

**DIOT Clément**

**BTS SIO 1<sup>ère</sup> année**

**Lycée Jean Rostand**

## **AMELIORATION DES RESULTATS DE LA COUPE FEUTRE POUR LE SERVICE QUALITE**

Rapport de stage de BTS SIO



Stage : 31/05/2021 – 25/06/2021

Société : BIC Rasoirs

Tuteur : Pascal Hennion – Responsable logistique et informatique – BIC Rasoirs

Tuteur : Jérémy Brassart – Chef de projets informatiques – BIC Rasoirs

Responsable pédagogique : Agnès Kintzler – Enseignante – Lycée Jean Rostand

## Table des matières :

I.	Présentation de l'entreprise .....	3
1.	BIC de 1945 à aujourd'hui.....	3
2.	BIC rasoirs, l'usine de production .....	5
3.	Organisation de BIC rasoirs.....	7
4.	Le MIS, le service informatique .....	9
II.	Mes Missions.....	11
1.	Liste des missions .....	11
III.	Description de la mission «réalisation d'un module web améliorant la consultation des résultats des coupes feutre de lames finies ».....	11
1.	Genèse du projet .....	11
2.	Environnement de travail .....	12
3.	Planning .....	13
4.	Développement du formulaire de consultations des résultats de coupe feutre .....	13
a.	Utilisation de BRREQ .....	14
b.	Présentation du formulaire.....	16
c.	Le code complet permettant d'obtenir le formulaire .....	17
d.	Présentation du code permettant d'afficher la liste déroulante contenant les types de lames ....	19
e.	Présentation du code permettant de choisir la période .....	21
f.	Affichage des données .....	22
5.	Développement de la partie « saisie d'un commentaire » .....	28
a.	Utilisation de BRISE.....	28
b.	Présentation des formulaires.....	29
c.	Modification de la base de données pour intégrer le commentaire.....	29
d.	Le code complet permettant d'obtenir les formulaires .....	31
e.	Présentation du code permettant de vérifier le numéro de lot et l'existence d'un commentaire	32
f.	Présentation du code permettant d'insérer un nouveau commentaire .....	36
g.	Présentation du code permettant de modifier un commentaire.....	37
h.	Présentation du code faisant le lien entre la partie BRREQ et la partie BRISE.....	39
IV.	Configuration d'un pc sur le domaine de l'entreprise et migration de données .....	42

1. Genèse de la mission .....	42
2. Réalisations .....	42
3. Bilan .....	43
4. Remerciements .....	44
5. Annexes .....	45
Annexe 1 : Schéma réseau de l'usine .....	45
Annexe 2 : Structure initiale de la base CoupeFeutre .....	46

## I. PRESENTATION DE L'ENTREPRISE

### 1. BIC de 1945 à aujourd'hui...

**BIC** est une société française fondée le 25 octobre 1945 par Marcel BICH et son associé Edouard BUFFARD, dont le siège social se trouve à Clichy.

En premier lieu nommée **PPA** pour **P**orte-plume, **P**orte-mines et **A**ccessoires et fondée dans le but de produire des pièces détachées pour stylos plume et porte-mines, dans une usine à Clichy, elle se renommera définitivement BIC en 1953 pour une meilleure identification et prononciation dans toutes les langues du monde.



Marcel BICH  
(1914-1994)



Edouard BUFFARD  
(1908-1996)



Premier logo de BIC  
(1953)

En 1950 BIC se fait une place parmi les plus grands acteurs du marché mondial du stylo grâce à la commercialisation d'un produit : le **BIC Cristal**. Vendu à plus de 100 milliards d'exemplaires, il reste encore aujourd'hui le stylo le plus vendu dans le monde. La société élargit son activité en lançant dès 1973 la fabrication et la commercialisation de briquets jetables, ensuite celle des rasoirs en 1975, suivi d'une nouvelle branche de société, BIC Sport, spécialisé dans la fabrication de planches à voile et de planches de surf. Pour finir, sa dernière innovation est la commercialisation de tablette numérique BIC en partenariat avec Intel, disponible en France depuis le premier trimestre 2013, elle est destinée principalement à un usage pédagogique dans les écoles primaires. Aujourd'hui la société s'articule autour de 3 domaines principaux : les articles de papeterie, les briquets et les rasoirs.



Vente BIC par jour dans le monde

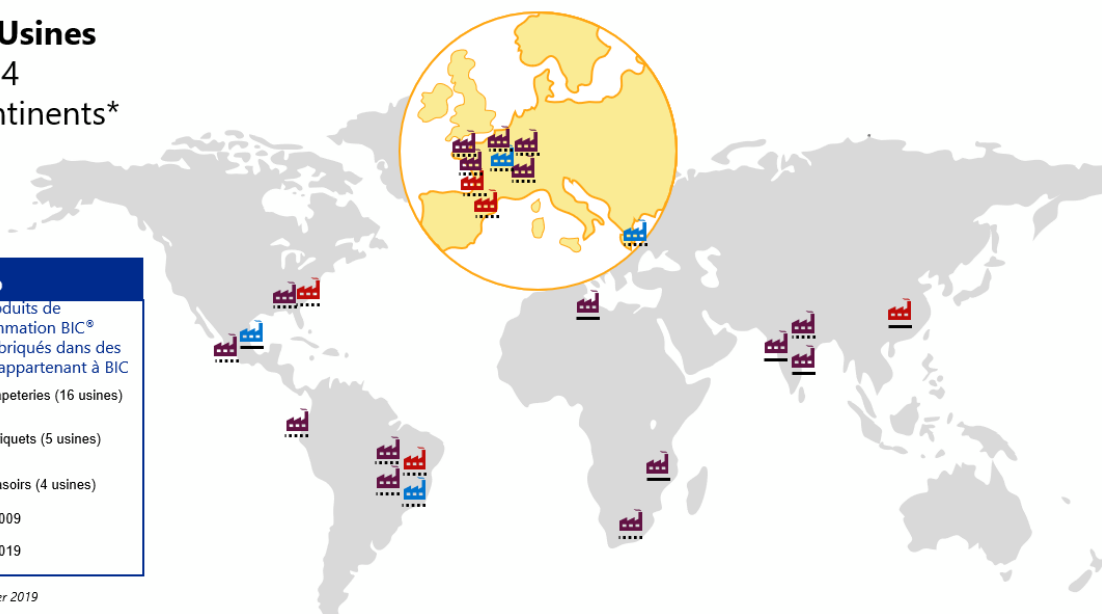
De plus, la société s'est toujours projetée sur le marché mondial en s'implantant dans différents pays au fil des années : en Italie en 1954, au Brésil en 1956, au Royaume-Uni, Australie, Nouvelle-Zélande et Scandinavie en 1957, aux États-Unis, au Moyen-Orient en 1958 et en Asie en 1997.

Aujourd'hui BIC est présent sur les cinq continents, dans plus de 162 pays employant ainsi au total plus de **9.600 personnes**, dont environ 3.900 en Europe. Elle possède également 3,2 millions de points de vente qui distribuent les produits BIC et 9 700 sociétés collaborent avec BIC à travers le monde.

**26 Usines**  
sur 4  
Continents\*



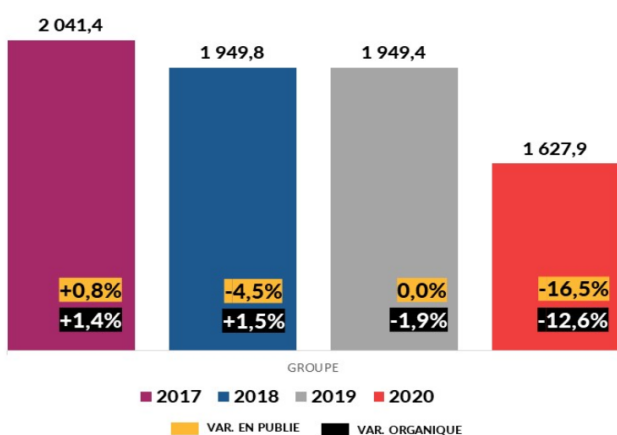
\*En Février 2019



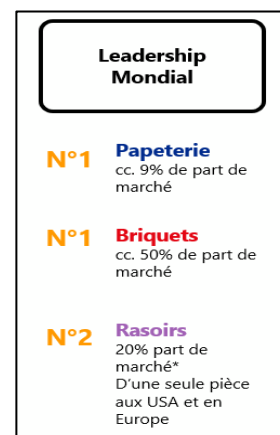
Empreinte de fabrication internationale

Les productions du groupe BIC sont généralement destinées au grand public et représentent 85 % de son chiffre d'affaires. Ses produits sont fabriqués dans 26 usines réparties sur 4 continents (dont 7 en France).

Réalisant en 2019 un chiffre d'affaires de 1 949,4 millions d'euros, BIC comme beaucoup d'autres entreprises n'a pas été épargnée par la crise sanitaire et a vu son chiffre d'affaires de 2020 chuter à 1 627,9 millions. Cela n'empêcha pas la société de rester leader sur le marché en étant n°1 en Papeterie en Europe, en Inde et Amérique latine ainsi qu'en Océanie par exemple.



Chiffres d'affaires Groupe (en millions d'euros)



Pour finir, le groupe BIC possède de nombreuses filiales comme BIC Rasoirs spécialisée dans la fabrication de rasoirs de la firme, Bic Kids qui est la marque de produits de coloriage du Groupe BIC destinée aux enfants ou encore BIC Graphic qui est le principal fournisseur de produits promotionnels de la marque.

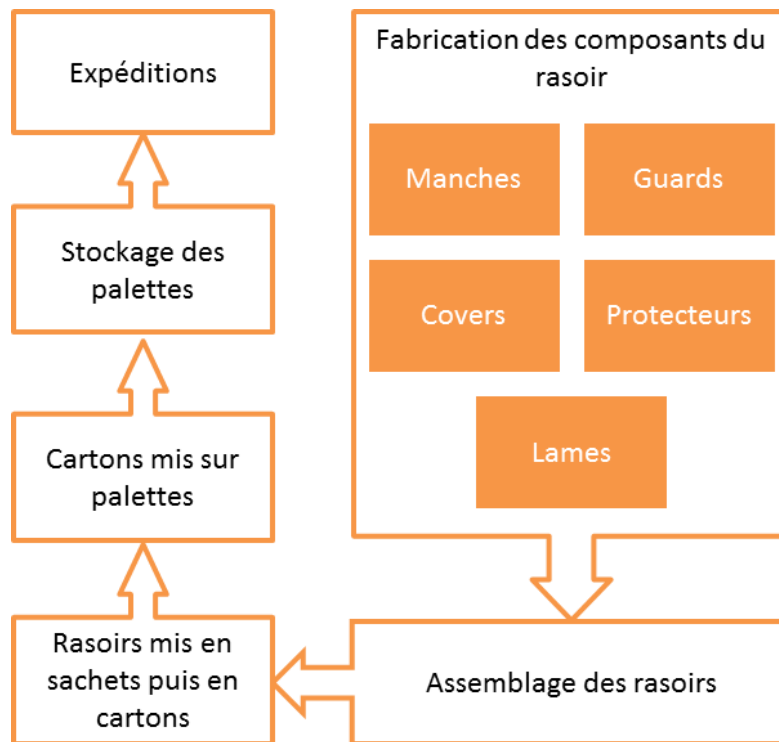
## 2. BIC rasoirs, l'usine de production

Mon stage s'est effectué dans l'entreprise **BIC Rasoirs**, une SASU (Société par Actions Simplifiée Unipersonnelle) filiale du groupe BIC. Cette filiale est située au 6 Rue du Port Salut, 60126 Longueil-Sainte-Marie.

D'abord, utilisée en 1973 pour la conception de pièces de briquet et de composant d'articles d'écriture, elle est aujourd'hui uniquement consacrée à la production de rasoirs depuis l'année 1975. En outre, l'usine couvre une superficie de 20 000 m<sup>2</sup> et comprend 4 tranches d'activités. Employant environ 200 employés, elle fonctionne 5 jours sur 7 avec 3 équipes de 8 heures.

L'usine est divisée en 4 secteurs d'activité :

1. Une activité **moulage** : première étape de la création d'un rasoir, elle consiste à transformer du polystyrène granulé en composant des rasoirs au moyen de presse à injecter. Ce service crée donc toutes les pièces moulées nécessaires aux rasoirs comme le manche ;
2. Une activité **lames** : deuxième étape de la création d'un rasoir, elle permet de créer les lames indispensables pour un rasoir à partir d'une bobine de feuillard en acier. Celle-ci, avant d'atteindre le statut de lame, est perforée avec des outils de découpe montés sur des presses. On réalise ensuite un traitement thermique à la bobine, appelé trempe, pour rigidifier les lames et en améliorer les propriétés mécaniques. Le feuillard circule ensuite entre des meules afin d'affûter l'acier et réaliser le fil de la lame. Quand les lames atteignent le poste de polissage, un couteau les sépare contre une enclume. Elles seront alors contrôlées manuellement et en ligne ;
3. Une activité **assemblage** : troisième et dernière étape de la création d'un rasoir, elle consiste à assembler les différentes pièces créées précédemment pour assembler le rasoir final. Les différents composants sont chargés dans des systèmes de distribution qui cadencent leur arrivée dans la machine d'assemblage. Les composants plastiques sont disposés dans des trémies de grandes capacités (alimentées au moyen du chariot élévateur) alors que les lames sont distribuées. La machine permet alors l'assemblage des composants entre eux et de façon suivante : la tête du rasoir est assemblée indépendamment puis soudée sur le manche.  
Une fois assemblés, les rasoirs sont emballés dans un sachet (appelé « pouch ») puis disposés dans un carton. Le carton est pesé afin de s'assurer que le bon nombre de rasoirs sont présents ;
4. Une partie **expéditions** et **entreposage** des produits finis : enfin les rasoirs sont ensuite stockés dans l'entrepôt avant d'être expédiés vers différentes destinations.



Cycle de production des rasoirs

Cette usine est le cœur de la production du rasoir BIC et de sa technologie en France. Les rasoirs sont exportés dans les pays de l'Union Européenne, Afrique, Moyen-Orient, Amérique du Nord et Centrale et certains pays d'Amérique du Sud et Latine. Le principal client de BIC RASOIRS est BIC S.A.

BIC Rasoirs a réalisé un chiffre d'affaires de 32 943 300,00 € sur l'année 2018.

Bic Rasoirs est la seule usine en France à fabriquer les rasoirs de la marque. Il existe 2 autres usines à l'étranger les produisant : une à Athènes en Grèce, et une à Manaus au Brésil. Elles communiquent entre elles et s'échangent leur production pour vérifier l'état et la conformité de la production (le visuel, le dimensionnel, le packaging) puisque les 3 usines ne produisent pas les mêmes types de rasoirs.

Pour finir, l'usine produit principalement 2 types de rasoirs : le **BIC 1** (630 M/an) et le **BIC Comfort Twin** (80 M/an). Ce sont des rasoirs jetables produits grâce aux différents secteurs d'activités présentées précédemment. 3 millions de rasoirs et 6 millions de lames sont produits chaque jour et ce sont 11 millions de rasoirs qui sont vendus chaque jour dans le monde, soit près de 7700 chaque minute.

Comme toute entreprise, BIC Rasoirs possède de nombreux concurrents. Ses principaux concurrents sur le marché sont la marque américaine de rasoirs Gillette et la marque anglaise Wilkinson. En ce qui concerne la recherche et le développement, BIC Rasoirs ne conçoit pas de nouveaux produits. Cette activité est centralisée à BIC Violex à Athènes.

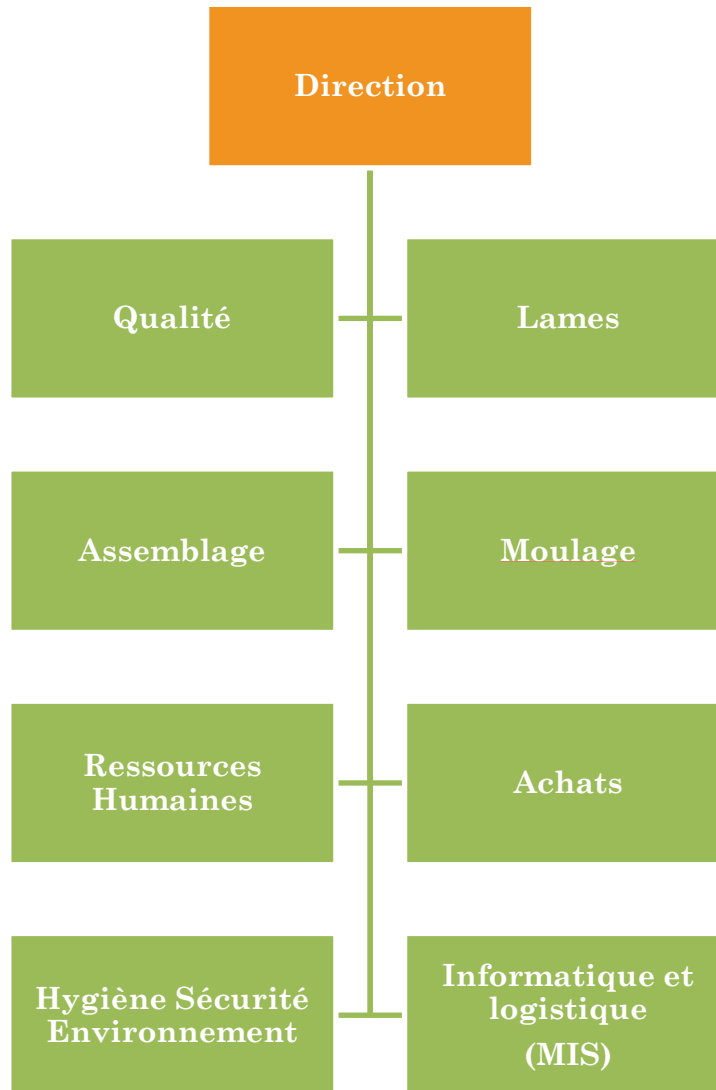
Par ailleurs, l'usine dispose de plusieurs certifications ISO de normalisation internationale et peut garantir à ses produits et services un label de référence reconnu. Par conséquent, grâce à la norme ISO, les produits de l'usine peuvent facilement se positionner sur le marché intérieur et international. Ces certifications sont les suivantes :

- ISO 9001/2000 : elle spécifie les exigences du système de management de la qualité où un organisme doit démontrer sa capacité à fournir systématiquement un produit qui répond aux exigences des clients et des réglementations ;
- ISO 14 001 : elle concerne le management environnemental et repose sur le principe d'amélioration continue de la performance environnementale par la maîtrise des impacts liés à l'activité de l'entreprise ;
- ISO 45001 : c'est une norme définissant des exigences précises concernant la sécurité et la santé au travail.

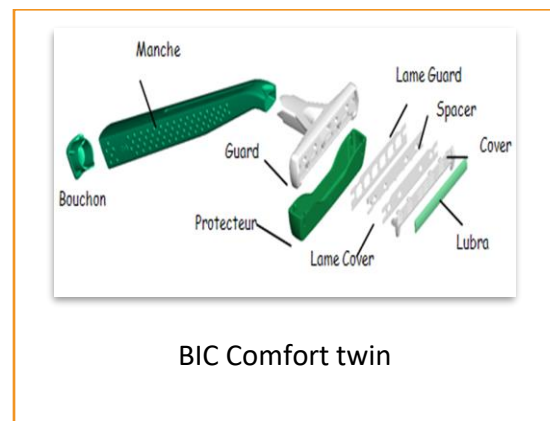
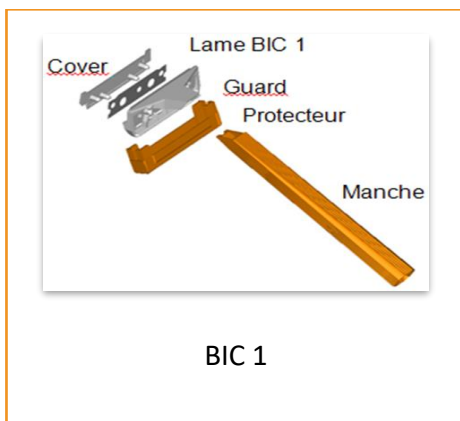
### 3. Organisation de BIC rasoirs

L'organigramme suivant montre la composition de l'entreprise avec ses différents services, dont le service informatique dans lequel je me trouvais durant le stage :





Organigramme BIC Rasoirs

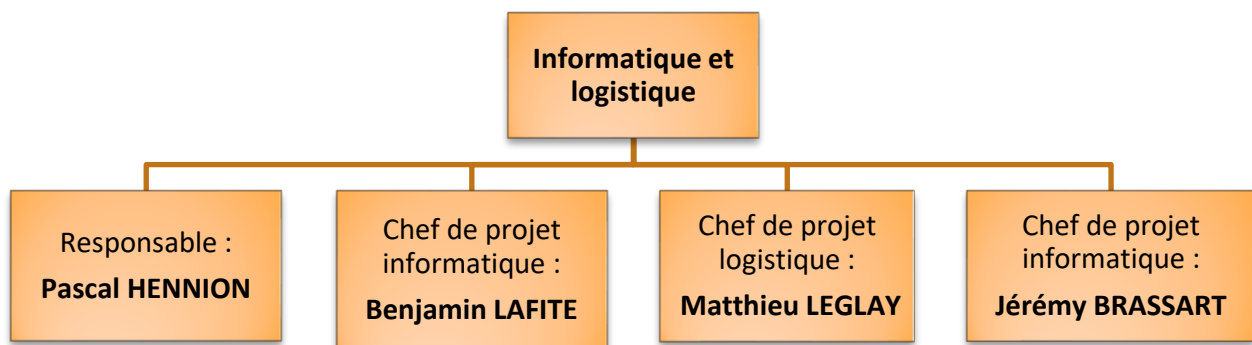


Vue éclatée des rasoirs produits

#### 4. Le MIS, le service informatique

Le service informatique ou MIS pour **Management Information System** assure la gestion et le fonctionnement du réseau sur le site ainsi que l'administration des bases de données. Il a la charge du développement de nouveaux outils utiles pour l'usine. Ces outils permettent des enregistrements, des consultations sur les bases de données. Il doit aussi assurer la sécurisation et l'intégrité des données pour l'ensemble des logiciels utilisés.

Voici un organigramme présentant les différentes personnes travaillant dans le service ainsi que leurs rôles :



Organigramme du MIS de BIC Rasoirs

Le MIS est là aussi pour former les employés et leur permettre un accès aux données dans les meilleures conditions possibles. Il est en communication avec l'ensemble des services de l'usine. De plus, il contribue à la réduction des impressions papier, action qui va dans le sens de la norme ISO 14 001 pour l'environnement. Le service n'a recours à aucun prestataire extérieur, la majeure partie se fait en interne.

Pour finir, les missions du MIS s'orientent autour de 3 domaines majeurs : le réseau et système, le développement et les projets.

##### Réseau et système :

- Installation et gestion du parc informatique (environ 200 postes) ;
- Configuration et installation d'équipements réseau (switch et bornes). Voir schéma réseau en annexe ;
- Administration des serveurs, essentiellement Windows server 2012 à 2019 (environ 20 serveurs) :
  - Serveurs Web, Applicatifs, Impression, Active Directory, Fichiers, GMAO, SQL, ... ;
  - Serveurs virtualisés sous VMware ;
- Administration des bases de données :
  - Création de base de données ;
  - Sauvegarde ;
  - Mise à jour ;
- Administration de la téléphonie.

##### Développement :

- Création d'appliquatif web pour l'ensemble des services (outils de développement présentés dans ma partie environnement de travail) :
  - AP2M, outils de supervision des machines de production pour l'assemblage ;
  - SYCODOM, système de collecte de données moulage ;
  - SS2L, systèmes des lots lames.
- Formation aux utilisateurs.

Projets :

- Etablissement de note de cadrage ;
- Démarchage de prestataire ;
- Chef de projet MOE/MOA ;
- Conduite du changement.

## II. MES MISSIONS

### 1. Liste des missions

- Réalisation d'un module web améliorant la consultation des résultats des coupes feutre de lames finies
- Configuration d'un pc sur le domaine de l'entreprise et ses données depuis l'ancien pc

## III. DESCRIPTION DE LA MISSION «REALISATION D'UN MODULE WEB AMELIORANT LA CONSULTATION DES RESULTATS DES COUPES FEUTRE DE LAMES FINIES »

### 1. Genèse du projet

La production de lame est le pôle d'activité le plus important et prédominant au sein de l'usine. On ne dénombre pas moins de 12 étapes pour passer du matériau de base (bobine de fer) à sa forme finale de lame. La plupart des étapes portent sur la qualité de la lame, la contrôlant ainsi tout au long de sa transformation par des AAQL (**A**gent d'**A**ssurance **Q**ualité **L**ame).

Afin d'assurer une bonne durabilité de la lame, un contrôle qualité est effectué vérifiant le tranchant de la lame par l'intermédiaire d'une machine. La machine utilise la lame pour couper du feutre une dizaine de fois. Les résultats de mesures de coupe de cette machine ne doivent pas dépasser une certaine tolérance sur les premiers et derniers coups. De plus, le feutre est utilisé ici car ce matériau est similaire aux poils sur la peau. Ainsi lors du test, la valeur mesurée correspond à une résistance délivrée pour couper le feutre. Si la résistance de la lame à la coupe du feutre est trop grande dépassant ainsi la tolérance définie, les AAQL signaleront immédiatement la lame et le problème à la zone d'affûtage.

Enfin, les AAQL collectent les résultats de ces coupes de feutre sur une feuille de papier pour les saisir ensuite dans un fichier Excel. C'est à ce moment-là du processus que je suis intervenu.

Les besoins du projet sont les suivants :

- Afficher les données sous forme de tableau comprenant la date du contrôle, le type de lame, la machine, le Top/Bottom (positionnement de la lame), le 1<sup>er</sup> coup, le 10<sup>ème</sup> coup, les commentaires, les champs de tolérance du 1<sup>er</sup> coup et du 10<sup>ème</sup> coup ;
- Possibilité d'ajouter et de modifier des commentaires ;
- Ajouter le numéro de bobine qui n'existait pas dans le fichier Excel ;
- Visualiser les données sous forme de graphique en courbe contenant les valeurs du 1<sup>er</sup> coup, du 10<sup>ème</sup> coup, de la Tolérance au 1<sup>er</sup> coup ainsi qu'au 10<sup>ème</sup> coup ;
- Filtrer les données du tableau et du graphique par date, type de lame, numéro de bobine, machine et enfin par le Top ou Bottom.

Le projet achevé permettra aux AAQL d'économiser environ 15 minutes de temps de travail par jour, soit 55 heures de temps de travail par an. De sorte que les AAQL auront plus de temps pour mesurer et augmenter les inspections.

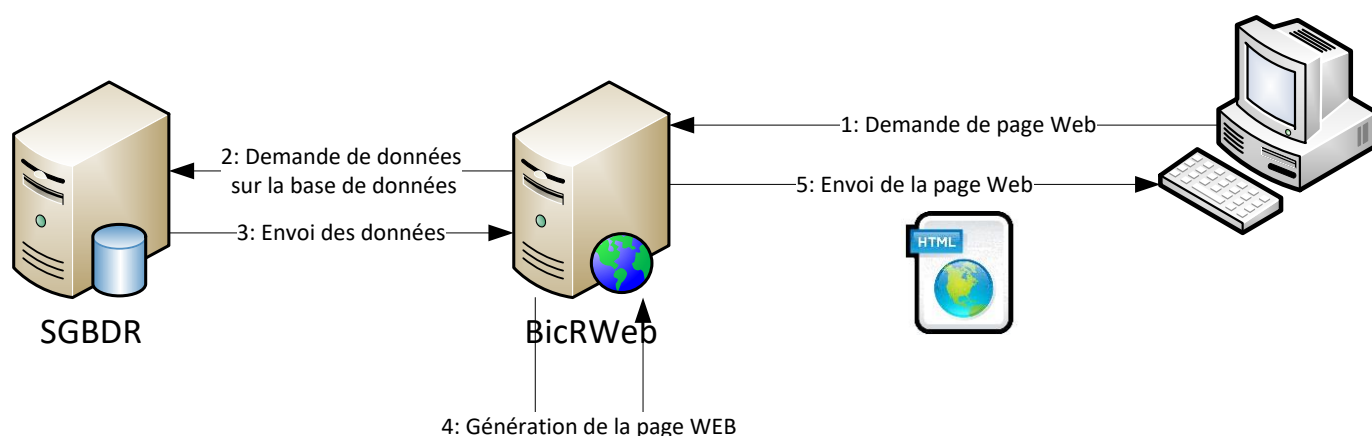
## 2. Environnement de travail

Tout d'abord, pendant le stage, j'ai reçu un ordinateur portable HP pour pouvoir mener à bien le projet et pouvoir utiliser la base de données des coupes feutre. Ensuite, j'ai reçu un cahier des charges préparé en amont expliquant le contexte ainsi que les besoins du projet. Pour mener à bien le projet j'ai utilisé **BicRWeb** et **SQL Server 2012**.

BicRWeb est un framework de développement web utilisé par le MIS. Il a été développé en interne avec le langage PHP et il utilise les principes de la POO (Programmation Orienté Objet). BicRWeb s'appuie sur des fichiers XML dans lesquels on y trouve nos codes. L'avantage de cet outil repose surtout sur la maintenance, l'évolution des scripts et le déploiement des applicatifs. En effet, la maintenance et l'évolution des scripts sont facilitées car toute l'équipe utilise le même outil. Pour le déploiement des applicatifs sur les postes clients, il suffit juste d'un navigateur web assurant une interopérabilité entre les systèmes.

Cependant, utiliser BicRWeb n'est pas aussi performant que des framework développés avec des langages compilés comme le Java ou le C#. Et il n'est pas aussi souple que des framework connus comme Symfony car chaque nouvelle fonctionnalité doit être implémentée par l'équipe. Une plateforme Web comme Xampp peut être utilisée pour ce framework « maison ». La configuration de Xampp est basique et ce logiciel est gratuit.

SQL Server 2012 est un SGBDR (Système de Gestion de Base de Données Relationnelles) et apporte une meilleure gestion de la sécurité, de la protection des données, de la traçabilité et de la performance que les autres SGBDR sur le marché. Ce SGBDR permet de gérer les informations collectées à travers les applicatifs web au sein de l'usine et de les stockées dans une base de données. Cette base de données n'a pas pour but d'être consultée depuis l'extérieur, et chaque personne travaillant au MIS en possède une copie afin de réaliser des environnements de test avant la mise en production.



Fonctionnement générale de BicRWeb

Le choix d'un autre serveur SGBD (tel qu'Oracle) aurait été surdimensionné pour les besoins interne car au sein du site, il y a très peu de requêtes par jour (environ 7000). De plus, le choix d'utiliser le SGBD SQL Server est une décision prise par le siège.

Enfin pour programmer sur BicRWeb, **Notepad++** est utilisé. C'est un éditeur de texte libre générique gratuit qui intègre la coloration syntaxique de code source pour les langages et fichiers comme XML, SQL, PHP, etc donc parfait pour l'exploitation de BicRWeb et de la base de données gérée par SQL Server 2012.

### 3. Planning

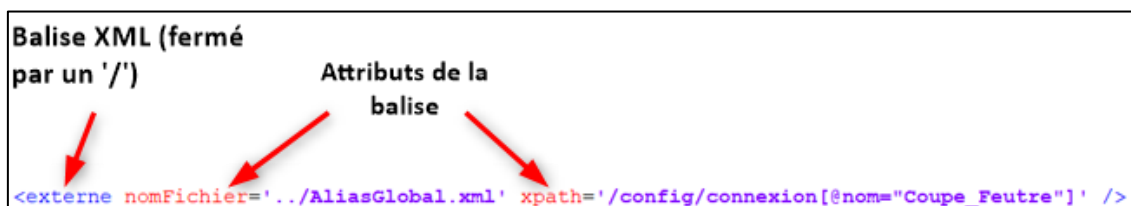
Pour mener à bien mon projet, un planning a été défini :

Taches :	Semaines :			
	1	2	3	4
Installation & Formations				
Développement consultation sur BRREQ				
Formation développement sur BRISE				
Présentation, formation & mise en prod				
Rédaction du rapport de stage				

### 4. Développement du formulaire de consultations des résultats de coupe feutre

Pour commencer, j'ai reçu une base de données en local nécessaire à la mise en place du projet. Baptisée CoupeFeutre, cette base de données contient 4 tables utilisées par le service qualité. Leur structure est expliquée en détail en annexe 2.

Tout d'abord, avant de commencer à réfléchir au code, il est primordial de comprendre les deux bibliothèques PHP de BicRWeb que j'ai utilisées tout au long du projet. La première bibliothèque se nomme **BRREQ** pour **Bic Rasoirs Requêteur** et est dédiée à la définition des sources de données, des traitements et des mises en forme à effectuer. En résumé, c'est par le biais de cette bibliothèque que j'ai effectué mes consultations. Enfin, la deuxième bibliothèque se nomme **BRISE** pour **Bic Rasoirs Interrogation Saisie et Enregistrement** et est dédié à la définition de formulaire. Donc cette bibliothèque m'a permis de faire saisir une donnée et de l'insérer ou de la mettre à jour dans une table de la base de données. Pour me familiariser correctement à ses deux bibliothèques, je me suis formé à leur utilisation par le biais de documentation ainsi que d'exemples fournis pour mieux comprendre leur fonctionnement et leurs balises XML respectives. Cette documentation précise toutes les balises XML utilisables pour chaque bibliothèque et fournit les attributs obligatoires ou facultatifs de ses balises.



La description des tâches que j'ai réalisées pour exécuter ce projet sera divisée en trois parties. La première partie se concentrera sur mon utilisation de BRREQ, la deuxième partie introduira l'utilisation de BRISE, et la dernière partie reliera ces deux bibliothèques.

### a. Utilisation de BRREQ

BRREQ permet d'importer les données utiles à ma consultation, de les afficher sous forme tabulaire et graphique, et enfin de pouvoir filtrer les données.

En premier lieu pour pouvoir utiliser BRREQ j'ai dû créer 2 fichiers : l'un de type PHP pour faire appel à la bibliothèque BRREQ et l'autre de type XML qui contient la partie code. Le fichier PHP sert à appeler la bibliothèque PHP et le fichier XML permet l'utilisation des balises XML faisant référence à la bibliothèque. Mon premier fichier s'appelle **Coupe\_feutre.php**. Ce script en PHP contient seulement l'inclusion du BRREQ de la librairie BicRWeb (étape 1) et son appel avec le fichier XML (étape 2).

```
<?php
    /// Librairie BicRWeb /// 1
    require_once '..\..\lib-php\bfp.inc.php';
    require_once REP_BRREQ.'brreq.php';

    /// Lancer la consultation 2
    brreq('Coupe_Feutre.xml');
?>
```

Le fichier **Coupe\_Feutre.xml** configure toute la consultation désirée dans BRREQ. La première balise du fichier XML permet de configurer BRREQ :

```
<!--
  Elément "brreq" : Définition des sources de données, des traitements et des mises en forme à effectuer.
  Attribut (facultatif) "titre" : Donne un titre à la requête. Il sera affiché dans l'entête de la page web.
  Attribut (facultatif) "debug" : A 'oui', permet d'afficher les étapes intermédiaires du traitement des données
-->
<brreq titre='Résultats : Coupes feutre lame finie ' debug='non' >
</brreq>
```

L'ajout de l'attribut titre à la balise brreq permet de fournir un titre à la requête et de l'afficher dans l'entête de la page web. Et l'attribut debug permet d'afficher les étapes intermédiaires du traitement des données, pour l'activer il faut simplement remplacer le 'non' par 'oui'. Le code présenté par la suite sera situé entre la balise <brreq> et la balise </brreq>.

Une fois BRREQ configuré, je dois accéder à la base de données CoupeFeutre pour y effectuer mes consultations. J'ai également en ma possession un fichier extérieur **AliasGlobal.xml** contenant le script de connexion à la base de données. Par conséquent, j'utilise la balise « alias » et « externe » qui vont me permettre d'accéder à ce fichier externe nommé **AliasGlobal.xml**. Les attributs de la balise « externe » vont indiquer le chemin vers ce fichier et récupérer dans ce fichier le script correspondant à la connexion de la base de données CoupeFeutre :

```
<alias>
<!--
  Elément "externe" : Alias externe
  Attribut (obligatoire) "nomFichier" : Nom du Fichier (xml) contenant l'alias.
  Attribut (obligatoire) "xpath" : Chaîne de recherche de type XPATH permettant de récupérer un élément XML.
-->
<externe nomFichier='../AliasGlobal.xml' xpath='/config/connexion[@nom="Coupe_Feutre"]' />
</alias>
```



Le contenu du fichier AliasGlobal.xml est le suivant :

```
<config>
<!--
  Elément "connexion" : Définition du type et de la connexion à la source.
  Attribut (obligatoire) "type" : Type de source.
    "type='SQL'" : Requête SQL.
    Connexion au serveur avec les attributs (obligatoires) : "utilisateur":nom d'utilisateur
                                                             "motDePasse":mot de passe utilisateur
                                                             (facultatifs) : "serveur":nom du serveur
                                                             "base":nom de la base
-->
  <connexion nom='Coupe_Feutre' type='sql' serveur='VERW0652\SQLEXPRESS' base='CoupeFeutre' utilisateur='sa' motDePasse='Pa$word' />
</config>
```

La balise « config » contient la configuration de la connexion et la balise « connexion » est utilisée pour définir le type et la source de connexion. Plusieurs types de sources sont disponibles mais celle qui nous intéresse ici est la source SQL utilisé pour accéder à la base de données. Ensuite, les autres attributs indiquent toutes les informations utiles à la connexion de la base de données comme le nom du serveur, le nom de la base, le nom de l'utilisateur et de son mot de passe ayant les habilitations requises sur cette base pour effectuer des consultations, des insertions, des modifications...

Enfin, pour aller plus loin dans l'étape liée à la base de données. Voici la bibliothèque de fonctions PHP utilisée pour se connecter à la base de données. Ce script permet via ODBC la manipulation des bases de données mises à disposition par le système de gestion de base de données comme SQL Server dans notre cas. Ici, le script reçoit les informations de connexion à la base de données venant de notre fichier de connexion AliasGlobal en XML dans une variable « \$fichierxml ». Et la variable est ensuite utilisée dans une fonction qui retournera une variable de connexion ODBC vers la base de données si bien évidemment toutes les informations de connexion sont correctes et renseignées.

```

/*//////////////////// bfp_odbc_connect //////////////////////
Ouverture de la base de donnée
////////////////////////////////////
Reçoit : $fichierxml = Fichier XML contenant les informations de connexion
        $tagDataBase = Tag de la base de donnée dans le fichier XML (pour récupération des paramètres de connexion)
        $dbserver, $dbbase, $dbuser, $dbpass : Paramètres de connexion (actifs si $tagDataBase =null)
        $driver : Driver ODBC d'accès à la base de donnée. ("SQL Server" ou "SQL Server Native Client 10.0"
        $dns : Chaine DNS facultative
Retourne : $variable de Connection ODBC vers la base de donnée
////////////////////////////////////*/
// 20/11/2012 : (PH) : bfp_odbc_connect : Ajout traitement du passage des paramètres de connexion sans fichier XML
// 2013-03-18 : (PH) : bfp_odbc_connect : Ajout du paramètre $driver.
// 2014-04-09 : (PH) : bfp_odbc_connect : Ajout du paramètre $dns.
function bfp_odbc_connect($fichierxml,$tagDataBase,$dbserver=null,$dbbase=null,$dbuser=null,$dbpass=null,$driver=null,$dns=null) {
    if($fichierxml!=null){
        $xml = simplexml_load_file($fichierxml);
        $dbserver = $xml->$tagDataBase->server;
        $dbbase = $xml->$tagDataBase->base;
        $dbuser = $xml->$tagDataBase->user;
        $dbpass = $xml->$tagDataBase->password;
    }else{
        // La chaine DNS est prioritaire
        if($dns!=null){
            // echo $dns.HR;
            $conn=odbc_connect($dns,$dbuser,$dbpass,SQL_CUR_USE_ODBC);
        }else if($dbserver!=null && $dbbase!=null && $dbuser!=null && $dbpass!=null ) {
            if($driver==null) {
                /// Valeur par défaut
                ///$driver="SQL Native Client";      /// Il faut installer "sqlncli_x64.msi"
                $driver="SQL Server";
                /// $driver="SQL Server Native Client 10.0";
            }
            // $conn=odbc_connect("Driver={SQL Server};Server=".$dbserver.";Database=".$dbbase, $dbuser,$dbpass,SQL_CUR_USE_ODBC);
            $conn=odbc_connect("Driver={$driver};Server=".$dbserver.";Database=".$dbbase, $dbuser,$dbpass,SQL_CUR_USE_ODBC);
        }
        return $conn;
    }
}

```



## b. Présentation du formulaire


**IMPORTER LES DONNÉES**


les critères de choix à renseigner par l'utilisateur

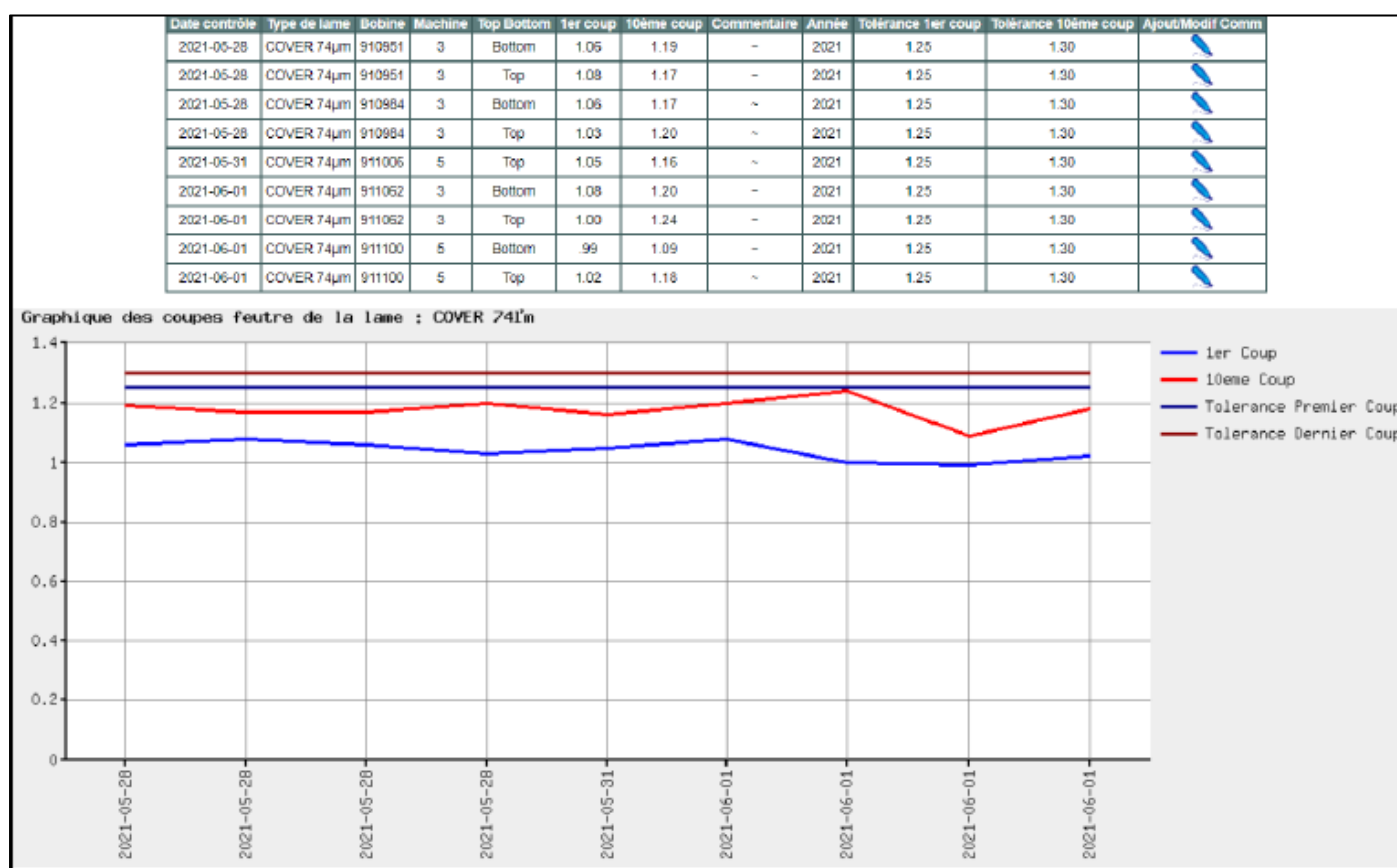
Selection du type de lame

Date de début

Date de fin

 Importer



Les captures d'écran ci-dessus montrent l'utilisation et le résultat obtenu par le formulaire de consultation. Tout d'abord pour visualiser les résultats des coupes feutres de lame finies, l'utilisateur doit renseigner le type de lame ainsi que la période de date qu'il souhaite. Les données importées de la base de données sont ensuite présentées sous forme de tableau et de graphique à l'utilisateur, données qu'il peut filtrer. Dès à présent, je vais rentrer plus en détail sur le fonctionnement du formulaire et l'affichage des données.

c. Le code complet permettant d'obtenir le formulaire

```
<?xml version='1.0' encoding='utf-8'?>
<brreq titre='Résultats : Coupes feutre lame finie ' debug='non' >
  <alias>
    <externe nomFichier='../AliasGlobal.xml' xpath='/config/connexion[@nom="Coupe_Feutre"]' />
  </alias>
  <parametres>
    <source nom='Type de lame'>
      <connexion type='alias' nom='Coupe_Feutre'>
        SELECT TypeLameNom AS TypeLame
        FROM (SELECT *,
              case WHEN TypeLameCle = 15 then '4'
                   WHEN TypeLameCle=16 then '8'
                   WHEN TypeLameCle=17 then '9' END AS premier_digit
              FROM TypesLames
             ) S1
        WHERE premier_digit IS NOT NULL
      </connexion>
    </source>

    <source nom='Date'>
      <connexion type='alias' nom='Coupe_Feutre'>
        SELECT
          CONVERT (DATE, GETDATE()-1) AS DateDebut,
          CONVERT (DATE, GETDATE()) AS DateFin;
      </connexion>
    </source>

    <parametre type='liste' nom='lame' label='Selection du type de lame' >
      <champ nom='' val='' />
      <champSrc nom='TypeLame' valSrc='TypeLame' />
    </parametre>

    <parametre type='date' nom='dateDeb' label='Date de début' valSrc='DateDebut' />
    <parametre type='date' nom='dateFin' label='Date de fin' valSrc='DateFin' />
  </parametres>
```

**1** La liste déroulante permettant de sélectionner un type de lame. Code expliqué dans la partie d)

**2** La période de dates a choisir. Code expliqué dans la partie e)

```
<donnees>
<source nom='Résultats des coupes feutres'>
  <connexion type='alias' nom='Coupe_Feutre' >
    SELECT
      LEFT(LotDateCtrl, 4) AS Annee
      , CONVERT(DATE, LotDateCtrl) AS DateCtrl
      , TypeLameNom AS TypeLame
      , LotNoLot AS Bobine
      , CASE LotTB
        WHEN 'T' THEN 'Top'
        ELSE 'Bottom'
      END AS TB
      , LotResultatPrem*0.01 AS [1er Coup]
      , LotResultatDer*0.01 AS [10eme Coup]
      , TypeLameResultatMaxPrem*0.01 AS [Tolerance Premier Coup]
      , TypeLameResultatMaxDer*0.01 AS [Tolerance Dernier Coup]
      , CL_Commentaire as Commentaire
      , LotCle
      , LotMachine
    FROM Lots
    INNER JOIN (
      SELECT *,
      case WHEN TypeLameCle = 15 then '4'
      WHEN TypeLameCle=16 then '8'
      WHEN TypeLameCle=17 then '9' END AS premier_digit
      FROM TypesLames
    ) Sous_requete ON premier_digit=LEFT(LotNoLot,1)
    LEFT OUTER JOIN Commentaire_Lot_CL ON lotcle=cl lotcle
    WHERE (TypeLameNom #EG# '$lame$' OR '..Tous..' #EG# '$lame$')
    AND CONVERT(Date, LotDateCtrl) #SUPEG# '$dateDeb$'
    AND CONVERT(Date, LotDateCtrl) #INFEG# '$dateFin$'
  </connexion>
</source>
```

3 Affichage des données.  
Code expliqué en partie f)

4 Ajout d'une colonne pour ajouter à un  
commentaire pour une donnée.  
Code expliqué plus tard

```
<traitement type='calculer'>
  <champ nom='lien' val='modifier.png?Cliquez ' lien='..\Insertions\Coupe_Feutre_Commentaire.php' dest='local' >
  <lienParam nom='LotCle' valCalc='[LotCle]' />
  </champ>
</traitement>
</donnees>
```

```
<affichage>

<format nom='Tri par Date, Type Lame et Bobine (sans graphique)' code='1' />
<format nom='Tri par Type Lame, Date et Bobine (inverse et sans graphique)' code='2' />
<format nom='Tri par Date (Avec graphique)' code='3' />
<tri format='1'>
  <champ nom='DateCtrl' ordre='asc' />
  <champ nom='TypeLame' ordre='asc' />
  <champ nom='Bobine' ordre='asc' />
  <champ nom='TB' ordre='desc' />
</tri>
<tri format='2'>
  <champ nom='TypeLame' ordre='desc' />
  <champ nom='DateCtrl' ordre='desc' />
  <champ nom='Bobine' ordre='desc' />
</tri>
<tri format='3'>
  <champ nom='DateCtrl' ordre='asc' />
</tri>

<tableau>
  <champ nom='DateCtrl' label='Date contrôle' />
  <champ nom='TypeLame' label='Type de lame' />
  <champ nom='Bobine' label='Bobine' />
  <champ nom='LotMachine' label='Machine' />
  <champ nom='TB' label='Top Bottom' />
  <champ nom='1er Coup' label='1er coup' >
    <condition test='[1er Coup]#SUP#[Tolerance Premier Coup]' aspect='erreur blanc gras' />
  </champ>
  <champ nom='10eme Coup' label='10ème coup'>
    <condition test='[10eme Coup]#SUP#[Tolerance Dernier Coup]' aspect='erreur blanc gras' />
  </champ>
  <champ nom='Commentaire' label='Commentaire' />
  <champ nom='Annee' label='Année' />
  <champ nom='Tolerance Premier Coup' label='Tolérance 1er coup' />
  <champ nom='Tolerance Dernier Coup' label='Tolérance 10ème coup' />
  <champ nom='lien' type='image' lien='oui' label='Ajout/Modif Comm' />
</tableau>
```

```
<filtre nom='DateCtrl' label='Date' />
<filtre nom='TypeLame' label='Type de lame' />
<filtre nom='TB' label='Top Bottom' />
<filtre nom='Annee' label='Année' />

<graphique format='3' titre='Graphique des coupes feutre de la lame : $lame$' type='courbe' echYPrin='0.20' echYMax='1.40' imageL='1250' >
  <etiquette nom='DateCtrl' />
  <champ nom='1er Coup' traitCoul='blue' traitEpaisseur='3' aspect='ligne' />
  <champ nom='10eme Coup' traitCoul='red' traitEpaisseur='3' aspect='ligne' />
  <champ nom='Tolerance Premier Coup' traitCoul='darkblue' traitEpaisseur='3' aspect='ligne' />
  <champ nom='Tolerance Dernier Coup' traitCoul='darkred' traitEpaisseur='3' aspect='ligne' />
</graphique>
<filtre nom='LotMachine' label='Machine' />
</affichage>
</breq>
```

d. Présentation du code permettant d'afficher la liste déroulante contenant les types de lames

```
<source nom='Type de lame'>
  <connexion type='alias' nom='Coupe_Feutre'>
    SELECT TypeLameNom AS TypeLame
    FROM (SELECT *,
      case WHEN TypeLameCle = 15 then '4'
      WHEN TypeLameCle=16 then '8'
      WHEN TypeLameCle=17 then '9'  END AS premier_digit
    FROM TypesLames
    ) S1
    WHERE premier_digit IS NOT NULL
  </connexion>
</source>
```

La connexion à la base de données étant faite, il est maintenant possible de choisir les paramètres d'importation des données. Pour cela, on utilise la balise « parametres ».

Pour commencer, les données seront afficher selon le type de lame sélectionné par l'utilisateur.

Tout d'abord, le premier paramètre d'import de données dépendra du type de lame sélectionné. Pour cela, je dois faire une consultation dans la base de données pour récupérer le type puis l'afficher sous forme de liste. Le choix du type listé semble plus confortable pour le type de données à afficher et à sélectionner par l'utilisateur. L'utilisation de la balise « source » me permet d'ouvrir une source de données dans mon code qui sera utilisable dans les paramètres d'importation et je le nomme 'Type de lame'. Ensuite, il me faut définir le type et la source de cette connexion avec la balise « connexion » : le type de source indique que la connexion est définie en alias et le nom du script de connexion à la base de données dans cet alias. Ces étapes effectuées, il m'est maintenant possible d'effectuer ma requête dans la table TypesLames pour récupérer mes types de lames. Or là où normalement une simple requête SQL pour extraire les enregistrements de la colonne TypeLameNom de la table TypesLames (Cf annexes) aurait été possible. Ici un problème se posa dans notre base de données, la colonne TypeLameCle et la colonne LotCleTypeLame partagent les mêmes valeurs pour permettre d'effectuer une jointure entre la table Lots et la table TypesLames faisant ainsi correspondre les lots avec les types de lames.

Cependant durant le projet nous avons remarqué qu'en sélectionnant un type de lame, celui-ci ne récupérer pas tous les enregistrements dans la table Lots le correspondant car des valeurs non référencées dans la colonne TypeLameCle exister dans la colonne LotCleTypeLame. Rendant impossible la récupération de toutes les données sur les différents lots. De plus, cette erreur s'explique par le fait que la base de données est un SGBD non relationnel donc ne contenant aucune clé primaire et étrangère. Enfin, il est impossible de toucher à l'intégrité des 4 tables de la base de données étant donné qu'elle n'a pas été développée par le MIS et il est impossible d'avoir accès au code source donc de connaître les requêtes d'INSERT dans les tables. En résumé, il est impossible de toucher à l'intégrité des tables pour ne pas perturber tout le programme existant car nous ne savons pas comment sont écrits les INSERT dans cette base de données.

Pourtant, nous avons réussi à passer outre ce problème par un palliatif assez simple. Le premier chiffre de chaque numéro de bobine correspond obligatoirement à un type de lame, par exemple tous les numéros de bobine commencent par le chiffre 4 sont associés à la lame de type BIC1 76µm et ainsi de suite. Voici comment en utilisant le premier numéro de lot je récupère les types de lames :

TypeLameCle	TypeLameNom
15	BIC1 76µm
16	GUARD 74µm
17	COVER 74µm

Table TypesLames

LotCle	TypeLame	LotNoLot
16		80502
15		410393
15		410393
15		410346
15		410346
17		90511

Table Lots

C'est ce qu'effectue donc notre requête SQL (en jaune) avec l'utilisation du « CASE ... WHEN ... » qui permet d'utiliser des conditions de type « si / sinon » (cf. if / else) similaire à un langage de programmation pour retourner un résultat. Le SELECT récupère les types de lames dans la colonne TypeLameNom et je lui assigne l'alias TypeLame pour me simplifier le code. Enfin, j'ajoute une condition pour ne pas récupérer une valeur de données n'existant pas dans la base de données.

La consultation étant terminée, il nous faut maintenant mettre les données extraites sous forme de liste. Pour ce faire il suffit d'utiliser la balise « parametre » qui permet de définir un paramètre de consultation. Je lui assigne le type 'liste' créant ainsi une liste dont les valeurs sont paramétrées par les balises « champSrc » et « champ ». La valeur et le nom de la balise champ sont vides ce qui permet de créer une ligne vide à la sélection du type de lame montrant l'obligation de sélectionner un type pour afficher les données. En revanche, c'est la balise « champSrc » qui permet de définir des champs issus des données provenant des sources. Je lui attribue un nom et lui indique la donnée à récupérer dans la requête SQL donc l'alias TypeLame qui contient tous les noms des types de lame. Enfin, le label correspond à l'étiquette proposée à l'utilisateur.

Voici le résultat de nos scripts :

## IMPORTER LES DONNÉES

Selection du type de lame

▼

✓ Import

BIC1 76µm

GUARD 74µm

COVER 74µm

valeur de la balise "champ"

valeur de la balise "champSrc"

Et voici ce que contenait les balises utilisées pour ce formulaire :

```
<parametre type='liste' nom='lame' label='Selection du type de lame' >
  <champ nom='' val=''/>
  <champSrc nom='TypeLame' valSrc='TypeLame'/>
</parametre>
```

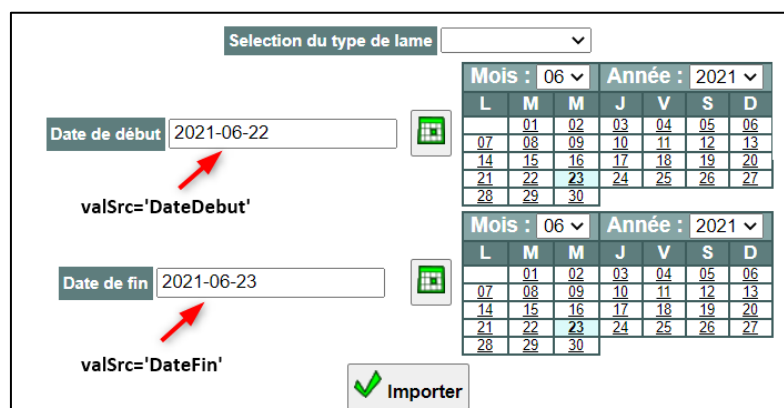
e. Présentation du code permettant de choisir la période

```
<source nom='Date'>
  <connexion type='alias' nom='Coupe_Feutre'>
    SELECT
      CONVERT (DATE, GETDATE()-1) AS DateDebut,
      CONVERT (DATE, GETDATE()) AS DateFin;
  </connexion>
</source>
```

Ce paramètre sera de type date pour permettre une meilleure sélection de la date. Pour commencer même procédé que pour les types de lames, il me faut définir une ouverture de la source de données qui sera utilisable dans les paramètres d'importation que je nomme tout simplement Date. La balise connexion exerce la même fonction que pour le paramètre précédent et maintenant il m'est possible d'effectuer ma requête. La fonction GETDATE () renvoie la date et l'heure du système de base de données en cours donc du jour dans un format 'AAAA-MM-JJ hh:mm:ss.mmm'. Cependant, il nous faut seulement la date au format 'AAAA-MM-JJ' donc il me suffit de convertir au format date cette fonction et voici ce que cela donne en SQL : CONVERT (Date, GETDATE ()). De plus, le format 'AAAA-MM-JJ' est un format ISO 8601 dissipant toute confusion en établissant, pour la représentation de la date, un format internationalement reconnu. Donc l'étape 1 permet de récupérer la date de la veille par l'utilisation du (-1) avec l'alias DateDebut et l'étape 2 permet de récupérer la date du jour assigné à l'alias DateFin.

La définition des valeurs à affecter aux paramètres date de début et date de fin étant réalisée, il est nécessaire de définir le nom et le type des paramètres avec la balise <paramètre>. Ils permettront à l'utilisateur de choisir de récupérer les données sur une période de son choix. Ensuite, je définis les paramètres et leur attribue un titre à afficher à l'utilisateur à l'aide de l'attribut label. Enfin la partie la plus importante, l'utilisation de valSrc qui va prendre la première valeur de la source comme valeur par défaut donc ici les alias DateDebut et DateFin définis dans la balise source.

Voici le résultat du script (partie écrite le 23/06/2021) :


















Et voici ce que contenait les balises utilisées pour ce contrôle de périodicité :

```
<parametre type='date' nom='dateDeb' label='Date de début' valSrc='DateDebut' />
<parametre type='date' nom='dateFin' label='Date de fin' valSrc='DateFin' />
```

#### f. Affichage des données

Après que la description des paramètres, il est temps d'aller extraire toutes les données correspondant aux paramètres pour les afficher sous forme de tableau et de graphique.

Voici le résultat de la requête SQL et de l'affichage au format tabulaire :

nom='DateCtrl'	nom='TypeLame'	nom='Bobine'	nom='LotMachine'	nom='TB'	nom='1er coup'	nom='10ème coup'	nom='Commentaire'	nom='Année'	nom='Tolérance Premier Coup'	nom='Tolérance Dernier Coup'	nom='lien'
Date contrôle	Type de lame	Bobine	Machine	Top Bottom	1er coup	10ème coup	Commentaire	Année	Tolérance 1er coup	Tolérance 10ème coup	Ajout/Modif Comm
2021-03-03	COVER 74µm	910252	5	Top	1.00	1.14	~	2021	1.25	1.30	
2021-03-03	COVER 74µm	910252	5	Bottom	.96	1.16	~	2021	1.25	1.30	
2021-03-03	COVER 74µm	910404	3	Top	.93	1.07	~	2021	1.25	1.30	
2021-03-03	COVER 74µm	910404	3	Bottom	.97	1.09	~	2021	1.25	1.30	
2021-03-04	COVER 74µm	910391	3	Top	1.00	1.18	~	2021	1.25	1.30	
2021-03-04	COVER 74µm	910391	3	Top	1.01	1.16	~	2021	1.25	1.30	
2021-03-05	COVER 74µm	910363	3	Top	.91	1.28	~	2021	1.25	1.30	
2021-03-05	COVER 74µm	910363	3	Top	.97	1.24	~	2021	1.25	1.30	
2021-03-08	COVER 74µm	910318	3	Top	.97	1.12	~	2021	1.25	1.30	
2021-03-08	COVER 74µm	910349	5	Top	.88	1.17	~	2021	1.25	1.30	
2021-03-08	COVER 74µm	910349	5	Top	.98	1.15	~	2021	1.25	1.30	
2021-03-09	COVER 74µm	910335	3	Top	.98	1.20	~	2021	1.25	1.30	
2021-03-09	COVER 74µm	910335	3	Bottom	1.09	1.31	~	2021	1.25	1.30	

condition 10ème coup > tolérance 10ème coup,  
1.31>1.30, aspect=erreur blanc gras

```
<donnees>
<source nom='Résultats des coupes feutres'>
  <connexion type='alias' nom='Coupe_Feutre' >
    SELECT
      LEFT(LotDateCtrl, 4) AS Année
      , CONVERT (DATE, LotDateCtrl) AS DateCtrl
      , TypeLameNom AS TypeLame
      , LotNoLot AS Bobine
      , CASE LotTB
          WHEN 'T' THEN 'Top'
          ELSE 'Bottom'
        END AS TB
      , LotResultatPrem*0.01 AS [1er Coup]
      , LotResultatDer*0.01 AS [10ème Coup]
      , TypeLameResultatMaxPrem*0.01 AS [Tolérance Premier Coup]
      , TypeLameResultatMaxDer*0.01 AS [Tolérance Dernier Coup]
      , CL_Commentaire as Commentaire
      , LotCle
      , LotMachine
    FROM Lots
    INNER JOIN (
      SELECT *,
        case WHEN TypeLameCle = 15 then '4'
        WHEN TypeLameCle=16 then '8'
        WHEN TypeLameCle=17 then '9' END AS premier_digit
      FROM TypesLames
    ) Sous_requete ON premier_digit=LEFT(LotNoLot,1)
    LEFT OUTER JOIN Commentaire Lot_CL ON lotcle=cl_lotcle
    WHERE TypeLameNom #EG# '$lame$'
    AND CONVERT (Date, LotDateCtrl) #SUPEG# '$dateDeb$'
    AND CONVERT (Date, LotDateCtrl) #INFEG# '$dateFin$'
  </connexion>
</source>
```

La requête SQL nécessaire pour extraire les données de la base de données et l'affichage des données sont placés dans une balise « donnees ». La source de données est contenue la balise « source » qui contient un nom « Résultat des coupes feutres » obligatoire. Ici, la balise « connexion » a le même fonctionnement que pour les paramètres me permettant de se connecter à la base de données. La requête SQL se trouve dans cette balise <connexion>.

L'étape 1 permet de récupérer l'année ainsi que la date de contrôle du lot. Dans la base de données, la date de contrôle est au format ISO « AAAAMMJJ » ; pour récupérer l'année je vais utiliser la fonction LEFT () qui permet d'extraire un certain nombre de caractères d'une chaîne en commençant par la gauche. Donc en indiquant de récupérer les 4 premiers caractères de gauche, j'extrait uniquement l'année de la date de contrôle que je nomme Annee avec un alias pour faciliter la compréhension de la requête. Puis j'extrait la date de contrôle mais pour une meilleure compréhension aux yeux de l'utilisateur dans le tableau, je convertis en Date la colonne passant du format « AAAAMMJJ » au format « AAAA-MM-JJ » et je la nomme DateCtrl avec un alias.

L'étape 2 me permet d'extraire les autres données souhaitées. Il y a dans l'ordre le type de lame, le numéro de bobine, le top ou le bottom. Pour cette donnée je réutilise le CASE...WHEN... pour ajouter une meilleure compréhension aux yeux de l'utilisateur car dans la base de données, cette donnée est exprimée seulement par deux valeurs : 'T'et 'B' pour Top et Bottom. J'extrait la donnée avec un CASE pour que si la valeur enregistrée 'T'alors Top soit affiché et sinon il sera affiché Bottom.

J'extrait les résultats de la 1<sup>ère</sup> coupe ainsi que sa tolérance et les résultats de la 10<sup>ème</sup> coupe ainsi que sa tolérance que je multiplie par 0,01 pour me donner des nombres décimaux à 2 chiffres après la virgule car dans la base de données ces enregistrements sont des nombres entiers. Pour finir sur les données extraites, il reste le commentaire, LotCle qui correspond au numéro de lot nous servira plus tard et le numéro de machine. On récupère principalement les informations de la table Lots mais certaines données ne proviennent pas de cette table. Donc pour récupérer les types de lames associé au numéro de bobine (en bleu), on effectue la même requête SQL que pour obtenir les valeurs affichées dans la liste déroulante avec un INNER JOIN. Cette commande INNER JOIN est un type de jointure permettant de lier plusieurs tables entre-elles et retournant les enregistrements lorsqu'il y a au moins une ligne dans chaque colonne qui correspond à ma condition. Et le LEFT(LotNoLot,1) me permet de récupérer le premier chiffre de chaque bobine pour effectuer parfaitement ma jointure.

La deuxième jointure utilise la commande LEFT OUTER JOIN (jointure externe à gauche) pour pouvoir récupérer tous les lots même s'ils n'ont pas de commentaire. Cette jointure externe porte sur les deux tables Lots et Commentaire\_Lots\_CL, et retourne tous les lots de la table de gauche (table Lots) même s'ils n'ont pas de commentaires dans la table Commentaire\_Lots\_CL.

La clause WHERE de la requête SQL fait une restriction sur les enregistrements à sélectionner. Les paramètres sont encadrés par \$...\$ dans la balise connexion et sont appelés par les noms de leurs paramètres. La requête utilise les mots-clés suivants pour les tests d'égalité : #EG# pour 'égal à', #SUPEG# pour 'supérieur ou égal à' et #INFEG# pour 'inférieur ou égal à'. En résumé, les conditions imposent que les données extraites soient égales au type de la lame sélectionné et que la plage de données soient compris entre la date de début et de fin sélectionné par l'utilisateur.



Maintenant que nos données sont extraites, il nous reste plus qu'à les afficher sous forme de tableau et de graphique. Pour ce faire, une balise le permet et c'est la balise « affichage ». Le code se trouvera donc dans cette balise.

La balise « tableau » qui permet d'afficher les données sous forme de tableau. Les balises « champs » définissent les champs à afficher dans le tableau. L'ordre des champs donne l'ordre d'affichage. L'attribut nom donne le nom de la donnée provenant de la requête SQL au champ à afficher. Par exemple, j'ai donné à la colonne LotDateCtrl l'alias DateCtrl dans la requête SQL aussi pour l'afficher dans le tableau j'indique l'alias DateCtrl dans l'attribut nom et j'indique ensuite le label « Date contrôle » qui sera le titre de la colonne. Comme on peut le voir mon tableau va afficher dans l'ordre la date du contrôle, le type de lame, le numéro de bobine, la machine, le Top/Bottom, le 1<sup>er</sup> coup, le 10<sup>ème</sup> coup, le commentaire, l'année, la tolérance premier coup, la tolérance dernier coup et enfin un lien vers BRISE que j'introduirais dans ma dernière partie. Des conditions 1 et 2 permettent de conditionner l'aspect des valeurs affichées. Par exemple, la 1<sup>ère</sup> condition indique si un résultat est supérieur à sa tolérance la valeur affichée aura un aspect visuel différent des autres valeurs. Cette démarcation visuelle va permettre aux AAQL de remarquer plus rapidement sur le tableau s'il y a un problème avec une lame.

```
<affichage>

<format nom='Tri par Date, Type lame et Bobine (sans graphique)' code='1' />
<format nom='Tri par Type Lame, Date et Bobine (inverse et sans graphique)' code='2' />
<format nom='Tri par Date (Avec graphique)' code='3' />
<tri format='1'>
  <champ nom='DateCtrl' ordre='asc' />
  <champ nom='TypeLame' ordre='asc' />
  <champ nom='Bobine' ordre='asc' />
  <champ nom='TB' ordre='desc' />
</tri>
<tri format='2'>
  <champ nom='TypeLame' ordre='desc' />
  <champ nom='DateCtrl' ordre='desc' />
  <champ nom='Bobine' ordre='desc' />
</tri>
<tri format='3'>
  <champ nom='DateCtrl' ordre='asc' />
</tri>

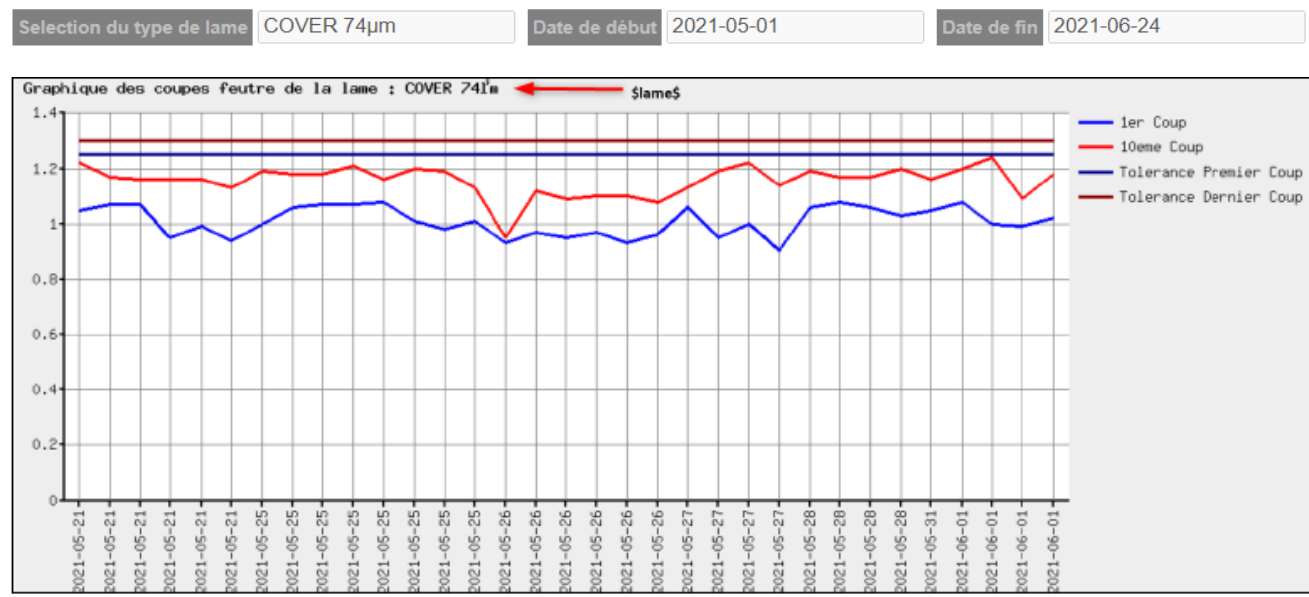
<tableau>
  <champ nom='DateCtrl' label='Date contrôle' />
  <champ nom='TypeLame' label='Type de lame' />
  <champ nom='Bobine' label='Bobine' />
  <champ nom='LotMachine' label='Machine' />
  <champ nom='TB' label='Top Bottom' />
  <champ nom='1er Coup' label='1er coup' >
    <condition test='[1er Coup]>SUP#[Tolerance Premier Coup]' aspect='erreur blanc gras' />
  </champ>
  <champ nom='10eme Coup' label='10eme coup'>
    <condition test='[10eme Coup]>SUP#[Tolerance Dernier Coup]' aspect='erreur blanc gras' />
  </champ>
  <champ nom='Commentaire' label='Commentaire' />
  <champ nom='Annee' label='Année' />
  <champ nom='Tolerance Premier Coup' label='Tolérance 1er coup' />
  <champ nom='Tolerance Dernier Coup' label='Tolérance 10eme coup' />
</tableau>
```

Dans un second temps, j'ai dû afficher un graphique en courbe comportant les données du 1<sup>er</sup> et 10<sup>ème</sup> coup ainsi que leurs tolérances respectives.

```
<graphique titre='Graphique des coupes feutre de la lame : $lame$' type='courbe' echYPrin='0.20' echYMax='1.40' imageL='1250' >
  <etiquette nom='DateCtrl' />
  <champ nom='1er Coup' traitCoul='blue' traitEpaisseur='3' aspect='ligne' />
  <champ nom='10eme Coup' traitCoul='red' traitEpaisseur='3' aspect='ligne' />
  <champ nom='Tolerance Premier Coup' traitCoul='darkblue' traitEpaisseur='3' aspect='ligne' />
  <champ nom='Tolerance Dernier Coup' traitCoul='darkred' traitEpaisseur='3' aspect='ligne' />
</graphique>
```

Comme pour le tableau, une balise <graphique> est dédiée à la construction de graphique tracé à partir de données. Cette balise a un titre, un type de graphique donc en courbe et enfin une taille et l'image du graphique sur l'axe Y avec les attributs echYPrin, echYMax, imageL pour un meilleur rendu sur la page web. On peut remarquer dans le titre l'apparition de \$lame\$ qui contient le type de lame choisi. Ensuite, la balise étiquette définit le champ servant d'abscisse, ici la date de contrôle. Pour finir, la balise « champ » va permettre de tracer la série de données grâce à l'attribut nom qui définit le nom du champ correspondant à la série. L'attribut aspect permet de définir le type de tracé (ici en ligne) et les attributs traitCoul et traitEpaisseur permettent respectivement de régler la couleur du trait et son épaisseur.

Et voici, le résultat du graphique avec les paramètres d'importation suivant :



Pour finir, il nous reste à implémenter l'option pour filtrer les données. Ce filtre sera effectif sur le tableau ainsi que le graphique.

Voici ce que donne la sélection du filtre pour le tableau et le graphique :

DateCtrl  Type de lame  Top Bottom  Année  Machine

Et voici le code le permettant :

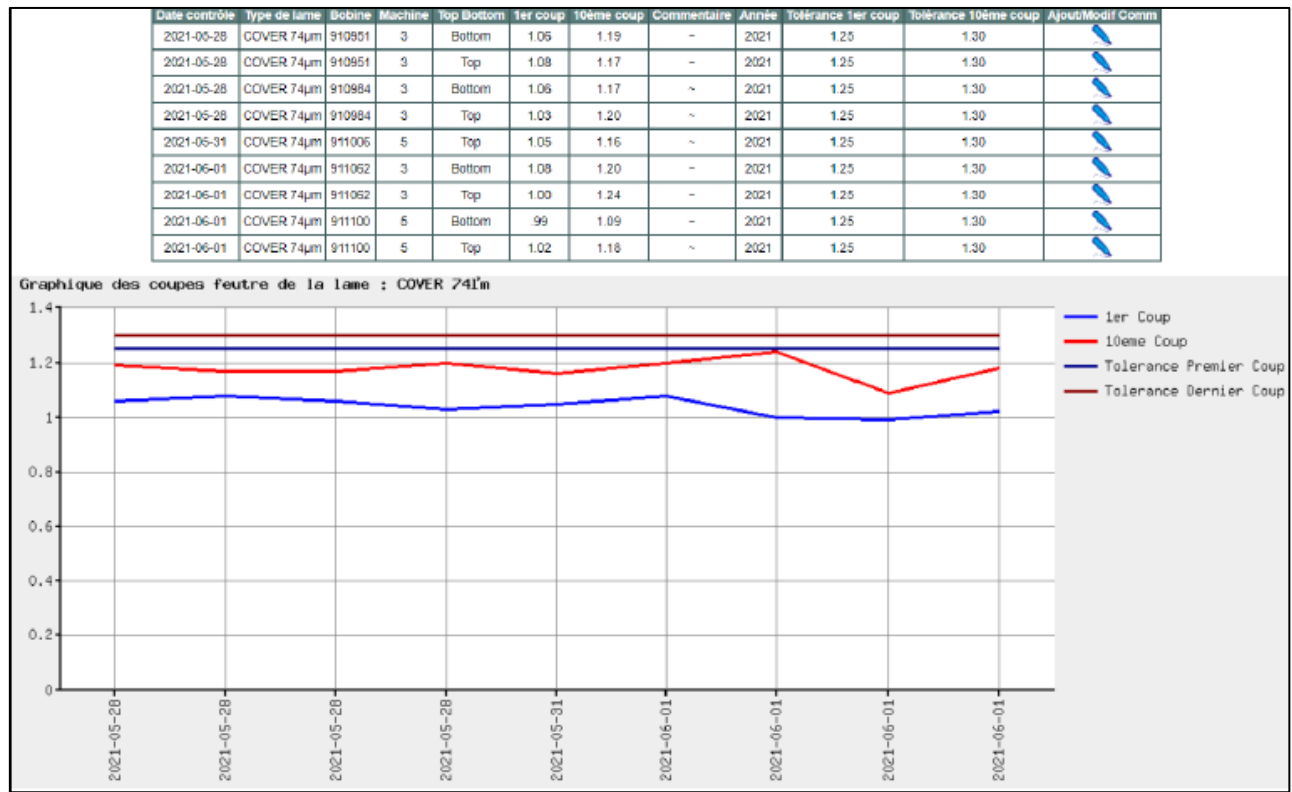
```
<filtre nom='DateCtrl' />
<filtre nom='TypeLame' label='Type de lame' />
<filtre nom='TB' label='Top Bottom' />
<filtre nom='Annee' label='Année' />
<filtre nom='LotMachine' label='Machine' />
```

Pour ce faire, il faut utiliser la balise « filtrer » qui se trouve également dans la balise « affichage ». Cette balise permet de définir un paramètre de filtrage des données affichées : l'attribut nom permet de donner le nom du champ du tableau sur lequel portera le filtrage et le label donne le titre du paramètre qui sera proposé à l'utilisateur. Je donne la possibilité de filtrer les données affichées selon la date de contrôle, le type de lame, si c'est du top ou du bottom, par année et enfin par machine.

Ensuite prenons un exemple concret pour vérifier le bon fonctionnement général de la page ainsi que l'option filtrage.

Selection du type de lame	COVER 74µm	Date de début	2021-05-28	Date de fin	2021-06-24
---------------------------	------------	---------------	------------	-------------	------------

Voici le tableau des données ainsi que son graphique :



Et maintenant choisissons quelques paramètres de filtrage pour ne voir que les données du 1 juin 2021 sur la machine 5 :

2021-06-01

Type de lame

Top Bottom

Année

5

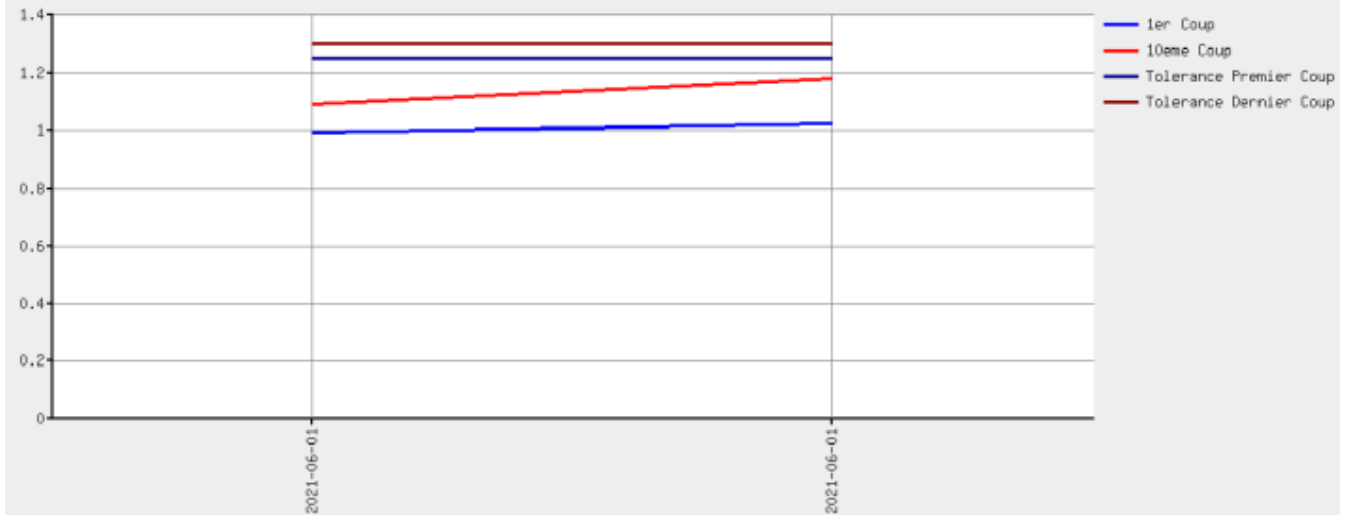
DateCtrl

Machine

Et voici l'impact sur le tableau et le graphique, les données ont été filtrés avec succès :

Date contrôle	Type de lame	Bobine	Machine	Top Bottom	1er coup	10ème coup	Commentaire	Année	Tolérance 1er coup	Tolérance 10ème coup	Ajout/Modif Comm
2021-05-01	COVER 74µm	911100	5	Bottom	.99	1.09	-	2021	1.25	1.30	
2021-06-01	COVER 74µm	911100	5	Top	1.02	1.18	~	2021	1.25	1.30	

Graphique des coupes feutre de la lame : COVER 74µm



Pour conclure, par l'utilisation de BRREQ j'ai pu réaliser la plus grande partie du projet. En résumé, j'ai pu afficher les données sous forme tabulaire ainsi que graphique et y ajouter une option de filtrage des données. BRREQ une bibliothèque extrêmement puissante car par de simples balises XML, elle permet d'afficher un graphique, un tableau, ...

Cependant, il reste encore une partie à effectuer concernant la saisie du commentaire qui ne peut se faire avec BRREQ qui permet exclusivement la définition des sources de données, des traitements et des mises en forme. Pour réaliser la saisie du commentaire, j'ai utilisé une deuxième bibliothèque appelée BRISE. Cette bibliothèque permet de définir des formulaires. Le commentaire permettra de comprendre la cause du dépassement d'une des tolérances.

## 5. Développement de la partie « saisie d'un commentaire »

### a. Utilisation de BRISE

L'utilisation de la bibliothèque BRISE ne peut être contenue dans un fichier utilisant BRREQ. J'ai dû créer 2 fichiers, un fichier de type PHP pour faire appel à la bibliothèque BRISE et un autre fichier XML permettant l'utilisation des balises XML faisant référence à la bibliothèque. Mon premier fichier s'appelle **Coupe\_feutre\_Commentaire.php**. Ce script en PHP contient seulement l'inclusion de BRISE de la librairie BicRWeb (étape 1) et son appel le fichier XML permettant ainsi de l'utiliser (étape 2).

```
<?php
    //// Librairie BicRWeb ////
    require_once '..\..\lib-php\bfp.inc.php';
    require_once REP_BRISSE.'brise.php';


    //// Lancer la consultation
    brise('Coupe_Feutre_Commentaire.xml');
?>
```

Mon deuxième fichier se nomme **Coupe\_Feutre\_Commentaire.xml** et configure toute la saisie désirée via BRISE. La première chose à faire lorsque j'atteins le fichier XML est d'appeler BRISE, grâce à une balise XML afin de pouvoir l'utiliser. L'attribut titre de la balise brise permet de fournir un titre à la requête et de l'afficher dans l'entête de la page web. Le code se trouve dans les balises brise. L'attribut debug lui permet d'afficher des informations de debug et pour l'activer il suffit de remplacer le 'non' par un 'oui'. Enfin, l'attribut debut contient le nom de l'étape de démarrage du formulaire car BRISE fonctionne par algorithme regroupé en plusieurs étapes se faisant référence. BBREQ et BRISE possèdent des balises XML ayant le même fonctionnement, cependant ils possèdent aussi des balises exclusivement exploitables dans leurs bibliothèques respectives.

```
<brise titre='Commentaire : Coupes feutre lame finie ' debug='non' debut='debut' >
</brise>
```

## b. Présentation des formulaires

Capture d'écran du formulaire de consultation montrant le lien vers le formulaire de saisie de commentaire :

Date contrôle	Type de lame	Bobine	Machine	Top Bottom	1er coup	10ème coup	Commentaire	Année	Tolérance 1er coup	Tolérance 10ème coup	Ajout/Modif Comm
2021-05-20	BIC1 76µm	400000	1	Top	1.00	.97	~	2021	1.25	1.30	

Capture d'écran du cheminement du formulaire de saisie de commentaire dans le cas d'un ajout de commentaire :

Commentaire

teststage

Retour

Valider

Enregistrement effectué du commentaire : 'teststage' sur le numéro de lot 89768

Retour

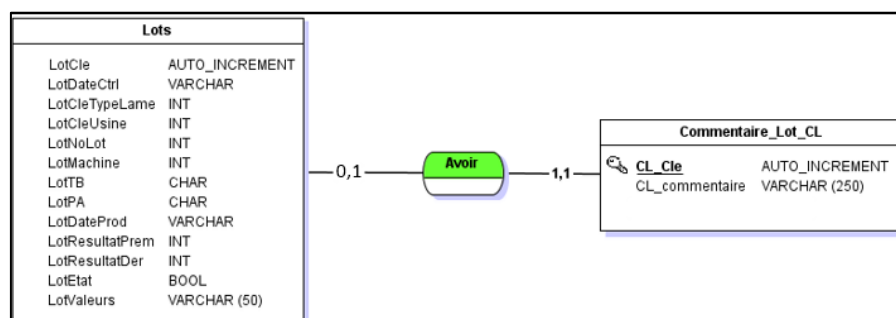
Date contrôle	Type de lame	Bobine	Machine	Top Bottom	1er coup	10ème coup	Commentaire	Année	Tolérance 1er coup	Tolérance 10ème coup	Ajout/Modif Comm
2021-05-20	BIC1 76µm	400000	1	Top	1.00	.97	teststage	2021	1.25	1.30	

Dans la cas d'une modification de commentaire, le cheminement de formulaire de saisie est le même. Sauf l'apparition de l'ancien commentaire dans le champ de saisie du commentaire. Permettant sa modification au cas où la personne se serait trompée de quelques caractères dans son premier commentaire

## c. Modification de la base de données pour intégrer le commentaire

Dans la base de données CoupeFeutre, aucune colonne stockant le commentaire existe. Par logique, ce qui serait attendu ici serait de rajouter une colonne Commentaire dans la table Lots pour l'associer au numéro de lot. Or comme j'ai expliqué dans la première partie, il nous est impossible de modifier les tables de la base de données donc de rajouter cette colonne. Pour permettre de stocker le commentaire, j'ai dû créer une nouvelle table dans la base de données. Cette table contiendra donc le commentaire saisi ainsi qu'une colonne contenant le numéro de lot faisant ainsi référence à la table Lots.

Pour m'aider à la conception de cette table, un MEA a été réalisé :



La conception de ce MEA m'a permis de mieux visualiser la création de la table Commentaire\_Lot\_CL de même que de connaître la table devant stocker la clé étrangère. L'une des règles du MEA vue en cours indique que s'il y a des cardinalités de type 0, n-1,1 une clé étrangère est générée dans la table se trouvant du côté de la cardinalité 1,1.

Et voici le script SQL permettant de créer la Commentaire\_Lot\_CL. CL\_cle est la clé primaire de la table, CL\_commentaire contiendra le commentaire saisie et CL\_LotCle contiendra un numéro de lot faisant la liaison avec la table Lots. Nous pouvons aussi remarquer la contrainte UNIQUE sur la colonne CL\_LotCle car nous souhaitons un unique commentaire pour chaque numéro de lot.

```
CREATE TABLE Commentaire_Lot_CL (
    CL_cle int PRIMARY KEY NOT NULL identity(1,1),
    CL_commentaire varchar(250) NULL,
    CL_LotCle INT NOT NULL UNIQUE
);
```

La création de la clé étrangère de la table Commentaire\_Lot\_CL sur la table Lots n'a pas fonctionné car la table Lots ne contient aucune clé primaire. Or pour générer une clé étrangère la colonne de la table référencée doit être clé primaire. Voici l'ordre permettant de créer la clé étrangère qui n'a pas fonctionné :

```
ALTER TABLE Commentaire_Lot_CL ADD CONSTRAINT FK_Commentaire_Lot_Lots_LotCle
FOREIGN KEY (CL_LotCle) REFERENCES Lots (LotCle)
```



#### d. Le code complet permettant d'obtenir les formulaires

```

<?xml version='1.0' encoding='utf-8' ?>
<brise titre='Commentaire : Coupes feutre lame finie ' debug='non' debut='debut' >
  <alias>
    <externe nomFichier='../AliasGlobal.xml' xpath='/config/connexion[@nom="Coupe_Feutre"]' />
  </alias>

  <source nom='VerifLots'>
    <connexion type='alias' nom='Coupe_Feutre' >
      SELECT COUNT( * ) AS nb, $lot$ as lot, LotNoLot FROM Lots WHERE lotcle=$lot$ GROUP BY LotNoLot
    </connexion>
  </source>

  <source nom='VerifCommentaire'>
    <connexion type='alias' nom='Coupe_Feutre' >
      SELECT LotCle AS nb, COUNT(CL_commentaire) AS commentaire
      FROM Lots
      INNER JOIN Commentaire_Lot_CL ON lotcle=cl_lotcle
      WHERE lotcle=$lot$
      GROUP BY LotCle
    </connexion>
  </source>

  <source nom='ExisteCommentaire'>
    <connexion type='alias' nom='Coupe_Feutre' >
      SELECT CL_commentaire AS commentaire
      FROM Lots
      INNER JOIN Commentaire_Lot_CL ON lotcle=cl_lotcle
      WHERE lotcle=$lot$
    </connexion>
  </source>

  <var nom='lot' valParam='LotCle' />
  <var nom='nbLot' val='0' />
  <var nom='numLot' type='texte' val='' />
  <var nom='erreur' />
  <var nom='commentaire' />
  <var nom='nbCommentaire' val='0' />

  <etape nom='debut'>
    <saut test='$lot$' #EG# ' ' etape='saisieLot' />
    <saut test='$lot$' #DIF# ' ' etape='verifLot' />
  </etape>

  <etape nom='saisieLot' >
    <saisie var='lot' type='ligne' label='Numéro de lot' />
    <bouton nom='Continuer' label='Continuer' etape='verifLot' />
  </etape>

  <etape nom='verifLot' >
    <calcul var='commentaire' val='' />
    <calcul var='nbCommentaire' val='0' />
    <recherche var='commentaire' source='ExisteCommentaire' trouve='commentaire' />
    <chargement source='VerifLots' />
    <recherche var='nbLot' source='VerifLots' quoi='$lot$' cle='lot' trouve='nb' />
    <recherche var='numLot' source='VerifLots' trouve='LotNoLot' />

    <calcul var='erreur' val='' />
    <calcul var='erreur' test='$nbLot$' #EG# 0' val='LotDif' />
    <calcul var='erreur' test='$lot$' #EG# ' ' val='LotVide' />
    <calcul var='erreur' test='!is_numeric('$lot$')' val='LotNonNum' />

    <saut test='$erreur$' #EG# ' ' etape='verifComm' />

    <affichage test='$erreur$' #EG# "LotDif" val='Le numéro de bobine ($numLot$) est incorrect' aspect='erreur' />
    <affichage test='$erreur$' #EG# "LotVide" val='ERREUR : Un numéro de lot doit être saisi' aspect='erreur' />
    <affichage test='$erreur$' #EG# "LotNonNum" val='Le numéro doit être en saisie numérique' aspect='erreur' />

    <bouton nom='Retour' label='Retour' etape='fin' />
  </etape>

```

1 Vérification du numéro de lot et de l'existence d'un commentaire pour ce numéro de lot.  
Code expliqué dans la partie e)

2 Formulaire dans le cas d'un ajout de commentaire.  
Code expliqué dans la partie f)

3 Formulaire dans le cas d'une modification de commentaire.  
Code expliqué dans la partie g)

4 Envoi de la partie BRISE à la partie BRREQ.  
Code expliqué dans la partie h)



```

<etape nom='verifComm' >
  <affichage val='Commentaire du lot : $lot$' />
  <chargement source='VerifCommentaire' />
  <recherche var='nbCommentaire' source='VerifCommentaire' quoi='$lot$' cle='nb' trouve='commentaire' />

  <saut test='$nbCommentaire$ #EG# 0' etape='saisieNouvComm' />
  <saut test='$nbCommentaire$ #SUP# 0' etape='saisieCommMAJ' />
</etape>

<etape nom='saisieNouvComm' >
  <saisie var='numLot' testM='false' label='N° Bobine' />
  <saisie var='commentaire' type='bloc' label='Commentaire' />
  <bouton nom='Retour' label='Retour' etape='fin' />
  <bouton nom='OUI' label='Valider' etape='commEnreg' />
</etape>

<etape nom='commEnreg' >
  <enregistrement>
    <connexion type='alias' nom='Coupe_Feutre' >
      INSERT INTO Commentaire_Lot_CL (CL_commentaire,CL_LotCle)
      VALUES ('$commentaire$', '$lot$')
    </connexion>
  </enregistrement>
  <affichage val='Enregistrement effectué du commentaire : '$commentaire$' sur le numéro de bobine $numLot$' aspect='vert gras' />
  <bouton nom='Retour' label='Retour' etape='fin' />
</etape>

<etape nom='saisieCommMAJ' >
  <chargement source='ExisteCommentaire' />
  <!-- <affichage type='source' val='ExisteCommentaire' /> -->
  <recherche var='commentaire' source='ExisteCommentaire' trouve='commentaire' />
  <saisie var='numLot' testM='false' label='N° Bobine' />
  <saisie var='commentaire' type='bloc' label='Commentaire' />
  <bouton nom='Retour' label='Retour' etape='fin' />
  <bouton nom='OUI' label='Valider' etape='commMAJ' />
</etape>

<etape nom='commMAJ' >
  <enregistrement>
    <connexion type='alias' nom='Coupe_Feutre' >
      UPDATE Commentaire_Lot_CL
      SET CL_commentaire='$commentaire$'
      WHERE CL_LotCle=$lot$
    </connexion>
  </enregistrement>
  <affichage val='Mis à jour effectué du commentaire : $commentaire$ sur le numéro de bobine $numLot$' aspect='vert gras' />
  <bouton nom='Retour' label='Retour' etape='fin' />
</etape>

<etape nom='fin' >
  <branchement lien='../Consults\Coupe_Feutre.php' />
</etape>
</brise>

```

### e. Présentation du code permettant de vérifier le numéro de lot et l'existence d'un commentaire

Le formulaire de gestion des commentaires contient 2 champs de saisie :

Numéro de lot <input type="text"/> <input type="button" value="Continuer"/>	Commentaire <input type="text"/> <input type="button" value="Retour"/> <input type="button" value="Valider"/>
--	--

Pour commencer, le programme va demander à l'utilisateur de saisir le numéro de lot, puis un contrôle sera effectué pour vérifier la validité de celui-ci (s'il existe dans la base de données). S'il est incorrect le programme s'arrête avec un message d'erreur sinon le programme va regarder s'il existe déjà un commentaire pour ce numéro de lot. Si oui, le programme récupère et affiche ce commentaire et permet à l'utilisateur de modifier ce commentaire dans une zone de saisie. Enfin, un UPDATE est fait sur le commentaire pour le mettre à jour dans la base de données et un message de confirmation est affiché mettant fin au programme. Mais s'il n'y a aucun commentaire pour le numéro de lot alors une zone de saisie apparaît pour saisir un nouveau commentaire. Puis un INSERT est effectué pour insérer ce nouveau commentaire dans la base de données et un message de confirmation est affiché mettant également fin au programme.

Pour commencer comme pour BRREQ, il nous faut déclarer la base de données utilisée c'est le même procédé expliquée pour BRREQ.

```
<alias>
  <externe nomFichier='../AliasGlobal.xml' xpath='/config/connexion[@nom="Coupe_Feutre"]' />
</alias>
```

Ensuite, la balise « source » qui contient les requêtes SQL à effectuer sur la base de données CoupeFeutre. La première balise source 'VerifLots' sert à vérifier si le numéro de lot saisi par l'utilisateur existe dans la base de données en utilisant la fonction COUNT qui permet de compter le nombre d'enregistrements. Cette requête va compter le nombre d'enregistrement existant du numéro de lot saisi par l'utilisateur dans la base de données. Si la requête ne retourne la valeur 0 cela veut dire que le numéro de lot n'existe pas, sinon le numéro de lot existe dans la base de données donc il est valide.

```
<source nom='VerifLots'>
  <connexion type='alias' nom='Coupe_Feutre' >
    SELECT COUNT( * ) AS nb, $lot$ as lot, LotNoLot FROM Lots WHERE lotcle=$lot$ GROUP BY LotNoLot
  </connexion>
</source>
```

La requête SQL permet de vérifier si un commentaire existe déjà pour un numéro de lot (LotCle) l'utilisation d'un COUNT. Je nomme cette source VerifCommentaire pour la réutiliser plus tard dans la balise « chargement ». Cette requête permet de dénombrer le nombre d'enregistrement de commentaires par rapport au numéro de lot

```
<source nom='VerifCommentaire'>
  <connexion type='alias' nom='Coupe_Feutre' >
    SELECT LotCle AS nb, COUNT(CL_commentaire) AS commentaire
    FROM Lots
    INNER JOIN Commentaire_Lot_CL ON lotcle=cl_lotcle
    WHERE lotcle=$lot$
    GROUP BY LotCle
  </connexion>
</source>
```

Enfin voici la dernière requête SQL permettant d'obtenir le commentaire déjà existant dans la pour le modifier. Je nomme la source ExisteCommentaire et cette requête SQL permet d'extraire le commentaire en fonction du numéro de lot saisi par l'utilisateur.

```
<source nom='ExisteCommentaire'>
  <connexion type='alias' nom='Coupe_Feutre' >
    SELECT CL_commentaire AS commentaire
    FROM Lots
    INNER JOIN Commentaire_Lot_CL ON lotcle=cl_lotcle
    WHERE lotcle=$lot$
  </connexion>
</source>
```

L'autre grande différence avec BBREQ est que BRISE offre la possibilité de déclarer des variables qui seront ensuite utilisées plus tard au sein du code. Une variable est déclarée avec la balise « var » qui permet de définir le nom et la valeur initiale.

```
<var nom='lot' />
<var nom='nbLot' val='0' />
<var nom='erreur' />
<var nom='commentaire' />
<var nom='nbCommentaire' val='0' />
```

Comme expliqué dans le début de ma seconde partie, BRISE fonctionne par étape, chaque étape correspondant à une étape du formulaire de saisie du commentaire. Les étapes du formulaire sont appelées par la balise « etape » et l'attribut nom permet de nommer cette étape. Pour commencer, je nomme 'debut' la première étape de ce programme. Cette étape se compose d'une partie sur la saisie du numéro de lot par le biais de la balise « saisie », cette balise permet la création d'un champ de saisie. L'attribut var donne le nom de la variable où sera stockée la valeur saisie et l'attribut type définit le type de champ donc ici le type 'ligne' permet une saisie simple en ligne. Puis le label sera le texte affiché devant la zone de saisie pour l'utilisateur. Enfin, la balise « bouton » permet la création d'un bouton d'action qui permet de rediriger vers une étape donnée qui sera donc ici 'verifLot' et le label contient le texte affiché dans le bouton.

```
<etape nom='debut' >
  <saisie var='lot' type='ligne' label='Numéro de lot' />
  <bouton nom='Continuer' label='Continuer' etape='verifLot' />
</etape>
```

Et voici ce que donne l'étape debut :

Numéro de lot

Après avoir cliqué sur le bouton 'Continuer', nous nous retrouvons dans l'étape 'verifLot' :

```
<etape nom='verifLot' >
  <calcul var='commentaire' val='' />
  <calcul var='nbCommentaire' val='0' />
  <recherche var='commentaire' source='ExisteCommentaire' trouve='commentaire' />
  <chargement source='VerifLots' />
  <recherche var='nbLot' source='VerifLots' quoi='$lot$' cle='lot' trouve='nb' />

  <calcul var='erreur' val='' />
  <calcul var='erreur' test='$nbLot$ #EG# 0' val='LotDif' />
  <calcul var='erreur' test='$lot$' #EG# '' val='LotVide' />
  <calcul var='erreur' test='!is_numeric('$lot$)' val='LotNonNum' />

  <saut test='$erreur$' #EG# '' etape='verifComm' />

  <affichage test='$erreur$' #EG# 'LotVide' val='ERREUR : Un numéro de lot doit être saisi' aspect='erreur' />
  <affichage test='$erreur$' #EG# 'LotNonNum' val='Le numéro doit être en saisie numérique' aspect='erreur' />

  <bouton nom='Retour' label='Retour' etape='fin' />
</etape>
```

Cette étape va permettre de vérifier si le numéro de lot saisi existe dans la base de données sinon le programme renverra un message d'erreur. La balise « calcul » permet d'effectuer un calcul dans une variable, ici la variable 'commentaire' se voit devenir une chaîne de caractères vide à chaque renouvellement de cette étape. Dans l'étape suivante sur la saisie du commentaire, si un commentaire

est déjà associé à un numéro de lot, le programme l'affiche dans le champ de saisie pour être modifié. nbCommentaire est également réinitialisé à chaque renouvellement de l'étape. Ensuite, la balise « recherche » permet de rechercher une clé dans une table source et d'écrire cette valeur dans la variable commentaire. L'attribut source indique le nom de la source de données qui est notre requête SQL permettant de récupérer le commentaire et l'attribut trouve indique le nom du champ dans lequel se trouve la valeur à retourner.

Puis il est enfin de temps de vérifier si le numéro de lot saisi est bon. Alors j'utilise la balise « chargement » qui déclenche le chargement des données de la source 'VerifLots' et la balise « recherche » est réutilisée pour aller extraire le numéro de lot dans la requête avec l'attribut 'quoi' qui donne la valeur à rechercher dans le champ clé (les "\$xxx\$" forcent le remplacement par la valeur de la variable 'xxx'). L'attribut 'cle' indique le nom du champ dans lequel rechercher le 'quoi' et enfin avec l'attribut trouve je lui indique de trouver 'nb' l'alias de LotCle dans VerifLots. J'ai créé des messages d'erreur lorsque la zone de saisie est vide (1) ou encore si le numéro de lot saisi n'est pas au format numérique (2) avec la balise « affichage » qui permet la création d'un champ d'affichage. Enfin l'étape 3 est la plus importante car c'est elle qui nous fait changer d'étape. Le test est le suivant : si l'erreur est égale à une chaîne de caractères vide cela veut dire qu'il n'y a aucune erreur pour le numéro de lot alors on se rend sur l'étape suivante 'verifComm'. Pour finir, un bouton 'Retour' est mis à disposition lorsque l'on se trouve sur la page d'erreur pour se rendre à l'étape 'fin' que j'expliquerai plus tard.

Par exemple, si dans la saisie du numéro je saisis une chaîne de caractères au lieu d'un nombre grâce au script ci-dessus un message d'erreur apparaît :

Numéro de lot

Continuer

Le numéro doit être en saisie numérique

Retour

Si le numéro de lot est bon, on se retrouve ici dans l'étape 'verifComm'. Cette étape permet de vérifier s'il existe déjà un commentaire ou non associé au numéro de lot et provoque le branchement à la bonne étape suivant cette condition (l'insertion d'un nouveau commentaire ou mise à jour du commentaire). Tout d'abord avant de vérifier l'existence d'un commentaire, j'affiche le numéro du lot concerné pour la saisie du commentaire à l'utilisateur avec la balise « affichage ».

Pour vérifier s'il existe un commentaire pour ce lot, on charge tout d'abord la requête SQL 'VerifCommentaire' écrite préalablement qui est chargée par la balise « chargement ». Puis j'utilise la balise « recherche » et ses attributs pour aller chercher la colonne 'commentaire' retournée par la requête SQL. Cette requête permet d'obtenir le nombre de commentaires du numéro de lot puis cette valeur est stockée dans la variable 'nbCommentaire'. Si 'nbCommentaire' est égale à zéro cela veut dire qu'il n'y a aucun commentaire pour ce numéro de lot alors je passe à l'étape 'saisieNouvComm' pour insérer un nouveau commentaire grâce la balise « saut » (1). Et si nbCommentaire est supérieur à zéro cela veut dire qu'il existe déjà un commentaire et je passe à l'étape 'saisieCommMaj' pour mettre à jour le commentaire (2).

```
<etape nom='verifComm' >
  <affichage val='Commentaire du lot : $lot$.' />
  <chargement source='VerifCommentaire' />
  <recherche var='nbCommentaire' source='VerifCommentaire' quoi='$lot$' cle='nb' trouve='commentaire' />
  <saut test='$nbCommentaire$ #EG# 0' etape='saisieNouvComm' /> 1
  <saut test='$nbCommentaire$ #SUP# 0' etape='saisieCommMAJ' /> 2
</etape>
```

#### f. Présentation du code permettant d'insérer un nouveau commentaire

Je vais maintenant vous expliquer l'étape 'saisieNouvComm' qui permet de saisir un nouveau commentaire.

Voici ce que donne l'étape 'saisieNouvComm' :



La balise « saisie » me permet comme pour le numéro de lot de faire saisir dans un champ un commentaire récupéré dans la variable commentaire. L'attribut type contient bloc ce qui permet la saisie d'un bloc de texte sur plusieurs lignes et l'attribut label permet d'afficher un texte à cette saisie. Enfin, les deux balises « bouton » permettent de créer deux boutons. L'un nommé 'Valider' emmenant à l'étape 'commEnreg' lorsque que la saisie du commentaire est finie et l'autre nommer 'Retour' emmenant à l'étape fin que nous verrons à la fin de ce programme.

```
<etape nom='saisieNouvComm' >
  <saisie var='commentaire' type='bloc' label='Commentaire' />
  <bouton nom='Retour' label='Retour' etape='fin' />
  <bouton nom='OUI' label='Valider' etape='commEnreg' />
</etape>
```

En cliquant sur le bouton 'Valider', on se retrouve dans l'étape 'commEnreg' qui va permettre d'insérer le commentaire saisi dans la base de données. La balise « enregistrement » permet l'enregistrement des données et la balise « connexion » définit le type et la connexion à la base. Après avoir défini la connexion, on effectue la requête SQL qui insère le commentaire ainsi que le numéro de lot associé dans la table Commentaire\_Lot\_CL (en jaune). Ensuite j'affiche un message à l'utilisateur pour lui indiquer que son commentaire a bien été enregistré pour le numéro de lot avec la balise « affichage ». Enfin un bouton 'Retour' est mis à disposition à l'utilisateur, en cliquant dessus cela emmène directement l'utilisateur à l'étape fin qui sera expliqué plus tard. C'est ainsi que se finit la partie BRISE lorsque l'on souhaite ajouter un nouveau commentaire.



```
<etape nom='commEnreg' >
  <enregistrement>
    <connexion type='alias' nom='Coupe_Feutre' >
      INSERT INTO Commentaire_Lot_CL (CL_commentaire,CL_LotCle)
      VALUES ('$commentaire$', '$lot$')
    </connexion>
  </enregistrement>
  <affichage val="Enregistrement effectué du commentaire : '$commentaire$' sur le numéro de lot $lot$" aspect='vert gras' />
  <bouton nom='Retour' label='Retour' etape='fin' />
</etape>
```

#### g. Présentation du code permettant de modifier un commentaire

Cette étape 'SaisieCommMAJ' va permettre d'afficher le commentaire déjà existant et de le modifier. Tout d'abord, je charge la source 'ExisteCommentaire' contenant la requête SQL qui permet d'extraire le commentaire du numéro de lot saisi par l'utilisateur. Avec la balise « recherche » qui sera stocké dans la variable 'commentaire' grâce à la balise « recherche ». Puis comme pour l'insertion d'un nouveau commentaire, je crée une zone de saisie de type bloc contenant la variable commentaire obtenue avec dans la balise « recherche ». Il sera possible de modifier. Pour finir, les deux balises « bouton » permettent de créer deux boutons. L'un nommé 'Valider' emmenant à l'étape 'commMAJ' lorsque que la modification du commentaire est finie et l'autre nommé 'Retour' emmenant à l'étape fin que nous verrons à la fin de ce programme.

```
<etape nom='saisieCommMAJ' >
  <chargement source='ExisteCommentaire' />
  <recherche var='commentaire' source='ExisteCommentaire' trouve='commentaire' />
  <saisie var='commentaire' type='bloc' label='Commentaire' />
  <bouton nom='Retour' label='Retour' etape='fin' />
  <bouton nom='OUI' label='Valider' etape='commMAJ' />
</etape>
```

Puis en cliquant sur le bouton 'Valider', on se retrouve ici dans l'étape 'commMAJ' qui a la même structure et fonction que l'étape 'commEnreg' mais la requête et le message affiché sont différents. La requête SQL utilise un UPDATE pour mettre à jour la colonne contenant le commentaire contenu dans la variable commentaire (en jaune). Enfin j'affiche un message expliquant que la mise à jour du commentaire s'est bien déroulé. Enfin, un bouton 'Retour' est mis à disposition de l'utilisateur, en cliquant dessus cela l'emmène directement à l'étape fin qui sera expliqué plus tard. C'est ainsi que se finit la partie BRISE lorsque l'on souhaite mettre à jour un commentaire.

```
<etape nom='commMAJ' >
  <enregistrement>
    <connexion type='alias' nom='Coupe_Feutre' >
      UPDATE Commentaire_Lot_CL
      SET CL_commentaire='$commentaire$'
      WHERE CL_LotCle=$lot$
    </connexion>
  </enregistrement>
  <affichage val="Mis à jour effectué du commentaire : $commentaire$ sur le numéro de lot $lot$" aspect='vert gras' />
  <bouton nom='Retour' label='Retour' etape='fin' />
</etape>
```

Prenons un exemple pour illustrer les étapes de mise à jour d'un commentaire. Par exemple sur le numéro de lot '89846', il existe déjà un commentaire dans la base de données. Ce commentaire est le suivant : 'test123' et je souhaite le modifier. Pour cela, je rentre le numéro de lot correspondant et je clique sur le bouton 'Continuer' :

COMMENTAIRE : COUPES FEUTRE LAME FINIE	
Numéro de lot	<input type="text" value="89846"/>
<input type="button" value="Continuer"/>	

Ensuite, une zone de commentaires apparait avec le commentaire déjà présent dans la base de données :

Commentaire	<input type="text" value="test123"/>
	<input type="button" value="Retour"/> <input type="button" value="Valider"/>

Je le modifie en ajoutant le chiffre '4' et je clique sur le bouton 'Valider' :

Commentaire	<input type="text" value="test1234"/>
	<input type="button" value="Retour"/> <input type="button" value="Valider"/>

Et voilà un message s'affiche m'indiquant que la mise à jour du commentaire s'est effectuée avec succès :

COMMENTAIRE : COUPES FEUTRE LAME FINIE
Mis à jour effectué du commentaire : test1234 sur le numéro de lot 89846
<input type="button" value="Retour"/>

Pour finir, voici la dernière étape de ce programme. J'arrive sur cette étape lorsque je clique sur le bouton 'Retour' et tous les boutons 'Retour' du formulaire amènent à cette étape. Nommé 'fin', cette étape redirige l'utilisateur sur la consultation BRREQ par le biais de la balise « branchement » qui permet un branchement sur un autre script en indiquant le chemin de destination, ici notre programme BRREQ.

```
<etape nom='fin' >
  <branchement lien='../Consults\Coupe_Feutre.php' />
</etape>
```

Pour conclure, la bibliothèque BRISE m'aura permis de faire saisir, d'insérer ou de mettre à jour un commentaire. C'est une bibliothèque très puissante car de simples balises « etape » permet d'aboutir à un vrai formulaire. Le projet est terminé remplissant ainsi tous les besoins du cahier des charges. Cependant pour une meilleure expérience, il est préférable de lier mes 2 programmes pour une utilisation

plus fluide. Nous avons pu voir les prémices de ce lien avec la dernière étape de la bibliothèque BRISE. C'est donc ce que je vais développer dans ma dernière partie.

#### h. Présentation du code faisant le lien entre la partie BRREQ et la partie BRISE

Dans ma dernière partie, je vais présenter comment, à partir du tableau donnant les caractéristiques des coups de lame il est possible de saisir ou de modifier un commentaire. Pour ce faire, je vais créer une colonne dans mon tableau contenant un lien qui mène directement à mon script BRISE permettant la saisie du commentaire ou de sa modification. BRREQ enverra à BRISE le numéro de lot associé à la ligne du tableau que l'on souhaite commenter, il ne sera donc plus nécessaire besoin de le saisir.

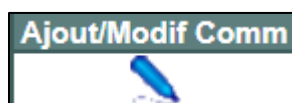
Tout d'abord dans le script BRREQ, il nous faut faire un lien vers notre script BRISE. Pour cela j'ai ajouté une balise « traitement » dans la balise données. Ensuite, je définis le type de traitement avec l'attribut type en 'calculer' qui permet de calculer des valeurs pour de nouveaux champs ou de modifier des champs existants. Dans les calculs, les noms des champs à utiliser sont écrits entre [] (ex : [Date] pour le champ de nom 'Date'). Puis la balise « champ » correspond au champ traité qui contiendra le lien. Je nomme ce champ 'lien' avec l'attribut nom et définis comme valeur pour le paramètre une image en png avec un '\*' servant de séparateur pour spécifier une info bulle par le biais de l'attribut val qui correspond au fichier image. J'attribue également un lien hypertexte dans le champ vers mon programme BRISE avec l'attribut lien. Remarque : le lien doit être aussi paramétré au niveau de l'affichage tableau afin d'apparaître correctement. L'attribut dest en 'local' indique que notre lien pour le programme BRISE s'ouvrira sur la page web actuelle et non sur un nouvel onglet. Enfin la balise « lienParam » permet de passer un paramètre au lien hypertexte. Ce paramètre est bien évidemment le numéro de lot (LotCle), le script BRISE peut directement récupérer le numéro de lot de ligne du tableau sélectionné.

```
<traitement type='calculer'>
  <champ nom='lien' val='modifier.png*Cliquez ' lien='../Insertions\Coupe_Feutre_Commentaire.php' dest='local' >
  <lienParam nom='LotCle' valCalc='[LotCle]' />
  </champ>
</traitement>
```

De plus, la dernière étape qui nous reste à effectuer au sein de BBREQ est de faire apparaître ce lien dans notre tableau. Pour ce faire, on crée un autre champ dans le tableau à la suite de ceux déjà existants avec la balise « champ ». Mais ici on lui donne le nom de notre paramètre de lien donc 'lien' et lui indique un type 'image'. L'image affichée sera le fichier correspondant au nom de la cellule (chemin & fichier). Enfin l'attribut lien permet d'activer dans le champ le lien hypertexte et l'attribut label correspond au titre de la colonne.

```
<champ nom='lien' type='image' lien='oui' label='Ajout/Modif Comm' />
```

Voici ce que donne donc ces deux scripts dans BBREQ :



Pour finir il ne nous reste plus qu'à modifier notre script BRISE pour qu'il récupère le numéro de lot venant de BRREQ, et ainsi ne plus effectuer l'étape de saisie de lot. Pour ce faire, j'affecte



directement à la variable 'lot' la valeur 'LotCle' provenant de la ligne de données du tableau BBREQ choisie par l'utilisateur. C'est l'attribut valParam qui me permet d'indiquer le nom du paramètre contenant la valeur d'initialisation. Dans notre cas 'LotCle' donc le numéro de lot.

```
<var nom='lot' valParam='LotCle' />
<var nom='nbLot' val='0' />
<var nom='erreur' />
<var nom='commentaire' />
<var nom='nbCommentaire' val='0' />
```

Enfin, la dernière étape consiste à passer la saisie du numéro de lot car nous le récupérons déjà par le biais du script BBREQ. L'utilisation de la balise « saut » permet de sauter à une autre étape et si plusieurs sauts sont définis, le dernier dont le test est bon sera effectué. Son attribut test est la condition à vérifier permettant le saut ou non vers une autre étape. Ici les conditions sont assez simples : si la variable lot est égal à une chaîne de caractères vide alors l'utilisateur se retrouve à l'étape 'saisieLot' pour saisir un numéro de lot sinon on se rend à l'étape verifLot pour vérifier de la validité du numéro de lot.

```
<etape nom='debut'>
  <saut test='"$slot$" #EG# ""' etape='saisieLot' /> 1
  <saut test='"$slot$" #DIF# ""' etape='verifLot' /> 2
</etape>
```

Je sélectionne dans mon tableau une ligne de données dans laquelle je souhaite insérer un commentaire. Ici j'ai choisi cette ligne et vous pouvez voir qu'elle ne contient aucun commentaire dans sa colonne commentaire :

Date contrôle	Type de lame	Bobine	Machine	Top Bottom	1er coup	10ème coup	Commentaire	Année	Tolérance 1er coup	Tolérance 10ème coup	Ajout/Modif Comm
2021-05-20	BIC1 76µm	400000	1	Top	1.00	.97	~	2021	1.25	1.30	

Puis je clique sur l'image dans le champ 'Ajout/Modif Comm' qui cache notre lien vers le script BRISE pour y ajouter un commentaire. Une fois cliqué, je me retrouve sur une zone de saisie. Ensuite, je saisis un commentaire :

Commentaire

Retour

Valider

Je valide en cliquant sur le bouton 'Valider' puis un message apparaît montrant que l'opération s'est effectuée avec succès :


Enregistrement effectué du commentaire : 'teststage' sur le numéro de lot 89768

Retour

Au niveau de la base de données le commentaire a bien été inséré dans la table CL\_Commentaire\_Lots :

CL_de	CL_commentaire	CL_LotCle
1002	teststage	89768
	Commentaire	numéro de lot

Le tableau propose à présent le commentaire saisi :


Date contrôle	Type de lame	Bobine	Machine	Top Bottom	1er coup	10ème coup	Commentaire	Année	Tolérance 1er coup	Tolérance 10ème coup	Ajout/Modif Comm
2021-05-20	BIC1 76µm	400000	1	Top	1.00	.97	teststage	2021	1.25	1.30	

Pour conclure, la liaison entre les deux bibliothèques est opérationnelle rendant ainsi le module web plus fluide pour l'utilisateur. Cette dernière partie met un terme au développement de l'application. L'étape finale a été la présentation du projet lors d'une réunion qui sera développée dans ma partie bilan.

En bonus, je vais vous présenter une autre bibliothèque PHP différente de BRISE et de BRREQ que je n'ai pas utilisée lors de mon projet. Cette bibliothèque nommée **BFRAME** permet de définir le menu sur une page web. Et voici son script XML exploitant mes deux scripts pour les afficher dans un menu web. Ici, la balise « brame » contient tout le code, l'attribut titre correspond au titre affiché à l'utilisateur et son attribut val définit le texte de l'affichage. Puis l'attribut retour définit la page cible pour le bouton 'Retour'. Ensuite, la balise « affichage » va permettre de créer l'affichage des deux projets dans le menu. Enfin, l'attribut lien indique le lien vers les différents projets et l'attribut dest en 'EXT' indique que les projets s'ouvriront à l'extérieur donc sur un nouvel onglet.

```
<brame titre="Coupe-feutre lame finie" retour='' login='oui' >
  <affichage val="Modules" />
  <bouton nom="Consultation Coupe Feutre" lien='Consults/Coupe_Feutre.php' dest='EXT' />
  <bouton nom="Commentaire Coupe Feutre" lien='Insertions/Coupe_Feutre_Commentaire.php' dest='EXT' />
</brame>
```

Et voici le rendu de ce script BFRAME :

 COUPE-FEUTRE LAME FINIE			
		MODULES	
Consultation Coupe Feutre	Commentaire Coupe Feutre		

#### IV. CONFIGURATION D'UN PC SUR LE DOMAINE DE L'ENTREPRISE ET MIGRATION DE DONNEES

Dans cette partie, je vais vous parler brièvement d'une courte mission qui m'a été confiée. Cette mission est axée sur les notions réseaux diversifiant ainsi mon stage.

##### 1. Genèse de la mission

Un employé au sein de l'entreprise avait besoin de changer son pc portable car il devenait obsolète. En amont, un nouveau pc avait été commandé et sa réception est intervenue durant mon stage. Ma mission était la suivante :

- Configurer et effectuer les mises à jour au démarrage ;
- Mettre ce pc sur le domaine de l'entreprise ;
- Migrer les données de l'ancien sur le nouveau ;
- Installez toutes les applications utiles à l'employé.

##### 2. Réalisations

Pour commencer, j'ai configuré Windows 10 au démarrage et effectué toutes les dernières mises à jour possible du système. Ensuite, j'ai renommé le nom du pc avec un nom spécifique à l'entreprise et paramétré le pc sur le domaine de celle-ci. Pour joindre l'ordinateur au domaine de l'entreprise, je suis allé dans le « panneau de configuration » puis dans « système et sécurité » pour ensuite cliquer sur « système ». Sous « Paramètres de nom d'ordinateur, de domaine et de groupe de travail », j'ai cliqué sur « Modifier les paramètres ». Et sous l'onglet « Nom de l'ordinateur », j'ai choisi « Modifier » et sous « membre de », j'ai cliqué sur « domaine ». J'ai ensuite fourni le nom du domaine de l'entreprise auquel cet ordinateur doit être joint et j'ai redémarré le pc.

Une fois le pc configuré, il ne me restait plus qu'à attendre l'arrivée de l'employé pour pouvoir migrer ses données. Pour ce faire, l'employé devait se munir de son ancien pc branché en câble Ethernet et se connecter avec sa session au nouveau pc. Ensuite, j'ai exécuté en faisant Windows + R la commande suivante : '\\nom ancien pc\C\$'. Cette commande m'a permis de me positionner dans le disque local (C:) de l'ancien pc et je me suis rendu dans le dossier utilisateur de l'employé afin de copier-coller tous les dossiers nécessaires indiqués par l'employé dans le nouveau pc. La même manipulation a été réalisée sur le disque (D:) en changeant bien évidemment dans la commande d'exécution Windows le 'C\$' par 'D\$'. Pour finir, j'ai paramétré les différents services du pc en activant la licence Word, installation de l'imprimante, la configuration du VPN, ...

Pour conclure, cette mission m'a permis de faire une courte pause dans mon projet. Et de pouvoir consolider mes acquis et d'acquérir de nouvelles compétences sur les domaines systèmes et réseaux d'une entreprise.

### 3. BILAN

Pour commencer, ce projet m'a beaucoup apporté sur l'aspect technique et humain. Tout d'abord sur l'aspect technique, j'ai pu découvrir de nouveaux outils (SQL Server 2012, Xampp) et langage (XML) que je n'avais pas encore vu en cours. Augmentant ainsi mon portefeuille de compétence et ma curiosité. J'ai pu découvrir la notion de Framework maison et son utilisation avec BicRWeb. De plus durant ce projet, j'ai pu consolider mes bases en SQL et apprendre de nouvelles commandes comme CASE...WHEN...

Ensuite sur l'aspect humain, j'ai pu découvrir ce que c'était de travailler en équipe sur un projet commun dans une entreprise. Moi qui n'avais effectué que des projets à plus petite échelle au lycée. Le fait de savoir que son projet va servir à l'entreprise rajoute une plus-value ainsi qu'une très grande satisfaction pour son travail. J'ai appris également que le travail personnel et l'autonomie permettent d'augmenter ses compétences personnelles. Enfin, j'ai pu découvrir ce que c'était d'être chef de projet temporairement et à faible échelle en concevant un projet à partir d'un cahier des charges pour finalement le présenter aux clients. J'ai également pu apprendre que le besoin du client est le plus important et que parfois le client exprime d'autres besoins non contenus dans le cahier des charges initial. Par exemple durant la réunion, qui s'est très bien passée, des modifications et améliorations ont été exprimées et certaines n'étaient pas initialement convenues dans le cahier des charges comme le fait d'ajouter un bouton « supprimé » servant à supprimer une ligne du tableau dans certains cas. Néanmoins, ce point-là m'a beaucoup plu et m'a permis d'améliorer mon contact humain ainsi que mon élocution orale.

Pour finir, la plus grande difficulté de ce projet a été la base de données. Il était impossible de pouvoir modifier les tables déjà existantes car n'étant pas développée par le MIS. En résumé, il est impossible de toucher à l'intégrité des tables pour ne pas perturber toutes les applications existantes. Cependant, nous avons réussi à passer outre ces problèmes par des palliatifs expliqués dans mon développement mais ce problème m'a permis de comprendre la difficulté de travailler à plusieurs car nous ne contrôlons pas le code à 100% et c'est là que l'esprit d'équipe est très important.

#### 4. REMERCIEMENTS

Pour commencer, je tiens à remercier Pascal HENNION pour m'avoir permis d'effectuer mon stage dans son service. De tous ses bons conseils précieux sur le monde de l'informatique et de la bonne humeur ainsi que l'accueil offert tout au long de mon stage.

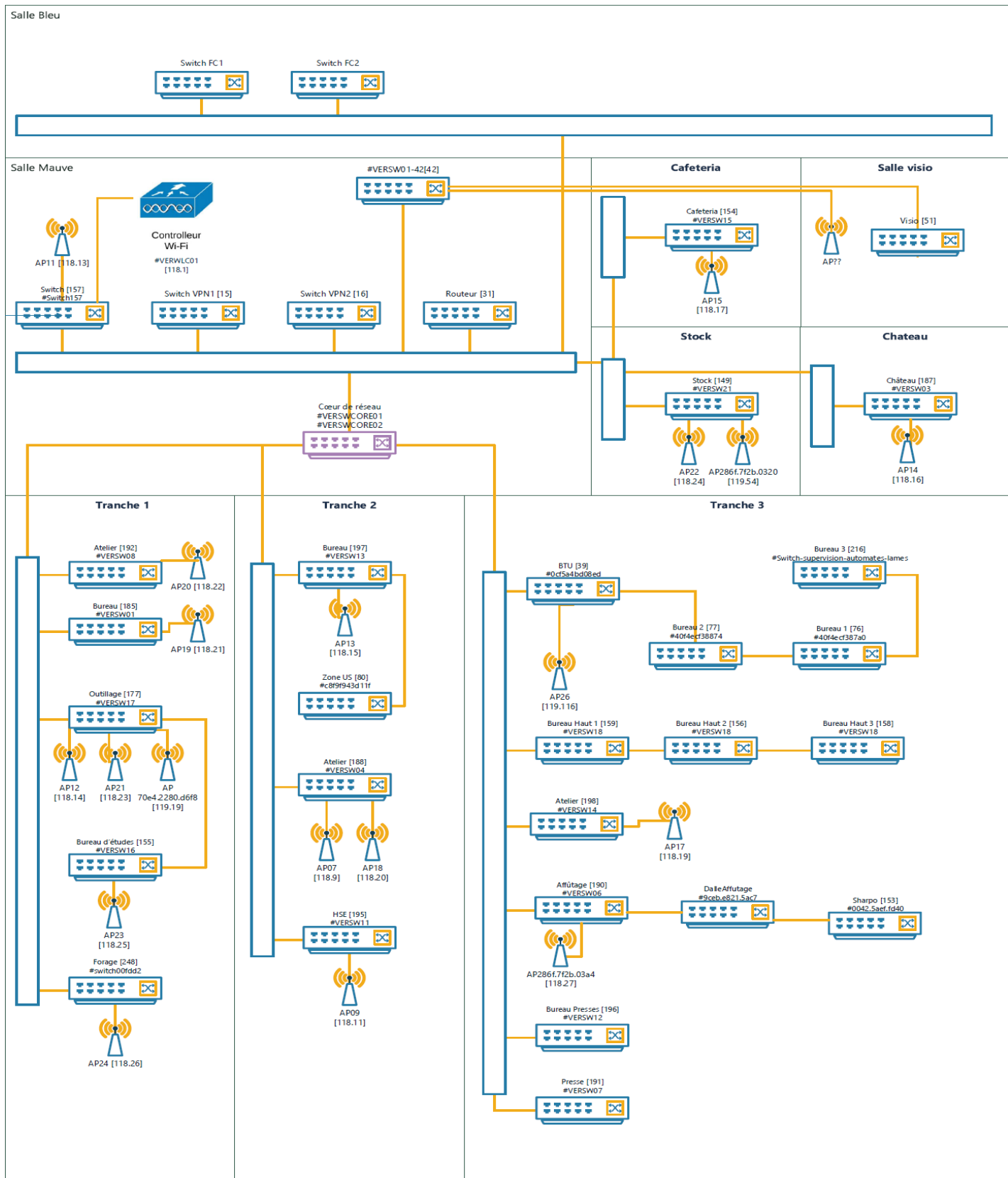
Ensuite, je voudrais tout particulièrement remercier Jérémy BRASSART pour m'avoir accompagné tout au long de ce stage. De m'avoir conseillé et aidé sur la conception du projet ainsi que de m'avoir appris de nouvelles choses sur l'informatique consolidant mes bases en développement et réseau. Et enfin d'avoir pris le temps de répondre à toutes mes questions et de m'avoir aidé à concevoir ce rapport.

Enfin, je tiens également à remercier tous les collaborateurs de BIC Rasoirs pour leur accueil, leur gentillesse et leur humour, qui ont rendu mon stage très agréable.

P.S. : Merci pour toutes les pauses café offertes.

## 5. ANNEXES

### Annexe 1 : Schéma réseau de l'usine



## Annexe 2 : Structure initiale de la base CoupeFeutre

Table **Calib** (elle contient le calibrage de la machine de coupe feutre)

CalibCle  
Numéro incrémenté croissant

CalibDate  
Date sous la forme "AAAAMMJJ"

CalibValAvanceFeutre  
Entier positif 2 digits

CalibCapteurForceDyn  
Entier positif 4 digits

CalibCapteurForceCarte  
Entier positif 4 digits

CalibCalibrationOK  
Booléen

Table **Lots** (elle contient les lots de lames de l'usine)

LotCle  
Numéro incrémenté croissant

LotDateCtrl  
Sous forme "AAAAMMJJ"

LotCleTypeLame  
Entier positif 1 digit - correspond au choix dans 'TypeLameCle' de la table 'TypeLames'

LotCleUsine  
Entier positif 1 digit - correspond au champ choix dans 'UsineCle' de la table 'Usines'

LotNoLot  
Entier positif 6 digit - numéro de lot saisi

LotMachine  
Entier positif 1 digit - correspond au numéro de machine saisi

LotTB  
Texte Court ('T' ou 'B') - correspond à la saisie 'Top' ou 'Bottom'

LotPA  
Texte Court ('O' ou 'N') - correspond à la réponse à 'Prélèvement Affûtage ?' saisi

LotDateProd  
Date sous la forme "AAAAMMJJ", si LotPA = 'O'

LotResultatPrem  
Entier positif 3 digits

LotResultatDer  
Entier positif 3 digits

LotEtat  
Booléen

LotValeurs  
Chaîne contenant les mesures individuelles, en Newton, sous la forme de 7 séries de 10 fois 3 chiffres. Chaque série est terminée par une virgule  
Exemple :  
'114122138137147157166173178197,193134148154170179193199207229,  
137137150150167179182193210215,189185193200203228239236252268,  
135143149164162183184207198221,135147146156161176189202205224,  
110115117126136151145165167174,'



Table **TypesLames** (elle contient les différents type de lames ainsi que leurs tolérance à différents coup)

TypeLameCle

Numéro incrémenté croissant

TypeLameNom

Chaîne dénomination du type de lame

TypeLameFil

Chaîne type de fil de lame

TypeLameResultatMaxPrem

Limite de tolérance max pour la valeur du premier coup

TypeLameResultatMaxDer

Limite de tolérance max pour la valeur du dernier (10e) coup

Table **Usines** (elle contient le nom des différentes usines)

UsineCle

Numéro incrémenté croissant

UsineNom

Chaîne dénomination de l'usine