samedi 10 juin 2017



Application web de gestion de matériel

LAMPIN DENOVAN, Julien Pruvost

lICENSE 3 informatique

Table des matières

[Introduction : 2](#_Toc484866839)

[Répartition des tâches 2](#_Toc484866840)

[Analyse 2](#_Toc484866841)

[Réalisation du projet 4](#_Toc484866842)

[Conclusion 6](#_Toc484866843)

[Annexes 7](#_Toc484866844)

# Introduction :

Le projet consisté au développement d’une application WEB permettant la gestion de matériels. Cette application devait permettre aux utilisateurs de consulter la liste des objets et leur quantité mais aussi d’emprunter ou de rendre des objets.

Un administrateur devait être capable d’inscrire un nouvel objet dans la base de de données et d’en supprimer.

Pour ce projet, certaines technologies nous étaient imposées :

* Java EE ;
* Utilisation des Servlets et des .JSP ;
* Serveur TomCat ;

Il nous était donc demandé de créer en plus de l’application, une base de données.

Il s’agit d’un projet qui s’étend sur la totalité du semestre 6 (Licence 3 informatique) avec le soutien de notre tuteur Monsieur Mikael Desertot.

# Répartition des tâches

Nous avons d’abord chacun de notre côté effectué des recherches sur JEE et les moyens de développer des applications web modernes.

Puis nous avons réfléchi ensemble au modèle de données de l’application (la base de données, les fonctionnalités précises).

Suite à cela, nous nous sommes réparti les différentes fonctionnalités de l’application.

Les fonctions de gestion de la connexion/déconnexion, des sessions utilisateur et de l’inscription ont été principalement réalisés par Denovan.

Les fonctions de gestion admin (ajout, suppression) et la gestion des emprunts ont été principalement réalisés par Julien.

Une fois l’application fonctionnelle, nous avons pu l’améliorer : Denovan a pu améliorer l’ergonomie de l’application et a effectué des recherches sur Struts (pas de mise en place par manque de temps) et Julien a amélioré le code de l’application (passage en JPA, refactorisation).

# Analyse

Nous avons dû mettre en place une base de données pour l’application, cette analyse s’est faite en amont des autres tâches. La structure de cette base de données se compose de trois tables :

* Une table User, permettant de stocker les informations de l’utilisateur ;
* Une table Objet, permettant de stocker les différents objets ;
* Une table Emprunt, permettant de stocker toutes informations liées à un emprunt.

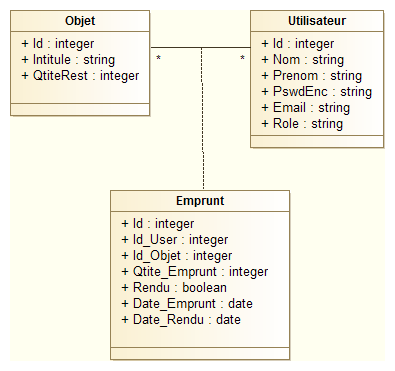
Voici le schéma de cette base de données.

Figure 1. Modèle MCD

Nous pouvons voir ici en détails la structure de la base de données ou nous avons dans la table utilisateur un identifiant (PRIMARY KEY), un nom, un prénom, un mot de passe encodé, un email (qui servira de login de connexion, donc UNIQUE) ainsi qu’un rôle qui est soit « user » soit « admin ».

La table objet ne contient qu’un identifiant (PRIMARY KEY), un intitule (le nom de l’objet donc UNIQUE) ainsi que la quantité qu’il reste. Une vérification est faite en SQL pour s’assurer que la quantité reste toujours positive.

Enfin, nous avons la table Emprunt qui est une table associative entre utilisateur et objet où nous retrouvons les identifiants de l’objet et de l’utilisateur (id\_user et id\_objet qui sont des FOREIGN KEY). L’emprunt est aussi indexé à l’aide de son id (PRIMARY KEY). Nous pouvons également observer la quantité empruntée par l’utilisateur ainsi que les dates de l’emprunt (qui s’initialise automatiquement grâce au SQL) et de rendu. Nous avons pour finir le champ Rendu qui permet de savoir si un objet est rendu ou non. *Bien que depuis que la date de rendu est enregistrée ce champ n’est plus nécessaire.*

Le choix du gestionnaire de base de données s’est fait entre 3 choix (car il fallait un gestionnaire léger pour cette application) :

* H2, qui est le plus performant ;
* Derby, moins performant mais reste dans le top ;
* HyperSQL, qui est plus stable et facilement pris en main ;

Les retours trouvés lors de nos recherches ont éliminé Derby.

Nous avons fait le choix d’HyperSQL (ou HSQL) car il semble, malgré une performance moins, plus répandu et donc plus facile à mettre en place pour un projet qui possède un temps limité. HSQL est souvent le mieux classés comparé aux deux autres sur les comparateurs de gestion de base de données relationnelles.

# Réalisation du projet

Nous avons tous deux commencé à analyser le problème ainsi que deux réfléchir à une structure possible de l’application.

Ensuite chacun de notre côté nous avons réalisé un tutoriel[[1]](#footnote-1) sur le WEB, nous permettant de connaître les bases du JEE. Nous avons utilisé l’IDE Eclipse JEE tout au long du projet.

Une première application a été créée qui nous a permis simplement de nous connecter, d’emprunter et de rendre un objet. A ce moment, il n’a avait aucune gestion des erreurs c’est-à-dire que lorsque qu’une erreur survenez, l’utilisateur connecté n’en avait aucune information. Ensuite cette application ne permettait pas l’inscription d’un nouvel utilisateur.

Cette application a été versionnée sur GitHub[[2]](#footnote-2) afin de pouvoir travailler chacun de notre côté plus efficacement.

C’est de cette base qu’est partie notre application, bien qu’après coup, nous aurions dû recommencer de zéro une fois les connaissances acquises, car le code réalisé en début de projet n’était pas forcément optimisé ou n’utilisait pas les dernières fonctionnalités du JEE.

Nous avons pu ensuite réaliser les différentes fonctionnalités.

Les templates JSP ont été réalisées en début de projet, nous avons utilisé la bibliothèque JavaScript Bootstrap et JQuery pour une implémentation simple et efficace du design, ainsi que la bibliothèque JavaScript DataTables pour gérer les listes.

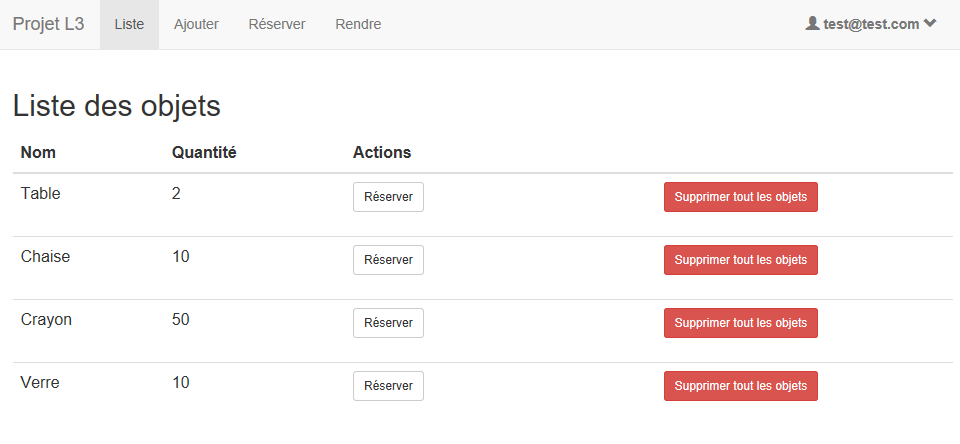


Figure 2. Page d'accueil en tant qu'admin

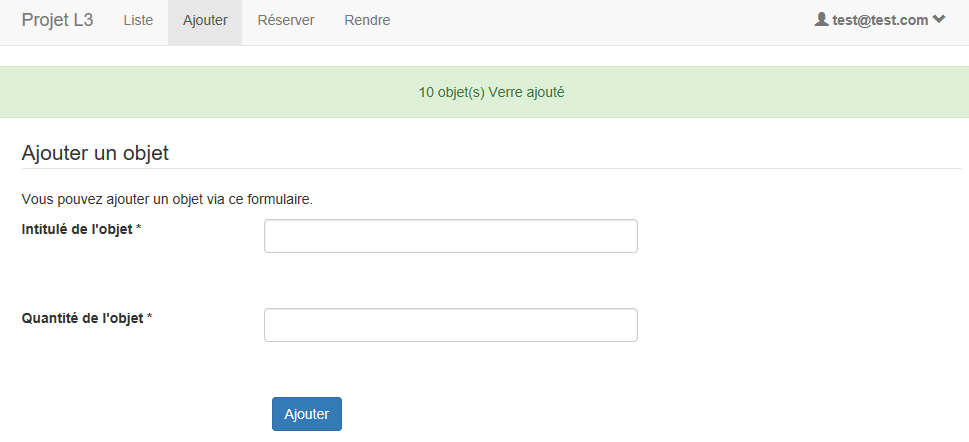


Figure 3. Page d'ajout d'un objet (admin)

Les servlets ont d’abord été enregistré via le fichier web.xml, puis via les annotations grâce aux nouvelles versions de JEE.

Les listener également ont été enregistré sur le web.xml puis directement dans les fichiers Java grâce aux annotations. Ceux-ci servent à ne pas avoir accès aux pages autres que l’accueil ou inscription si l’on n’est pas connecté, ainsi qu’à vérifier si un utilisateur à accès aux pages d’administration.

La gestion des formulaires se fait grâce à des classes dédiées, qui vont retrouver les champs du formulaire HTML et faire des vérifications personnalisées.

Les entités (Utilisateur, Objet et Emprunt) ont d’abord été créés en tant que Bean (classe Java sérialisable), avec des fonctions statiques qui accédées à la base de données via JDBC, une classe d’accès à la base avait été créée pour éviter la redondance de code dans l’application.

Puis dans un second temps, grâce à l’implémentation de JPA (« Java Persistence API »), les Beans ont été retiré du projet pour être séparées en Entités et DAO (« Data Access Object »). Les entités vont faire référence directement à la base de données en se connectant au serveur avec un fichier de configuration XML. On pourra donc utiliser un Entity Manager pour gérer les interactions simples avec la BDD.

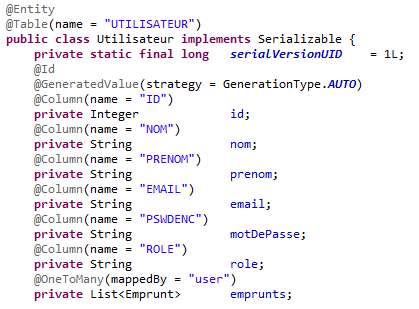


Figure 4. Exemple d'entité JPA

Et les DAO seront les classes qui vont servir à gérer les accès plus complexes avec la base de données (comme lister des Emprunts spécifique, trouver un certains Utilisateur).

# Conclusion

Pour conclure, notre application est fonctionnelle ; elle permet à un administrateur de créer ou de supprimer un objet dans la base de données. Elle permet également à un utilisateur de réserver un nouvel objet avec la quantité souhaité à la condition qu’il reste suffisamment d’objet. Un historique des emprunts est enregistré, et est affiché pour chaque utilisateur.

La gestion des erreurs est également mise en place. Si l’utilisateur commet une fausse manipulation, on retournera une erreur.

L’application n’est pas parfaite, nous n’avons pas eu le temps d’apprendre toutes les ficelles de JEE, une utilisation de Spring avec Maven aurait pu être judicieuse pour obtenir du code plus propre et une gestion des Entités plus efficace. Certaines parties comme la génération des listes d’objets pourrait être améliorées.

Ou nous pourrions à l’aide de struts2, gérer plus facilement les dépendances.

Nous avons pu apprendre de ce projet qu’il y a 1000 manières de développer une application web. Que le JEE en est une, qui est répandu et a également beaucoup de manière d’être appréhendé. Nous avons choisi une manière plutôt simple car notre application est simple, mais des Framework sont dédiées au développement d’applications complexes pour JEE également.

Nous remercions également M. Desertot qui a pu nous guider sur certaines pistes de développement et a répondu présent à nos demandes de rendez-vous.

# Annexes

Le versionnage du projet : <https://github.com/hqrd/ProjetL3>

Le tutoriel pour apprendre les bases : <https://openclassrooms.com/courses/creez-votre-application-web-avec-java-ee>

1. Voir annexes [↑](#footnote-ref-1)
2. Voir annexes [↑](#footnote-ref-2)