# Langage Web
# Gestionnaire de maintenance
# Rapport

Créateurs :
PIRES Théo
OSMONT Alexis

Encadré par :
Monsieur NICART

# Table des matières

# 1 - Présentation du projet

L’objectif de ce projet est de créer une application web permettant la gestion d’objet et de répertorier facilement les défaillances ou l’épuisement d’un produit. Pour ce faire, l’utilisateur voulant spécifier une défaillance, pourra scanner un code QR ou entrer une URL dans son navigateur pour l’emmener sur une page internet où il pourra expliquer la défaillance remarquée de l’objet, afin que le responsable de cet objet puisse être mis au courant pour traiter la défaillance.

# 2 - Fonctionnalités de l’application

Notre application offre la possible à n’importe quelle personne d’accéder à une page internet avec des informations telles que le nom de l’objet, sa localisation et une description de l’objet, ainsi que l’accès à un champ éditable pour lui permettre d’indiquer la défaillance d’un objet.

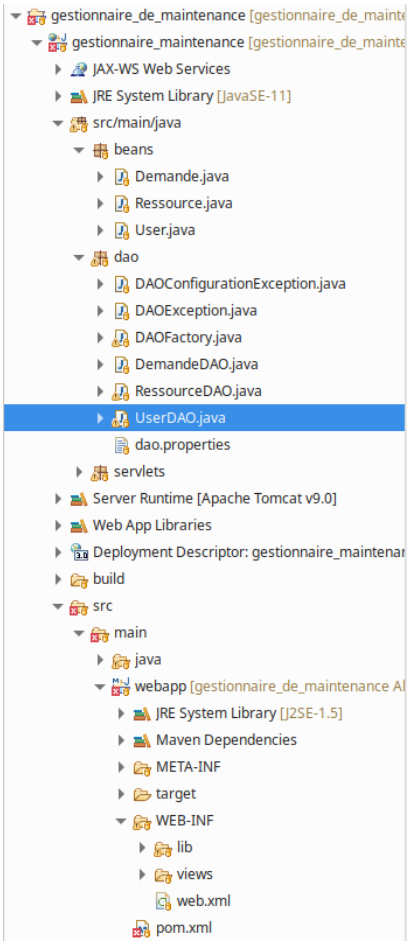
Chaque anomalie découverte sur un objet mettra à jour une base de données répertoriant les demandes (qui représente les anomalies détectées par les utilisateurs), les ressources (les objets que l’on souhaite maintenir à l’aide de notre application) et les utilisateurs (représentant le personnel qui utilise l’application pour maintenir les objets).

Si on utilise l’application en tant qu’administrateur, on obtient une liste des comptes qui sont responsables de la maintenance. Il est possible d’ajouter ou supprimer ces comptes.

Si on utilise l’application en tant que responsable de maintenance, nous avons une liste des demandes concernant les ressources dont nous sommes chargés de maintenir en bon état fonctionnel. Il est possible de valider les demandes que nous avons traité au préalable. Nous sommes aussi capable à l’aide du formulaire de rajouter des ressources que nous devrons maintenir.

# 3 - Structure du projet

Lors de la création du projet sur Eclipse, il génère une structure du projet :




Dans le dossier ***webapp***, la présence du dossier ***WEB-INF*** contient le dossier ***lib*** contenant l'ensemble des librairies et le dossier ***views*** contenant l'ensemble de nos pages JSP.

Dans le package ***src/main/java***, on y retrouvera trois packages, le package ***beans*** contenant les objets que nous allons utiliser lors de l'implémentation**,** le package ***dao*** (pour data access object) contenant les classes qui vont interagir avec notre base de données et le package ***servlet*** contenant des fichiers java faisant office de contrôleur pour notre application.

# 4 - Modèle MVC

Notre application a été implémenté de manière à respecter le modèle MVC (modèle, vue et contrôleur).

Nous accédons à notre application web par le biais d'un navigateur qui va envoyer des informations à notre serveur web qui suit un modèle MVC, les informations passent d'abord par le contrôleur qui va gérer le comportement de notre site internet. Il va falloir passer par le modèle pour récupérer nos données sur le serveur de base de donnés pour ensuite les renvoyer au contrôleur qui va se charger

d’envoyer les informations pour générer une vue, c’est cette vue qui est renvoyé au navigateur et devient disponible par l’utilisateur.

## 4.1- Le serveur web

Nous retrouvons une architecture MVC avec trois grands rôles différents :

- Le modèle
- La vue
- Les contrôleurs

Chacun de ces rôles a pour objectif d’effectuer des actions différentes pour ensuite interagir ensemble.

### 4.1.a) Les contrôleurs

C’est ici que le comportement du site est défini. En fonction des requêtes provenant du navigateur. Ils vont permettre d’interagir avec le modèle et modifier la vue en conséquence.
Le comportement de notre projet est localisé dans le package ***/src/main/java/servlets*** représenté par des classes Java.

### 4.1.b) Le modèle

Le modèle contiendra l’état de notre application à tout instant. C’est ici que les contrôleurs précédemment vus, vont chercher les informations que nous désirons traiter avant de les afficher à l’utilisateur.

Les classes Java représentant le modèle sont situées dans le package ***/src/main/java/dao***.

### 4.1.c) La vue

La vue permet la création de ce que l’on souhaite afficher à l’utilisateur.

Dans notre projet, les fichiers responsables de la vue sont des fichiers d’extension ***jsp (java server page)***. Ils se localisent dans le dossier ***/webapp/WEB-INF/views/*** .

# 5 - Structure Base de Données

Notre base de données tourne sous le système de gestion de bases de données relationnelles MySQL contenant 3 tables, la table Utilisateurs, la table Demande et la table Ressources :

La table demande :

| # | Nom | Type | Interclassement | Attributs | Null | Valeur par défaut | Commentaires | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | **idRequest** | int | | | Non | *Aucun(e)* | | AUTO_INCREMENT |
| 2 | **idUser** | int | | | Non | *Aucun(e)* | | |
| 3 | **idSource** | int | | | Non | *Aucun(e)* | | |
| 4 | **idManagerMaint** | int | | | Non | *Aucun(e)* | | |
| 5 | **state** | varchar(100) | utf8_general_ci | | Non | *Aucun(e)* | | |
| 6 | **description** | varchar(256) | utf8_general_ci | | Oui | *NULL* | | |

La table Ressources :

| # | Nom | Type | Interclassement | Attributs | Null | Valeur par défaut | Commentaires | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | **idSource** | int | | | Non | *Aucun(e)* | | AUTO_INCREMENT |
| 2 | **idManagerMaint** | int | | | Non | *Aucun(e)* | | |
| 3 | **localisation** | varchar(100) | utf8_general_ci | | Non | *Aucun(e)* | | |
| 4 | **description** | varchar(256) | utf8_general_ci | | Oui | *NULL* | | |
| 5 | **nom** | varchar(255) | utf8_general_ci | | Oui | *NULL* | | |

La table Utilisateurs :

| # | Nom | Type | Interclassement | Attributs | Null | Valeur par défaut | Commentaires | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | **idUser** | int | | | Non | *Aucun(e)* | | AUTO_INCREMENT |
| 2 | **username** | varchar(100) | utf8_general_ci | | Non | *Aucun(e)* | | |
| 3 | **pwd** | varchar(100) | utf8_general_ci | | Non | *Aucun(e)* | | |
| 4 | **job** | varchar(100) | utf8_general_ci | | Oui | *NULL* | | |

# 6 - Modèle DAO

Dans le modèle de notre MVC, on a vu que la partie modèle était représentée par une base de données. Pour interagir avec cette base de données, nous avons aussi suivi un modèle que l'on appel le modèle DAO (Data Access Object). Il consiste à créer des objets qui communiquent seulement avec une partie de notre base de données. L'ensemble de nos objets DAO sont localisés dans le package ***/src/main/java/dao*** .

# 7 - Java Beans

Nous avons créé des classes Java qui ont pour rôle d'interagir et sauvegarder l'état des objets ; ce sont des Java Beans. Ce sont des classes contenant des attributs, un constructeur, des requêtes pour accéder à l'état de l'objet et des commandes pour modifier l'état de l'objet.

Notre projet contient trois Java Beans qui se trouvent dans le package ***/src/main/java/beans*** :

- Demande : offre la possibilité d'interagir ou de récupérer les données d'une demande.

- Ressource : offre la possibilité d’interagir ou de récupérer les données d’une ressource.
- User : offre la possibilité d’interagir ou de récupérer les données d’un utilisateur.

Les Java Beans sont souvent utilisés lors d’une récupération d’une information dans la base de données afin de mettre toutes les valeurs retournées par nos requêtes SQL dans ces objets.

# 8 - Difficultés

Une des difficultés a été la recherche pour l’interaction avec une base de données en Java et son utilisation. La recherche de l’API JDBC a été plutôt rapide. C’est surtout son utilisation qui a demandé du temps.

Une autre difficulté : le travail essentiellement à distance avec l’utilisation de Git dans lequel nous avons encore quelques lacunes lorsque qu’il faut faire une fusion de nos travaux et que nous avions modifié le projet chacun de notre côté.

# 9 - Améliorations

Il y a plusieurs améliorations qui pourrait être envisagées dans notre projet.

Notamment la sécurité. Au niveau des comptes, tous les mots de passes sont stockés en clair. Ainsi que la connexion à notre base de données où l’on peut voir le nom utilisateur et le mot de passe en clair. Il faudrait trouver un moyen de stocker toutes ces informations de manière cryptées.

La possibilité de se connecter en tant qu’utilisateur n’apporte pas réellement de fonctionnalité supplémentaire dans notre projet actuel. Mais dans le futur, on pourrait penser à un système de récompense en fonction du nombre de signalements corrects qu’un utilisateur produit. Ce qui nous oblige d’être connecté pour récupérer ces récompenses.