Université de Franche-Comté
UFR Sciences et techniques

MASTER M2E2 parcours ESE
Année 2016

Rapport Final de Stage

pour l'obtention du Master 2 M2E2 parcours électronique et systèmes embarquées

XX, Maître de stage
Ingénieur responsable
Méthodes et industrialisation

M. Franck CHOLLET
Maitre de conférences
département électronique

TORRES ZETINO Juan Carlos
Stagiaire de Master

Besançon, le 23 Août 2016

1 Contenu

2 CONFIDENTIALITE

Le contenu de ce rapport de stage et les documents qui l'accompagnent sont la propriété de la société SIMU.
Il est interdit de les copier ou de les transmettre à des tiers (y compris la mise à disposition dans une bibliothèque).

Leur usage est strictement limité au cadre du stage effectué au sein de la société SIMU.

3 Introduction

Le présent travail montre les différents projets qui ont été réalisés au sein du département de méthodes et industrialisation de l'entreprise SIMU SAS fabricant de moteurs tubulaires pour les volets roulants, le premier projet consiste à numériser différentes procédures afin d'améliorer la saisie d'information et connaître la disponibilité des emplacements au magasin, d'ailleurs, le deuxième projet consiste à développer une application pour calculer les indicateurs de productivité d'un ensemble de machines afin d'obtenir le niveau de sa performance, ainsi que les causes de déperditions qui empêchent une production optimale de pièces .

Afin de réaliser les deux projets, il faudra choisir la technologie la plus adaptée pour répondre aux besoins du cahier des charges, en ce sens, nous présenterons les alternatives choisies pour développer le logiciel, ainsi que la structure matérielle (hardware) pour mettre en place l'application, en outre, concernant les étapes du système, il faudra développer une partie dédiée au traitement d'information et une autre partie correspondant au stockage de données provenant des différents utilisateurs.

A cet égard, je présenterai de façon détaillée les objectifs de chaque projet en insistant sur la structure matérielle du système et l'interaction du logiciel avec les utilisateurs, après j'expliquerai le schéma du logiciel développé avec les parties plus importantes qui ont été conçues pour répondre au cahier des charges. L'interface utilisateur développée sera analysée pour comprendre son interaction avec le monde réel.

Finalement, je présenterai mes conclusions et recommandations au niveau des évolutions futures de l'application, sécurité et contraintes qui devront être prises en compte afin d'améliorer la façon de répondre aux besoins du projet.

4 Présentation de la société et organigramme

SIMU c'est une société industrielle créée en 1952 spécialisée dans la conception, la fabrication et la commercialisation de moteurs, d'automatismes et d'accessoires pour les systèmes de fermeture comme par exemple les volets roulants, portes de garage, fermetures industrielles et commerciales.

C'est une société implantée dans tout le territoire français avec différentes branches par zone géographique, elle possède en plus une forte présence internationale avec des équipes de vente et centres de productions dans différents pays du monde.

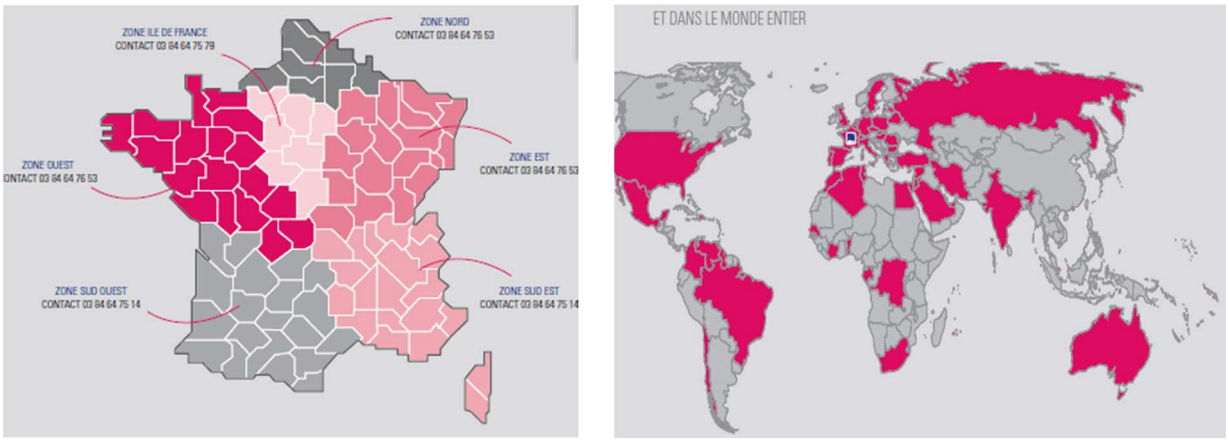


Fig1 Implantation de SIMU au niveau du territoire français et sa présence internationale.

Parmi les innovations qui ont eu lieu dans l'entreprise, nous pouvons mentionner la ligne de moteurs alternatifs, en effet, ils ont été les premiers à concevoir le moteur tubulaire, qui comporte notamment le condensateur, la paire stator-rotor, le frein, le système de transmission et le système de fin de course électronique, ces groupes de pièces sont ainsi adaptés pour être assemblés dans un cylindre à différents diamètres soit **40-50-80-90 mm** selon son utilisation dans la fermeture de portes.

Fig2: Moteur AC fabriqué en l'atelier diamètre 50

4.1 Organigramme de la société

Au niveau d'organisationnel, elle est composée d'une structure hiérarchique avec différent départements, dans la suite nous présentons un extrait simplifié de cette structure :

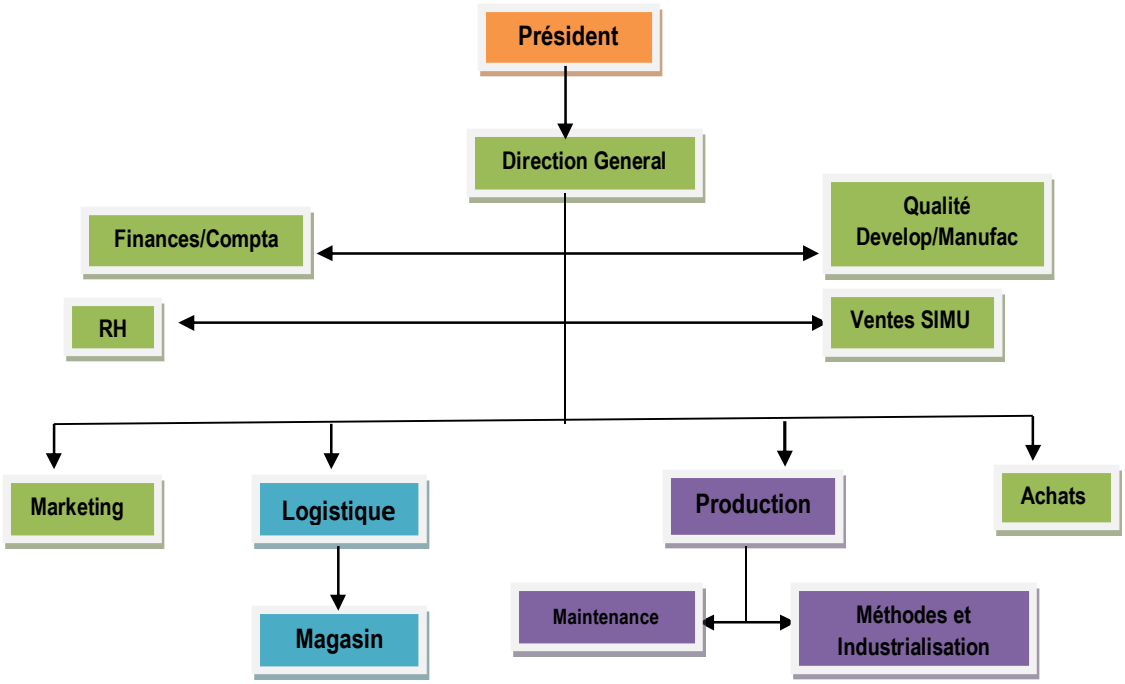


Fig3 : Organigramme simplifié de l'entreprise et départements concernés pendant le stage

Les activités réalisées dans le cadre du stage comportent deux départements différents, le premier sera au sein de la branche logistique notamment pour réaliser un essai de la technologie choisie, il a été développé dans le magasin principal et il consiste à développer une application pour améliorer la gestion des emplacements.

Le deuxième projet, avec plus de besoins et connaissances techniques, a été développé au sein de la branche production dans le département de méthodes et industrialisation, il consiste à réaliser un système qui permettra d'améliorer la saisie de paramètres de productivité d'un ensemble de machines.

Les deux projets ont servi pour introduire un premier essai d'amélioration future à réaliser dans l'usine notamment dans l'installation de systèmes numériques pour la gestion d'information, dans la suite je présenterai plus en détail et de façon chronologique les différents projets réalisés ainsi que les étapes pour les accomplir.

5 Technologie pour le développement de l'application logiciel

Comme dans tout projet que ce soit dans le domaine de l'informatique, électronique ou l'ingénierie civile il faut un temps pour analyser et choisir les meilleurs outils technologiques qui puissent répondre aux besoins d'un système complexe, à cet égard, afin de répondre de manière efficace au cahier des charges du département de méthodes, j'ai fait une étude approfondie des technologies les plus adaptées pour obtenir la solution désirée, celle-ci prend en considération des paramètres critiques comme le temps de développement, la connaissance nécessaire pour mettre en place une technologie, l'investissement au niveau d'argent, le temps de développement et aussi les évolutions futures à prévoir, dans la suite je présenterai les outils choisis ainsi que les caractéristiques principales.

5.1 Langage Java environnement JDK

C'est un langage de programmation orientée objet basée sur C qui possède les avantages suivants :

- C'est une technologie multiplateforme indépendante du système d'exploitation, il est composé d'un compilateur **javac** qui transforme le code source du programme **(.java)** en byte code **(.Class)**, et d'une machine virtuel **JVM** qu'exécute le byte code selon l'architecture du hardware ou l'environnement java est installé.
- il possède un grand nombre de bibliothèques pour implémenter fonctionnalités de calcul, traitement de données, communication web, stockage de données, formatage du temps, etc.
- Il possède une spécification JEE (Java Entreprise Edition) qui est un standard développé par Oracle avec la participation des organisations indépendantes, experts de l'industrie de software ,etc. qui permet le **développement des applications web** en utilisant le langage java comme référence, il peut être associé avec d'autres technologies comme par exemple HTML, CSS, JavaScript pour implémenter des fonctions complémentaires.

5.2 Serveur de Base de données MySQL

En ce qui concerne la base de données, j'ai décidé d'utiliser le système MySQL qui est un serveur de bases de données relationnelles SGBDR qui stocke l'information en forme de tableaux à 2 dimensions, il utilise le langage SQL pour la gestion et stockage d'information de manière persistante.
MySQL c'est un système de base de données Open source(GPL) développé par Oracle qui a un développement actif et stable.

Il possède de multiples mises à jour, avec des fonctionnalités riches pour l'automatisation de procédures, gestion de données, Trigger, etc.
Il possède en plus un paquet de connexion JDBC qui permet de communiquer avec des applications en langage Java.

5.3 Serveur Apache Tomcat

Au niveau du serveur pour héberger l'application informatique, j'ai choisi **Apache Tomcat** qui est un conteneur Web qu' implémente **les servlets** et **pages JSP** ainsi qu'un serveur HTTP pour recevoir et transmettre des requêtes du protocole de couches applicatives, il permettra d'héberger l'application Web développée sur Java JEE ainsi que de traiter les différentes requêtes provenant des clients pour générer des pages HTTP de façon dynamique .

Dans la suite nous voyons les composants d'une Application Web Java EE hébergée sur le Serveur Apache Tomcat

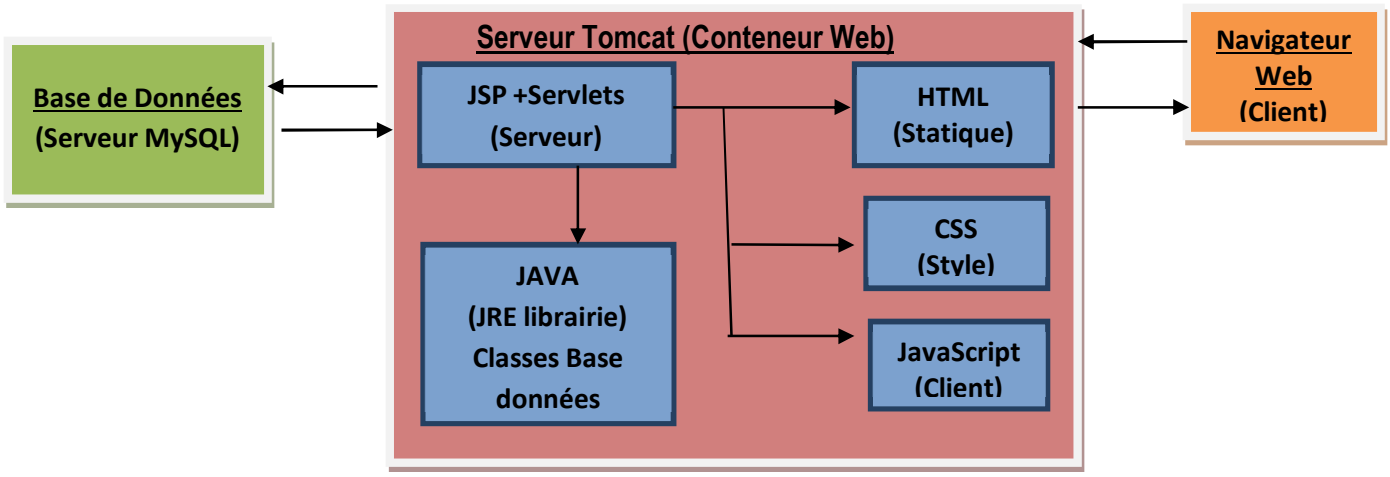


Fig4: Composants technologiques d'une application web développée en java et installée sur un serveur.

6 Structure matériel et logiciel pour le développement des applications en réseau.

Dans la suite nous présentons la structure utilisée pour développer les deux projets technologiques au sein du département de méthodes, ainsi que la mise en place des équipements pour déployer l'application en production.

La solution comprend l'installation d'une partie hardware et software dans l'atelier. Pour la première partie, j'ai décidé d'installer 2 ordinateurs en mode serveur pour exécuter les applications développées en java et aussi pouvoir installer le système de base de données dans le même endroit, au niveau connectivité j'ai décidé de relier ces 2 ordinateurs au réseau local déjà en fonctionnement sur l'atelier afin réduire l'investissement dans un serveur commercial en web hosting ou bien le développement d'un réseau externe.

Pour le software, un des besoins du cahier des chargés a été le fait d'utiliser des logiciels libres pour développer l'application, donc j'ai décidé d'utilisé les solutions OpenSource disponibles en industrie. Cela m'a permis d'offrir une performance élevée, souplesse pour évoluer les technologies et réduction de dépenses en licences et soutien technique des solutions commerciales.

Structure matériel et logiciel d'un système informatique en réseau

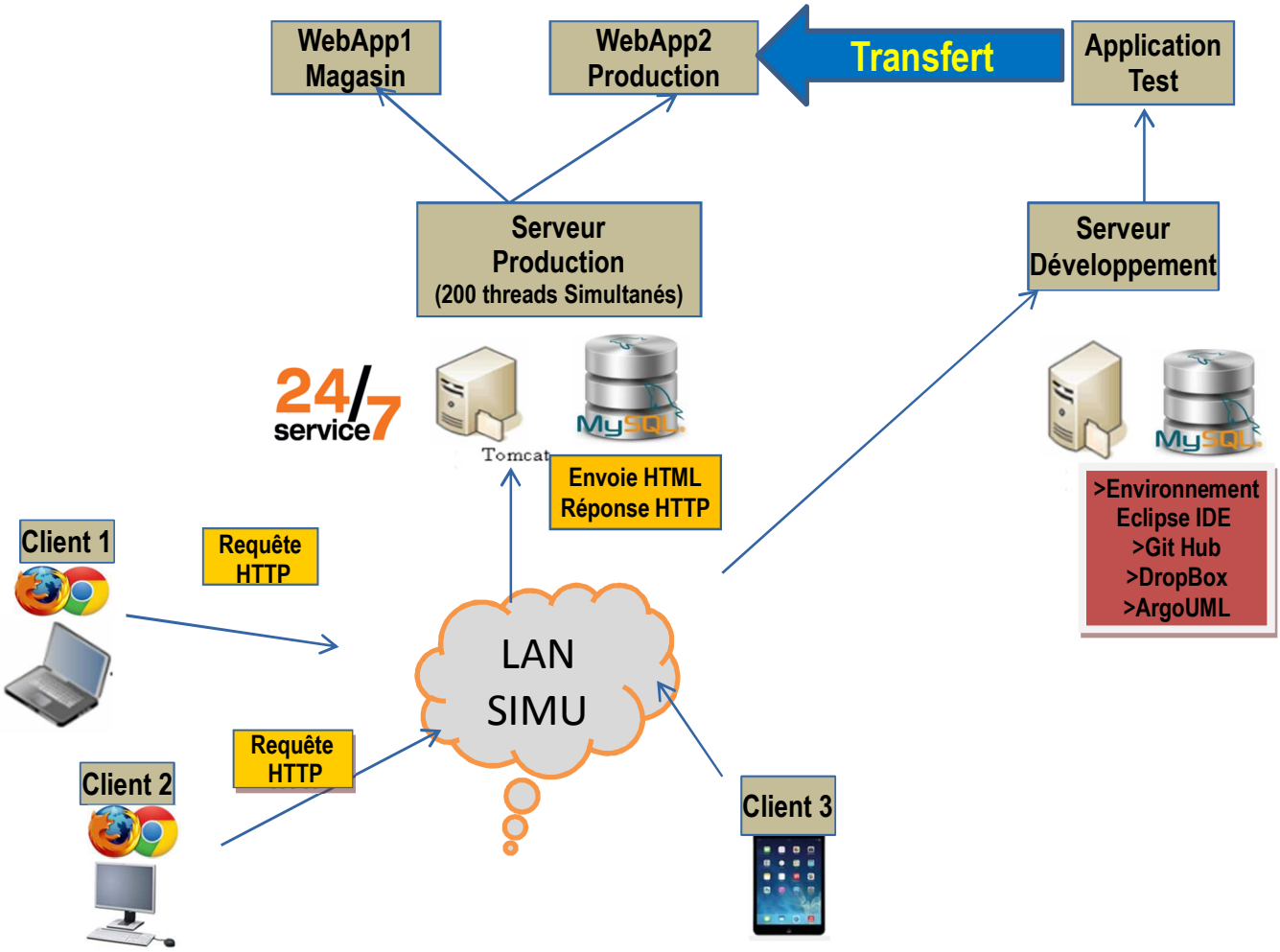


Fig5 : Description matériel et logiciel d'un projet informatique dans le réseau local de l'atelier

Serveur de développement : c'est un ordinateur avec SO Windows ou GNU/LINUX qui contient l'environnement de développement Eclipse pour la réalisation de l'application Web, il contient en plus l'environnement Java JDK avec la JVM qui est capable d'interpréter le byte code résultant de la compilation du code source.

L'ordinateur comporte aussi **un serveur de base de données MySQL** en mode hôte local(Local host) qui permet le stockage des données de façon persistante, cette configuration permet la réduction du temps de communication car le serveur Web se trouve dans la même machine, nous aurons finalement un Workbench de travail MySQL qui permet de créer, vérifier et exécuter les fichier de requêtes SQL de façon plus aisée que dans la console.

Nous constatons que nous aurons aussi **un serveur Web** (conteneur Web) pour héberger et exécuter l'application Web selon les requêtes provenant des clients connectés (navigateur web), dans notre cas nous utiliserons la version **Tomcat 8.0** pour sa performance, stabilité et maintenance continue.

Serveur de Production : il est composé des mêmes outils logiciels installés sur le serveur de développement à différence qu'on déploiera les fichiers WAR de chaque application dans le conteneur Web Tomcat pour son exécution en permanence , cet ordinateur est lié au réseau local de l'entreprise et il est en fonctionnement de manière continue afin de rendre service sur demande. Il est composé d'une mémoire RAM 8G et d'un espace disponible de disque dure de 360GB.

7 Contexte du Projet 1 :gestion des emplacements du magasin

La présente section montre les objectifs du premier projet que j'ai réalisé dans le cadre du stage, il est divisé en différentes sous-sections qui permettront d'expliquer les points plus importants développés comme par exemple la structure du logiciel, l'information relative à la base de données ainsi que l'interface utilisateur mise en place dans le magasin, finalement nous verrons les conclusions et recommandations futures à prévoir dans l'application.

Objectif :
L'objectif du projet sera de connaitre l'état de disponibilité des emplacements dans les étagères du magasin de stockage à travers une application informatique en réseau qui est accessible depuis un ordinateur ou tablette. Le système doit permettre la modification et lecture de l'état des étagères et cela aura comme résultat de réduire le temps que les opérateurs prennent à récupérer les palettes ainsi que l'amélioration dans la recherche d'emplacements libres du magasin.

Points à développer :
Identifier et définir de façon précise et optimale les différents états d'un emplacement lorsqu'il est plein, il possède une palette ou bien il est vide, cette information permettra de concevoir l'application.
Par ailleurs il faudra choisir l'ensemble de technologies à utiliser pour développer l'application ainsi que le modèle logiciel (Framework) plus adapté pour répondre au besoin.
Finalement il faudra prévoir la mise en place d'une infrastructure composée d'ordinateurs à des endroits précis et reliés avec le serveur de production afin d'améliorer la saisie d'informations.

7.1 Structure du logiciel de l'application web

Nous avons utilisé le ***modèle 3 tiers*** qui permet de diviser une application en 3 parties indépendantes autant en termes de technologie que des objectifs (actions) à accomplir pour répondre au cahier des charges.

Architecture 3-tiers

Fig6: Architecture en 3 tiers d'une application logiciel.

Couche présentation : c'est l'interface utilisateur en langage HTML/JSP , il comporte aussi les feuilles de style et fichiers JavaScript pour valider les formulaires, elle permet d'afficher les pages en réponse provenant du serveur à travers son navigateur web, cette couche est utilisée pour envoyer les requêtes provenant du client pour la lecture ou écriture de données selon l'action choisie. Elle sera composée d'une partie coté serveur, le contrôleur qui permet d'exécuter les méthodes **POST** ou **GET** de la requête http.

Couche métier: Cette couche permet le traitement de données envoyées par le client, il comporte toutes les classes nécessaires pour calculer une valeur et préparer les données de travail à stocker ou afficher.
Couche données: Cette couche permet de gérer tous les traitements et code correspondant à l'enregistrement ou récupération de données dans la base de données à travers des requêtes SQL, en plus elle est composée de la base de données elle-même avec la structure des tableaux où seront enregistrés les valeurs de manière persistante.

A cet égard la structure de l'application développée sera la suivante :

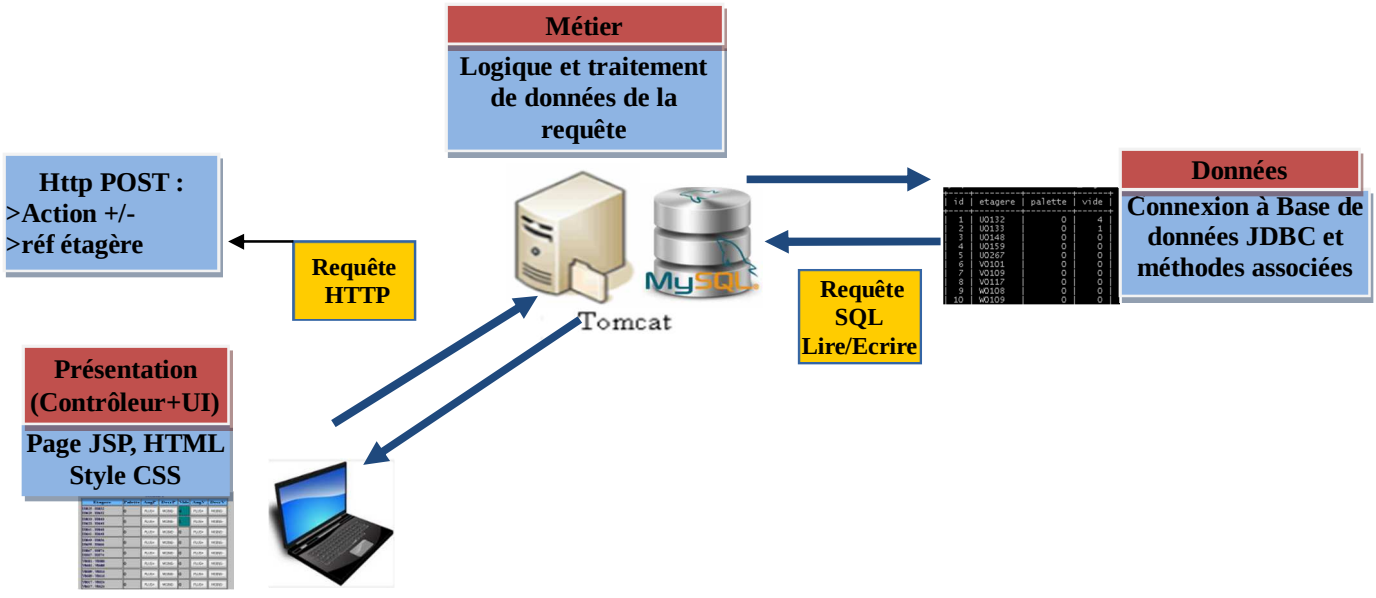


Fig7: Structure de l'application web modèle 3 tiers développée pour le projet1

7.2 Base de données MySQL couche de données

Le magasin de l'atelier est composé d'environ 3200 emplacements disponibles, afin de réduire le nombre de références à créer dans la base de données et améliorer la lisibilité de l'interface utilisateur. J'ai décidé de prendre uniquement la première référence de chaque étagère, ainsi nous avons obtenu 66 éléments au total qui représentent l'ensemble des emplacements de l'atelier.

La base de données créée comporte un tableau avec 4 colonnes principales qui ont les variables suivantes :

- **Id (INT):**Cette colonne sera utilisée pour identifier de façon précise une étagère
- **Etagère (CHAR) :** il comporte le nom et la chaine de caractères utilisée pour représenter une référence.
- **Palette (UNSG INT):** Il permet de stocker les nombres des palettes sans produit selon l'étagère choisi.
- **Vide (UNSG INT) :** il permet de stocker les nombres des emplacements vides sans produit ni palettes selon l'étagère choisie.

Cette structure permet ainsi de répondre aux besoins de couches métiers et présentation de l'application web, lors de la lecture ou écriture d'information dans la base de données.

id	etagere	palette	vide
1	UO132	0	4
2	UO133	0	1
3	UO148	0	0
4	UO159	0	0
5	UO267	0	0
6	VO101	0	0

Fig8 : un extrait du tableau utilisé pour stocker les références d'emplacements du magasin

Dans la suite nous voyons les différents types de variables qui comportent le tableau du magasin, vu la compatibilité entre les variables int au niveau de MySQL et java, il sera plus aisée la récupération et traitement de données stockées comme le nombre des espaces vides, palettes, etc. car l'application n'aura pas besoin de convertir ou modifier le type de variable lors de l'interaction avec la base de données.

Field	Type	Null	Key	Default	Extra
id	int(4)	NO	PRI	NULL	auto_increment
etagere	char(10)	NO		NULL	
palette	int(4) unsigned	YES		NULL	
vide	int(4) unsigned	YES		NULL	

Fig9 : Description de variables du tableau du magasin

7.2.1 Implémentation de la librairie JDBC pour la gestion de la base de données.

Au niveau logiciel dans l'application web Java EE, la couche de données permettra d'implémenter les requêtes SQL nécessaires pour modifier la base de données, en plus, elle permettra la récupération de l'information des étagères qui seront transmis aux couches supérieures pour son traitement.

Pour réaliser l'écriture de données j'ai décidé de créer une classe java capable d'implémenter différentes méthodes composées des requêtes SQL précises, afin d'interagir avec le serveur MySQL où se trouve l'information du tableau magasin, les méthodes ainsi implémentées sont les suivantes :

Pour les emplacements (espaces) avec palettes

```
public void decrementerP(String  nomEtagere) ;
public void incrementerP(String  nomEtagere) ;
```

Pour les emplacements (espaces) vides

```
public void decrementerV(String  nomEtagere) ;
public void incrementerV(String  nomEtagere) ;
```

Prototype de méthodes à implémenter contenant des requêtes SQL pour agir sur la base de données

Au niveau de lecture de données la seule méthode à implémenter sera une liste qui récupère l'ensemble de toutes les étagères sur le magasin, cela permettra de savoir et afficher à chaque requête http de lecture l'état le plus récent du tableau magasin.

```
public List<Etagere>listEtageres() ;
```

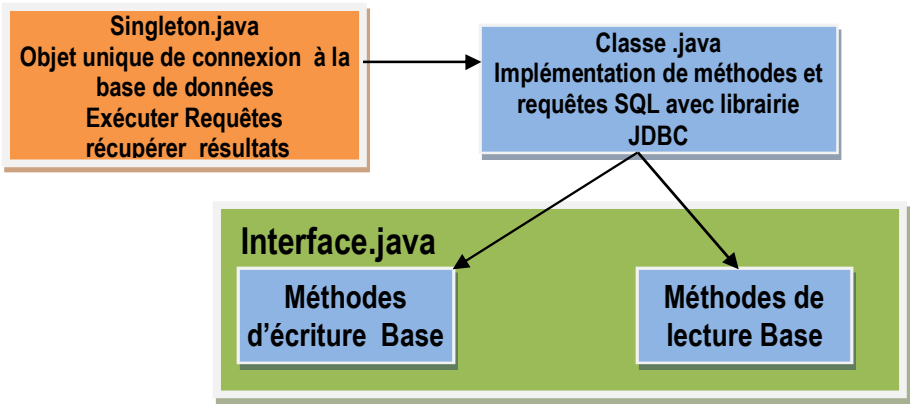


Fig10 : Interface de connexion à la base et l'implémentation de ses méthodes avec requêtes SQL

Dans la suite, je présenterai un extrait de l'implémentation d'une méthode contenant les requêtes SQL pour interagir avec la base, la classe dans laquelle cette méthode a été implémenté contient l'ensemble de méthodes définies par l'interface c.à.d. Les prototypes de méthodes de lecture et écriture.

```
public void incrementerP(String nomEtagere) {
    Connection conn=SingletonConnection.getConnection();//Creation Connection
    try {
        PreparedStatement  ps =conn.prepareStatement("UPDATE table_magasin SET palette=palette+1 WHERE etagere='"+nomEtagere+"'");
      //  ps.setString(0, nomEtagere);

        ps.executeUpdate();//Executer la requete
        ps.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Fig11 : Implémentation d'une méthode contenant une requête SQL

La structure de la méthode comporte 4 étapes principales :

- **La connexion à la base de données** : nous utiliserons un objet de connexion créé préalablement dans le singleton afin d'établir une seule session ou les requêtes SQL seront exécutées et les résultats de la base seront récupérés
- **Préparation de la requête** : nous allons préparer la syntaxe de la requête à envoyer au serveur SQL.
- **L'exécution de la requête :** une fois l'objet « statement » créé, on peut envoyer et exécuter la requête désirée sur la base de données
- **Fermeture de ressources de requête :** à chaque requête envoyée nous allons libérer les ressources de la JDBC afin de qu'ils soient disponibles pour d'autres demandes de l'application web, il faut noter que la connexion avec la base de données à travers le port 3306 restera

toujours ouverte car elle demande un coût de temps et ressources trop important pour la réaliser à chaque requêtes SQL.

Cette structure a été suivie afin d'implémenter l'ensemble de méthodes en lecture et écriture de l'interface java, j'ai dû faire des adaptations respectives selon l'action à réaliser dans la base afin de obtenir les résultats désirés.

7.3 Application java dans la couche Métier

La couche métier qui est composée d'un ensemble des classes java sert à traiter les paramètres provenant du client, ainsi, il implémente la logique à exécuter pour modifier la base de données par numéro de référence en agissant sur les colonnes des emplacements vides ou avec palettes .

```
//Recuperation de Parametres de Client
String  action =request.getParameter(CHAMP_ACTION1);
String  nomE =request.getParameter(CHAMP_NOM1);// No
String  message=request.getParameter(CHAMP_TEXT);
```

Fig12 : Méthodes de la couche métier pour la récupération des paramètres de la requête

Dans un premier temps nous récupérerons les paramètres provenant de la requête qui seront traités par l'application, ainsi nous obtenons les valeurs suivantes :

- **Action :** Ce paramètre représente le type d'action à exécuter sur la base de données, il peut s'avérer d'une décrémentation ou bien d'une incrémentation d'une valeur.
- **nomE** : Ce paramètre représente le nom de l'étagère à modifier, il contient le choix que l'utilisateur à fait parmi l'ensemble des étagères disponibles dans le magasin de stockage.
- **Champ_Text** : Ce paramètre permet de récupérer le message de texte envoyé par le client qui sera enregistré dans un Bean au niveau application, en ce sens il sera disponible à tous les clients connectés.

Un autre aspect intéressant du code développé dans cette partie c'est l'utilisation d'une structure conditionnelle **if else,** qui permet de choisir la méthode SQL correspondant à l'action désirée par l'utilisateur, ainsi un paramètre peut avoir différentes valeurs entre deux requêtes http. Dans la première, l'utilisateur pourra demander une incrémentation de nombre de palettes, tandis que dans la deuxième il demande une décrémentation des emplacements vides en utilisant une autre référence.

```
//Methodes metier pour execution d'action Utilisateur
 if(action.equals("incP")){
     metier.incrementerP(nomE);
     updateDate= metier.getUpdateTable();
     model2.setUpdateDate(updateDate);
 }else if(action.equals("decP")) {
     metier.decrementerP(nomE);
     updateDate= metier.getUpdateTable();
     model2.setUpdateDate(updateDate);
 }else if(action.equals("incV")) {
     metier.incrementerV(nomE);
     updateDate= metier.getUpdateTable();
     model2.setUpdateDate(updateDate);
 }else if(action.equals("decV")) {
     metier.decrementerV(nomE);
     updateDate= metier.getUpdateTable();
     model2.setUpdateDate(updateDate);
 }else if(action.equals("SEND")){
     model2.setMessage(message);
 }else{}

 //Recuperation  d'attributs depuis ListEtagere  paquet metier
     listEtagere =metier.listEtageres();
 for(int i =0;i<listEtagere.size();i++){
     nbrPalette.add(listEtagere.get(i).getNbrPalette());
     nbrVide.add(listEtagere.get(i).getNbrVide());
 }
```

Fig13 : Structure conditionnelle pour exécuter différentes méthodes SQL selon l'information de la requête

Finalement, il restera à récupérer l'état le plus récent de la base de données à la fin de chaque traitement de requête HTTP , pour le faire nous allons appeler une méthode SQL qui aura comme rôle de récupérer l'information de toutes les étagères présentes sur la base (c.à.d. les emplacements vides et palettes), ensuite j'enregistre l'ensemble dans une liste nommé **'listEtagere'** comme nous voyons dans le code au-dessus.

Cette liste contenant les données plus récentes de la base sera affichée par la couche présentation à chaque rafraichissement de la page en réponse http.

7.4 Interface Utilisateur et JSP de la couche présentation

Cette partie contient le code HTML qui permet de récupérer les paramètres provenant de l'utilisateur et au même temps elle permet l'affichage en réponse apres le traitement de données des couches bas niveau, pour cela j'ai utilisé notamment la balise FORM comme nous voyons dans le code ci-dessous :

```
<td class="tde">${tm[0]}</td>
<td class="${model.listNbrP[0]==0 ? 'td_rempli' :'td_dispo'}"><c:out value="${model.listNbrP[0]}"/> </td>
<td>
<form action=control  method="post">
<input type="hidden" value="${hm[0]}"  name="nomTag">
<button type="submit" value ="incP" name="actionTag">PLUS+</button>
</form>
</td>
<td>
<form action=control  method="post">
<input type="hidden" value="${hm[0] }"  name="nomTag">
<button type="submit" value ="decP" name="actionTag">MOINS-</button>
</form>
```

Fig14: balises HTML utilisé pour générer la couche présentation

L'élément HTML FORM est un des éléments indispensables dans la création des pages web notamment car il permet d'utiliser les balises auxiliaires input qui eux-mêmes peuvent stocker différents paramètres à transmettre vers le serveur à travers d' une requête Http POST ou GET, ainsi, j'ai utilisé cette caractéristique des balises input pour joindre plusieurs paramètres qui seront traiter par l'application web afin de modifier et agir sur la base de données selon les besoins de l'utilisateur après la validation du formulaire.

A cet égard l'interface utilisateur permet d'afficher les données mise à jour par l'application à travers une interface HTML comme nous voyons dans l'image ci-dessous.

Section 2

Etagere	Palette	AugP	DecrP	Vide	AugV	DecrV
X0023 - X0029 X0523 - X0529	0	PLUS+	MOINS-	0	PLUS+	MOINS-
X0033 - X0040 X0633 - X0640	0	PLUS+	MOINS-	0	PLUS+	MOINS-
X0041 - X0048 X0641 - X0648	0	PLUS+	MOINS-	0	PLUS+	MOINS-
Y0001 - Y0008 Y0601 - Y0608	0	PLUS+	MOINS-	0	PLUS+	MOINS-

Fig15 : Extrait de l'interface utilisateur développée en code HTML

L'information à joindre aux balises input lors de l'envoie du formulaire qui permettra d'agir sur la base de données peut être visualisée dans le code HTML, ses composants principaux sont les suivants :

- **nomTag:** balise input qui contient la variable hm[0] , celle-ci représente la chaine de caractères correspondant à la **référence dans la base de données .** J'ai décidé de créer une liste globale **hm** qui contient le nom de toutes les références en format chaine de caractères, cela m'a permis de gagner en souplesse et temps de programmation lors de la création de la page JSP ,en effet, il suffit de modifier un seul élément de la liste au lieu de modifier tous les emplacements dans lesquels j'ai utilisé la référence demandé.
- **actionTag:** cette balise input contient le type d'action à exécuter par la couche métier lors de l'envoie du formulaire, ainsi nous aurons 4 actions possibles à exécuter soit incrémentation/décrémentation de palette ou bien incrémentation/décrémentation des emplacements vides.

- **Buttons PLUS+ MOINS- :** ces boutons seront de type SUBMIT et ils servent à envoyer le formulaire respectif avec les paramètres associés lors de son activation.
- **Attribut action:** il permet de diriger la requête vers la servlet ou page jsp correspondant pour son traitement.
- **Attribut méthode :** il permet de choisir le type de requête http : POST, GET, PUT à générer lors de l'envoie du formulaire.

Finalement afin d'améliorer l'application nous avons ajouté deux boîtes de texte supplémentaires, la première sert à afficher la **date** de la dernière modification de la base de données après calcul, cela sert notamment pour que le personnel soit informé des dernières mise à jour des étagères et qu'ils puissent prend des décisions plus précises concernant la gestion des emplacements .

Et la deuxième il s'agit d'une boite **de message** qui permet l'enregistrement du texte dans une variable portée requête sur un bean(classe java). Celui-ci sera visible à tous les clients connectés avec le serveur étant donné qu'il ne soit pas arrêté, il faut noter qu' à différence de données persistantes qui sont enregistrées sur la base, les variables stockées dans le bean resteront disponibles pour l'utilisation par l'application web exécuté par le serveur tomcat.

Fig16 : boit de texte HTML utilisées pour aider au personnel dans gestion des emplacements du magasin.

8 Recommandations et Conclusions du projet 1

8.1 Recommandation

Comme dans tout projet qui est arrivé à sa mise en production, il faudra suivre un chemin d'amélioration continue (méthodologie de production Japonaise Kaizen) afin de adapter et ajouter des caractéristiques que n'ont pas été évidentes dans la conception, dans notre cas nous présentons les recommandations suivantes:

Au niveau de sécurité il faudra installer un système pare-feu sur le serveur de production si jamais il est connecté à un réseau externe relié à internet ou bien il faudra prendre soin de maintenir le serveur dans le réseau local de l'entreprise afin de prévenir des attaques de « déni de services » ces attaques informatiques visent à empêcher la capacité du serveur de rendre service aux utilisateurs, il en existe ainsi de deux types :

- **Par saturation** qui consiste à envoyer de requêtes HTTP de manière automatique, massive et continue pour empêcher au serveur de répondre aux clients qui ont vraiment un besoin réel.
- **Par exploitation de vulnérabilités,** qui cherche à exploiter une faille pendant le développement de l' application ou défaillance du serveur web utilisé pour hébergé l'application et de cette manière rendre la machine ou l'application inutilisable avec des logiciels ou outils de piratage .

8.2 Conclusions

Pour conclure je précise quelques points à prendre en considération afin d'améliorer l'application , à ce sujet étant donné que la saisie de données s'avère manuelle il faudra synchroniser la mise à jour de l'information entre l'équipe qui gère les colis de produit et l'équipe qui fait le dépôt de palettes

selon la procédure proposée en annexes dans la notice technique développée, cela afin d'obtenir des données réelles et fiables qui puissent être d'utilité pour le personnel .

Une deuxième approche pour résoudre ce problème serait de restructurer toute l'application pour réaliser une saisie automatique en utilisant différents événements technologiques notamment une interface **Capteur –PLC- Application Web –Base de données** qui permettront ainsi d'avoir l'information relative aux emplacements libres en temps réel.

9 Contexte du Projet 2 : gestion des indicateurs de productivité ERP.

Dans cette partie nous allons aborder les différentes activités réalisées pour le développement d'une deuxième application dans le département de méthodes et production, la réalisation de ce projet demande une étude plus approfondie du système à numériser ainsi que l'utilisation des autres outils informatiques que nous verrons dans le développement de ce rapport

Objectif :
L'objectif de cette mission sera de concevoir une application pour mesurer et enregistrer différents paramètres de productivité de façon persistante, les paramètres ainsi trouvés permettront de calculer le niveau de performance d'un ensemble de machines qui fonctionnent en production, mais dans laquelle la saisie est manuelle.

Finalement le système doit permettre de visualiser l'information de productivité selon un temps défini par le responsable de l'atelier (1 Jour- 30Jours) et avec les indicateurs de productivité les plus représentatives de la machine.

Points à prévoir :
Afin d'implémenter cette application il faudra considérer différents aspects au niveau logiciel, hardware, réseau Ethernet et procédures opérationnels, dans la suite nous présenterons les aspects suivants :

1. Choisir quels seront les paramètres à enregistrer dans la base de données (Déperditions, TRS, TRG, arrêts fonctionnels propres ou induits, arrêts structurelles, taux de performance, etc.).
2. Définir le modèle à implémenter pour le développement de l'application.
   - Choisir la façon d'écrire les données sur la base de données en utilisant une application web(Formulaire) ou un automate programmable, après avoir discuté la faisabilité technique de chaque solution avec mon responsable .
   - Choisir la façon d'afficher les données vers l'utilisateur en mode lecture à travers d'une interface graphique Web ainsi que la structure du mode d'écriture de paramètres
3. Définir la base du temps à utiliser pour calculer les paramètres de production ainsi que la manier de gérer le temps dans l'application au niveau de l'interface utilisateur et le traitement de données.
4. Définir la structure de la base de données (tableaux à créer pour le stockage de paramètres de façon persistante).

9.1 Calcul de TRS et indicateur de productivité selon la Norme NF E 60-182

L'indicateur le plus représentatif afin de connaitre le niveau de productivité et utilisation de ressources d'une machine est la TRS ou bien le taux de rendement synthétique, celui-ci il est composé de 3 sous-indicateurs qui permettent obtenir de manière plus précise et riche l'information sur l'état d'une machine, dans la suite nous les présentons en détail :

- **Taux de performance Tp (Tn/Tf) :** permet de connaitre la performance de la machine face au micro arrêts et écarts de cadence.
- **Taux de disponibilité Do (Tf/Tr) :** permet de connaitre la performance de la machine par rapport aux arrêts structurels propres et induits lors de son cycle de production.
- **Taux de Qualité Tq (Tu/Tn):** permet d'identifier la performance par rapport au nombre de pièces bonnes ou rebutées qui ont été produit par la machine.

Afin de calculer les indicateurs de productivité d'une machine je me suis basé sur la norme NF E 60-182, elle indique les formules à utiliser pour obtenir les indicateurs et sous-indicateurs nécessaires pour calculer le TRS, le but du projet sera ainsi de calculer par moyen de l'application web l'ensemble de ces indicateurs et les enregistrer de manière persistante.

Dans la suite nous voyons le diagramme du temps de productivité qui permet de visualiser les interactions entre les différents évènements de panne, cadence et arrêt qui interviennent dans un cycle de travail, nous pouvons voir aussi les formules des indicateurs de productivité à implémenter par notre application .

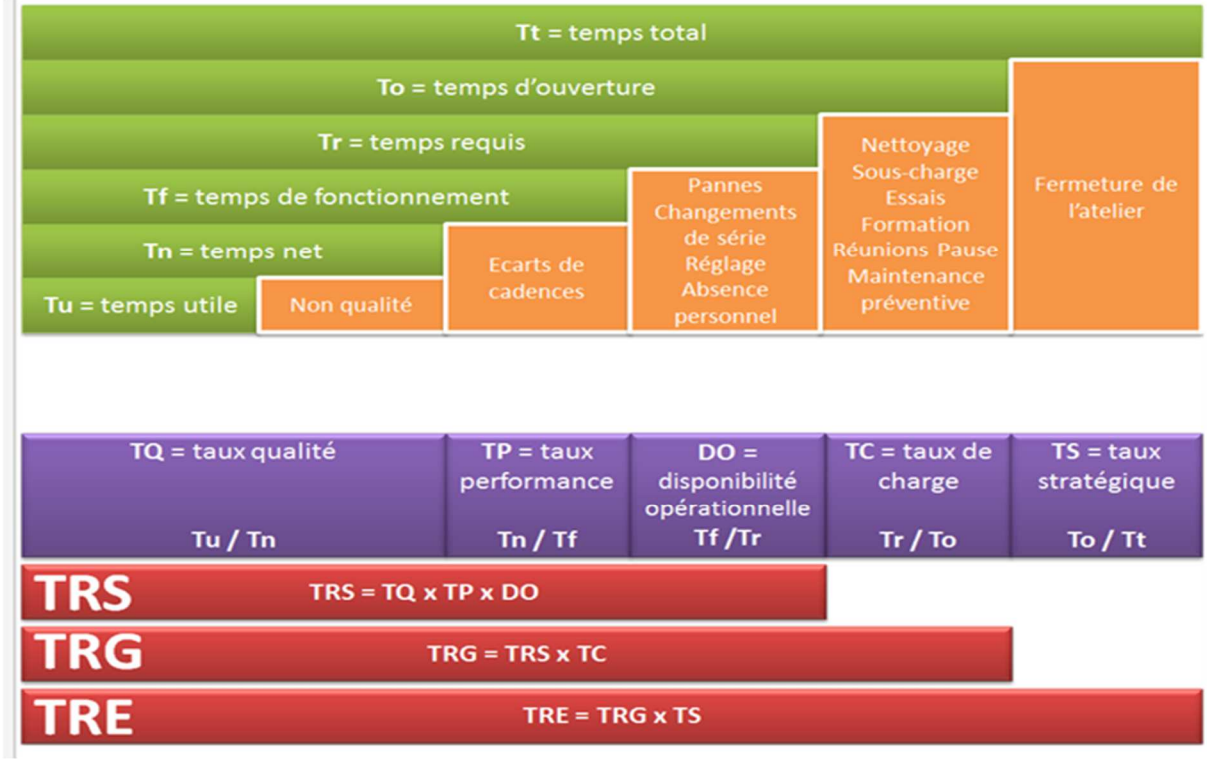


Fig17 : Diagramme du temps des indicateurs de productivité

9.2 Architecture du Software et Hardware de l'application

Afin de répondre au cahier de charges nous avons décidé de diviser le système en deux phases, la première phase sera une application web en écriture qui permettra de saisir les paramètres de production envoyés par l'opérateur à travers d'un formulaire pour son prétraitement par l'application web, ensuite les données seront envoyées vers une base de données pour le calcul des indicateurs de productivité selon les contraintes de temps et références du cahier de charges.
Cette application doit permettre le traitement de données au niveau de la couche applicative ainsi que de la base de données afin de pouvoir générer des tableaux résultants contenant les indicateurs de productivité recherché.

Phase ECRITURE :

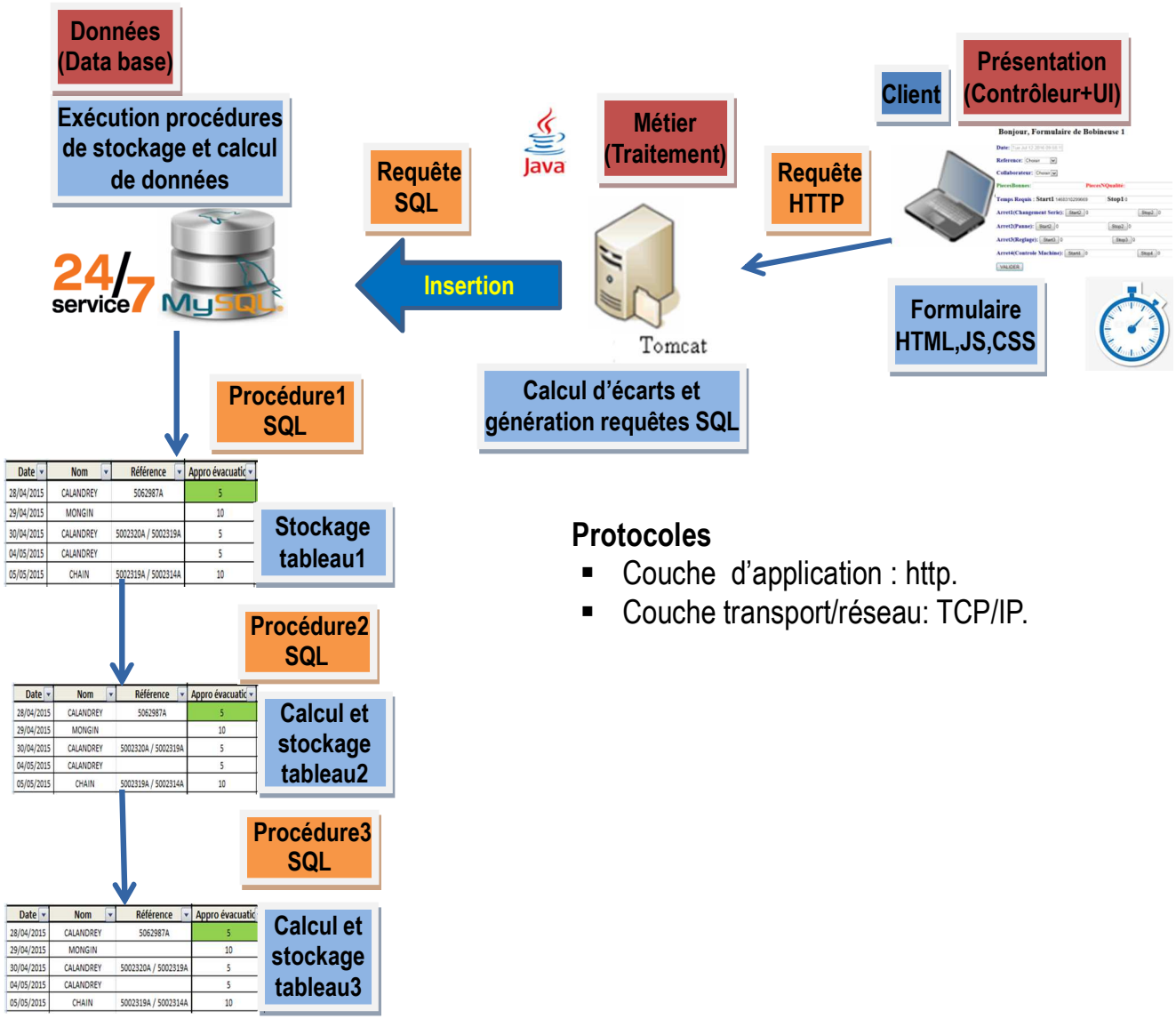


Fig18: Structure de l'application en phase d'écriture

La deuxième phase sera concernant la lecture de paramètres, elle sera composée d'une **application web** à travers laquelle le client se connectera à la base de données, celle-ci composée des sous-tableaux contenant les indicateurs de productivité.

Les tableaux qui contiendront déjà l'information pré-calculée serviront comme source pour afficher l'information selon les dates requises par l'utilisateur de l'application web.
La structure à implémenter sera basée sur le modèle 3 tiers qui permet de diviser l'application en 3 couches interdépendantes, soit couche de données, métier et présentation comme nous verrons dans la suite de ce rapport

Il faut préciser que cette dernière phase ne sera pas abordée dans le stage afin de pouvoir compléter et garantir une gestion du temps efficace pour compléter l'application d'écriture qui demandera beaucoup plus de connaissances technologiques et organisation de classes java et procédures SQL .

Phase LECTURE :

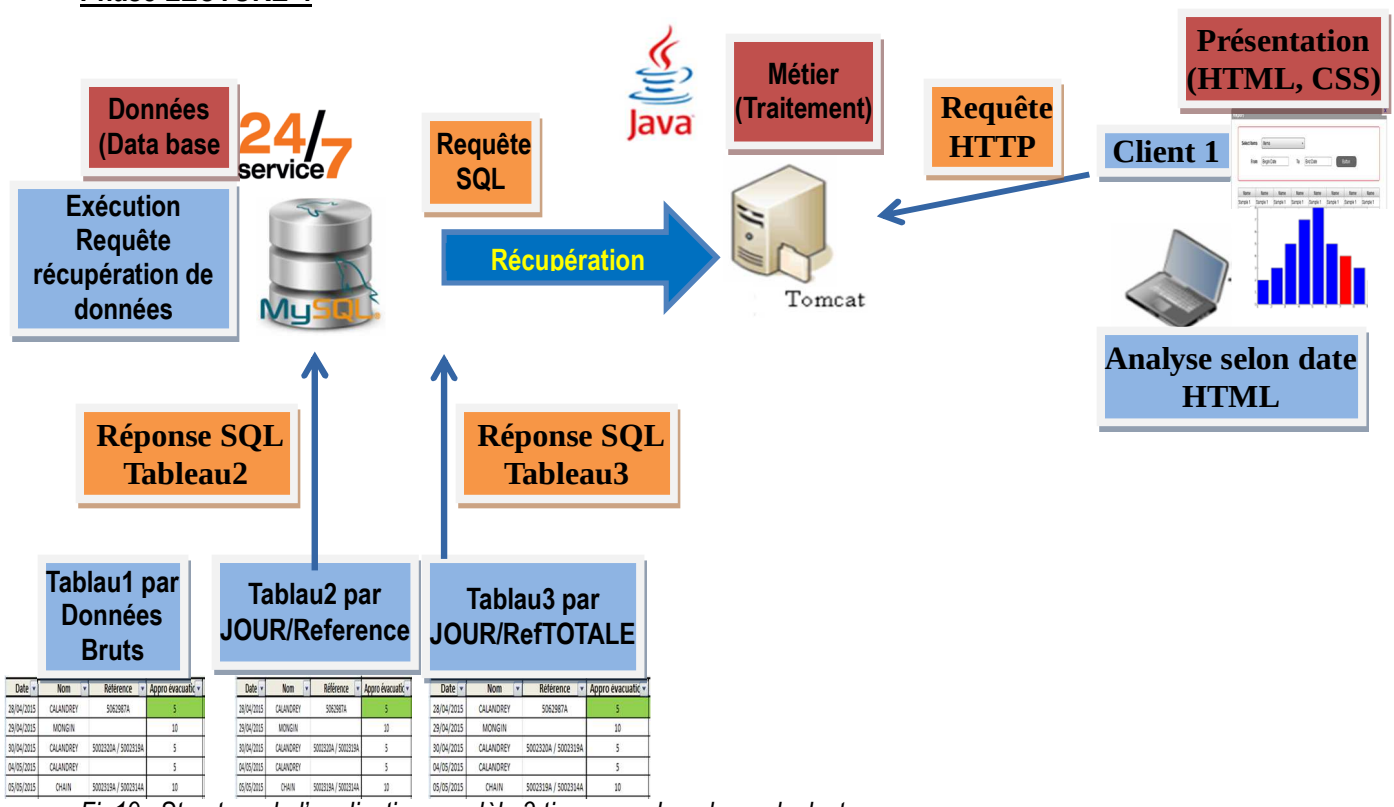


Fig19 : Structure de l'application modèle 3 tiers pour la phase de lecture.

9.3 Structure logiciel de l'application en java

Afin de réaliser l'application nous avons utilisé le model 3 tiers, celui-ci permet de diviser l'application web en 3 étapes indépendants qui interagissent entre eux selon des méthodes java que nous devrons codé , dans la suite nous présentons la structure de l'application Java EE.

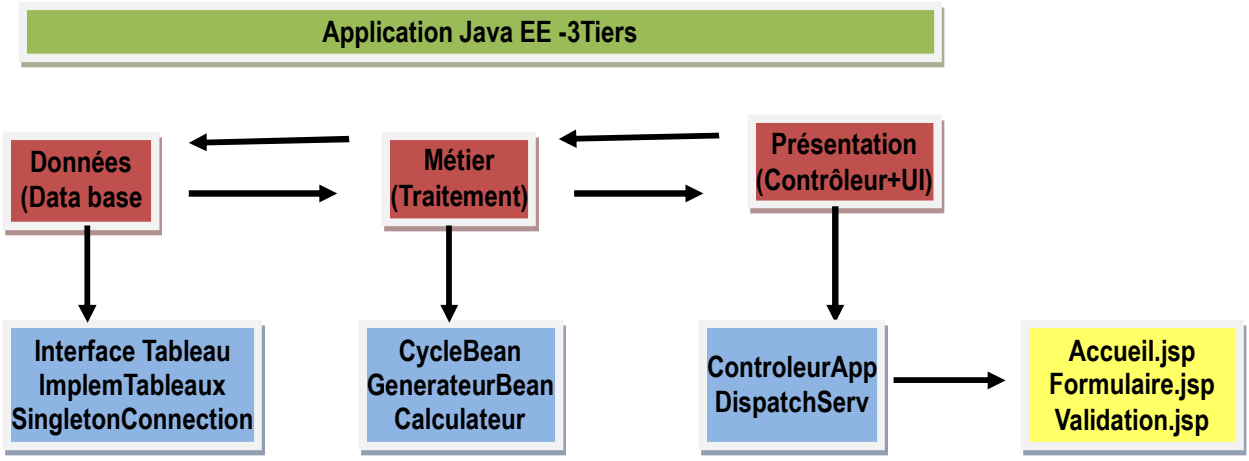


Fig. 20: Application Java EE avec une structure à 3 Tiers.

Dans la suite je présenterai les fonctions que chacun de ces parties devras exécuter lors de la mise en production sur le serveur.

Couche présentation : cette couche contiendra les classes nécessaires pour traiter la **requête http** provenant du client, elle est composé d'un **ensemble des pages JSP** qui servent comme l'interface utilisateur et qui seront visualisées sur le navigateur en format HTML , elle est composée aussi des

servlets de contrôle qui sont des classes Java permettant de rediriger les requêtes http vers les pages JSP correspondants .

Egalement, étant donné que le servlet doit pouvoir interagir avec le navigateur client, elles doivent hériter de la classe mère HttpServlet qui contient les méthodes pour traiter ce type requête.
Les classes dans cette couche permettront ainsi de réorienter la requeté http vers la page html correspondant au cycle de travail, c.à.d. les étapes que l'utilisateur devra suivre afin d'insérer l'information de productivité de la machine .

Couche métier : cette couche permet le traitement de paramètres de la requête http reçu par la couche présentation, il contient les classes pour calculer les écarts du temps de production ainsi que les arrêts globaux du cycle de travail, pour pouvoir réussir à cette fonction elle doit adapter les données à un format reconnu pour la base de données lors de son stockage .
La couche métier sera à charge de générer des objets (bean) avec l'information en format correcte qui sera utilisé pour la couche bas niveau ainsi que pour la couche présentation.

Couche données : cette couche permet le traitement du Bean généré par la couche métier pour pouvoir l'enregistrer dans la base de données, il contient les classes nécessaires pour appeler les procédures SQL qui créent les tableaux respectifs des indicateur de productivité, en plus elle contient les méthodes pour gérer la connexion et l'envoi des requêtes SQL en utilisant l'implémentation de la librairie JDBC(Java Dat base connectivité).

9.3.1 Couche métier de l'application web

Entre les classes qui comportent la couche métier nous présentons les suivantes :
Calculateur.java : Ensemble de fonctions qui permettent d'effectuer le traitement de données provenant de couches supérieures, par exemple calculer l'écart du temps entre deux valeurs, conversion de temps Unix vers un format différent, formater une date selon un modèle SQL, modifier une chaine de caractères.

GenerateurBean.java : c'est une classe qui permet le traitement de l'ensemblede paramètres provenant de la requête http pour son enregistrement dans l'objet **CycleBean,** celui-ci représentera les données du tableau1 dans la base MySQL. Il faut préciser que cette classe utilisera les fonctions de la sous classe Calculateur afin d'accomplir sa mission.

CycleBean.java : c'est un bean contenant l'ensemble de données à enregistrer sur le tableau1 de la base de données, en outre le format de ces attributs correspond au type de variables qui comportent la base de données, c'est grâce à cette similitude que les requêtes SQL de la couche de données sont plus simples à implémenter.

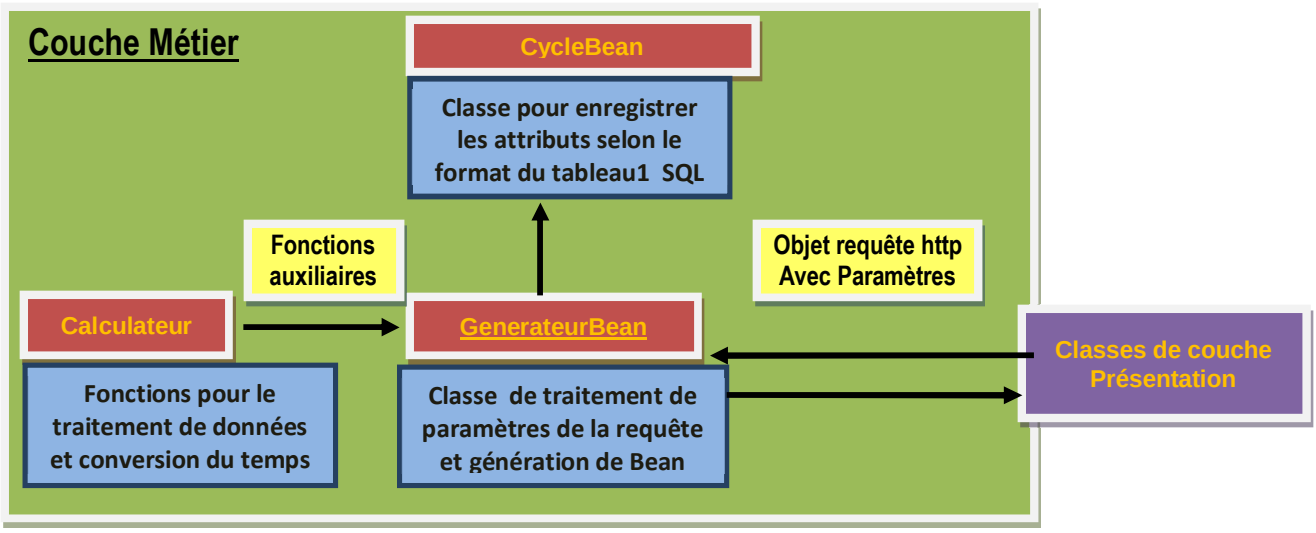


Fig.21: Structure de la couche métier et son interaction avec les autres couches dans l'application

9.4 Base de données MySQL (couche de données)

La base de données conçue pour cette application possède une structure hiérarchique à 3 tableaux, en ce sens les données d'un tableau dépendront d'un seul tableau parent, dans la suite nous présentons la structure à développer en utilisant la dernière version du logiciel MySQL.

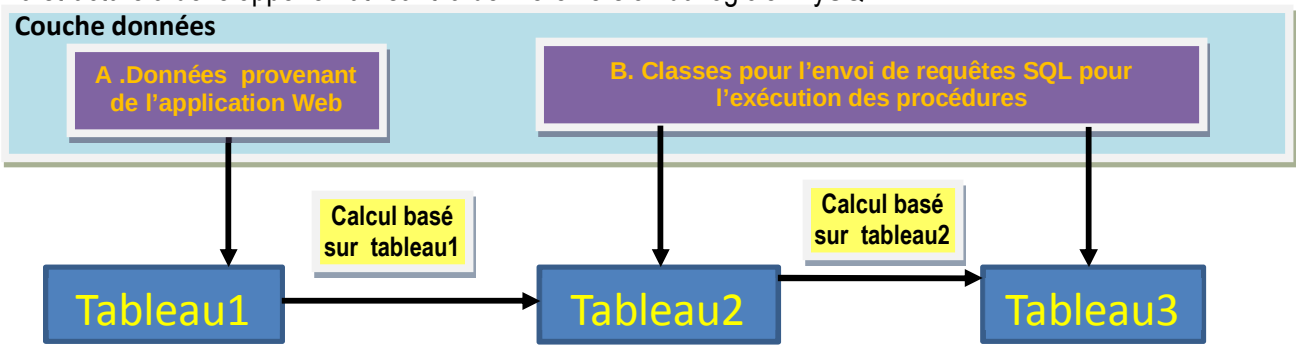


Fig22: Structure de la base de données de production et son interaction avec la couche de données.

D'après le graphe nous observons que dans un premier temps les données brutes proviendront de l'application web sur java, ces données contenant l'information du cycle de travail seront enregistrées dans le tableau1 qui sert de référence pour générer les 2 autres sous tableaux , en outre la couche de données fournit les requêtes SQL nécessaires pour exécuter les procédures automatisées qui permettront de créer les sous tableau 2 et 3, ce système aura les avantages suivants :

- **Stocker un nombre important de paramètres de productivité qui ne sont pas forcément dépendant entre eux**, (par exemple les cycles de travail par référence produit ou bien le nom d'opérateur et date de cycle de travail, etc.) dans un **seul lieu centralisé et persistant** qui pourra être utilisé pour générer des tableaux plus spécifiques selon le besoin.

- **L'amélioration du temps de traitement des requêtes en mode écriture** : Pour cela j'ai décidé de créer des procédures automatisées SQL qui sont capables de remplir le tableau 2 ou 3 directement depuis le serveur de données au lieu de faire les calculs dans l'application web elle-même, en utilisant des requêtes successives de connexion et déconnexion.

- **l'amélioration du temps de réponse en mode lecture** : puisque nous aurons des tableaux avec l'information déjà calculée très précis sur les indicateurs de productivité, par ailleurs

l'application web externe devra uniquement utiliser des requêtes SQL simples en lecture au lieu de recalculer les indicateurs à chaque fois qu'on veut connaitre les données pour une date spécifique, en utilisant de requêtes SQL complexes.

9.4.1 Tableau1 : donnés bruts de l'application Web

Le tableau1 forme partie intégrale de la base de production, il permet de récupérer l'information provenant de l'application web sur Java à chaque cycle de travail, parmi les paramètres que nous allons stocker nous présentons notamment :

- **Date :** date d'enregistrement du cycle de travail sur la machine.
- **Référence de travail :** type de référence travaillé sur la machine
- **Nom d'opérateur :** nom de la personne qu'a travaillé sur la machine
- **Le temps requis :** la durée totale en minutes de son cycle de travail
- **Arrêts propres (4 causes) :** la durée totale en minutes de différents types d'arrêts lors du cycle de travail.
- **Nombre de pièces bonnes** : le total de pièces produit de bonne qualité pendant le cycle de travail.
- **Nombre de pièces rebutées** : le total de pièces produit de mauvaise qualité pendant le cycle de travail.
- **Temps Net de Travail :** temps pendant lequel la machine travaille à son temps de cycle nominal
- **Temps utile de Travail :** temps pendant lequel la machine travaille à son temps de cycle nominal en produisant uniquement pièces bonnes.

Ces paramètres serviront donc de base pour calculer les indicateurs de productivité du tableau2

```
mysql> SELECT * FROM tableau1_a;
+----+---------------------+-----------+--------+--------+---------+---------+---------+---------+----+----+---------+---------+
| id | datetimes           | reference | nom    | tr     | arret1  | arret2  | arret3  | arret4  | nb | nr | tnet    | tu      |
+----+---------------------+-----------+--------+--------+---------+---------+---------+---------+----+----+---------+---------+
| 43 | 2016-07-16 21:56:44 | ref1      | Carlos | 0.3991 | 0.0277  | 0.0194  | 0.0000  | 0.0000  | 40 |  2 | 40.3200 | 38.4000 |
| 44 | 2016-07-16 22:29:42 | ref1      | Louis  | 0.3437 | 0.0133  | 0.0000  | 0.0000  | 0.0000  | 40 | 10 | 48.0000 | 38.4000 |
| 45 | 2016-07-16 22:34:35 | ref2      | Marie  | 0.2137 | 0.0000  | 0.0000  | 0.0000  | 0.0000  | 50 |  3 | 54.0600 | 51.0000 |
| 46 | 2016-07-16 22:34:59 | ref2      | Louis  | 0.2314 | 0.0000  | 0.0000  | 0.0000  | 0.0000  | 40 |  1 | 41.8200 | 40.8000 |
| 47 | 2016-07-16 22:35:26 | ref3      | Marie  | 0.2476 | 0.0124  | 0.0000  | 0.0136  | 0.0111  | 50 |  2 | 54.6000 | 52.5000 |
| 48 | 2016-07-16 22:36:32 | ref3      | Marie  | 0.2744 | 0.0132  | 0.0000  | 0.0000  | 0.0000  | 50 |  2 | 54.6000 | 52.5000 |
+----+---------------------+-----------+--------+--------+---------+---------+---------+---------+----+----+---------+---------+
```

Fig20: Structure du tableau1 avec les de données bruts provenant de l'application web

9.4.2 Tableau2 : paramètres et indicateurs de productivité par référence/Jour

Ce tableau permet de stocker les calculs résultants contenant l'information de productivité de la machine, d'autre part il permet le tri des indicateurs puisque il enregistre les données par référence de produit fabriqué, par conséquent tout l'information nécessaire pour réaliser les calculs vient du tableau1 que nous avons mentionné précédemment, en effet nous trouverons un peu près les mêmes données du tableau1 mais avec des paramètres et indicateurs de productivité additionnels ; dans la suite nous les présentons :

- **Nombre total de pièces (bonnes et rebutées) :** Quantité de pièces totale produit par référence pendant toute la journée de travail. (plusieurs cycles)
- **Temps total d'arrêt :** Temps total d'arrêt que la machine a expérimenté pendant toute la journée de travail et trié par type la cause de l'évènement.
- **Temps de fonctionnement** : Temps qu'on obtient après soustraire le temps total d'arrêt du temps total requis pour une journée complète.
- **Taux de Disponibilité** : permet de connaitre l'impact des arrêts structurels dans la performance de la machine
- **Taux de performance :** permet de connaitre l'impact d'une basse de cadence dans la performance de la machine

- **Taux de qualité :** permet de connaitre l'impact du rapport entre les pièces bonnes et mauvaises dans la performance de la machine
- **Le TRS de la machine par référence :** indicateur de productivité composé par différents taux de performance pour donner une valeur globale de la bonne utilisation de ressources d'une machine.

```
sql> SELECT * FROM tableau2_a;
+-------+------------+--------------+---------+-----------+-----------+-----------+-----------+--------+--------+----------+----------+----------+----------+----------+----------+----------+----------+
| t2_id | t2_date    | t2_reference | t2_tr   | t2_arret1 | t2_arret2 | t2_arret3 | t2_arret4 | t2_nb  | t2_nr  | t2_tnet  | t2_tu    | t2_nt    | t2_ta    | t2_tf    | t2_do    | t2_tp    | t2_tq    | t2_trs   |
+-------+------------+--------------+---------+-----------+-----------+-----------+-----------+--------+--------+----------+----------+----------+----------+----------+----------+----------+----------+----------+
|  10   | 2016-07-16 | ref1         | 0.7428  |  0.0410   |  0.0194   |  0.0000   |  0.0000   |   80   |   12   | 88.3200  | 76.8000  |    92    | 0.0604   | 0.6824   | 0.9187   | 129.4256 | 0.8696   | 103.39   |
|  11   | 2016-07-16 | ref2         | 0.4451  |  0.0200   |  0.0194   |  0.0000   |  0.0000   |   96   |   4    | 95.9839  | 91.8800  |    94    | 0.0587   | 0.2451   | 1.0800   | 210.4423 | 0.9574   | 206.23   |
|  12   | 2016-07-16 | ref3         | 0.5220  |  0.0256   |  0.0000   |  0.0136   |  0.0111   |  100   |   4    | 109.2000 | 105.0000 |   104    | 0.0503   | 0.4717   | 0.9036   | 231.5031 | 0.9615   | 201.13   |
+-------+------------+--------------+---------+-----------+-----------+-----------+-----------+--------+--------+----------+----------+----------+----------+----------+----------+----------+----------+----------+
```

9.4.3 Tableau3 : paramètres et indicateurs de productivité totale /jour

Ce tableau sert à calculer la somme de tous les paramètres de productivité pendant la journée de travail à différence du tableau précèdent nous allons utiliser les données de toutes les références de produit que la machine à produit pendant la journée, ensuite nous allons calculer les indicateurs de productivité comme par exemple le taux de disponibilité, taux de performance, taux de qualité et TRS totaux pendant la journée de travail .

```
sql> SELECT * FROM tableau3_a;
+-------+------------+--------------+---------+-----------+-----------+-----------+-----------+--------+--------+----------+----------+----------+----------+----------+----------+----------+----------+----------+
| t3_id | t3_date    | t3_reference | t3_tr   | t3_arret1 | t3_arret2 | t3_arret3 | t3_arret4 | t3_nb  | t3_nr  | t3_tnet  | t3_tu    | t3_nt    | t3_ta    | t3_tf    | t3_do    | t3_tp    | t3_tq    | t3_trs   |
+-------+------------+--------------+---------+-----------+-----------+-----------+-----------+--------+--------+----------+----------+----------+----------+----------+----------+----------+----------+----------+
|   1   | 2016-07-16 | ref_total    | 1.7099  |  0.0666   |  0.0194   |  0.0136   |  0.0111   |  270   |   20   | 293.4000 | 273.6000 |   290    | 0.1107   | 1.5992   | 0.9353   | 183.4667 | 0.9325   | 160.01   |
+-------+------------+--------------+---------+-----------+-----------+-----------+-----------+--------+--------+----------+----------+----------+----------+----------+----------+----------+----------+----------+
```

Fig 22: Structure du tableau3 utilisé pour l'enregistrement des paramètres et indicateurs de productivité calculés par total de toutes les références

9.5 Interface Utilisateur et JSP (couche présentation)

L'interface utilisateur est composée de 3 pages JSP qui seront transformées en servlets lors du traitement de la requête http selon le **flux de travail**, ainsi ,**l'application transformera le code source de la page jsp en servlet** de code **source .java** ,celle-ci sera ensuite compilée pour produire **un fichier . class exécutable par la JVM** , une fois ces procédures finies, la servlet ainsi compilé (qui représente la page jsp original) réalisera le traitement respectif de données (avec les classes auxiliaires de l'application web) et une fois le calculs faits, **l'application exécutera la servlet déjà compilé à travers la JVM qui créera une page HTML statique** pour envoyer en réponse au navigateur du client.

Dans la suite nous présentons le cycle de vie d'un page JSP dans l'application web :

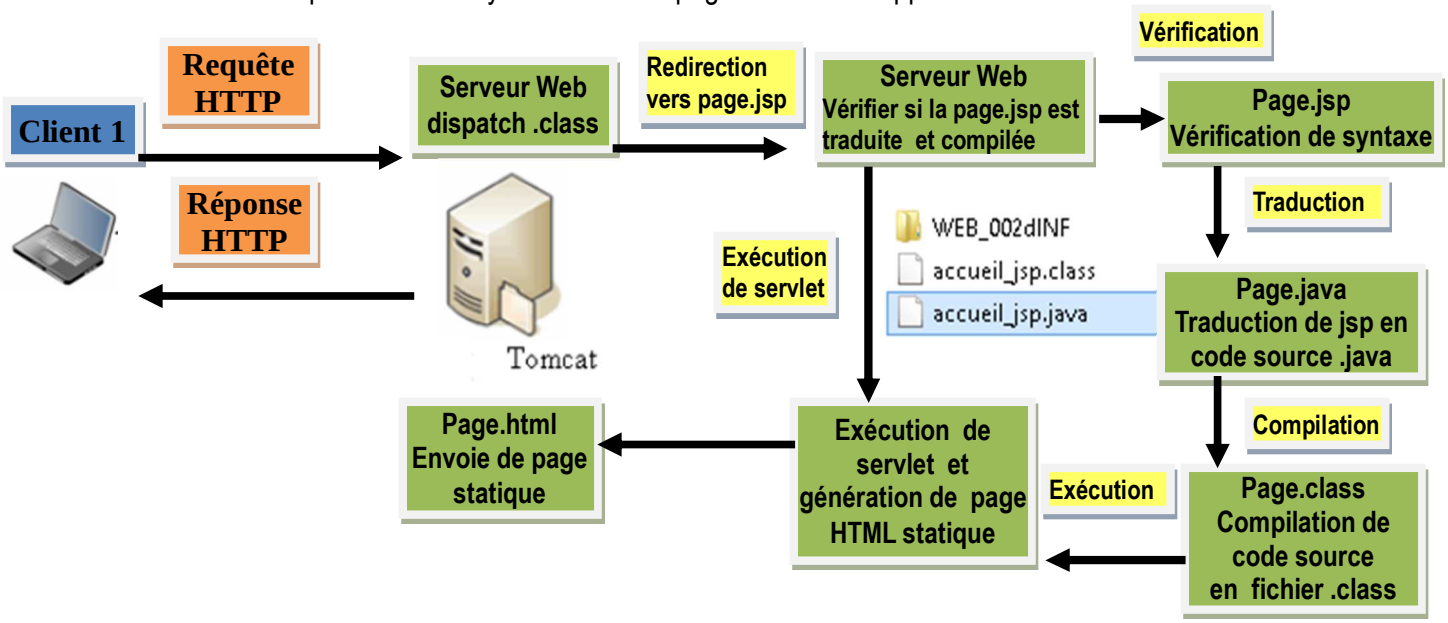


Fig. 23: Structure de pages JSP dans l'application web en interaction avec le contrôleur et classes auxiliaires.

9.6 Flux et insertion de données phase d'écriture

En ce qui concerne l'insertion d'information dans la base de données, l'opérateur en charge d'utiliser l'application devra suivre un flux de travail préétabli qui est composé des étapes suivantes :

Page d'accueil : cette page permet le choix des paramètres préliminaires du cycle de travail de la machine, à savoir, elle sera la première page que l'utilisateur de l'application visualisera dans son navigateur.

Page Formulaire : Dans cette page l'opérateur pourra insérer les données de production à travers un formulaire, elle contient différentes sections, notamment les paramètres de la date, référence , pièces produits, temps requis et temps d'arrêts ,de même, tous les paramètres pourront être envoyés à l'application web pour son traitement postérieur .

Page de validation : cette page permet de montrer les résultats de toutes les données qui ont été enregistrées sur la base de données, elle sert de vérification et confirmation pour savoir si les paramètres de productivité ont été bien calculées .

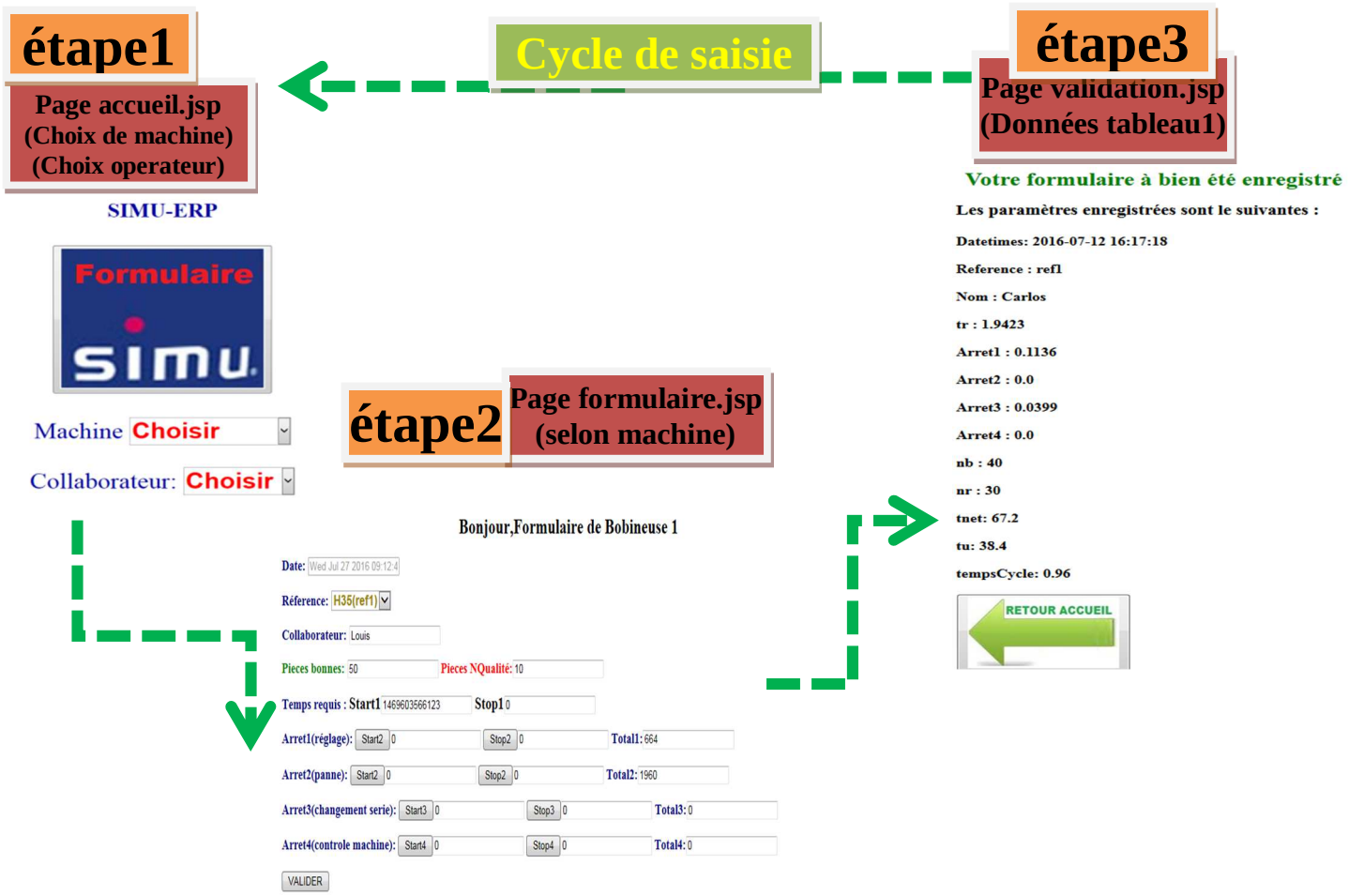


Fig24 : Structure de flux de travail avec les différents pages JSP parcourus par l'utilisateur

Page d'accueil : http://localhost:8080/MVCProductionSIMU/dispatch

Le flux de travail est conçu de façon cyclique afin que l'operateur puisse retourner à la page d'accueil après avoir réussi à insérer les données dans la base de données.

10 Recommandations et Conclusions projet 2

Afin de mettre en exécution ce projet de manière efficace et durable, il faudra prévoir certaines recommandations qui devront être adaptées aux caractéristiques propres de la machine et l'endroit d'utilisation.

10.1 Recommandation

Au niveau de hardware il faudra l'installation d'un ordinateur avec accès au réseau SIMU, cet ordinateur doit être le plus proche de la machine pour prévenir des erreurs dans le calcul du TRS.
Autant dire que cela permettra de réduire le temps que l'opérateur perde en se déplaçant pour remplir le formulaire notamment quand il se produit un arrêt imprévisible lors de son cycle de travail.

Dans le cas qu'une tablette soit utilisée comme dispositif client, il faudra la relier au réseau wifi local afin de pouvoir accéder aux services du serveur de production tomcat qu'exécute l'application de manière continue.

En ce qui concerne la documentation technique, il faudra rédiger des notices techniques, ainsi qu'organiser séances d'information pour renseigner le personnel à charge de la machine sur comment utiliser l'application lors de son travail quotidien en production.

Au niveau de sécurité afin de prévenir les attaques de « déni de service » DoS, il faudra installer un logiciel pare-feu sur le serveur de production, notamment s'il est relié directement à un réseau internet (WAN) extérieur à l'entreprise.

10.2 Conclusion

Ce projet nous a permis de mettre en place un système à 3 tiers pour développer une application web qui permet d'améliorer la saisie de paramètres de productivité.
En ce qui a trait à la technologie, il a servi pour valider l'utilité et versatilité des logiciels libres open sources face aux versions propriétaires plus couteaux au niveau d'argent et temps d'implémentation.
Afin de continuer le développement du système ERP, il faudra établir une équipe de travail pour rendre la maintenance de l'application et au même temps pour faire évoluer le système en ajoutant des modules logiciels qui puissent répondre aux besoins du département de méthodes, la maitrise des outils logiciels comme java , SQL HTML et similaires doit être renforcée dans le département afin de distribuer le travail de façon plus efficace, équitable et avoir une résilience face au bugs ou erreurs de conception.

11 Annexes

11.1 Temps de cycle

Le temps de cycle c'est une constante importent pour tout calcul de performance en production , d'une façon générale ,elle permet de savoir combien du **temps par pièce individuelle** peut produire une machine dans des conditions optimales de travail c.à.d. sans perte de cadence ou sous-performance du au vieillissement de la machine .
La bobineuse 1 modélisé dans l'application comporte 3 temps de cycle différents, en effet, elle est capable de produire 3 tailles de pièces selon son réglage, à ce sujet nous avons utilisé le tableau suivant pour calculer les sous-paramètres de productivité, notamment les constantes en unité de **minutes/pièces**

Bobineuse	id	reference_tempsCycle	dmH	H/piéce	mn/piéce	s/piéce
H35	1	**ref1**	**160**	**0,016**	**0,96**	**57,6**
H40	2	**ref2**	**170**	**0,017**	**1,02**	**61,2**
H55	3	**ref3**	**175**	**0,0175**	**1,05**	**63**
	4	**ref123**	**168,3333333**	**0,016833333**	**1,01**	**60,6**

Fig25 : tableau contenant les constantes de temps du cycle de la bobineuse1

11.2 Composition d'un projet web sur Java EE

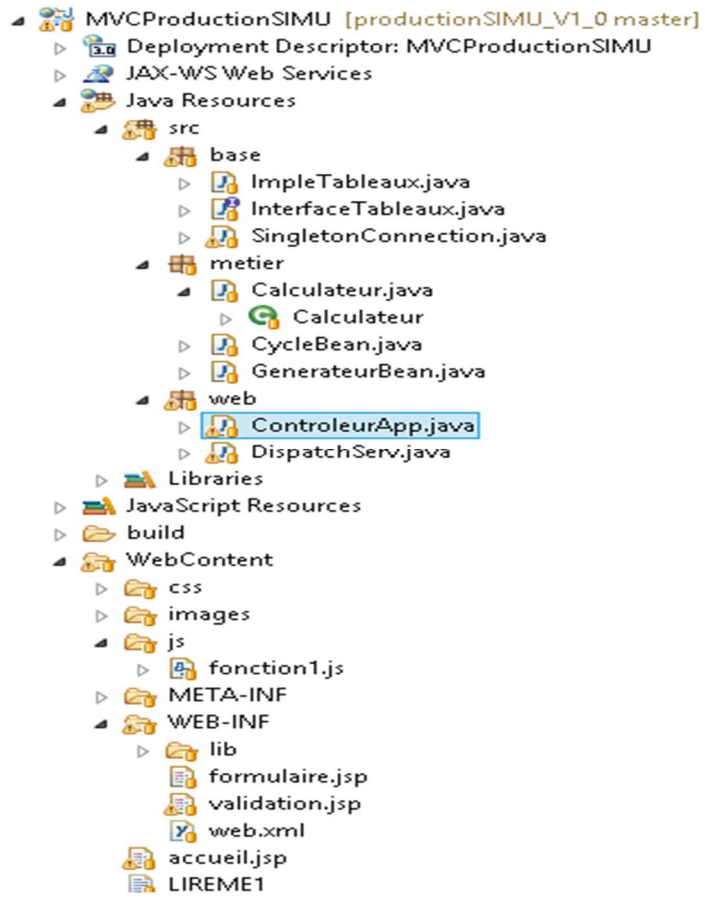
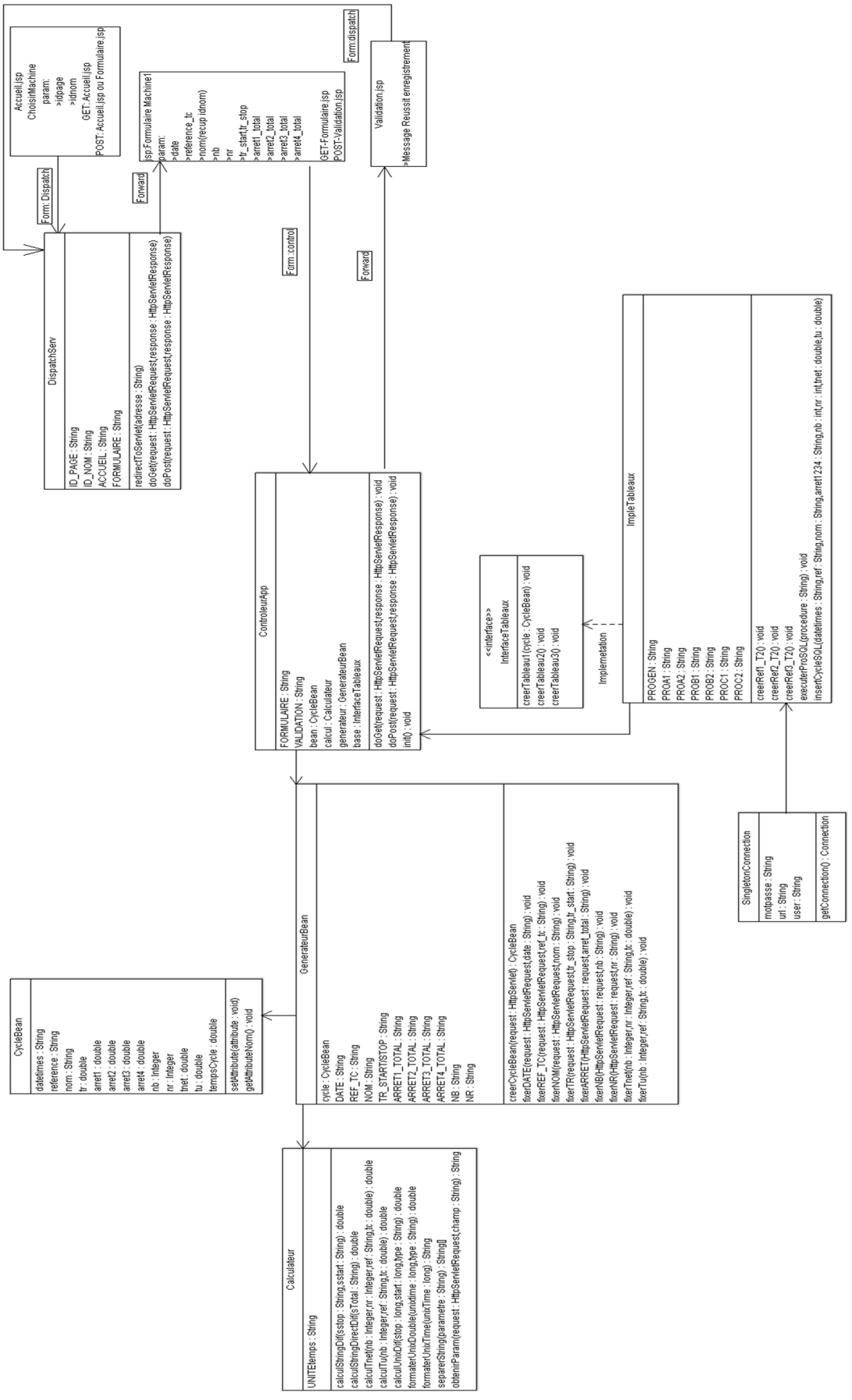


Fig26 : Structure du projet web implémenté pour réaliser le projet 2 (Gestion de productivité)

11.3 Diagramme de Classes de l'application 2 phase d'écriture.



11.4 Extrait du code de la application 2

D'ailleurs je présent un extrait de la classe *controleurApp* appartenant à la **couche présentation** qui permet de récupérer la requête http avec les paramètres envoyés dans le formulaire pour son traitement postérieur.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws Ser
    //Etape0: Instantiation des Objet
    bean=new CycleBean();
    generateur= new GenerateurBean();
    calcul=new Calculateur();
    base= new ImpleTableaux();
    //Etape1: envoie de requete pour generation d' Objet  Bean rempli.
    bean=generateur.creerCycleBean(request);
    //Etape2: enregistrement de Bean recuperé  dans la base de Données  creation tableau1
    base.creerTableau1(bean);
    //Etape3: creation tableau2
    try {
        base.creerTableau2();
    } catch (Exception e) {
        e.printStackTrace();
    }
    //Etape4: creation tableau3
    base.creerTableau3();
    //Etape5 envoi de Bean enregistrer vers page validation
    request.setAttribute("cycle", bean);

    //Forward avec Attribute  vers page VALIDATION
    this.getServletContext().getRequestDispatcher(VALIDATION ).forward(request, response);
    }
}
```

Fig27 : méthode doPost de la classe ControleurApp dans la couche présentation

Conséquemment je présent un extrait de la classe *GénérateurBean* appartenant à **la couche métier** qui permet de traiter les différents paramètres provenant de la requête http pour son enregistrement dans l'objet Bean.

```
//CREER BEAN  du premier ligne du tableau1 Base de données avec tous les types de columns s
public CycleBean   creerCycleBean (HttpServletRequest request){
    this.fixerDATE(request,DATE);
    this.fixerREF_TC(request, REF_TC);
    this.fixerNOM(request, NOM);
    this.fixerTR(request, TR_STOP, TR_START);
    try {
        this.fixerARRET(request, ARRET1_TOTAL);
        this.fixerARRET(request, ARRET2_TOTAL);
        this.fixerARRET(request, ARRET3_TOTAL);
        this.fixerARRET(request, ARRET4_TOTAL);
    } catch (Exception e) {
        e.printStackTrace();
    }
    this.fixerNB(request, NB);
    this.fixerNR(request, NR);
    //POST-CALCUL de indicateurs
    this.fixerTnet(bean.getNb(), bean.getNr(), bean.getReference(), bean.getTempsCycle());
    this.fixerTu(bean.getNb(), bean.getReference(), bean.getTempsCycle());


    return bean;
}
```

Fig28 Méthode implémenté pour générer un objet bean contenant les données du tableau1

12 Bibliographie

- MEDERIC Munier (2013), Créer *des applications web avec Java EE,* France, ISBN: 979-10-90085-32-9
- LAFOSSE Jérôme (2009), *Guide de développement d'applications web en Java*, France, ENI Editions
- GROUSSARD Thierry(2014), Java *8, Les fondamentaux*, ENI Editions
- LANGLET Etienne (2008), Apache Tomcat 6 : Guide d'administration du serveur Java EE sous Windows ou Linux, France, ENI Editions.
- HONDERMARCK(2009), JavaScript : Le guide complet, troisième édition, paris, Micro Application.
- HOHMANN Christian TRS indicateur clé de performance http://christian.hohmann.free.fr/index.php/lean-entreprise/la-boite-a-outils-lean/60-trs-indicateur-cle.
- Calcul d'OEE(TRS) formules et définitions http://www.oee.com/calculating-oee.html