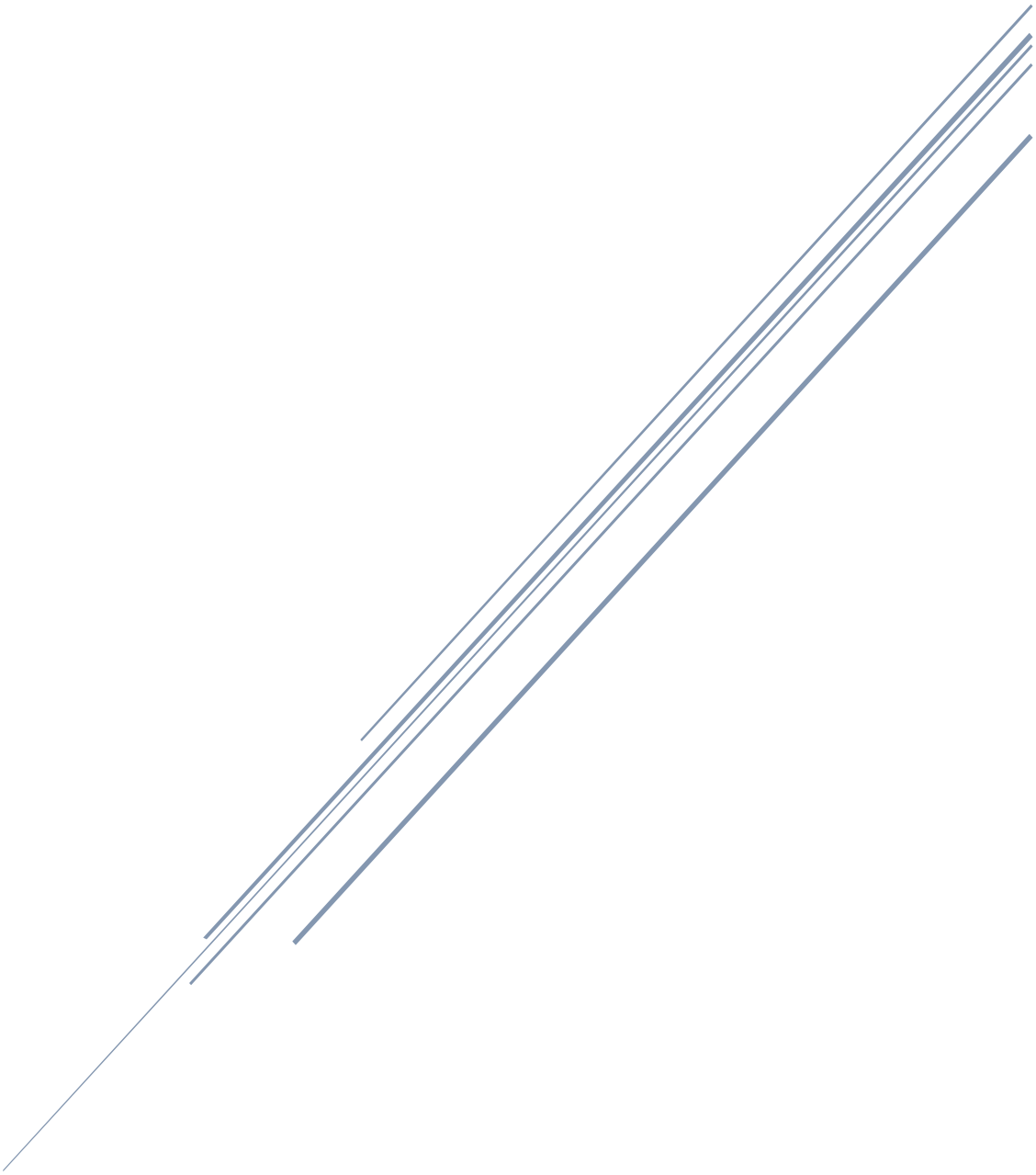
RAPPORT DE STAGE

VINOT Mathieu



2022

IUT Nancy Charlemagne
Université de Lorraine
2 ter boulevard Charlemagne
BP 55227
54052 Nancy Cedex

Département informatique

Réalisation d'un système d'affichage de défauts du papier

Rapport de stage de DUT informatique
Entreprise : Papeteries de Clairefontaine

VINOT Mathieu
Tuteur : COUTRET Stéphane
Année universitaire 2021-2022

Remerciements

Je tiens en premier lieu à remercier Stéphane COUTRET, mon tuteur, qui m'a été d'une grande aide sur toute la période du stage en entreprise, qui a toujours été à l'écoute et présent pour moi. Il m'a énormément apporté et j'ai beaucoup appris à ses côtés.

Je veux aussi témoigner ma reconnaissance aux professeurs de l'IUT, pour m'avoir enseigné de nombreuses choses qui m'ont servi tout au long de cette période de stage et notamment à Alain LONGHAIS, mon professeur référent, qui s'est toujours montré disponible par e-mail et qui s'est même donné la peine de me rendre visite aux papeteries.

Pour finir, je souhaite remercier l'ensemble du personnel de l'usine que j'ai pu côtoyer et qui se sont montré accueillant et qui m'ont guidé.

Un grand merci à Virginie KLAK et Thierry ALEXANDRE, sans qui je n'aurai pas eu ce stage qui m'a énormément plu.

Un dernier merci à Denis et Laura qui m'ont rendu service par leurs aides diverses.

Table des matières

1) Introduction

« Il est impossible de fabriquer du papier ne contenant aucun défaut ». Voici la première chose que j'ai apprise en discutant de papier avec Virginie, travaillant au laboratoire d'analyses des Papeteries de Clairefontaine.

Ce qui m'intéresse dans l'informatique, c'est la création d'applications et de systèmes qui aident au quotidien, donc quand j'ai reçu cette proposition de stage consistant à réaliser une interface utilisateur de visualisation des défauts présent dans le papier en **HTML***, **CSS***, **JavaScript*** et **C# ***, j'ai tout de suite accepté. Il correspond parfaitement à mes envies et est concret : je peux voir l'ensemble du système gravitant autour de l'interface, d'un échantillon de papier contenant les défauts relevé plus tôt à la personne qui se sert de l'application.

Je commencerai par présenter succinctement l'entreprise, puis je présenterai mon sujet d'étude. Je continuerai avec mon arrivée dans l'entreprise et la présentation de mon travail effectué lors du stage. Enfin, je dresserai un bilan de cette expérience.

Mon travail s'est déroulé selon le planning en Annexe.

Cf Annexe 1 : Planning

2) Déroulement du stage

2.1) Présentation de l'entreprise

Les papeteries de Clairefontaine sont situées à Etival Clairefontaine, elles produisent du papier haut de gamme aux caractéristiques variées, du papier pour l'impression au cahier scolaire. Malgré le savoir-faire de l'entreprise, il est impossible de produire de grandes quantités de papier sans aucun défaut. Pour fournir la qualité attendue par les clients, des contrôles qualité réguliers se font tout au long de la fabrication du papier.

Cf Annexe 2 : Présentation détaillée de l'entreprise

2.2) Présentation du sujet d'étude

Le système Viconsys permet la visualisation des défauts du papier en sortie **de la machine à papier*** défilant sur les **bobines***. Cela fonctionne à l'aide de 16 caméras et 2 types d'éclairages. L'un éclaire par le haut et permet de voir les défauts de surface, les taches, les dépôts d'huile ou autre. L'autre éclaire par le bas et permet de voir principalement des trous et des déchirures.

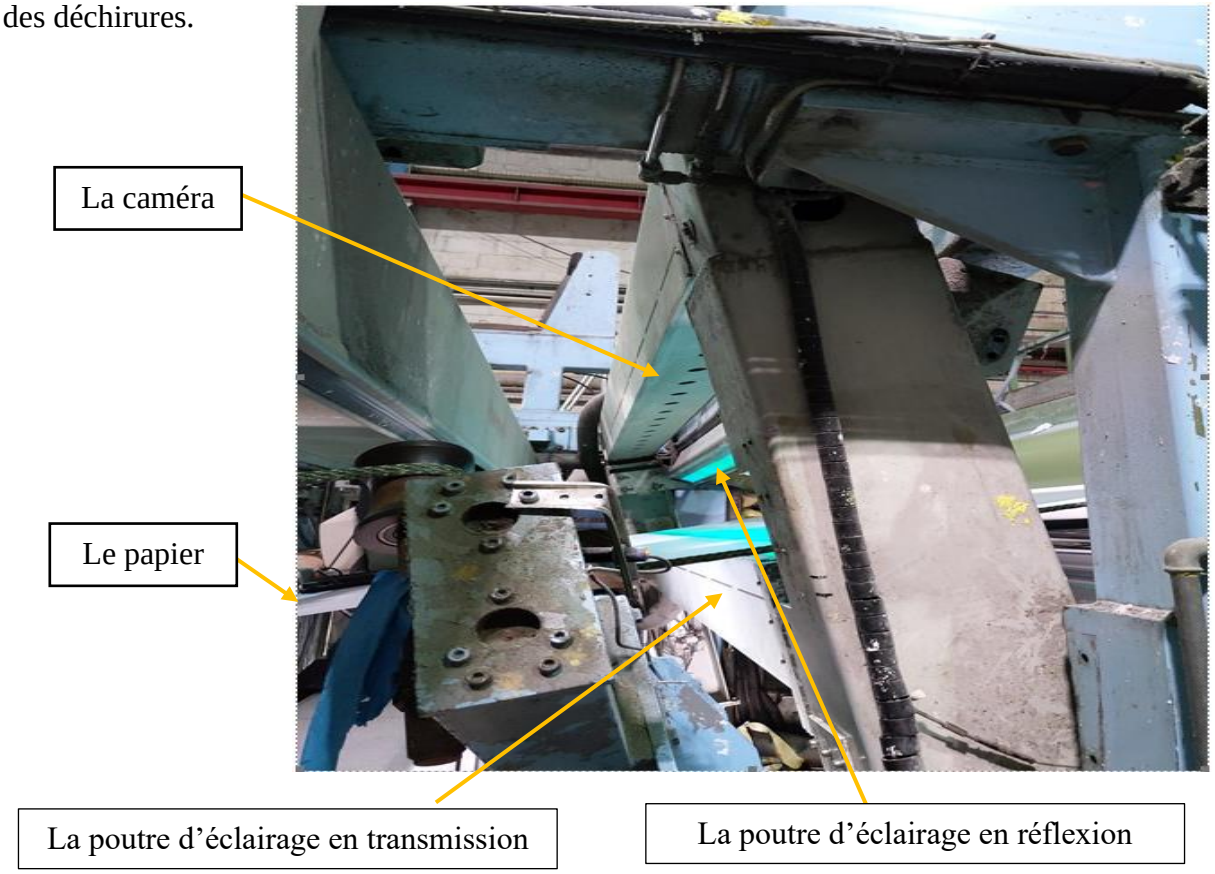


Figure 1: Photographie du système Viconsys, prise à l'usine

Une image sur deux est prise avec un type d'éclairage, ceux-ci alternent entre eux. Le but de voir les différents défauts à l'avance est de pouvoir repérer et identifier le plus rapidement possible les défauts pour corriger le problème et faire **refondre le papier*** dans le **pulpeur***.

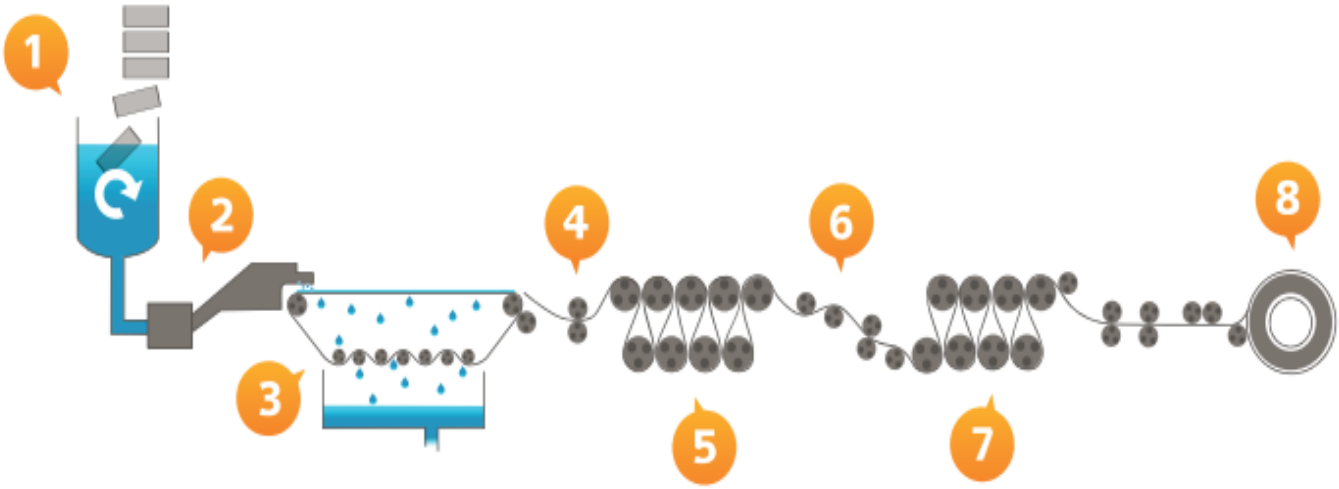


Figure 2 : Schémas des étapes de la fabrication du papier, ce schéma provient du rapport de stage [Imane, 2021]

Légende :

1. **Mélange de l'eau et de la pâte à papier**
2. **Raffinage* et épuration**
3. **Table de formation avec récupération de l'eau pour recycler**
4. **Presses et cylindres absorbants**
5. **Séchage par cylindres chauffés**
6. **Dispositif de couchage (Size-press)**
7. **Séchage par cylindres chauffés**
8. **Bobine mère**

Limiter le temps de présence du défaut permet de ne pas transformer le papier défaut et donc de limiter les pertes. De plus, un simple trou peut déchirer toute la bobine quand celle-ci est déroulée à plus de 1500 mètres par minute, les **bobineurs*** peuvent donc adapter la vitesse de rotation avec ses informations et éviter le problème.

Malheureusement, le système Viconsys est vieillissant et l'entreprise l'ayant installé n'existe plus (elle a été rachetée par une autre entreprise uniquement intéressée par ses brevets, et n'assure donc pas le suivi des anciennes installations). Il faut donc retravailler l'ensemble du système pour qu'il continue à exister.

La partie logicielle est une part importante du système et tout comme les composants électroniques, l'interface doit être actualisée. La version actuelle du système Viconsys fonctionne sur d'anciens systèmes d'exploitation, Windows 7 pour le plus récent et avec Internet Explorer uniquement.

Il faut donc recréer l'application en étant le plus fidèle possible à l'ancienne version pour garder toutes les fonctionnalités déjà présentes et ne pas dérouter les utilisateurs, en reproduisant notamment la disposition des éléments dans l'interface et en reproduisant les icônes. Évidemment, il faut faire une application qui soit fonctionnelle sur les systèmes d'exploitation actuels, la plupart des ordinateurs de l'usine fonctionnant sous Windows 11 et s'utilise avec différents navigateurs comme Edge, Chrome ou encore Firefox. Il est nécessaire que cela soit durable dans le temps et reste valable sur les prochaines mises à jour du système et de navigateurs. De plus, un nouveau système de caméra et d'éclairage sera implémenté.

En effet, comme sur la photographie ci-dessous, de permettront de voir de nouveaux problèmes comme les problèmes d'**azurant*** qui est un produit permettra d'accentuer l'impression de blanc de la feuille. Si le produit est mal réparti dans la feuille, on ne peut le voir à l'œil nu mais cela se remarque à l'impression.

CF Annexe 3 : Photo de la maquette du futur montage

Il faut donc, en plus simplifier l'ajout de défaut dans l'interface, en limitant le nombre de fichiers où intervenir.

Concernant le service dans lequel je travaille, il est important de préciser qu'il ne s'agit pas du service informatique mais des services techniques de Clairefontaine et même plus précisément je suis au contact du service de régulation et du service électrique.

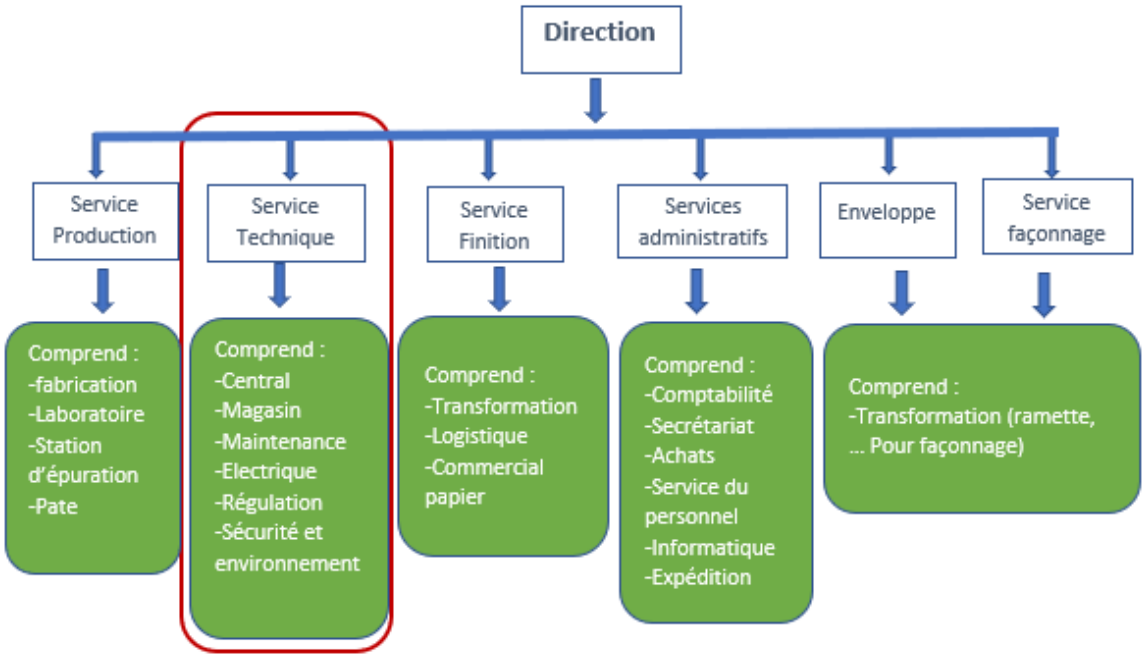


Figure 3 : Organigramme des différents services des Papeteries de Clairefontaine

Ce service s'occupe des machines, de développer des systèmes pour le pilotage et l'affichage d'informations.

À titre personnel, il est donc convenu que je m'occupe uniquement de tâches relatives à l'informatique. De ce fait, j'ai le même emploi du temps que ce service, donc ma journée débute à 7 heures jusqu'à 11 heures puis reprend à 13 heures et se termine à 16 heures 30 le lundi, mardi et mercredi et à 16 heures le jeudi et vendredi. Je suis situé dans les bureaux du service électrique, sur un bureau partagé en 8 postes. Mon tuteur est dans un autre bureau situé à quelques mètres à peine du mien.

Restant dans les bureaux du service, je ne suis pas concerné par le port d'une tenue spécifique, notamment d'équipement personnel de sécurité (EPI) tel que des chaussures de sécurité ou des protections auditives.

Le port des EPI est cependant indispensable partout dans l'usine où se trouvent les machines. Je ne peux donc pas aller et venir librement voir les bobineurs pour discuter de l'interface.

Pour pouvoir travailler, il m'a été fourni un ordinateur fixe afin d'avoir accès au réseau local avec les différentes données. Une adresse mail m'a été attribuée afin de communiquer avec mon tuteur, mais aussi de recevoir les différentes informations telles que l'arrêt éventuel de machine, un problème technique, …

Pour réaliser la partie application de Viconsys, j'ai accès à des captures d'écran de l'ancien système, j'ai eu une présentation complète par mon tuteur et le chef de service. J'ai aussi pu échanger aussi avec les bobineurs, qui ont su me faire part de leurs attentes et j'ai surtout bénéficié du squelette de l'application que mon tuteur avait déjà réalisé. Mon objectif principal est de réaliser précisément la partie d'affichage des défauts.

Cf Annexe 4 : Capture d'écran ancienne application Viconsys

Cf Annexe 5 : Capture du squelette de l'application

L'objectif principal est donc de reproduire les différentes fonctionnalités de cette partie telles que l'affichage complet de la bobine et de ses défauts, la possibilité de zoomer et déplacer, ajouter les défauts en temps réel et afficher ou cacher certains défauts au choix de l'utilisateur. Cette application est à réaliser en HTML, CSS, JavaScript et C#.

Je réalise ce travail en grande partie en autonomie. Mon tuteur s'occupe d'autres aspects du système, principalement le **backend***. Je peux tout de même lui demander de l'aide et des mises en commun se font tout au long de la réalisation.

2.3) Arrivée dans l'entreprise

À mon arrivée dans l'entreprise, je me suis rendu à l'accueil du service technique, j'ai fait la rencontre en personne de mon tuteur, que je n'avais vu jusqu'alors qu'en visioconférence, lors de mon entretien à distance. J'ai rejoint un petit groupe de deux personnes, qui étaient comme moi de nouveaux arrivants, l'un était un alternant l'autre une stagiaire. Nous avons été guidés vers une salle équipée d'ordinateurs. Lors de cet accueil sécurité de nombreux documents concernant l'entreprise avec en particulier le règlement intérieur des Papeteries et différentes consignes de sécurité nous ont été remis. Une présentation des différentes consignes de sécurité et autres informations utiles concernant l'entreprise, à l'aide d'un des postes de la salle. À l'issue, j'ai dû répondre à un court questionnaire, de contrôle des connaissances des consignes et l'ai réussi.

Une fois cela terminé, j'ai rejoint mon tuteur qui m'a fait visiter la partie de l'usine où se trouvent les 2 machines à papier et les 2 machines de bobinage appelées bobineuses. J'ai donc eu une vision concrète du système Viconsys, en découvrant la position des caméras, la bobineuse et en examinant la partie de traitement des images renvoyées par les caméras. Mon tuteur m'a ensuite présenté les bobineurs, qui m'ont fait part de leurs attentes concernant le programme et j'ai eu un aperçu de l'utilité du système.

Une fois cette étape de découverte passée, mon tuteur m'a indiqué mon bureau puis m'a aidé à m'installer. Il m'a montré le code déjà réalisé et m'a présenté la mission principale : elle consiste se concentre sur la partie représentation de la bobine.

Cette première journée fut consacrée aux présentations et mises en place. J'ai surtout analysé la structure du code, c'est-à-dire comprendre la fonction de chaque fichier, ce qu'ils contiennent et quels sont les liens entre ces différents fichiers. De plus, je n'ai seulement eu la possibilité de télécharger un **IDE*** que le lendemain. J'ai téléchargé le même IDE que mon tuteur (**Visual Studio**), ce qui nous a permis de mieux nous comprendre et il a eu plus d'opportunité pour m'aider grâce à cela.

2.4) Présentation du travail réalisé

2.4)1. Adapter le dessin à l'espace disponible

Je débute ma mission en adaptant l'espace qu'occupe ma partie à la taille de l'écran. L'objectif est de pouvoir visionner la totalité de la bobine, peu importe la taille de l'écran.

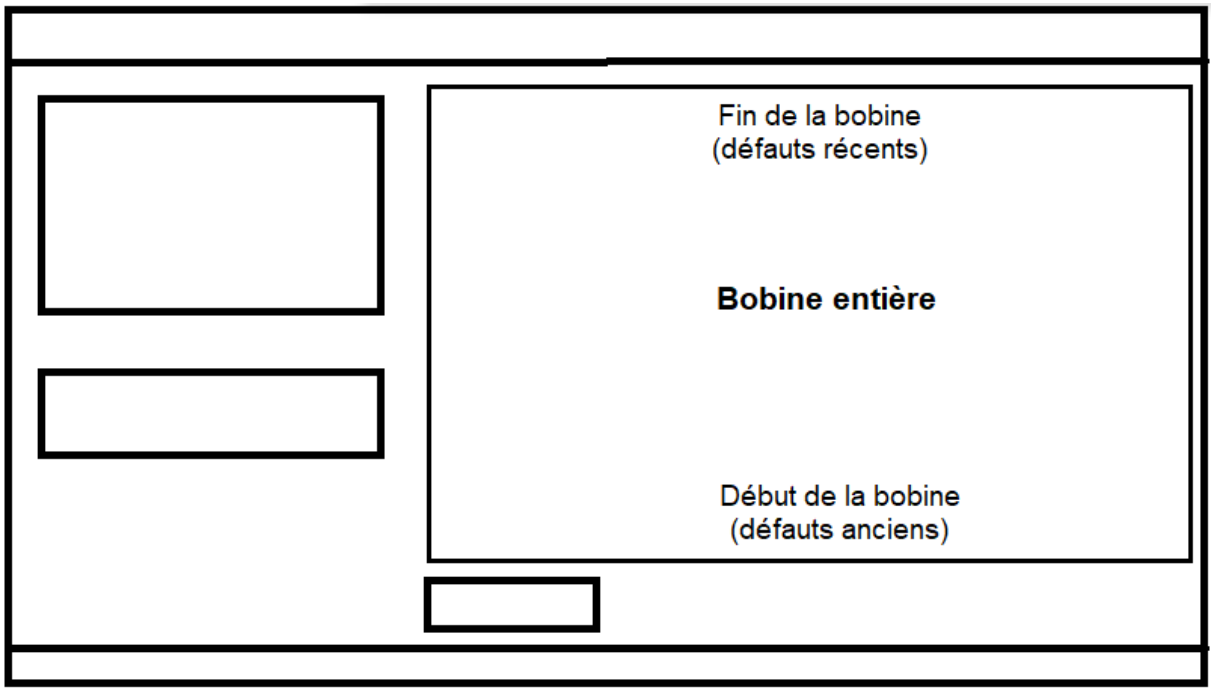


Figure 4 : Schéma du résultat attendu

Il me faut donc calculer l'espace disponible pour ma partie de représentation de la bobine, il s'agit de la différence entre la taille de l'écran et les différents éléments apparaissant à l'écran en même temps. Pour récupérer ces éléments, je fais usage de **JQuery***. Je calcule ensuite le coefficient qui permet de passer des dimensions de la bobine initiale aux dimensions finales. Avec ça, j'utilise l'attribut *translate* des éléments **SVG*** pour les redéployer tout en gardant leurs positions initiales pour recalculer en continue. Une fois terminée, j'étends cette fonctionnalité aux défauts ajoutés en temps réel et recalcule donc à chaque mise à jour. Cela me permet de réadapter à la surface de dessin lors du changement de la taille de la fenêtre. Il suffit simplement d'ajouter mon calcul d'espace disponible dans la fonction de rafraîchissement appelé automatiquement et qui ajoute les potentiels nouveaux défauts. Avec l'ajout en temps réel, je me rends compte d'un oubli, le SVG prend son origine en haut à gauche, ce qui fait que mes défauts les plus récents se retrouvent en bas de page et le plus ancien en haut, alors que l'inverse est souhaité. J'inverse donc la valeur des ordonnées.

Alors que je teste, je reçois un mail expliquant que la machine après laquelle se situe mon système de surveillance va s'arrêter. L'arrêt provoque un grand message d'erreur qui est évident : s'il n'y a pas de bobine, il n'y a pas de dimensions, je divise donc par zéro à un moment dans mes calculs. Je rajoute donc une condition pour empêcher l'irrégularité de se produire. Afin de prévenir l'utilisateur, je décide de créer une alerte (boite de dialogue s'ouvrant au-dessus de la page). En testant après avoir ajouté le code, je me rends compte que l'alerte se lance en boucle ce qui me force à fermer la page. J'imagine donc un système de **cookie*** « alerteVu » qui se crée à chaque lancement d'alerte, et qui dure 30 secondes. Le lancement d'alerte ne se lance pas si un cookie « alerteVu » existe, j'empêche de ce fait la création abusive d'alerte, forçant la fermeture du navigateur. En présentant à mon tuteur, il m'explique que les utilisateurs n'ont pas besoin de cette information et que bloquer suffit. Je mets donc le code de l'alerte en commentaire.

Pour continuer avec l'ajout de défauts en temps réel, j'applique les différents événements des défauts (le clic qui fait apparaître une fenêtre d'informations concernant le défaut et un affichage qui met en évidence le défaut que la souris survol) sur les défauts fraîchement apparus. Il suffit simplement d'ajouter les événements des nouveaux défauts lors de l'ajout ceux-ci.

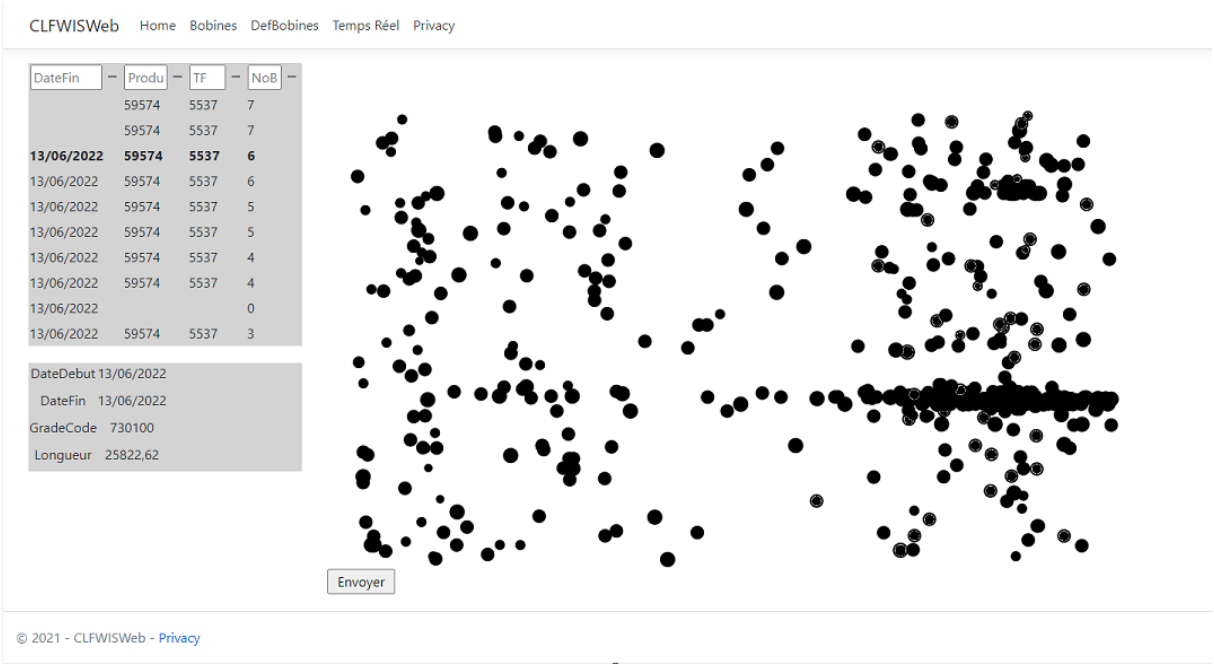


Figure 5 : Capture d'écran du résultat, à la fin de la semaine 2, défauts réels

2.4)2. Création du Zoom

Je continue avec la création d'un zoom qui doit être centré sur la souris. Il ne doit pas agrandir les défauts, seulement l'espace entre chaque.

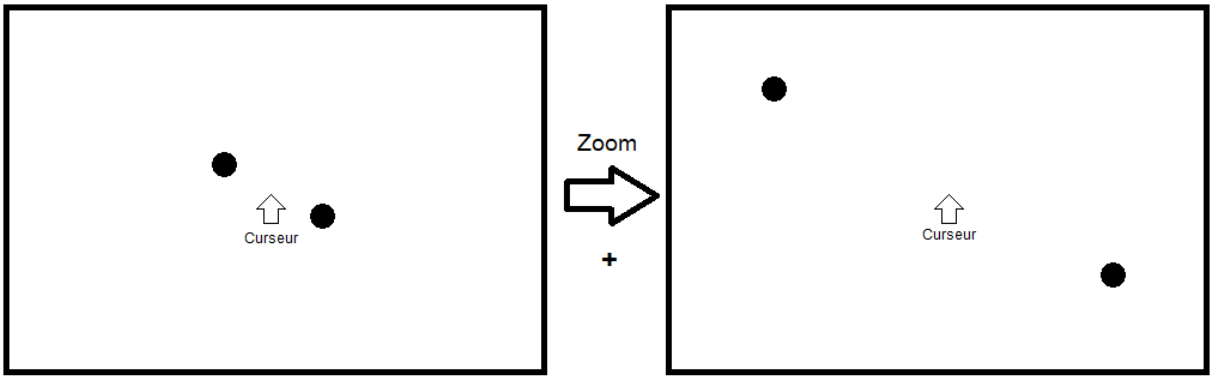


Figure 6 : Schéma du système de zoom souhaité

Pour la réaliser, je découpe la fonctionnalité en trois parties :

- Le zoom se fait en fonction de la molette.
- Le zoom doit espacer les défauts entre eux.
- Le zoom doit être centré sur la souris.

Il faut donc commencer par créer un événement qui, lorsque l'on utilise la molette sur la partie représentative de notre bobine, applique un zoom en fonction du sens de rotation (molette vers le haut est un zoom et vers le bas un dézoom).

Pour pouvoir comptabiliser l'action de la molette il faut qu'il y ait un élément qui le supporte. Or je veux pouvoir zoomer même dans la partie blanche qui représente la portion de bobine sans défaut. Je crée donc un rectangle blanc en SVG qui va occuper toute la surface en fond et qui prendra en compte la molette et appliquera le zoom.

Pour zoomer, il existe une option en SVG qui permet de modifier l'échelle du dessin, mais ce n'est malheureusement pas une solution, car augmenter l'échelle augmente la taille des défauts au passage. Pour simplement augmenter l'espace entre les défauts, je décide donc de déplacer les défauts à l'aide de l'attribut déplacement des éléments SVG. Pour réussir, je

multiplie par le coefficient de zoom la distance défaut à origine. De cette manière, je vais garder les proportions de ma bobine, et l'écartement entre chaque défaut sera respecté.

Il reste le zoom centré sur la souris, qui est assez évident. Le zoom actuel se fait par rapport à l'origine de mon espace SVG, il faut donc considérer le curseur de ma souris comme la nouvelle origine, en déplaçant chaque défaut en fonction du vecteur $\overrightarrow{Curseur\ Origine}$. Pour ce nouveau déplacement qui doit être le même et unique partout, j'utilise le patron de conception **Singleton***, qui me servira aussi lors du déplacement dans la partie représentation. À chaque zoom, je récupère la position de ma souris dans l'espace SVG, puis je convertis les coordonnées pour qu'elles correspondent à la position réelle dans la bobine. Je modifie ensuite l'attribut de déplacement de mon Singleton et mets à jour mon dessin, en appliquant les modifications à l'ensemble des défauts.

Il me reste à rajouter une limitation du dézoom. En effet, si l'on peut zoomer à volonté car cela pourrait se révéler utile, il est inutile de diminuer le zoom au-delà du zoom initial, qui permet déjà de voir toute la bobine. Je rajoute donc une simple condition pour bloquer ce cas.

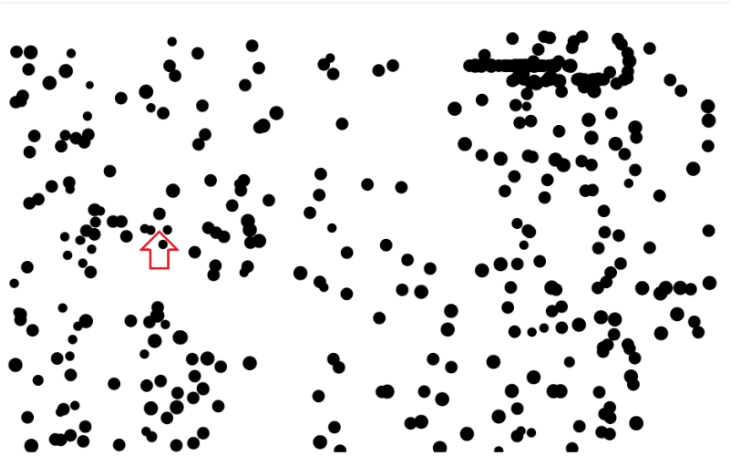


Figure 7 : Capture d'écran de la bobine avant le zoom, défauts réels

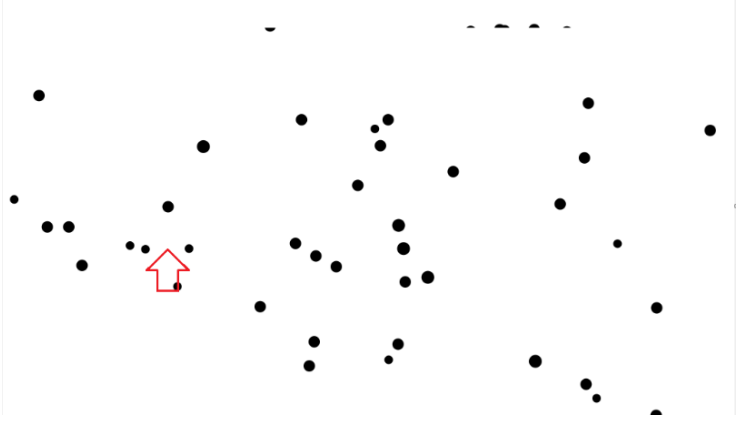


Figure 8 : Capture d'écran de la bobine après le zoom, défauts réels

2.4)3. Déplacement dans la bobine

Suite à ce zoom, je travaille sur le déplacement de ma représentation de la bobine. Sachant que je possède déjà un Singleton qui permet de déplacer l'ensemble des défauts, il me suffit de rajouter des attributs pour ce déplacement.

Je dois donc me concentrer sur ma manière de déplacer le dessin. Dans l'interface de l'ancien système, le déplacement se faisait à l'aide de barres de défilements, ce que je trouve peu intuitif. Je prends donc la liberté de changer cette solution par d'un glissé-déposé de la souris. Il suffit d'enfoncer le clic pour commencer le déplacement, on suit le déplacement de la souris et on arrête lors du relâchement du clic. J'ai donc la liste de mes trois événements à prendre en compte pour le déplacement.

Le premier clic me permet donc d'initialiser l'origine du déplacement, à chaque déplacement, je récupère la nouvelle position pour effectuer le déplacement et mets à jour l'origine, relâcher effectue la dernière mise à jour et réinitialise l'origine.

Le résultat est globalement satisfaisant mais quelques erreurs de réalisation se font ressentir. Tout d'abord, je remarque plusieurs parties de mon code se répètent, plus particulièrement pour les mises à jour de la représentation de la bobine. Afin de limiter les copier-coller et la casse, je dois factoriser mon code, c'est-à-dire réduire à une fonction une même partie qui se répète.

Suite à cela, je constate un problème de conception : les différents déplacements ne s'appliquent seulement lors des rafraîchissements et donc à chaque appel au serveur. Ce qui signifie que plus le déplacement est fluide, plus le serveur est sollicité. Néanmoins, il faudrait concilier un déplacement sans interruption ni saccade avec le moins de requêtes au serveur possible. Je revois donc ma façon de déplacer et sors les variables de déplacement du rafraîchissement. Celles-ci seront modifiées et appliqueront les changements lors de l'événement avec la souris. Grâce à la factorisation de la mise à jour précédemment réalisée, cette correction est plus simple à mettre en œuvre. Le plus dur a été de revoir ma façon d'aborder le problème.

Une fois que le déplacement est correct, il faut le limiter de la même manière que pour le zoom. Je commence par imaginer un système dont le principe est simple : je récupère les bornes du dessin, qui sont les défauts avec la valeur des abscisses la plus grande, la plus petite et la même chose avec les ordonnées. L'implémentation du système se passe sans problème et fonctionne comme je l'ai imaginé en amont.

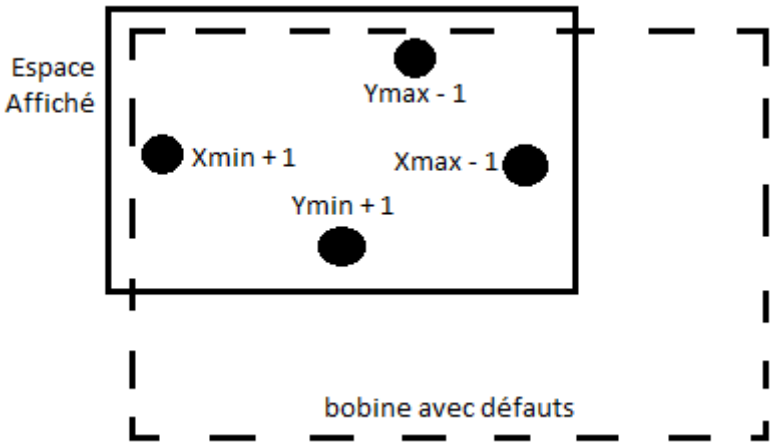


Figure 9 : Schéma du système de bordure

Par la suite, j'observe un ennui dans ma limitation du déplacement. En effet, lors d'un déplacement en diagonale, en rencontrant uniquement une limite, c'est la totalité de la manœuvre qui est bloquée. Mais si j'étais un utilisateur je préférerais que le déplacement devienne une ligne. Je modifie donc rapidement le blocage, en prenant en compte plus spécifiquement la borne bloquante au lieu de me contenter de tout annuler au moindre contact avec une limite.

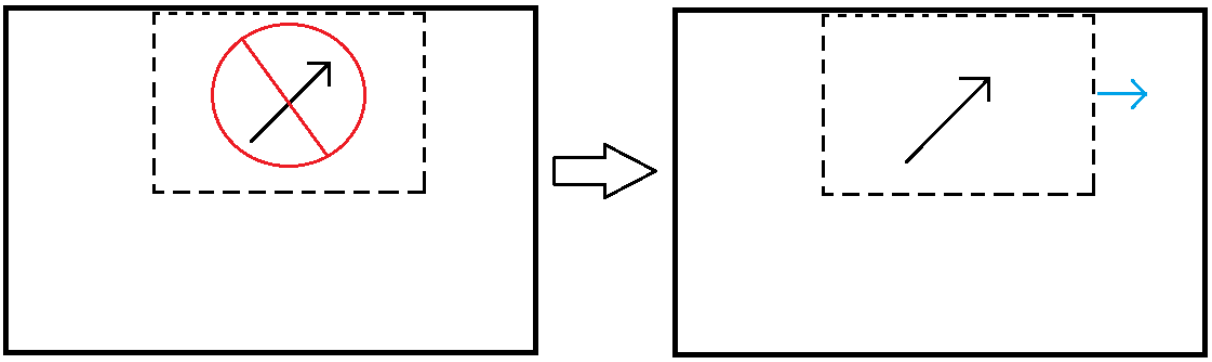


Figure 10 : Schéma du déplacement en diagonale, réalisé sur Paint

À gauche, le déplacement en diagonale précédent, la flèche noire représente le déplacement souhaité.

À droite, le déplacement en diagonale corrigé, la flèche bleue représente le déplacement effectif.

2.4)4. Dessins des défauts

Certains défauts sont déjà représentés tels que les taches sombres ou les trous, mais il en manque d'autres comme la rafale de taches ou l'huile. Je m'attelle donc au travail, je commence par imaginer la manière dont seront composés les défauts en SVG (de ronds, de traits ou autres) en utilisant comme modèle ceux de l'ancien système.

Cf Annexe 6 : Liste des défauts de l'ancienne application

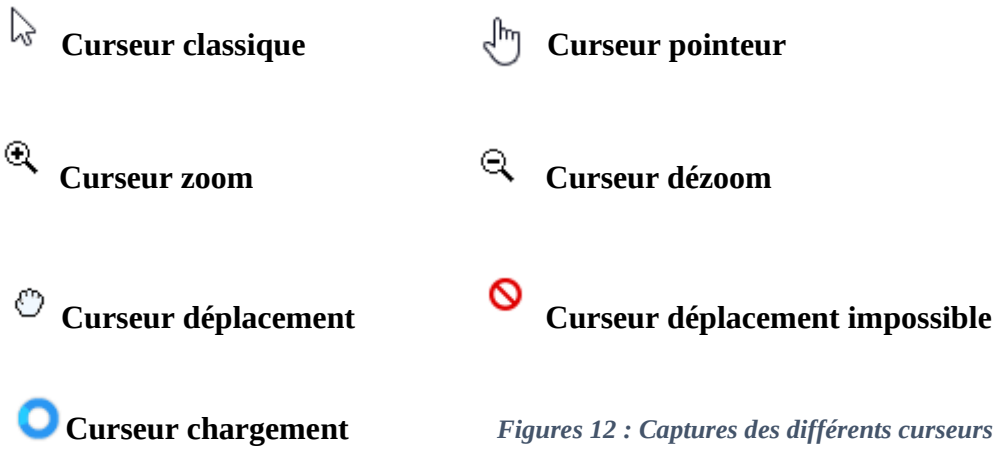
Le but de recréer les dessins le plus à l'identique possible est de ne pas déboussoler les utilisateurs du programme, qui retrouveront directement les symboles qu'ils connaissent. J'ai uniquement pris la liberté de changer le dessin de l'insecte, qui était un simple rond vert dans l'ancien système. Changer le symbole a permis de le rendre plus lisible dans la partie de représentation de la bobine et cela permet d'éviter de le confondre avec une tache d'huile. Je les crée ensuite sur des fichiers tests avec des valeurs en brut.

Insecte **Huile** **Trait** **Goutte**

Figures 11 : Exemples de défauts réalisés

Une fois avoir obtenu un résultat satisfaisant, j'ajoute le code dans un ***switch**** en adaptant les coordonnées et la taille du défaut par rapport aux informations obtenues. La totalité des défauts existants sont maintenant traités et apparaissent.

J'embellis ensuite mon application à l'aide d'indications simples et intuitives. Afin de montrer que le clic est enfoncé et qu'il est possible de déplacer mon dessin, je change le curseur. Je reprends ce schéma pour d'autres indications à l'aide du curseur : zoom, dézoom, déplacement impossible et chargement. Je souhaite rendre l'interface beaucoup plus cohérente et mieux présenter certains événements comme le chargement d'une nouvelle bobine.



Figures 12 : Captures des différents curseurs

Pour réaliser cela, je crée une fonction récursive qui applique le curseur à mon élément et tous ses enfants. Mais après des recherches, je constate les limites de ma conception (lorsque le CSS applique déjà un curseur personnalisé sur un élément), je change donc mes plans et utilise du CSS, en appliquant le curseur à l'ensemble de la page HTML à l'aide de classe qui applique son changement en priorité. C'est une solution plus simple et j'ai découvert des options en CSS qui m'étaient inconnues jusqu'alors.

À nouveau, je complète l'idée d'une meilleure lisibilité lors du changement de bobine en supprimant instantanément l'ancien affichage pour accentuer la compréhension du chargement par l'utilisateur et qu'il ne fasse pas plusieurs requêtes au serveur en appuyant à répétition sur le bouton.

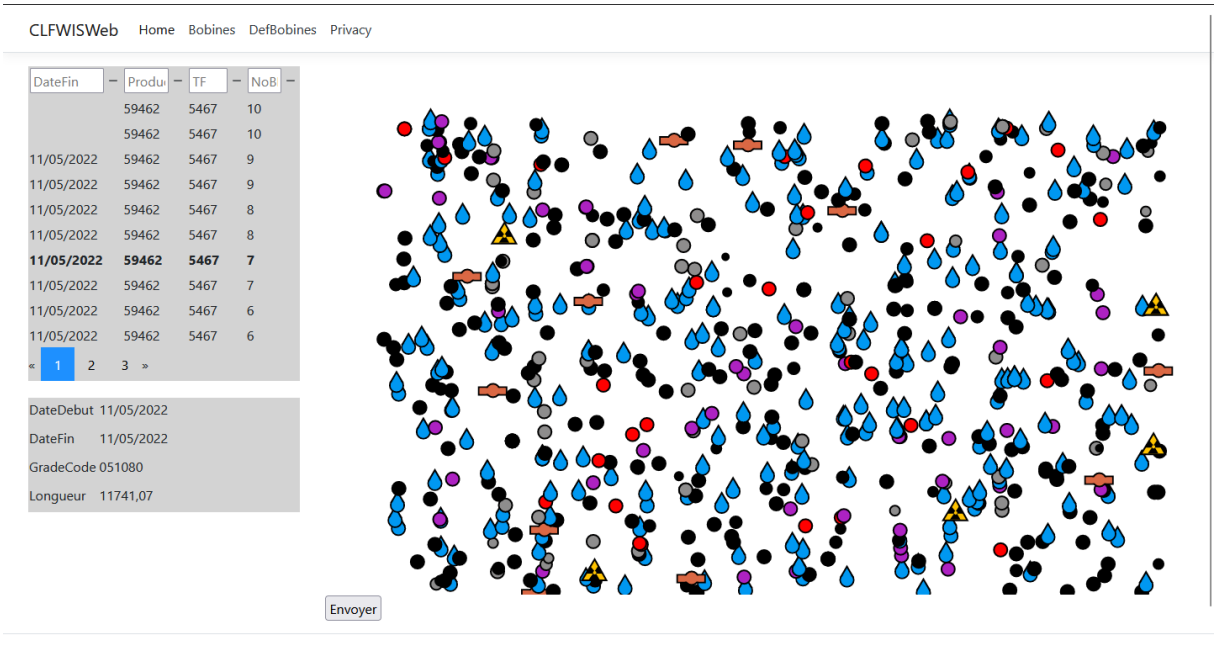


Figure 13 : Capture d'écran du résultat, à la fin de la semaine 4, défauts simulés

2.4)5. Bouton de « remise à zéro »

Je me décide à créer une option de « remise à zéro » du zoom et du déplacement, pour tout annuler simplement. Il faut donc d'une part créer l'option et d'une autre part l'utiliser.

La fonction est plutôt simple à réaliser, il me suffit de remettre les différents déplacements présents dans le déplacement à zéro puis de mettre à jour mon dessin.

Pour utiliser cette fonction, je commence par créer un événement sur une touche du clavier. Cependant, mon tuteur souhaiterait faire apparaître la fonctionnalité sous forme de bouton dans l'interface. Il me présente une manière pour créer dans un fichier C# un bouton avec une icône SVG, à l'aide d'une classe qu'il a créé. Il me montre comment utiliser le bouton dans mon fichier JavaScript principal

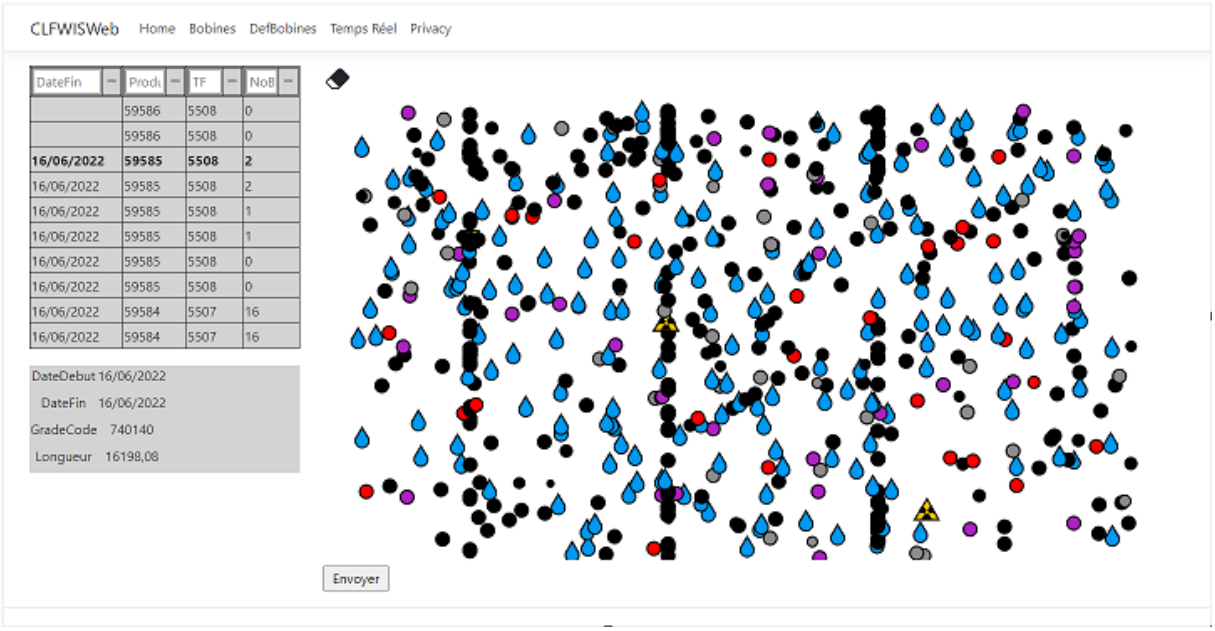


Figure 14 : Capture du résultat, défauts simulés

Le problème de cette solution est que l'on est énormément limité pour l'apparence des boutons, qui doivent être des icônes de **Bootstrap***. Je tente donc de modifier ce système de bouton pour rajouter un cas où je peux choisir mon propre logo.

Figure 15 : Capture du menu réalisé, seules les taches sombre sont sélectionnées (rond noir)

Je mets donc en œuvre mes boutons, mais deux problèmes se posent :

- Chaque bouton est créé à la main, il est impossible de faire une boucle qui, à chaque type de défaut va créer le bouton associé. Cela complique donc l'ajout d'un nouveau défaut, allant à l'encontre de mes consignes.
- Il n'y a pas d'indicateur visuel pour la sélection et changer le bouton semble impossible.

J'en conclus que je dois plutôt utiliser des boutons directement en HTML. J'imagine l'apparence de mon futur menu.

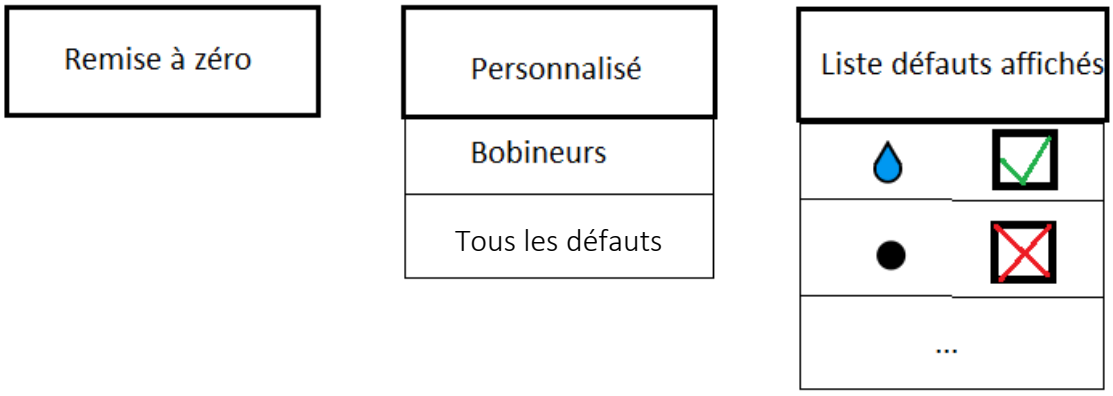


Figure 16 : Schéma du menu de boutons

Puis j'ajoute mon bouton de « remise à zéro ». N'ayant pas accès à un fichier HTML, je peux néanmoins en ajouter dans un fichier C# comme m'a montré mon tuteur.

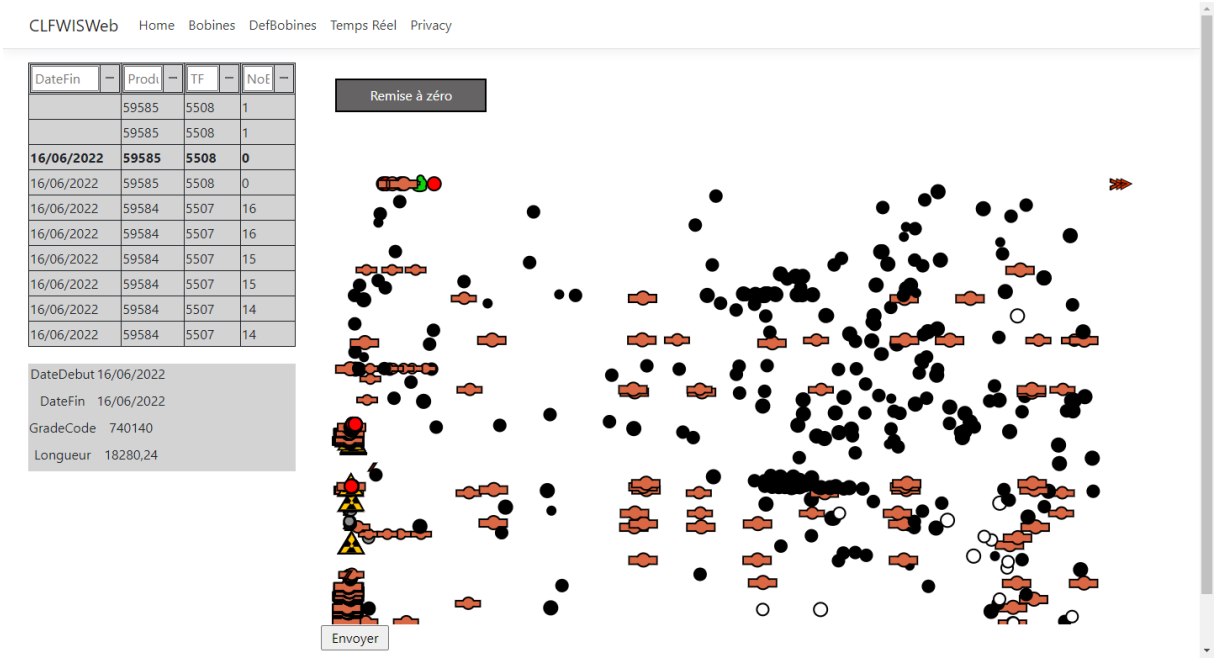


Figure 17 : Capture d'écran du résultat, défauts réels

2.4)6. Profils de visionnage des défauts

Le bouton de profil personnalisé est donc un menu déroulant. Pour le réaliser, je m'inspire de boutons de Bootstrap.

Je crée le bouton de profil à l'aide de la balise *select* en HTML. Je modifie ensuite un peu le CSS pour que la disposition des boutons soit meilleure et les rendre plus lisibles (profils de couleurs différents, espaces entre les boutons).

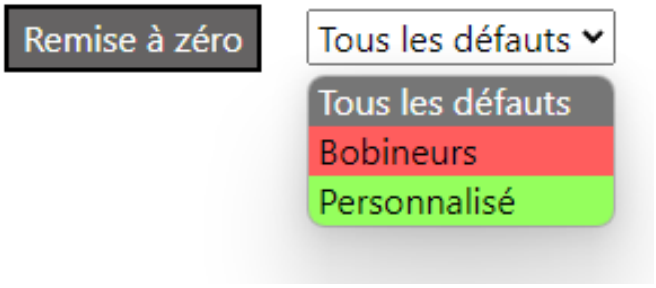


Figure 18 : Capture d'écran des boutons de remise à zéro et de profil

La fonctionnalité doit permettre d'afficher une certaine sélection de défauts. Le profil personnalisé servira lors du choix personnalisé de défauts.

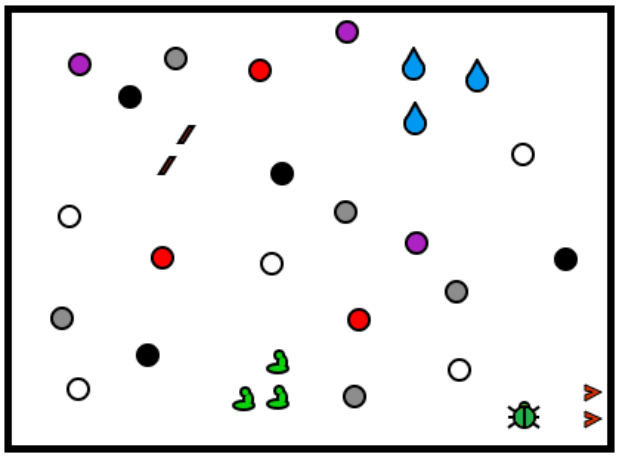


Figure 19 : Schéma vue bobine profil « Tous les défauts »

»

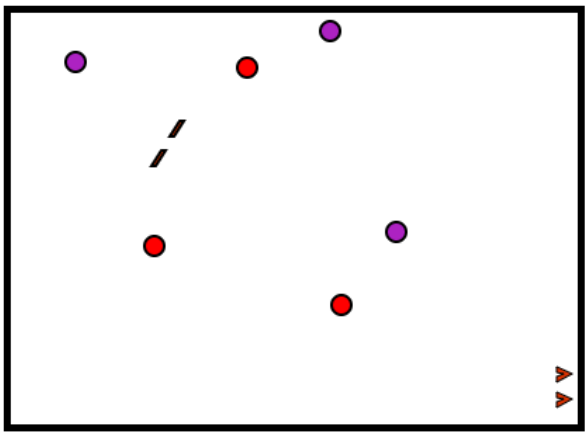


Figure 20 : Schéma vue bobine profil « Bobineurs »

Pour cacher ou afficher les défauts, je me sers de l'attribut *visibility* qui peut prendre soit la valeur *visible* soit *hidden*. Afin d'avoir une sélection persistante, il faut garder dans la mémoire du navigateur les choix. Pour ce faire, j'utilise le Local Storage de JavaScript. Pour chaque type de défaut, j'initialise à *visible*. J'initialise le profil à « Tous les défauts ».

Pour ce bouton de choix de profil, je choisis d'afficher celui en cours de sélection, en récupérant la valeur de **profil** dans le *Local Storage*. J'initialise donc cette valeur sur le profil par défaut, qui est l'affichage de tous les défauts. À chaque changement de profil, je récupère la valeur, l'écrit dans le *Local Storage* et affiche la valeur du *Local Storage* sur le bouton. De cette façon, je m'assure de toujours afficher la valeur utilisée, dans le cas où le changement de valeur du profil dans le *Local Storage* ne fonctionne pas.

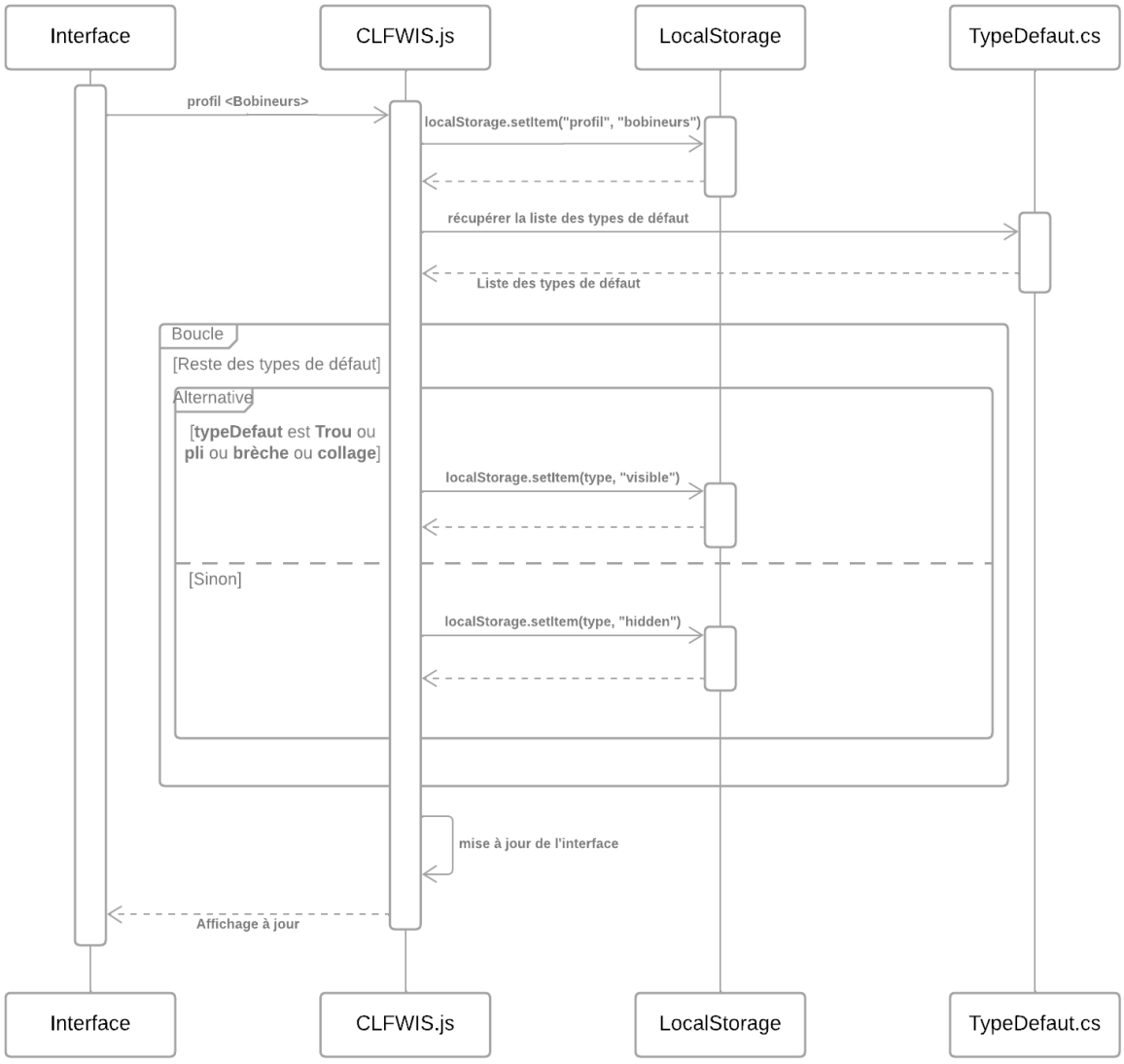


Figure 21 : Diagramme de séquence du changement de profil, réalisé avec Lucidchart

Avec le changement de profil vient le changement des défauts affichés, comme son nom l'indique, « Tous les défauts » boucle sur la liste des types de défaut et les rends visibles.

Le profil « Bobineurs » boucle de la même façon sur tous les défauts pour les cacher, puis rends certains types de défauts visibles (**Brèche***, trou, pli et **collage***). Le profil personnalisé quant à lui ne change rien, mais il servira lors de l'implémentation du bouton de liste des défauts affiché. Le but est de passer sur le profil personnalisé dès lors que l'on fait un changement dans la liste des défauts.

2.4)7. Simplification de l'ajout d'un nouveau type de défaut

Avec l'aide de mon tuteur, j'ai réussi à récupérer la liste des différents types de défauts sous forme d'une grande chaîne de caractères séparée par des virgules. Il me conseille d'ailleurs d'utiliser la fonction *split* en JavaScript pour avoir un tableau de valeurs. Avec cette indication, je factorise l'initialisation des valeurs dans le *Local Storage*.

Pour correctement limiter le travail à faire lors de l'ajout d'un nouveau défaut, je commence par énumérer les fichiers avec les endroits qui devront subir des modifications. Fâcheusement, il faut actuellement travailler dans 2 fichiers pour ajouter un nouveau défaut, et il est impossible de faire moins. Il est cependant concevable de simplifier l'implémentation en factorisant, ce qui permet tout de même de réduire le nombre de lignes à écrire.

Figure 22 : Dessin de l'apparence par défaut

J'imagine même permettre d'implémenter des nouveaux défauts sans leur représentation graphique, en attribuant un générique dans ce cas. Cela permettrait d'implémenter plus vite un nouveau défaut en laissant le temps de choisir une icône.

Étant donné qu'il faudra utiliser les dessins de défauts dans la partie représentation de la bobine et dans le bouton « liste défauts affichés », je factorise la création d'icône dans une fonction, qui prend en paramètres la taille et les coordonnées du centre.

Au final, les étapes pour rajouter un nouveau défaut sont :

- Il faut ajouter le nom du nouveau défaut dans la liste des différents types de défaut du fichier *TypeDefaut.cs*.

Une fois fait, l'ajout est fonctionnel et apparaît sous les traits de l'icône générique.

- Pour changer l'apparence, il faut modifier la fonction qui permet de créer le SVG, en ajoutant un cas dans le *switch,* avec le nom du nouveau défaut

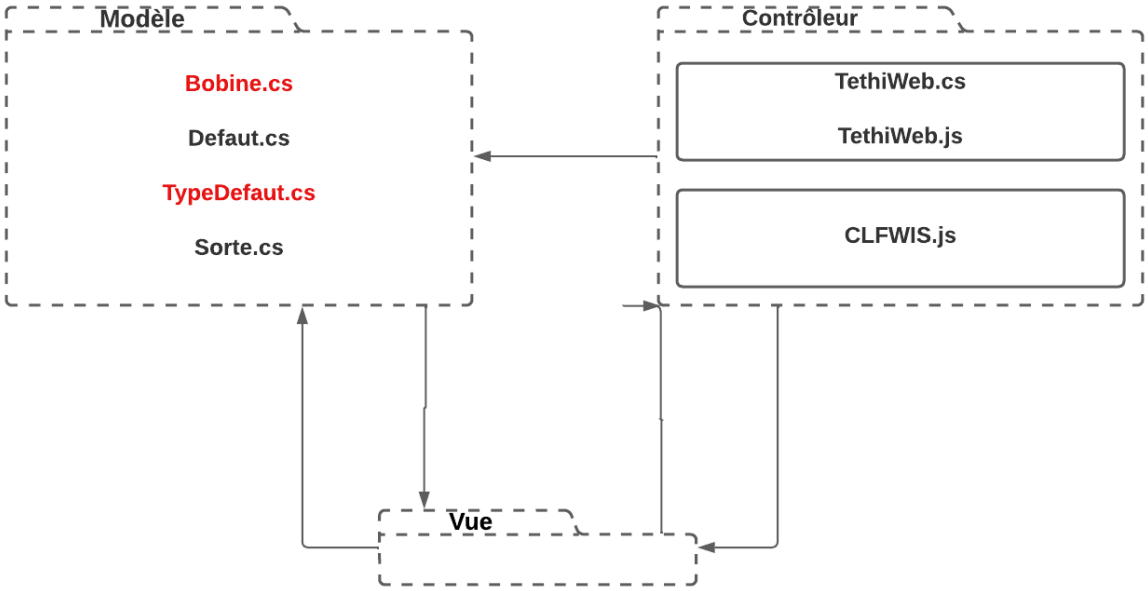


Figure 23Diagramme du modèle MVC réalisé avec Lucidchart

CLFWIS.js est le fichier JavaScript principal

Ce diagramme à n'est plus tout à fait exact à la fin du stage
Mais l'organisation se base toujours sur une architecture MVC

2.4)8. Choix personnalisé des défauts

Je continue avec le bouton de la liste des défauts affichés. Pour cela, j'explore différentes solutions pour trouver la bonne balise HTML, en regardant si Bootstrap offrait le résultat espéré par rapport à mon schéma. Ne réussissant pas à trouver ce que je voulais, je décide de le créer moi-même. J'écris donc un bouton qui permettra d'afficher la liste et un **élément de division de contenu*** (`<div>`) qui contiendra la liste. Le bouton va appliquer ou enlever la classe permettant de montrer la **div** à l'aide la fonction *toggle*. J'utilise des **éléments ancres*** (balise `<a>`) qui contiennent une balise **svg**, pour afficher le dessin du défaut et une case à cocher pour afficher la sélection.

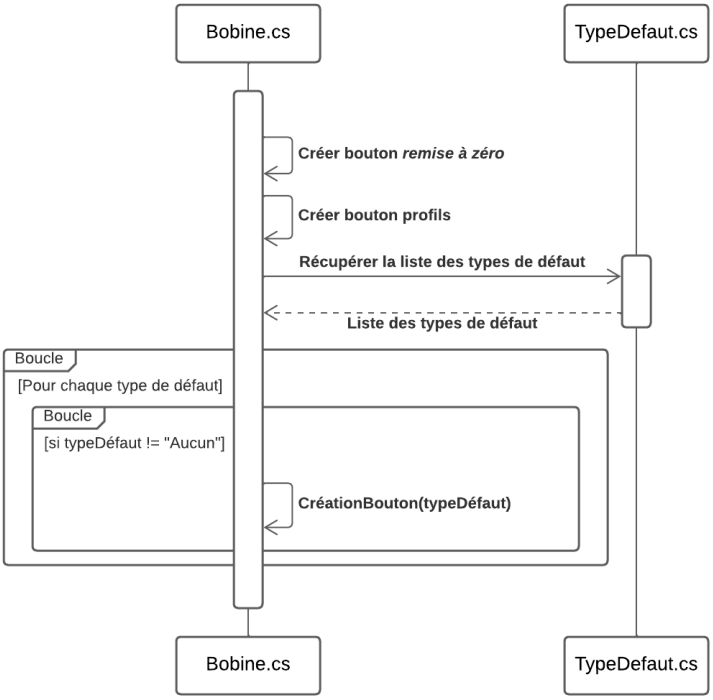


Figure 24 : Diagramme de séquence de la création du menu, réalisé avec Lucidchart

Je m'occupe du placement des éléments avec du CSS et réalise un affichage clair. Avec JavaScript, j'empêche la fermeture de mon menu lorsque l'action se fait dans celui-ci. En effet, si l'on souhaite modifier l'état de plusieurs défauts, il est plus agréable si le menu reste ouvert. Pour le fermer, j'utilise un clic en dehors. Dans le but d'afficher correctement la sélection de mes défauts, je récupère à chaque ouverture du menu l'état du défaut dans le L*ocal Storage* et en fonction, je coche la **case** ou non. Pour changer la valeur du L*ocal Storage*, je reprends l'état de la **case** modifiée. Je mets ensuite à jour mon affichage pour appliquer mes modifications. Je rajoute aussi une action sur mes éléments qui lors d'un clic, va modifier l'état de la **case à cocher** qu'il contient, pour ne pas à avoir à cliquer précisément dessus.

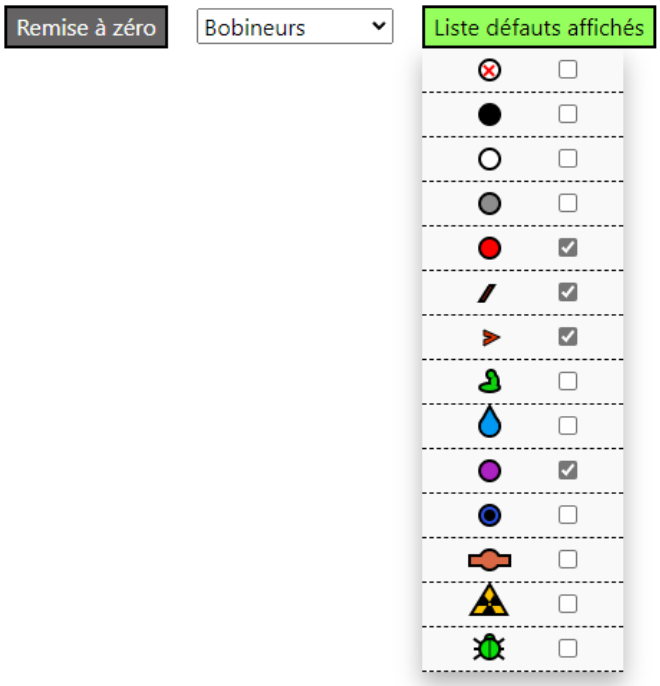


Figure 25 : Capture d'écran des boutons de l'application, profil « Bobineurs et liste des défauts affichés

Concernant le fonctionnement, il ressemble en principe à celui du profil. Lorsque l'on clique sur un type de défaut, on modifie son état dans le L*ocal Storage* et le profil devient le profil personnalisé.

Je présente mes boutons à mon tuteur, qui les trouvent corrects. Il m'explique par contre que les défauts traits ne sont pas intéressants à afficher. Je modifie donc les profils « Bobineurs » et « Tous les défauts ».

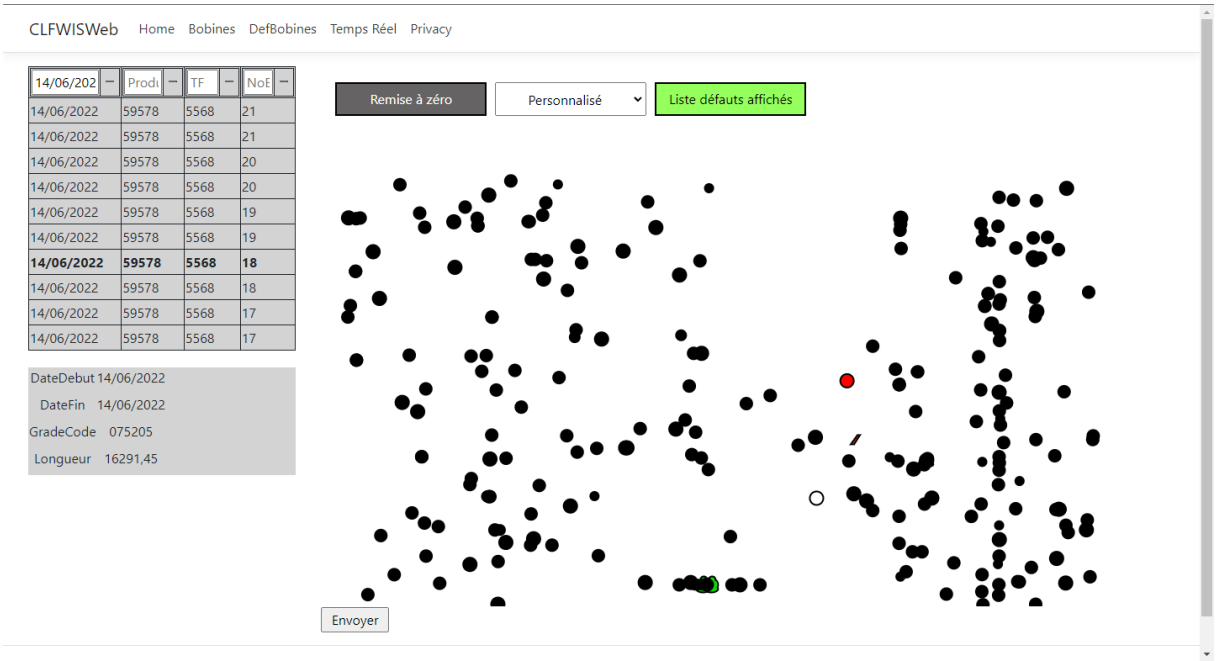


Figure 26 : Capture du résultat, défauts réels

2.4)9. Compteur de défauts par type

Un compter de défaut permet de connaître rapidement et précisément le nombre d'occurrences d'un certain type de défaut, ce qui pourrait souligner un éventuel problème lors de la production.

Pour le réaliser, j'utilise l'objet *Map*, qui me permet d'associer à une chaîne de caractères (le type de défaut), un nombre (le nombre d'occurrences). J'initialise le tableau à l'aide de la liste des types de défauts et je mets les occurrences à 0. J'incrémente le compteur à chaque défaut qui s'ajoute lors de chaque requête au serveur.

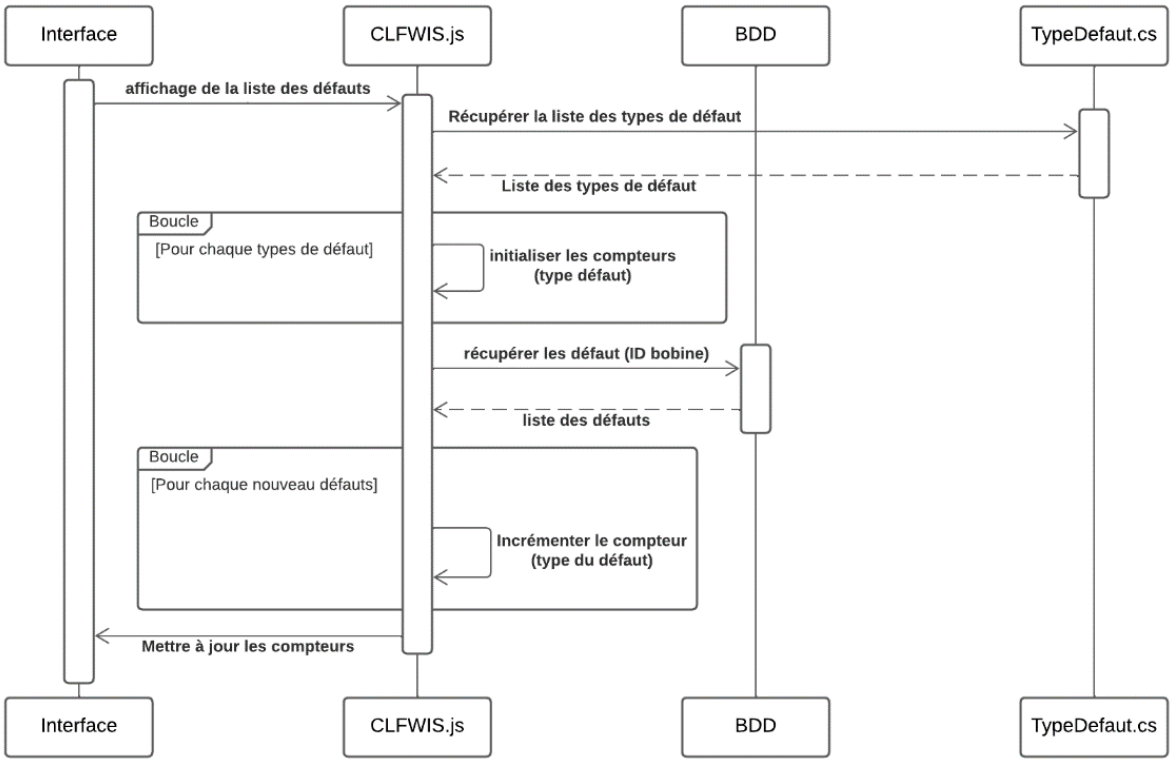


Figure 27 : Diagramme de séquence du compteur de type de défaut, réalisé avec Lucidchart

Pour afficher le compte, je rajoute une balise *Span* dans ma liste de défauts affichés. Il y a un identificateur propre pour pouvoir changer la valeur qui est de base à 0. Je mets à jour le contenu à chaque incrémentation des compteurs. Je réalise un peu de CSS pour rendre l'affichage plus lisible.

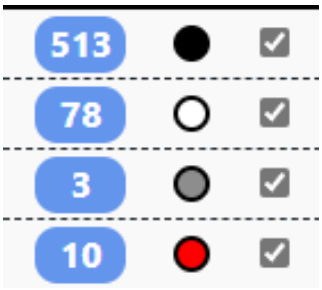


Figure 28 : Capture d'écran du compteur de défauts

2.4)10. Choix des couches

Après avoir présenté mon interface à mon tuteur, il m'explique le principe des couches, qui seraient à faire apparaître. Il n'y a pas différentes couches dans le papier, mais pour la récupération de défauts, de nombreuses prises de vue avec différents systèmes d'éclairage vont se superposer et former l'ensemble des défauts présents dans la bobine : ce sont les fameuses couches.

Une fois que la base de données a été mise à jour et contient l'information, je rajoute dans le fichier C# qui contient l'objet défaut l'attribut *Layer*. La fonction qui affiche le détail d'un défaut lorsque l'on clique dessus se met automatiquement à jour pour faire apparaître la ligne.

VICO_DEFECT_ID	33135617
CD	1593
MD	25504503
Length	2
Width	1
Layer	3

Figure 29 : Capture d'écran du détail d'un défaut
CD correspond au sens travers et MD au sens machine

Pour représenter visuellement sur quelle couche se trouve un défaut, je pense utiliser l'opacité du défaut. L'opacité va de 0 à 1, mais je ferai de 0.2 à 1 pour que les défauts de la dernière couche soient tout de même visibles. Il me suffit de diviser 0.8 par le nombre de couches pour obtenir le pas entre chaque.

Cependant, après avoir parlé avec Virginie, qui travaille au laboratoire, un tel affichage risquerait de perturber les utilisateurs et de leur faire rater certains défauts. Je vais donc laissez l'opacité maximum pour chaque couche et diminuer l'opacité des couches non sélectionnées pour les cacher en partie. Je crée à la suite le bouton en reprenant ce que j'ai fait sur la liste de défauts.

Je passe donc à la réalisation avec l'allocation du *Local Storage* pour chaque couche en donnant une valeur de 1 pour l'opacité.

Je passe ensuite à la création des différentes options du bouton (une pour chaque couche). Ne sachant pas comment récupérer les différentes couches dans mon fichier C# où j'ai ajouté le bouton des défauts, j'y ajoute juste une **div** qui contiendra mes options, que j'ajoute avec mon fichier *JavaScript* principal. J'ajoute donc dans ma fonction d'allocation mémoire, à condition que l'option correspondante au niveau de couche en paramètre n'existe pas déjà. Sinon j'ajoute une balise ancrée et une boite à cocher. Et de la même façon que pour mes choix de défauts, décocher met les défauts de cette couche en « inactif », en changeant leur opacité par 0,2. Je rajoute aussi une condition dans la fonction qui permet de cliquer sur un défaut et sur celle pour le mettre en surbrillance. Cette condition empêche de faire les actions si le défaut est sur une couche « inactive ».

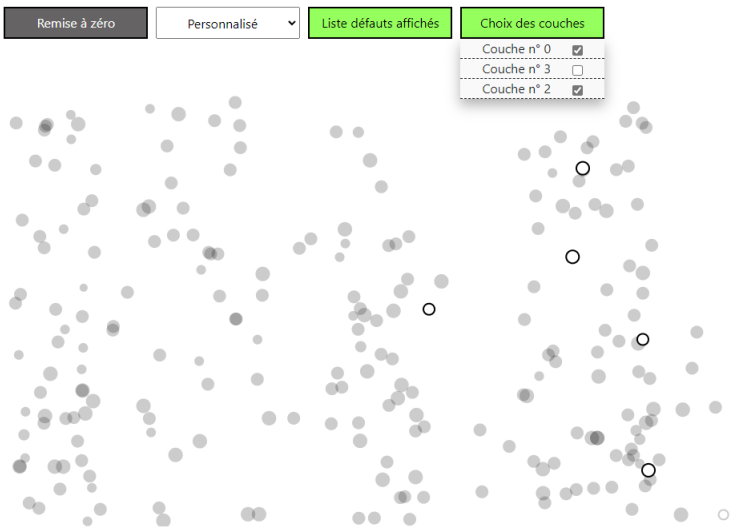


Figure 30 : Capture du résultat Couches 0 et 2 sélectionnées, défauts réels

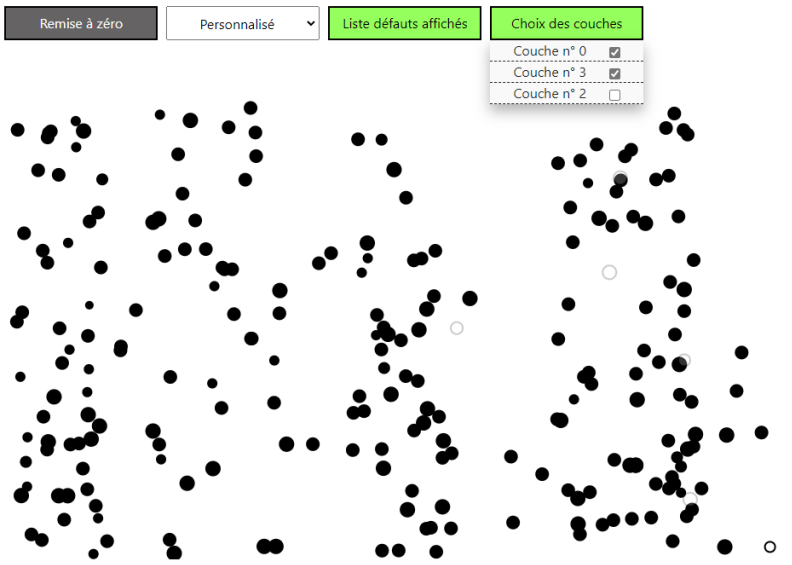


Figure 31 : Capture du résultat Couches 0 et 3 sélectionnées ; défauts réels

Je discute avec un automaticien au sujet de ce que je pourrai rajouter. Il m'explique que le détail d'un défaut contient des mots anglais (Layer pour couche par exemple) devrait être entièrement en français étant donné que cette interface ne sera utilisée que par le personnel de l'usine qui parle donc français. Je cherche alors un moyen de changer les labels pour utiliser des termes français et non anglais. Je tente d'abord de changer le nom dans les fichiers qui créent les objets (des fiches descriptives qui contiennent les diverses informations telles que le type ou bien la position) correspondant à un défaut mais cela pose un problème de référence, le lien entre la carte qui identifie la nature d'un défaut et le fichier ne se fait plus, je dois donc laisser les noms inchangés. Je me résous donc à changer les termes dans ma fonction d'affichage du détail lors du clic sur un défaut. Pour ça, je récupère avec JQuery la liste exacte des éléments qui composent le tableau et j'utilise un switch qui en fonction des mots-clés comme « *Layer* », traduit le terme (par « Couche » dans cet exemple). Une fois terminé, mon tuteur m'explique que malgré le fait que l'idée de traduire soit bonne, il faut utiliser les traductions au plus tôt, donc lors de l'acquisition des informations. Je ne peux donc pas appliquer mon code et le laisse en commentaire.

2.4)11. Amélioration de l'apparence de l'interface

J'améliore l'apparence de mes boutons pour pouvoir présenter quelque chose de beau. Je modifie ensuite le bouton « Envoyer ». J'augmente sa taille pour le rendre plus voyant, mais modifie aussi son apparence, pour qu'il n'attire pas trop le regard par rapport aux défauts. Le bouton est donc blanc avec une bordure et un texte bleu. J'ajoute une petite animation, lorsque l'on passe son curseur sur le bouton, il vire au bleu avec un texte blanc.

Sans le curseur **Avec le curseur**

Figures 32 : Captures d'écran du bouton « Envoyer » aux deux états

J'utilise donc le même principe pour le bouton de « remise à zéro », qui passe du noir au blanc.

Sans le curseur **Avec le curseur**

Figures 33 : Captures d'écran du bouton de « remise à zéro » aux deux états

Je termine mes boutons en ajoutant le changement de couleur progressif sur mes boutons « Liste défauts affichés » et « Choix des couches ». Ceux-ci passent du vert clair à un vert plus foncé.

Sans le curseur **Avec le curseur**

Figures 34 : Captures d'écran du bouton « Liste défauts affichés » aux deux états

Je présente ensuite à une personne travaillant au laboratoire que je connais des captures d'écrans de l'interface pour avoir son avis, elle m'explique que l'insecte en vert ne se voit pas assez et risque d'être confondu avec des taches d'huile. Je modifie en conséquence la couleur, en le mettant jaune clair. De cette façon, il se différencie un peu mieux du reste des défauts.

2.4)12. Affichage des coordonnées du curseur

En présentant l'interface à un futur utilisateur, il me demande de rajouter l'affichage des coordonnées du curseur dans la bobine.

Pour ce faire, rien de plus simple, je récupère déjà cette information lors de mon zoom centré sur le curseur. Je factorise alors le calcul de coordonnées puis j'ajoute deux événements : l'un sur le déplacement de la souris dans l'espace SVG, qui va permettre de mettre à jour l'affichage et l'autre événement sur la sortie du curseur de l'espace SVG qui va afficher un message dans la balise des coordonnées, pour indiquer que le curseur est en dehors de la bobine.

J'utilise une grille, pour que tous les éléments (les boutons et les coordonnées) fasse la même taille, s'alignent bien et se placent à distance égale.

Figure 35 : Menu de boutons avec un grand écran
Le curseur est dans la bobine, défauts réels

Je continue à réfléchir à la disposition de mon menu pour rendre l'interface ***responsive****. En cas d'écran avec une largeur inférieur à 1325 pixels, je rajoute une ligne dans ma grille, qui va contenir les coordonnées du curseur.

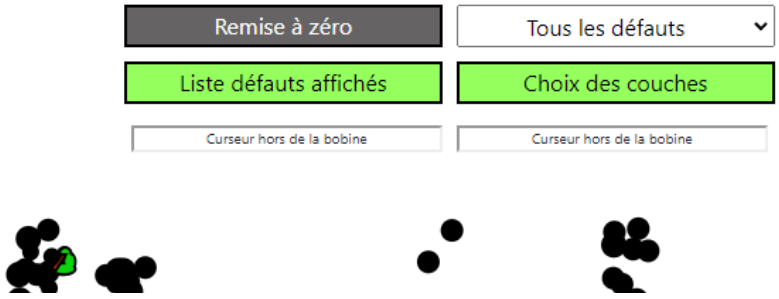


Figure 36 : Menu de boutons avec une longueur d'écran en dessous de 1325 pixels
Le curseur est en dehors de la bobine

2.4)13. Finalisation de l'interface

En présentant au chef de service, je remarque qu'il a tendance à choisir différentes bobines sans attendre la fin du chargement de la précédente, ce qui génère des problèmes lors de l'affichage (clignotements). Je cherche donc à bloquer les requêtes le temps du chargement.

Le tableau de choix de bobine n'étant pas composé de boutons, je ne peux pas utiliser l'instruction qui désactive un bouton. Je pense donc à créer une variable **booléenne*** initialiser à « **faux »** et qui prendrait la valeur « **vrai** » au début d'une requête, pour reprendre « **faux** » à la fin de la requête. Lors du clic, on vérifie alors si la valeur est à « **faux** » avant d'agir, sinon il ne se passe rien. La variable ne fonctionne pas correctement, je décide alors d'utiliser le ***session Storage****. Cette solution ne fonctionne pas avec des booléens, j'utilise donc des chaînes de caractères qui sont « non » pour « faux » et « enCours » pour « vrai ».

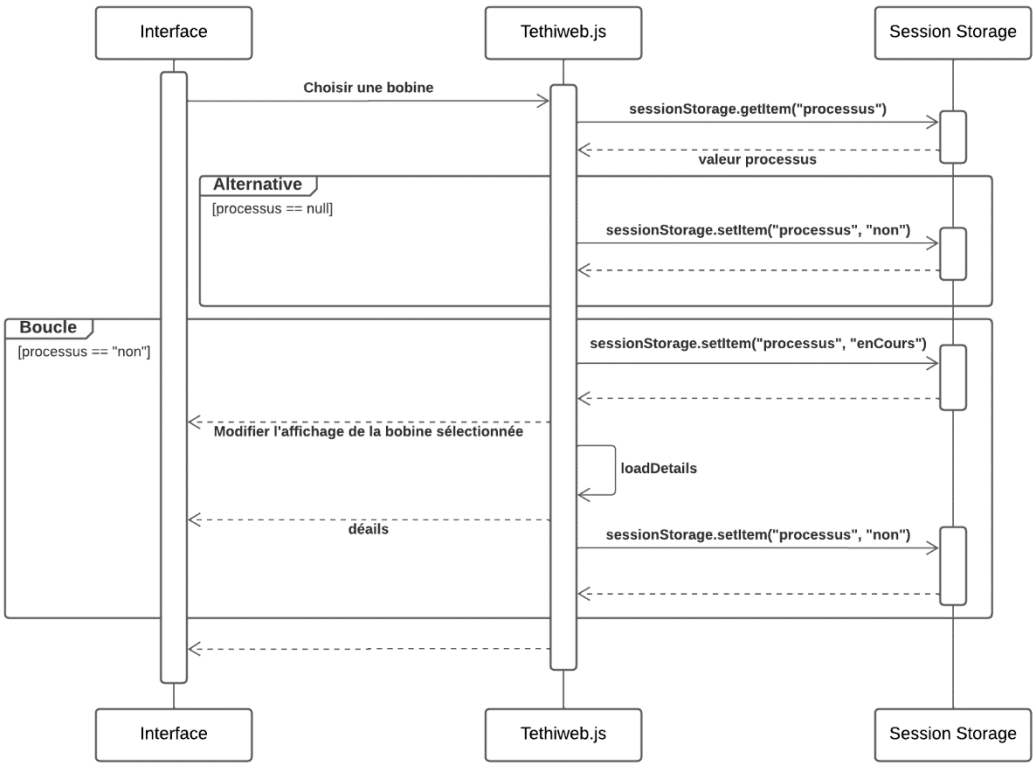


Figure 37 : Diagramme de séquence du changement de bobine

Il me faut ensuite supprimer le bouton « Envoyer » qui se révèle être inutile au final. Je ne peux pas le supprimer personnellement, même après avoir essayé d'utiliser une suppression en JavaScript. Je demande donc de l'aide à mon tuteur qui en profite aussi pour mettre à niveau mon code avec le sien. Il modifie aussi ma fonction qui recalcule l'espace disponible en utilisant une taille de 0 si le bouton « Envoyer » n'est pas trouvé.

2.4)14. Fonctionnalité annulée : connexion OPC

Afin de préparer la mise en place de ma nouvelle interface sur les machines, il faut établir une connexion entre les ordinateurs et l'automate, pour que mes données correspondent parfaitement à la machine que je souhaite surveiller.

Le système actuel fonctionne avec le **modèle AJAX***, qui fonctionne correctement, mais contient certaines limites (des limites de sécurité par exemple), que ne possède pas l'**OPC***. Le montage du système a été réalisé pour pouvoir correctement implémenter une connexion OPC.

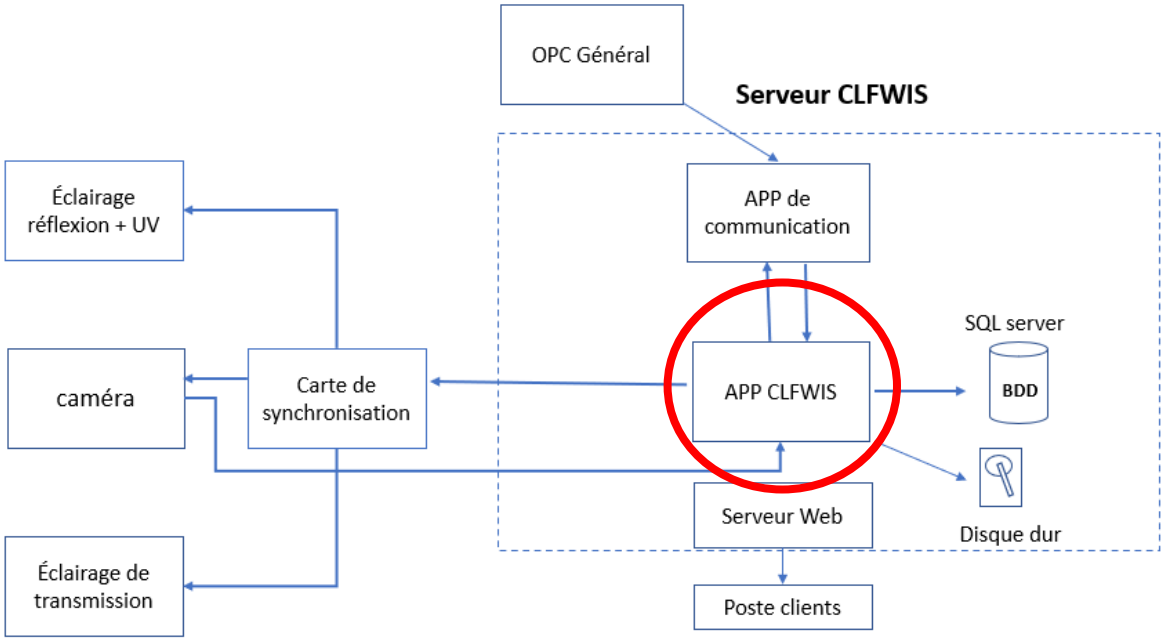


Figure 38 : Schéma du montage du système, [Imane, 2021]

Mon tuteur me demande donc de tester la technologie, voir si elle est applicable à notre projet. Pour cela, il me présente un projet sur **GitHub*** [Robert Rajakone, 2018] qui semble être un bon exemple. Je cherche donc à comprendre comment le modèle fonctionne.

Je finis par trouver qu'il est impossible de nous passer de Node JS dans notre cas et mon tuteur ne voulant pas de ça (il faudrait l'installer et le tenir à jour sur le secteur), nous abandonnons l'idée qui est trop contraignante et restons sur le modèle AJAX.

2.4)15. Travaux débutés

Quelques travaux ont été débutés mais n'ont pas pu être terminés. Lors d'une mise en commun du code avec mon tuteur, celui-ci a créé un nouvel onglet, appelé « Temps Réel », qui doit faire apparaître les derniers kilomètres de papier (longueur au choix de l'utilisateur).

Pour commencer, je reprends dans mon fichier C# qui permet l'affichage d'une bobine. Je sépare les fonctions pour mettre de côté la base commune qui me servira aussi dans l'affichage en temps réel (la création de la barre de menu et la création d'un défaut). Je mets ces méthodes en *static*, ce qui permettra de les utiliser dans d'autres objets qu'un objet « Bobine ». Je réfléchis aussi à la modification de mon fichier JavaScript, qui sert à redimensionner la représentation du papier et de ses défauts notamment. La plupart du code sera commun, seul le calcul de l'espace disponible va changer (étant donné que le tableau de choix de bobine n'apparaît pas en temps réel), ce qui me fais envisager le patron **Stratégie***. Je décide de me concentrer d'abord sur mon fichier C# et de créer l'espace avec les défauts avant de mettre les fonctionnalités. Concernant la récupération des défauts présents sur les n derniers mètres de papier, je pense à créer une liste de bobines correspondantes aux derniers mètres de papiers car il faut pour bien placer les défauts leurs positions et la longueur de la bobine précédente (si le défaut est en bas de la feuille mais correspond à la deuxième bobine, il ne doit pas apparaître sur le bas de ma partie de représentation du papier.

Je n'ai pas terminé cette fonctionnalité. Deux autres ont d'ailleurs été évoquées mais pas ajoutées :

- Un indicateur de zoom, qui permet de montrer quelle portion de la bobine est affiché à l'aide de barre de navigation.

- Un indicateur de concentration des défauts

3) Conclusion

Pour conclure, je suis assez satisfait du résultat final, j'ai globalement répondu aux différentes demandes qui m'ont été formulées et j'ai produit un code bien rédigé.

Cf Annexe 7 : Capture d'écran finale de l'interface créées

Je suis cependant légèrement déçu de ne pas avoir pu finir l'entièreté du projet lors de ces 10 semaines. Je pense que j'aurais eu la possibilité de faire plus si j'avais mieux travaillé à ma conception au début du projet où j'ai dû revoir plusieurs fois mon zoom par exemple.

J'ai su apprendre de mes erreurs et m'améliorer en termes de conception tout au long du projet. J'ai eu la possibilité de mettre en pratique ma capacité de travail en autonomie et je pense avoir progressé positivement. De plus, les nombreuses notions que j'ai découvertes et celles que j'ai su mettre en pratique ont renforcé mes compétences et il me tarde d'en apprendre davantage.

Cette expérience confirme mes choix d'orientation et j'ai hâte de retourner aux Papeteries de Clairefontaine cet été pour continuer à travailler et à enfin terminer l'interface du système Viconsys.

Ayant eu l'opportunité d'être embauché en Juillet, j'espère terminer mes travaux.

4) Glossaire

Azurant : un agent azurant ou azurant optique est une molécule qui absorbe les rayonnements électromagnétiques ultraviolets entre 300 et 400 nm de longueur d'onde et réémet ensuite cette énergie par fluorescence dans le visible entre 400 et 500 nm. Cela permet de faire paraître le blanc du papier encore plus blanc. D'après [Wikipédia, 2022a].

Backend : le backend correspond à la partie invisible pour l'utilisateur, mais qui permet le bon fonctionnement du système.

Bobine : correspond au papier enroulé sur un mandrin. Les dimensions avant découpe sont 3,50m de large et 3km de long.

Bobineurs : les bobineurs sont les personnes travaillant sur les bobines. Ils s'occupent de mettre en place les bobines vides, gérer la vitesse de rotation en fonction des trous et récupérer et étiqueter les bobines complétement enroulées.

Booléen : une valeur booléenne ne peut prendre que deux valeurs : **vrai** ou **faux**.

Bootstrap : Bootstrap est une collection d'outils pour la création de sites et d'applications web.

Brèche : une brèche aussi appelée fente est une déchirure sur le bord d'une feuille.

C# : est un langage de programmation orientée objet, commercialisé par Microsoft depuis 2002 et destiné à développer sur la plateforme Microsoft .NET. D'après [Wikipédia, 2022b].

Cookie (pour un navigateur) : c'est une information laissée par une page web dans le navigateur, qui contient certaines données et qui restent un temps défini.

Collage : un collage est un défaut de collage est le fait que deux feuilles soient donc collées l'une à l'autre.

CSS : « feuilles de style en cascade ». Il s'agit d'un des langages principaux du web et permet la mise en forme et le design du contenu d'une page web. D'après [MDN web docs, 2022a].

Elément ancre : crée un lien hypertexte vers des pages web, des fichiers, des adresses e-mail, des emplacements se trouvant dans la même page ou tout ce qu'un URL peut adresser. D'après [MDN web docs, 2022b].

Elément de division de contenu : l'élément HTML `<div>` (ou division) est le conteneur générique du contenu du flux. Il n'a aucun effet sur le contenu ou la mise en page tant qu'il n'est pas mis en forme d'une manière quelconque à l'aide de CSS. D'après [MDN web docs, 2021a].

GitHub : c'est une plateforme permettant d'héberger du code.

HTML : signifie *HyperText Markup Language* qu'on peut traduire par « langage de balises pour l'hypertexte ». Il est utilisé afin de créer et de représenter le contenu d'une page web et sa structure. D'après [MDN web docs, 2021b].

IDE : un environnement de développement intégré, ou IDE, est un logiciel de création d'applications, qui rassemble des outils de développement fréquemment utilisés dans une seule interface utilisateur graphique (GUI). D'après [Red Hat, 2019].

JavaScript : c'est un langage de script léger, orienté objet, principalement connu comme le langage de script des pages web. D'après [MDN web docs, 2021c].

JQuery : c'est une bibliothèque JavaScript libre et multi-plateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web. D'après [Wikipédia, 2022c].

Local Storage : il s'agit du stockage local d'un navigateur. Les données qu'il contient sont persistantes, elles restent tant qu'il n'y a pas d'action de suppression ou de réécriture par-dessus.

Machine à papier : il s'agit des différentes machines nécessaires à la transformation du papier tel que le pulpeur ou le sécheur.

Modèle AJAX : (*Asynchronous JavaScript + XML*) n'est pas une technologie en soi, mais un terme désignant une « nouvelle » approche utilisant un ensemble de technologies existantes, dont : HTML ou XHTML, CSS, JavaScript, DOM, XML, XSLT, et surtout l'objet XMLHttpRequest. Il permet de réaliser des applications web capable de mises à jour sans devoir recharger la page entière dans le navigateur. D'après [MDN web docs, 2022c].

OPC : l'**O**pen **P**latform **C**ommunications est une norme d'interopérabilité définie pour sécuriser les échanges de données d'automatisation industrielle. Elle est conçue pour dépendre des plateformes afin que les appareils des différents fournisseurs puissent échanger des informations. D'après [Automation sense, 2017].

Pulpeur : c'est une machine utilisée dans le recyclage du papier, constituée d'une cuve cylindrique dans laquelle un rotor, qui effectue la désintégration de la pâte, permet l'obtention de fibres individuelles, indépendantes les unes des autres et remises en suspension dans l'eau. D'après [Actu-Environnement, 2022].

Raffinage : acte de fibriller les fibres de cellulose, c'est-à-dire leur faire de petites fibrilles dans toute leur longueur. Ceci permet aux fibres de raccrocher l'une à l'autre pour former une feuille.

Refonte du papier : ce terme désigne le fait de retransformer le papier en pâte à papier

Responsive : terme anglais qui qualifie un site ou une application dont le code répond à des normes lui permettant de s'adapter à tous types de supports et à la taille de l'écran. D'après [L'internaute, 2021].

Session Storage : similaire au Local Storage, il a comme seule différence d'avoir des données non persistantes. Celles-ci s'effacent à la fermeture du navigateur.

Singleton : le patron Singleton est un patron de conception qui permet l'unicité d'un objet.

Stratégie : le patron Stratégie est un patron de conception de type comportemental grâce auquel des algorithmes peuvent être sélectionnés à la volée au cours du temps d'exécution selon certaines conditions. D'après [Wikipédia, 2022d].

SVG : (pour *Scalable Vector Graphics* en anglais, soit « graphiques vectoriels adaptables ») est un langage construit à partir de XML et qui permet de décrire des graphiques vectoriels en deux dimensions. D'après [MDN web docs, 2022d].

Switch : l'instruction switch évalue une expression et, selon le résultat obtenu et le cas associé, exécute les instructions correspondantes. D'après [MDN web docs, 2022e].

5) Bibliographie

[Imane, 2021] **Sadok Imane**. *Amélioration d'un système de détection de défaut*. Juillet – Août 2021. Diplôme d'ingénieur. INSA Hauts-de-France. 21 pages.

[Robert Rajakone, 2018] **GitHub**. Rajakone Robert (@robie2011) : *opc-ua-websocket-proxy*. https://github.com/robie2011/opc-ua-websocket-proxy. Consulté le 23 Mai 2022.

[Wikipédia, 2022a] **Wikipédia** (2022). Wikipédia *: Agent azurant*. https://fr.wikipedia.org/wiki/Agent_azurant. Consulté le 25 Mai 2022.

[Wikipédia, 2022b] **Wikipédia** (2022). Wikipédia : *C#.* https://fr.wikipedia.org/wiki/C_sharp. Consulté le 25 Mai 2022.

[MDN web docs, 2022a] **MDN web docs** (2022). MDN : *CSS : Feuilles de style en cascade*. https://developer.mozilla.org/fr/docs/Web/CSS. Consulté le 25 Mai 2022.

[MDN web docs, 2022b] **MDN web docs** (2022). MDN : *<a> : l'élément ancre*. https://developer.mozilla.org/fr/docs/Web/HTML/Element/a. Consulté le 25 Mai 2022.

[MDN web docs, 2021a] **MDN web docs** (2021). MDN : * : l'élément de division du contenu*. https://developer.mozilla.org/fr/docs/Web/HTML/Element/div. Consulté le 25 Mai 2022.

[MDN web docs, 2021b] **MDN web docs** (2021). MDN : *HTML (HyperText Markup Language)*. https://developer.mozilla.org/fr/docs/Web/HTML. Consulté le 25 Mai 2022.

[Red Hat, 2019] **Red Hat** (2019). Red Hat : *Un environnement de développement intégré, qu'est-ce que c'est ?* https://www.redhat.com/fr/topics/middleware/what-is-ide. Consulté le 25 Mai 2022.

[MDN web docs, 2021c] **MDN web docs** (2021). MDN : *JavaScript*. https://developer.mozilla.org/fr/docs/Web/JavaScript. Consulté le 25 Mai 2022.

[Wikipédia, 2022c] **Wikipédia** (2022). Wikipédia : *jQuery*. https://fr.wikipedia.org/wiki/JQuery. Consulté le 25 Mai 2022.

[MDN web docs, 2022c] **MDN web docs** (2022). MDN : *AJAX*. https://developer.mozilla.org/fr/docs/Web/Guide/AJAX. Consulté le 23 Mai 2022.

[Automation sense, 2017] **Automation sense** (2017). *Les serveurs OPC pour les nuls*. https://www.automation-sense.com/blog/automatisme/les-serveurs-opc-pour-les-nuls.html. Consulté le 23 Mai 2022.

[Actu-Environnement, 2022] **Actu-Environnement** (2022). *Dictionnaire environnement : pulpeur*. https://www.actu-environnement.com/ae/dictionnaire_environnement/definition/pulpeur.php4 Consulté le 25 Mai 2022.

[L'internaute, 2021] **L'internaute** (2021). *Dictionnaire français : responsive*. https://www.linternaute.fr/dictionnaire/fr/definition/responsive/. Consulté le 2 Juin 2022.

[Wikipédia, 2022d]. **Wikipédia** (2022). Wikipédia : *Stratégie (patron de conception)* https://fr.wikipedia.org/wiki/Strat%C3%A9gie_(patron_de_conception). Consulté le 15 Juin 2022.

[MDN web docs, 2022d] **MDN web docs** (2022). MDN : *SVG (Scalable Vector Graphics)*. https://developer.mozilla.org/fr/docs/Web/SVG. Consulté le 25 Mai 2022.

[MDN web docs, 2022e] **MDN web docs** (2022). MDN : *Switch*. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Statements/switch. Consulté le 25 Mai 2022.

6) Annexes

6.1) Planning

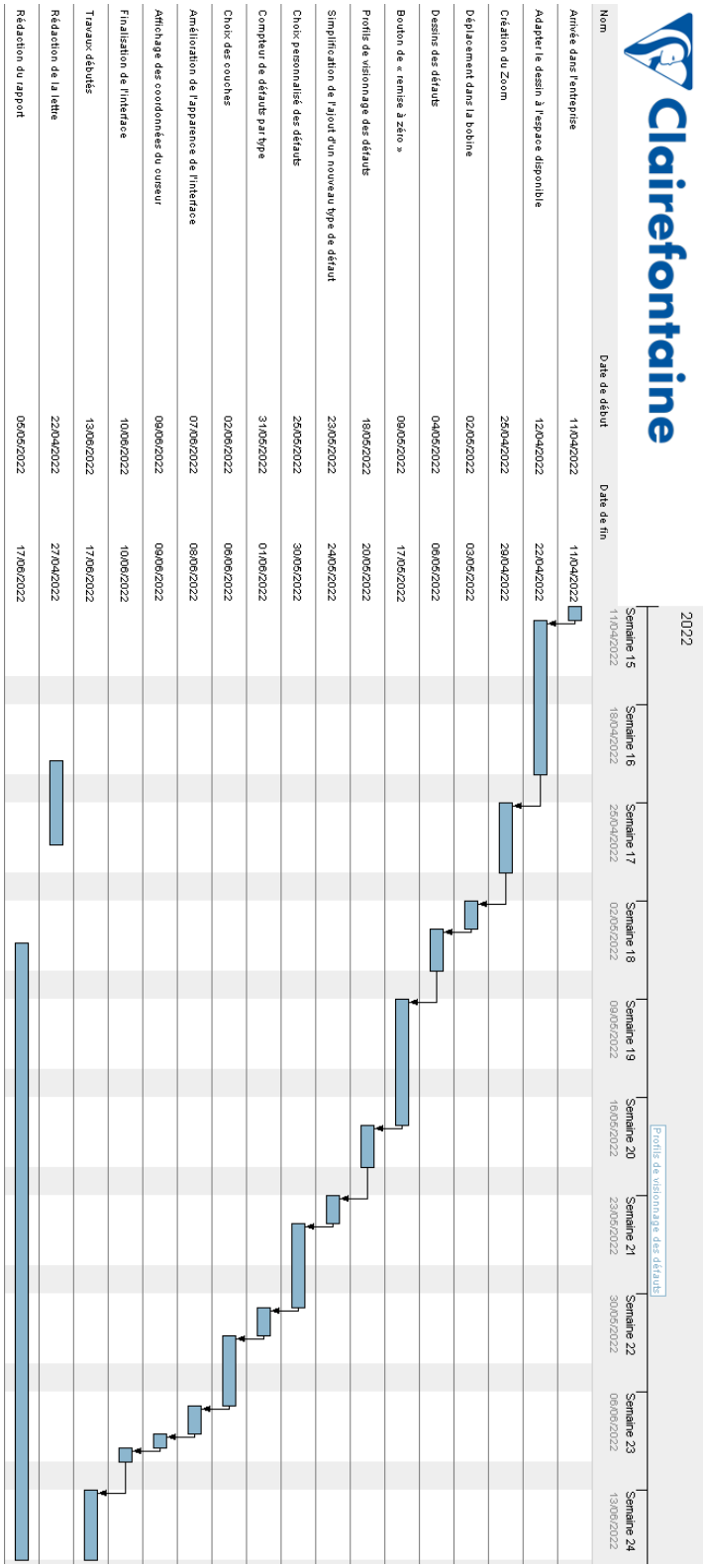


Diagramme réalisé avec GanttProject (http://www.ganttproject.biz/).

6.2) Présentation détaillée de l'entreprise

Pendant le XVIème siècle, les premiers moulins à papier sont créés, la première papeterie implantée sur le site d'Etival date 1512 : la tradition papetière remonte donc déjà plusieurs siècles pour le département des Vosges.

Ensuite, en 1799, la première machine à papier en continu est réalisée par le français luis Nicolas Robert. Cette machine appelée « machine Robert » marque la révolution de l'industrie papeterie et ce prototype inspirera les installations plus modernes.

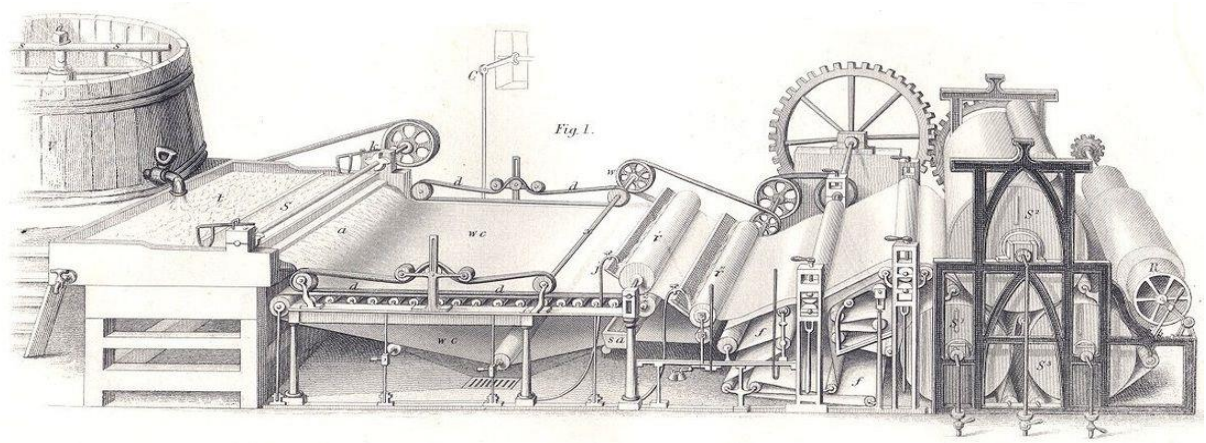


Figure 39 : Machine '' ROBERT"

En 1858, Jean Baptiste Bichelberger décide d'installer un nouvel établissement capable de créer du papier en continue. L'établissement devient par ce fait une papeterie moderne et le nom attribue est les Papeteries de Clairefontaine.

Figure 40 : Photographie de Jean Baptiste Bichelberger

En 1928, Charles Nusse, l'arrière-petit-fils de Jean-Baptiste Bichelberger, crée un atelier à Paris pour imprimer puis relier en série des registres de comptabilité. Des agendas sont également fabriqués dans cet atelier, ce qui inaugure avec Exacompta la politique de marque et de qualité du groupe.

Figure 41 : Photographie de Charles Nusse

Une vingtaine d'années plus tard, en 1950, Charles Nusse propose aux écoliers des cahiers dont le papier doux à l'écriture et les couvertures solides contrastent avec les articles de l'époque.

Par la suite, l'évolution des besoins et la volonté d'innovations mettent le groupe Exacompta Clairefontaine en position de leader sur de nouveaux marchés comme les papiers de photocopier, les articles de classement, les calendriers, les beaux-arts et les loisirs créatifs. Par ailleurs, des percées significatives sont également faites vers la bagagerie, les articles personnalisés ainsi que, plus récemment, le développement de photographies.

Figure 42 : Logo du groupe Exacompta

La production de papier est réalisée dans 3 usines utilisant des fibres vierges et une usine utilisant des fibres recyclées. Celle-ci représente 40% de l'activité totale du groupe. D'une manière générale, le groupe Exacompta Clairefontaine est composé de 5 départements :

- Exacompta (Paris,75), pour les articles de bureau et de classement. Ce département est également implanté dans plusieurs départements français ans qu'en Autriche, en Allemagne et au Royaume-Uni.
- Clairefontaine Rhodia (ottmarshein,68) produisant les papeteries scolaires, de bureau, de beaux-arts, ainsi que les loisirs créatifs avec des implantations dans les départements de Maroc, Rhône, en suède, au Danemark et au Royaume Uni.
- Papeteries de Clairefontaine (Etival-Clairefontaine,88), traitent de la production de papier, de ramettes, de bobines de papier, de cahiers et d'enveloppes dans les Vosges, Papeterue de Mandeure dans le Doubs pour les papiers épais, Everbal dans l'Aisne pour le papier recyclé et Schutt aux Pays-Bas pour le papier d'Arts.
- A.F.A (Paris,75), pour les agendas avec des implantations dans plusieurs départements français.
- Photo Web (Saint-Egrève,38) gérant principalement de photographie, les albums et les calendriers.

Le site d'Etival-Clairefontaine est spécialisé dans la production de papier d'impression et d'écriture. En effet l'usine se déploie le long de la Meurthe sur un terrain de 40 hectares dont 15 entre eux sont bâtis. Ces bâtiments abritent diverses unités : Fabrication de papier, Finition et emballage, centre Logistique, façonnage des cahiers et d'enveloppes, ateliers de maintenance et bureau d'études et services généraux.

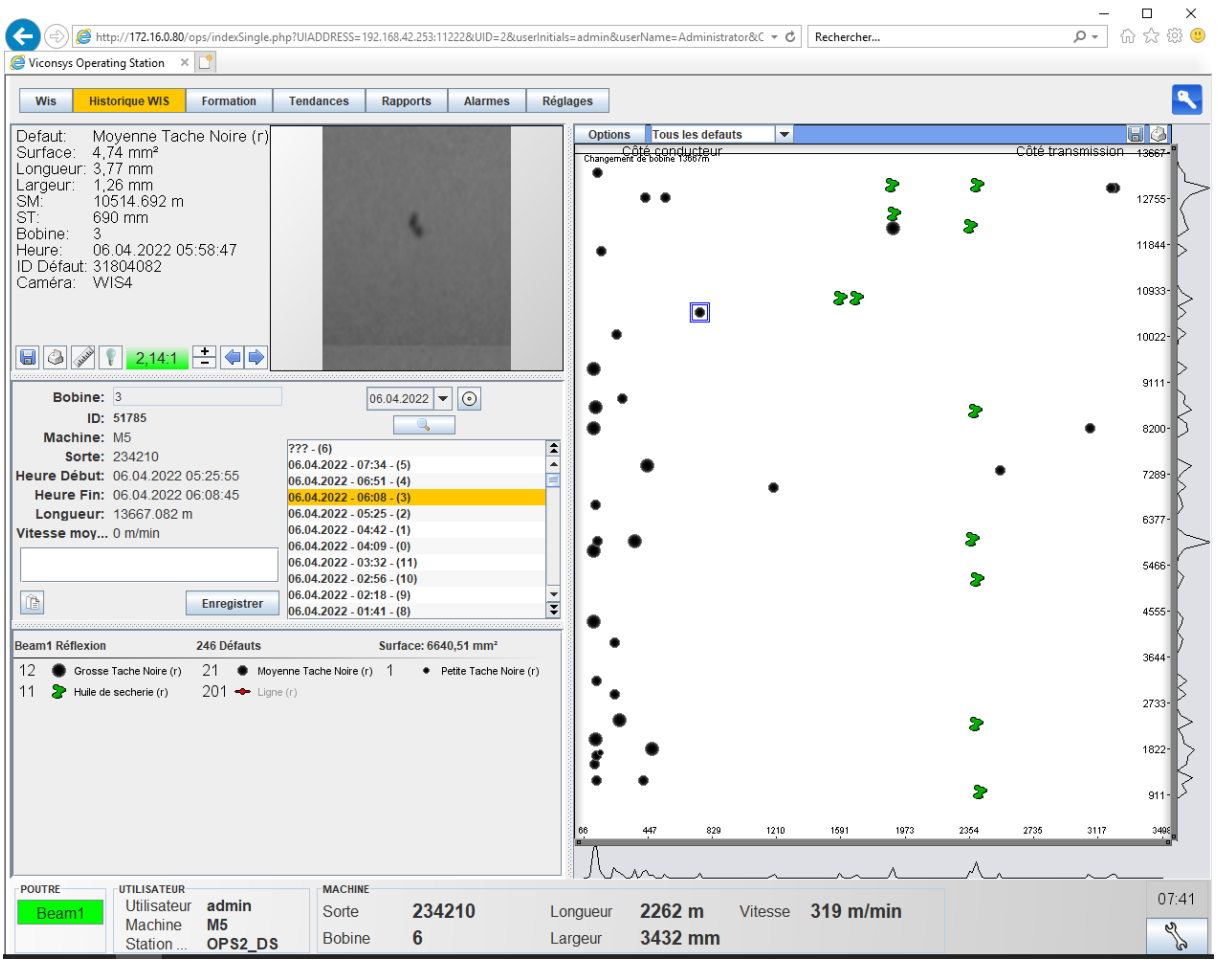
Les trois activités : papier, enveloppes et cahiers, sont toujours présentes sur le site. L'usine dispose de deux machines à papier de 3,40m de large. Pour l'une d'entre elles, la vitesse de production atteint 1000m par minute, ce qui représente 60km et 15 tonnes de papier par heure.

Les images et les informations proviennent du rapport de stage [Imane, 2021] et de l'usine.

6.3) Photo de la maquette du futur montage

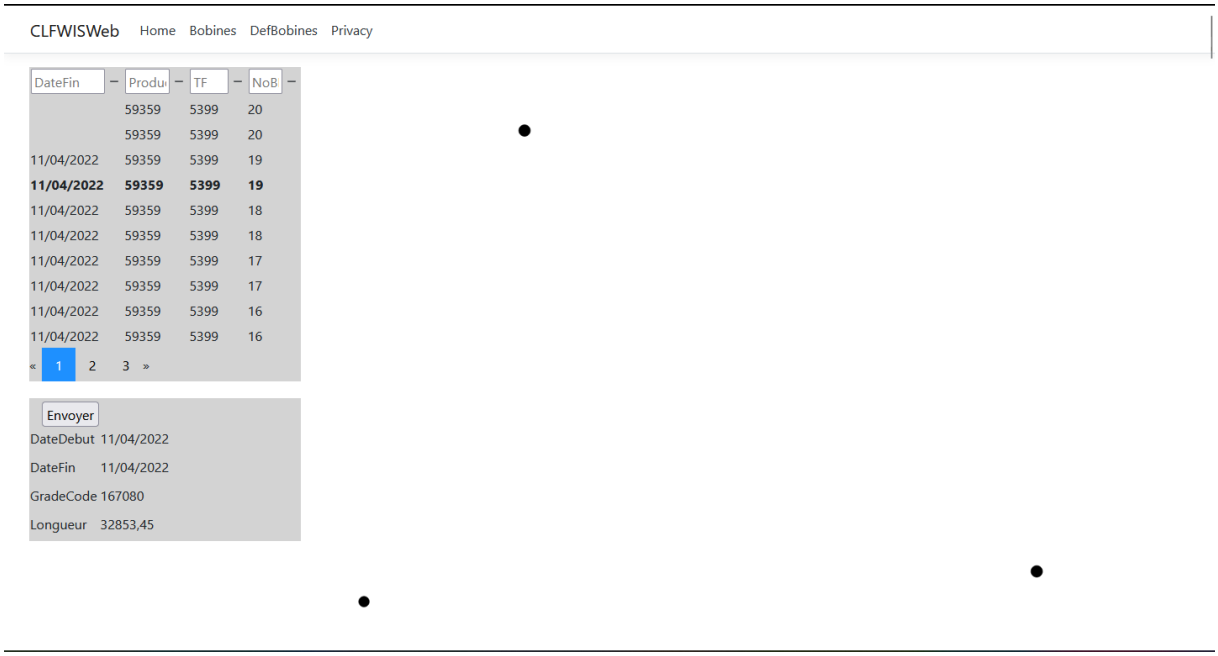
Les lumières éclairent l'une après l'autre de manière cyclique et les caméras récupèrent les images pour repérer les défauts. La caméra inclinée sert pour l'éclairage blanc et UV.

6.4) Capture d'écran de l'ancienne application Viconsys



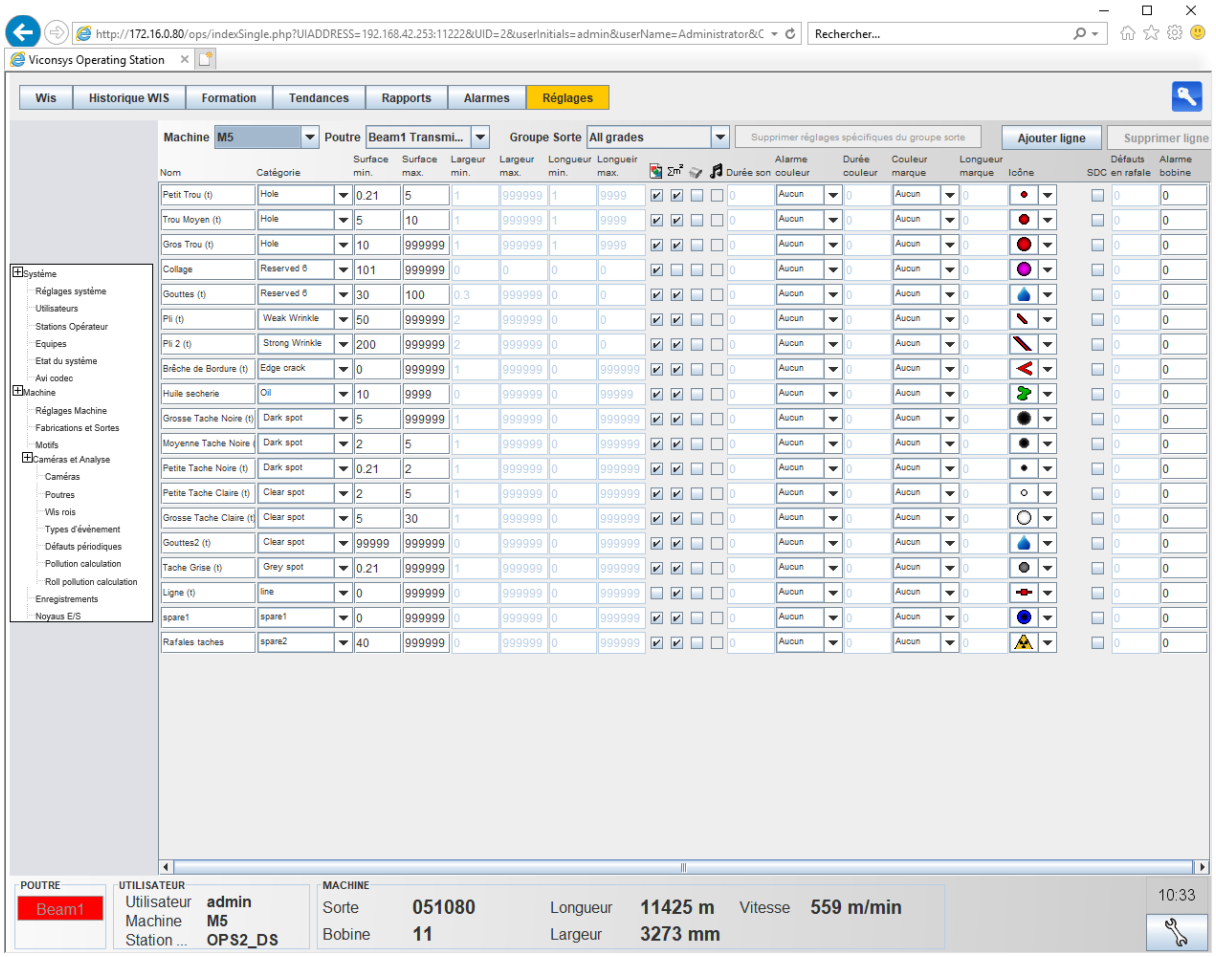
Cette capture d'écran provient de l'ancienne application du système Viconsys employé par les Papeteries de Clairefontaine. À droite se trouve la partie représentant la bobine avec ses défauts. C'est sur cette partie que se concentre ma mission principale. Sur la partie gauche, demeure les informations relatives à la bobine en cours de visionnage avec le choix d'un changement de bobine à l'aide d'un historique.

6.5) Capture du squelette de l'application



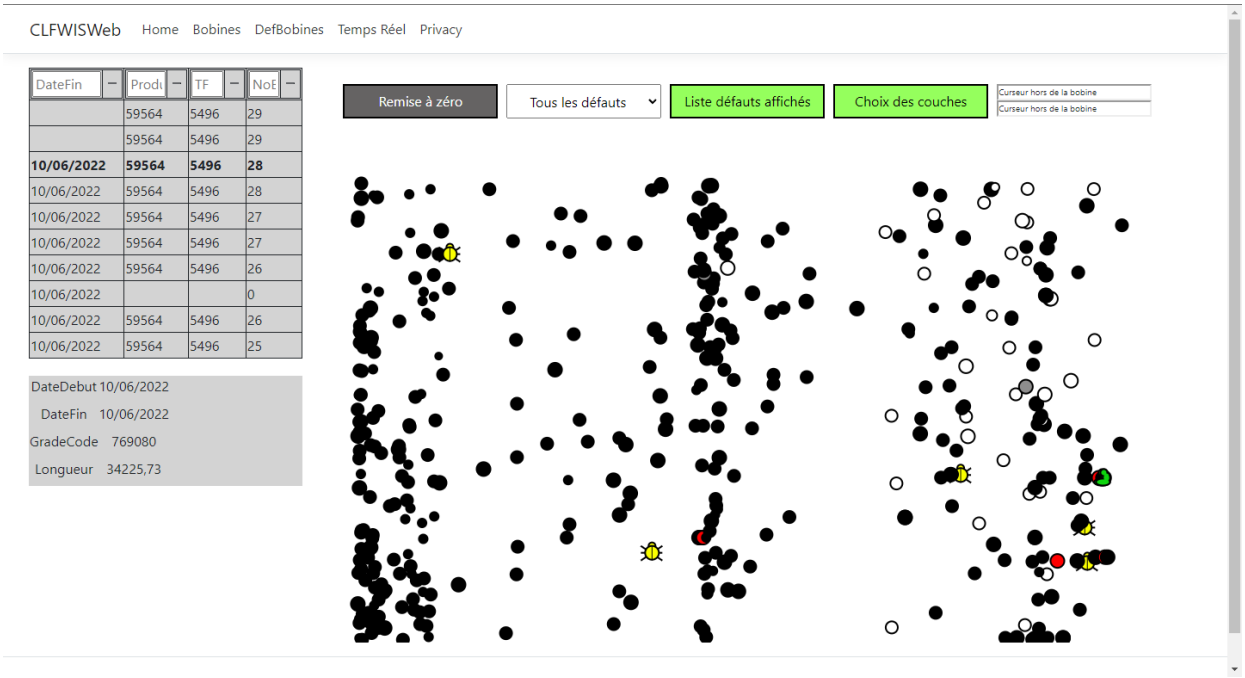
Cette capture est l'interface que mon tuteur m'a donnée à mon arrivée. La bobine y est affichée à échelle 1::1, c'est pour cela qu'il y a une barre de navigation à droite de l'écran. Il n'y aucune fonctionnalité à part le clic sur un défaut pour en voir les détails et la mise en avant (petit cadre jaune) du défaut quand le curseur de la souris le pointe. Il n'y a pas tous les défauts d'afficher, et ceux afficher ont tous l'apparence de la tache sombre (un point noir comme sur cette image). Les défauts sont les défauts réels de la bobine.

6.6) Liste des défauts de l'ancienne application



Cette capture d'écran provient de l'ancienne application du système Viconsys. On y voit une liste non-exhaustive des différents défauts avec les icônes qui les représentent.

6.7) Capture d’écran finale de l’interface créées



Cette capture d’écran provient du nouveau système Viconsys. Il s’agit de l’onglet « DefBobines », qui permet de voir l’historique des bobines. Les défauts sont les défauts réels de la bobine.

FICHE RAPPORT DESTINEE A LA BIBLIOTHEQUE
(à insérer à la fin du rapport)

RAPPORT CONFIDENTIEL ET NE DEVANT PAS FIGURER A LA BIBLIOTHEQUE :

☐ Oui ☒ Non

NOM ET PRENOM DE L'ETUDIANT : Vinot Mathieu

DUT : INFORMATIQUE

☒ S4 ☐ Année Spéciale

LICENCE PROFESSIONNELLE

☐ ASRALL ☐ CIASIE

_____

TITRE DU RAPPORT : Réalisation d'un système d'affichage de défauts du papier

Nom de l'Entreprise : Les Papeteries de Clairefontaine

Adresse : 19 Rue de l'Abbaye, 88480 Étival-Clairefontaine

Type d'activité (domaines couverts par l'entreprise) : Industrie papetière

Nom du parrain (enseignant IUT) : Longhais Alain

Mots-clés (sujets traités) : Création d'une interface en HTML, CSS, JavaScript et C#.

Résumé

Le sujet du stage est de refaire l'interface du système Viconsys, qui permet l'affichage des défauts présents dans le papier en sortie de machine. Il a fallu refaire des fonctionnalités comme un zoom, redessiner les défauts, créer des boutons pour choisir les types de défauts à afficher …