**Université de la Manouba**
**École Nationale des Sciences de l'Informatique**

## Rapport du
## Projet de Conception et de Développement

---

# Sujet : Implémentation d'une application pour la gestion et affectation des environnements aux enseignants

---

*Réalisé par :*

M^elle^Amira Ben Ahmed  M^elle^ Rahma Akaichi

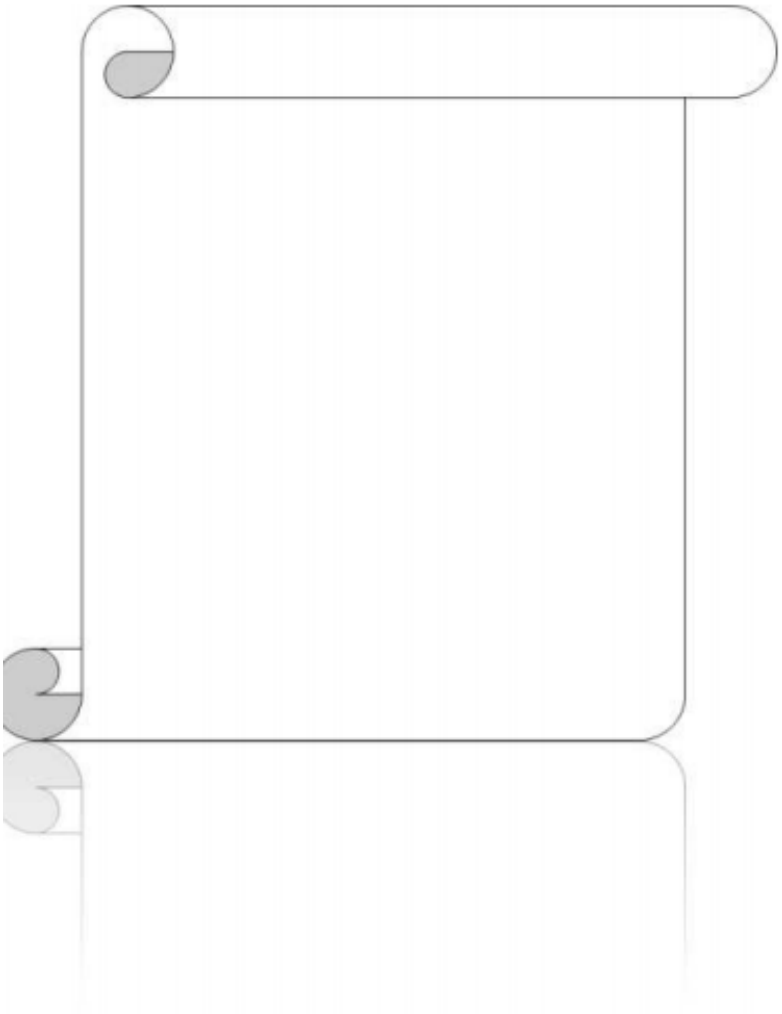
*Encadrante :*

Madame Leila Ben Ayed

---

Année Universitaire :2018 /2019

Signature de l'encadrant.



# Remerciement.

Nous ne pouvons pas laisser passer l'occasion de la présentation de ce projet sans exprimer nos plus vifs remerciements et notre grand respect à tous et à toutes les personnes qui,de près ou de loin, ont bien voulu apporter l'assistance nécessaire au bon déroulement de ce travail, qui présente une phase essentielle dans notre cursus scolaire.

Nous tenons à exprimer toute notre reconnaissance et notre gratitude à notre encadrante Madame Ben Ayed Leila pour son suivi régulier et les précieux conseils qu'elle nous a prodigué afin d'accomplir convenablement notre projet.

Qu'il nous soit permis d'exprimer notre respect et notre remerciement aux membres du jury pour l'honneur qu'ils nous ont fait d'avoir accepté d'examiner et évaluer cette modeste contribution, et à l'administration et tout le corps enseignant de l'ENSI pour la formation qu'il nous a assuré durant ces deux années.

Nos sincères remerciement s'adressent également à tous nos amis de l'ENSI pour leur soutien et encouragement continue.

# Table des matières

# Table des figures

# Introduction générale

# Chapitre 1

# Présentation du projet

## 1.1 Introduction

Avant d'entamer la conception de notre projet gérant l'affectation des environnements aux enseignants de l'ENSI, il est nécessaire d'avoir une vue claire de la démarche à suivre visant à proposer un plus vis-à-vis de l'existant, ajouter d'autres aspects, et améliorer les différentes fonctionnalités. Nous proposons ainsi une étude de l'existant partagée entre la stratégie d'affectation classique, totalement achevée par le responsable de l'administration, et celle qui adopte la solution Informatique. Ceci sera d'une grande utilité lors de la fixation des nouveaux apports à intégrer dans le but d'améliorer les conditions de la procédure d'affectation des environnements aux enseignants de l'ENSI.

## 1.2 Étude de l'existant

Nous avons rencontré le responsable de l'administration chargé de la mission de gestion des salles pour nous renseigner un peu de la procédure classique de distribution des salles vides en cas de rattrapage ou DS, de la méthode de gestion des absences des enseignants , comment se fait la modification des emplois du temps et de la procédure de renseignement des étudiants de toute modification affectant leur emploi du temps actuel. Et ceci dans le but d'avoir une idée sur les améliorations qu'il souhaite apporter.

## 1.3 Description de la procédure classique

A chaque fois qu'un enseignant à l'ENSI souhaite réaliser une séance de cours supplémentaire que ce soit pour faire un rattrapage ou pour passer un devoir surveillé il devrait demander au responsable de l'administration de lui chercher une salle disponible . En se basant sur les informations fournies par l'enseignant à savoir le jour et l'heur, Ce responsable cherche manuellement une salle disponible en consultant tous les emplois du temps de tous les niveaux. En ce qui concerne la gestion des absences des enseignants, ces derniers

envoient un mail à leurs étudiants pour leur informer au paravent. Ensuite, pour renseigner les étudiants de toute modification affectant leurs emplois du temps le responsable de l'administration rattache des notes á l'ENSI qu'ils doivent consulter quotidiennement afin de rester au courant de tout changement.

## 1.4 Problèmes dégagés

La procédure d'affectation d'environnements précédemment décrite s'est bien avérée d'un an à un autre, inefficace et non raisonnable avec une perte de temps pour l'enseignant, l'étudiant et aussi pour le responsable d'administration. Les défauts de la procédure classique se manifestes chaque jour sur le terrain impliquant non seulement un manque de communication entre l'administration et les autres acteurs,un retard dans les cours,un conflit lors de l'affection des salles mais encore des efforts inéluctable fournies par les personnels de l'école responsables du déroulement de ce processus.

## 1.5 Solutions proposées

Notre application est conçue pour pallier tous les problèmes rencontrés tous au long du cursus d'affectation classique. L'idée consiste à automatiser ce processus à partir d'une gestion efficace des données .

## 1.6 Conclusion

L'objectif de ce premier chapitre était principalement d'expliquer et de détailler les notions fondamentales à aborder dans notre sujet . Dans le second chapitre nous allons présenter notre perception d'alternatives dépassant les problèmes rencontrés dans le premier chapitre.

# Chapitre 2

# Analyse et Spécification

**Introduction**

**C**ette phase d'analyse et spécification des besoins est la première phase du cycle de développement du projet au cours de laquelle on va identifier les besoins des utilisateurs. En effet c'est une étape très importante dans la détermination de la qualité et des fonctionnalités qu'offre notre application.

## 2.1 Analyse des besoins

On va déterminer à ce niveau tous les besoins fonctionnels et non fonctionnels que doit répondre cette application.

### 2.1.1 Besoins fonctionnels

L'administrateur doit pouvoir assurer :
- Gérer les comptes.
- Maintenir l'application.
- l'administrateur de l'application peut ajouter d'autres fonctionnalités ou faire des modifications ergonomiques.
- Modifier la base de donnée.

L'enseignant a la possibilité de :
- Créer un compte : Pour se faire l'enseignant est sensé remplir certaines informations (nom,prénom, mot de passe...).
- Marquer son absence en fournissant les informations nécessaires à savoir la date et l'heure.
- Marquer un rattrapage ou un examen en recevant automatiquement un ensemble de suggestions adéquates générées par l'application (ces suggestions seront fixés en indiquant le nom du groupe, la date, l'heure et en recevant la salle disponible à cet heure).
- Demander la permission d'un autre enseignant pour enseigner dans sa salle dans le cas où les suggestions proposées par l'application ne sont pas satisfaisantes ou dans le cas où

il n'y a pas de salle vide.
- Modifier les informations relatives à une séance de cours (heure, salle...).

L' étudiant a la possibilité de :
- Créer un compte : Pour se faire l'étudiant est sensé remplir certaines informations (nom,prénom,niveau,groupe,filière, mot de passe ...).
- Consulter le fil d'actualités.
-Recevoir des notifications contenant les propositions des enseignants et toute modification affectant leurs emplois du temps.
-Confirmer ou rejeter les propositions des enseignants concernant les rattrapages ou les DS(faire un sondage).

### 2.1.2 Besoins non fonctionnels

A part les besoins fondamentaux, notre système doit répondre aux critères suivants :

BNF1 : L'extensibilité et la flexibilité :
Possibilité d'ajouter ou de modifier de nouvelles fonctionnalités. Ainsi l'application ne se limitera pas seulement aux services proposés au début de sa conception.

BNF2 : La sécurité et l'intégrité : L'application devrait avoir l'aptitude à protéger ses données contre les accès non autorisés. Autrement dit les données des utilisateurs devraient être protégés au maximum.
Le taux d'erreur doit être de l'ordre de 10 expo (-6)

BNF3 : Temps de réponse et Rapidité : Une requête nominale du numéro de la salle et de l'heure de la séance supplémentaire doit prendre moins de 5 secondes.

BNF4 : L'Ergonomie : L'application doit présenter une interface graphique simple et facile à manipuler.

## 2.2 Spécification des besoins

A ce niveau , nous allons modéliser notre application à partir des cas d'utilisation qui permettent de donner une vision globale du comportement fonctionnel d'un système et de représenter une unité discrète des interactions entre les utilisateurs(qui sont les acteurs) et le système.

### 2.2.1 Définition des acteurs

On distingue :
-Etudiant :
c'est un acteur qui profitera de quelques services offerts par cette application. En effet, il a accès à toutes les fonctionnalités qui lui sont dédiées.

-Enseignant :

c'est un utilisateur de l'application, il a la possibilité d'accéder directement afin de chercher l'heure et la date pour faire un rattrapage, un examen ou pour informer ses étudiants au préalable de son absence à une date donnée.

-Administrateur :

Est un utilisateur responsable principalement de la mise à jour des données des emplois du temps et a aussi accès total à tous les services offerts par l'application

### 2.2.2 Diagrammes de spécification

#### 2.2.2.1 Diagramme des cas d'utilisation général

Ce Diagramme représente l'ensemble des cas d'utilisations générales de notre application s'appuyant sur les différents fonctionnalités exigés dans notre système ,ainsi que l'interaction entre les acteurs et ces Use Case.
Comme le montre la figure suivante , on trouve 3 types d'acteurs possédant chacun un ensemble de cas d'utilisations qui le relie avec le système :
-L'administrateur
-L'enseignant
-L'étudiant

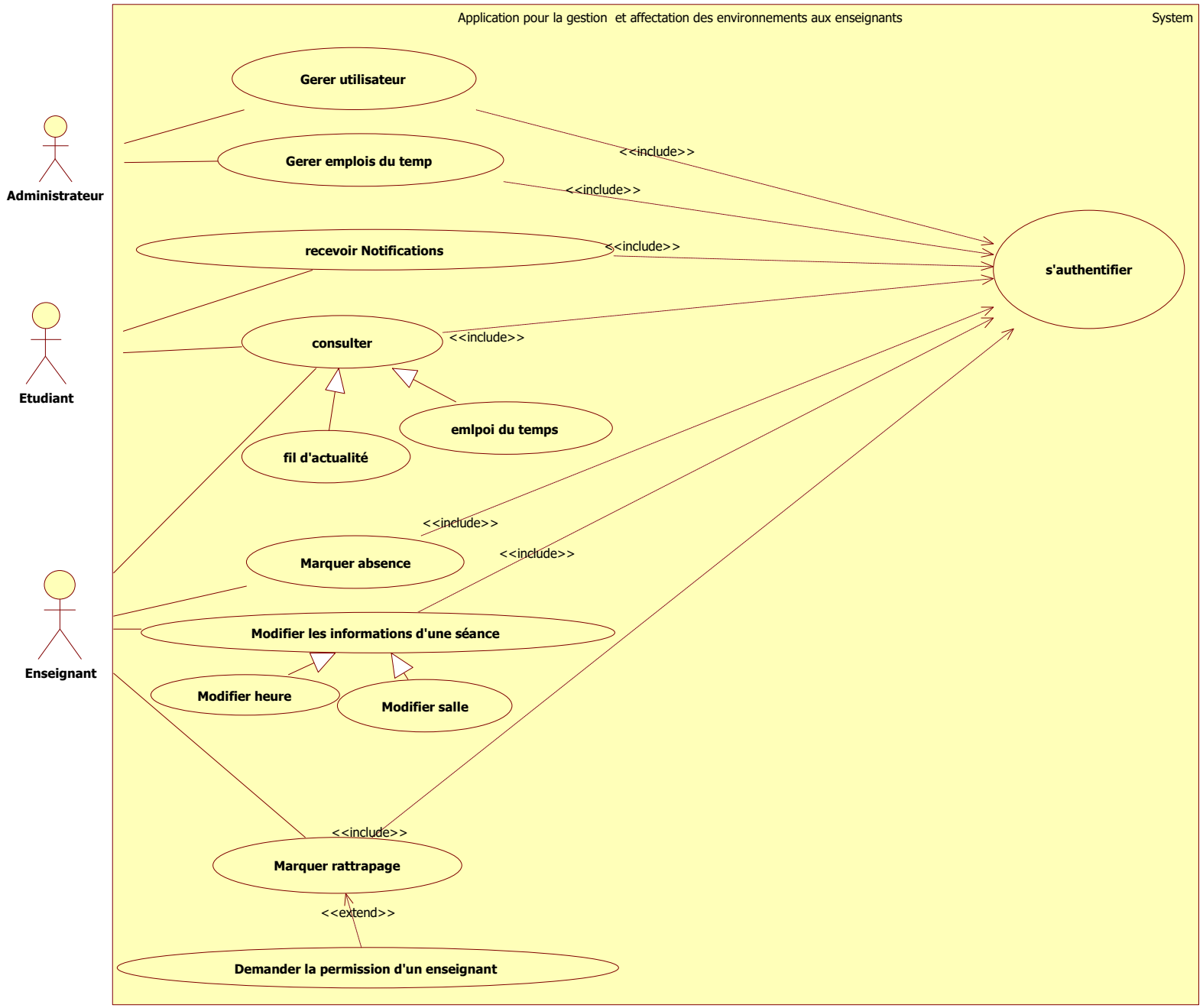



FIGURE 2.1 – Diagramme de cas d'utilisation général

# Chapitre 3

# Conception

**Introduction**

Après avoir analyser et modéliser les besoins,nous allons entamer la conception de notre présent projet. Dans ce chapitre, on va exposer deux vues complémentaires de l'application : • Une vue globale qui traduit son architecture globale . • Une vue détaillée qui explique les différents choix pris.

## 3.1 Conception Architecturale

Concernant l'architecture de notre application nous avons choisi de travailler avec celle de trois couches, et ceci par le billet du patron de conception MVC (modèle vue contrôleur).

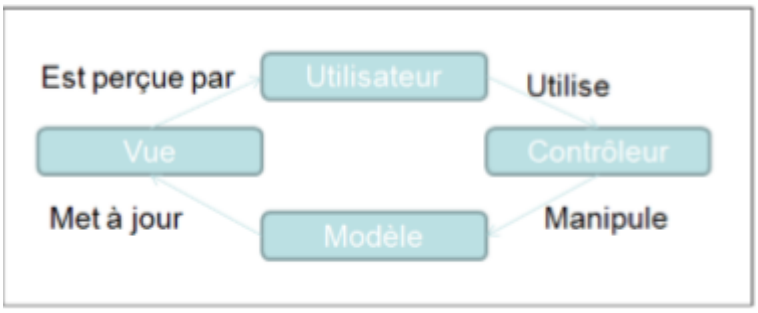



FIGURE 3.1 – Le modèle MVC

En effet, Le Framework Spring Web MVC fournit une architecture MVC (Model-View-Controller) et des composants prêts à utiliser pour développer des applications Web flexibles et peu couplées. Le modèle MVC permet de séparer les différents aspects de l'application (logique d'entrée, logique métier et logique UI), tout en fournissant un couplage en vrac entre ces éléments.

1. Le modèle :

-Représente la structure logique sous-jacente des données dans une application logicielle, la classe supérieure qui y est associée,l'interaction avec la base de données...
-S'intéresse à la représentation des données de la couche (business logic), i.e. données spécifiques à l'application
- Assure la gestion des données manipulées par l'application et garantit leur intégrité.
-Présente des méthodes assurant la mise à jour des données (insertion ,suppression, changement de valeur).et pour la récupération de ces données.

2. La vue :

concerne la forme (représentation) - Elle interagit avec le modèle
- la vue concerne principalement la représentation des données du modèle a l'ecran (ou sur tout autre périphérique de sortie).
-n'effectue aucun traitement.
- il peut donc exister plusieurs vues.

3. Le contrôleur :

-Le contrôleur gère les interactions avec l'utilisateur :
• s'occupe de la réécriture des URL.
• détermine quels traitements doivent être réalisées.
-Modifie les données du modèle et de la vue.
- Analyse la requête du client et appelle uniquement le modèle adéquat ensuite il renvoi la vue correspondante à cette demande.
-Le contrôleur dépend de la vue et du modèle :
la vue comporte des éléments visuels que l'utilisateur peut actionner. Le contrôleur répond aux actions effectuées sur la vue et modifie les données du modèle.

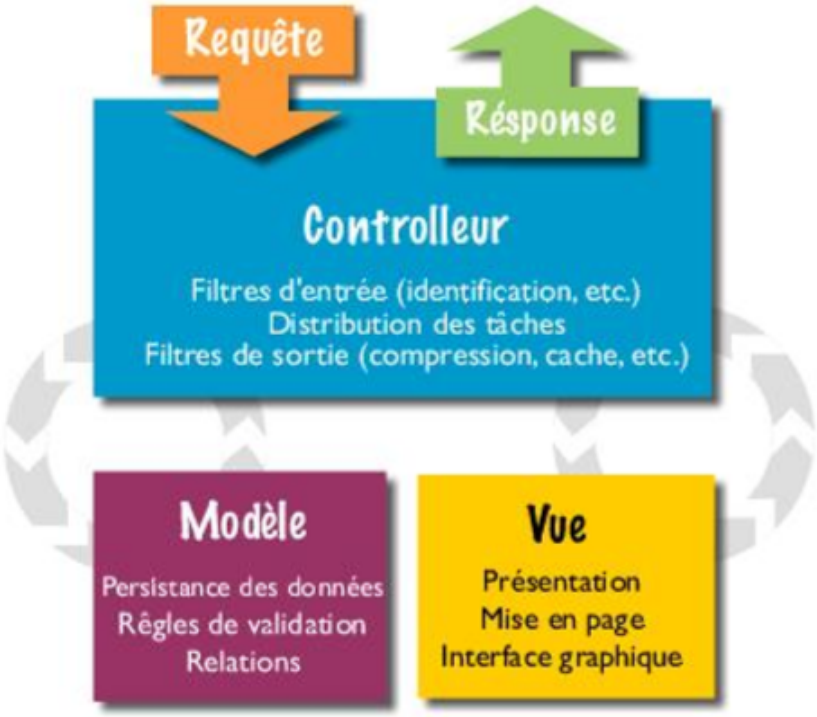



FIGURE 3.2 – Architecture MVC

En effet, une demande HTTP est envoyée au conteneur Java Servlet. La demande HTTP est d'abord interceptée par le contrôleur écrit en Java. Le contrôleur renvoie une page de serveur Java (JSP) qui est rendue sur le serveur et renvoyée au navigateur Web sous forme de code HTML statique. Le diagramme ci-dessous montre comment une application Java MVC interagit avec le côté client de l'application Web dans un navigateur Web.

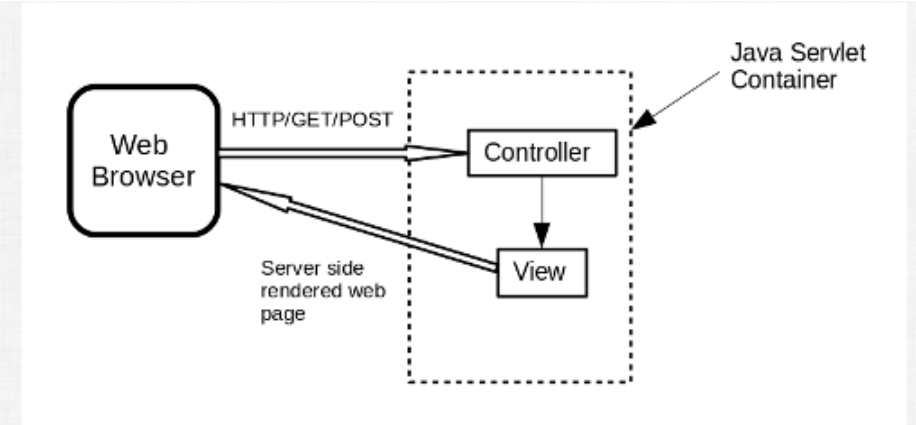



FIGURE 3.3 – interaction d'une application Java MVC avec le coté client

## 3.2 Conception Détaillé

## 3.3 Conception de la Base de Données

# Chapitre 4

# Réalisation

**Introduction**

**C**ette partie est consacrée à la présentation de tous les environnements logiciels que nous avons employé afin de réaliser notre projet ,et à sa mise en œuvre avec les tests d'intégration ainsi que les perspectives.

## 4.1 Architecture physique

L'architecture physique d'un logiciel décrit l'ensemble des entités matériels supportant l'application. Généralement nous parlons d'architecture physique quand nous travaillons sur une application appelée distribuée (ou répartie). En effet un système reparti est défini comme étant un ensemble de programmes distribués sur un réseau de communication et qui a pour but de coopérer et réaliser un service.
La Figure suivante décrit les entités matérielles de notre application :

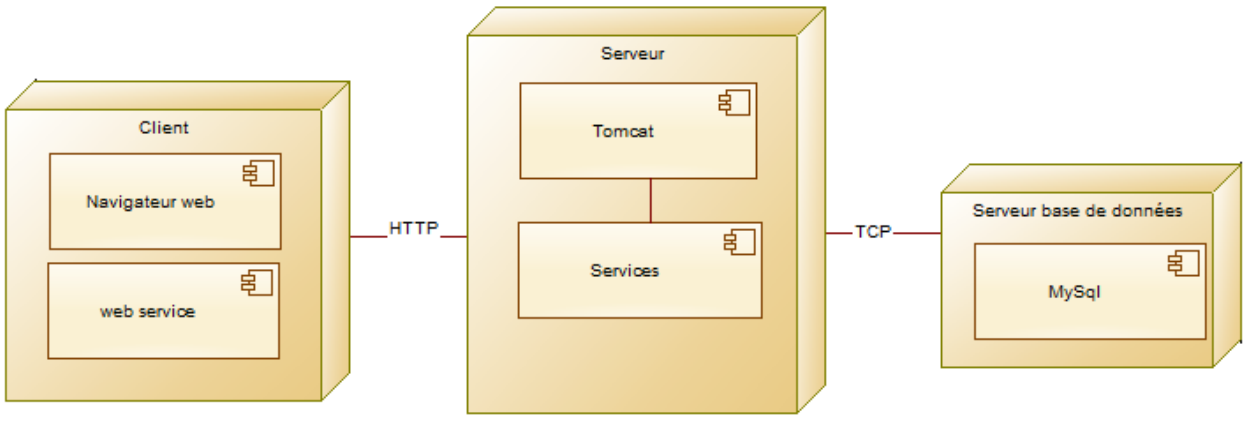



FIGURE 4.1 – Diagramme de déploiment

Dans notre projet nous avons choisi de travaillé avec le style architecturale physique 3 tiers qui est définis comme suivant :
-Tier client :
correspond à la couche de présentation s'occupant de la partie visible et interactive de l'application pour les utilisateurs. En effet le tier client va être un service web jouant le rôle d'intermédiaire entre notre système et le client mis en jeu.
-Tier applicatif :
correspond à la couche de traitement, c'est la partie fonctionnelle de l'application, celle qui implémente la logique métier, qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs effectuées au travers de la couche de présentation.En effet ce tier est composé d'un serveur d'application Tomcat faisant appel aux services de l'application
-Tier données :
correspond à la partie gérant l'accès aux données de l'application.
La figure suivante explique bien les relations entre les 3 couches dans une application utilisant le modèle MVC.

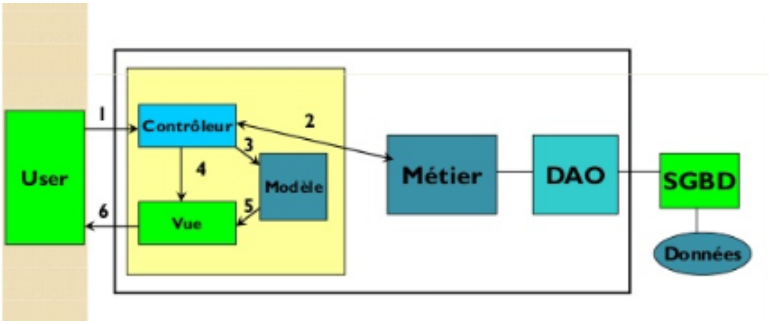



FIGURE 4.2 – Architecture 3 tiers d'une application respectant le Modèle MVC

En effet les numérotations dans cette figure indiquent :
1 * Le client fait une demande au contrôleur. Celui-ci voit passer toutes les demandes des clients. Dans notre cas le contrôleur est assuré par une servlet générique :
**org.springframework.web.servlet.DispatcherServlet**
2 * Le contrôleur principal[**Dispatcher Servlet**] fait exécuter l'action demandée par l'utilisateur par une classe implémentant l'interface **org.springframework.web.servlet.mvc.controller**
3 * Le contrôleur[**Controller**] traite une demande particulière de l'utilisateur.Pour ce faire , il peut avoir besoin de la couche métier. Une fois la demande du client traitée,celle-ci peut appeler diverses réponses.
4 * Le contrôleur choisit la réponse(qui est égale à la vue) à envoyer au client.
5 * Le Contrôleur[**DispatcherServlet**]demande à la vue choisie de s'afficher.Il s'agit d'une classe implémentant l'interface **org.springframework.web.servlet.View**
6 * Le générateur de vue View utilise le modèle **Map** préparé par le **Contoller** pour initialiser les parties dynamiques de la réponses qu'il doit envoyer au client.
7 * La réponse est envoyée au client.

## 4.2 Environnement de travail

Cette section est dédiée pour la présentation de l'environnement matériel et logiciel utilisés pour le développement de cette application.

### 4.2.1 Configuration matérielle

Afin de réaliser cette application on a à notre disposition un ordinateur caractérisé comme suit :
Machine : hp.
Disque Dur : 1To.
Processeur : Intel Core i5.
Type de système : Windows 10.

### 4.2.2 Configuration logiciel

A propos l'environnement logiciel nous présentons par la suite la liste des différents composants déployés et la raison pour laquelle on les a choisi :
-Système d'exploitation : Windows 10
-Environnement de développement intégré : Eclipse IDE for Java EE Developers


-SGBD : MySQL est un système de gestion de bases de données relationnelles (SGBDR). C'est un logiciel libre, open source et fait partie des logiciels de gestion de base de données les plus utilisés au monde.Il supporte le langage de requête SQL.


-Serveur :Apache Tomcat, souvent appelé Tomcat Server, est un conteneur de servlets Java open source développé par Apache Software Foundation (ASF). Tomcat implémente plusieurs spécifications Java EE, y compris Java Servlet, JavaServer Pages (JSP) et Web-Socket, et fournit un environnement de serveur web http « Java pur » dans lequel le code

Java peut s'exécuter.


- plateforme de développement Web :WampServerun est un environnement comprenant trois serveurs (Apache, MySQL et MariaDB) permettant de faire fonctionner localement (sans avoir à se connecter à un serveur externe) des scripts PHP.


-Outils de modélisation UML :
*PowerDesigner :
C'est un logiciel de conception créé par la société SAP, qui permet de modéliser les traitements informatiques et leurs bases de données associées. Nous avons utiliséce logiciel durant notre projet pour schématiser les diagrammes de séquence, le diagramme de classe et le modèle entité association.

*StarUml :
c'est un logiciel de modélisation UML gratuit,open source,flexible et extensible. Grace à ce logiciel nous avons pu représenter les différents diagrammes use case.

-Editeur de maquettes IHM : Balsamiq Mockups
c'est un outil permettant de créer facilement des prototypes d'IHM électronique. Avec Balsamiq Mockups il est ainsi possible de prototyper tout type d'applications (desktop, web, smartphone, . . . )


## 4.3 Captures d'écran

# Conclusion générale

# Bibliographie

# Netographie