

Réalisation d'une application de gestion de bibliothèque

Module : Cawa.

Fait par :

- Ferhat Yacine, TI 02.
- Ouali Mustapha TI02.

Gestion de Bibliothèque

1. Introduction :

De nos jours, l'informatique prend de plus en plus d'ampleur dans notre quotidien, où son importance ne cesse de s'accroître, où l'humain a tout simplement abandonné les méthodes archaïques et s'est mis à l'informatique, où le temps est devenu une ressource bien trop importante.

De nos jours, plusieurs possibilités s'offrent à nous les développeurs, des langages, des architectures et des environnements ont été créés afin de répondre à nos différents besoins, ainsi qu'aux besoins des utilisateurs.

C'est ici que Java Enterprise Edition(JEE) entre en scène, qui permet de développer une application web robuste basée sur une plateforme et architecture solide.

Cette application est basée sur le modèle MVC, que JEE utilise en particulier, nous avons utilisé le modèle MVC car il nous permet de séparer le traitement des données et des présentations, ainsi les tâches sont plus simples à réaliser, les fichiers sont organisés, donc on sait où trouver ce qu'on cherche, la distribution des tâches sur les différents membres de l'équipe ainsi que la maintenance.

2. Analyse et spécification des besoins :

L'informatique étant une science de traitement automatique de données s'avère bénéfique dans tous les domaines qu'ils soient scientifiques ou non. Ce projet vient dans le cadre d'un projet d'étude du module Cawa, il nous est demandé de créer une solution web pour la gestion d'une bibliothèque, qui connaît actuellement assez de difficultés liées à son mode de gestion manuelle et archaïque, entraînant un gaspillage de temps et de ressources.

Cette solution web nous permettra d'apporter les fonctionnalités suivantes :

- La consultation des œuvres disponibles dans la bibliothèque.
- La création des différents comptes
- La gestion de la bibliothèque par les administrateurs
 - La gestion des différents utilisateurs.
 - La gestion des ouvrages.
 - La gestion des écrivains.

Par conséquent, nous ressortons avec deux différents acteurs.

- L'administrateur, qui s'occupera de la gestion de notre plateforme.
- Le simple utilisateur, qui pourra consulter les ouvrages disponibles après s'être authentifier ou inscrit sur notre plateforme.

Les différents use cases sont représentés dans le **digramme de cas d'utilisation** suivant :

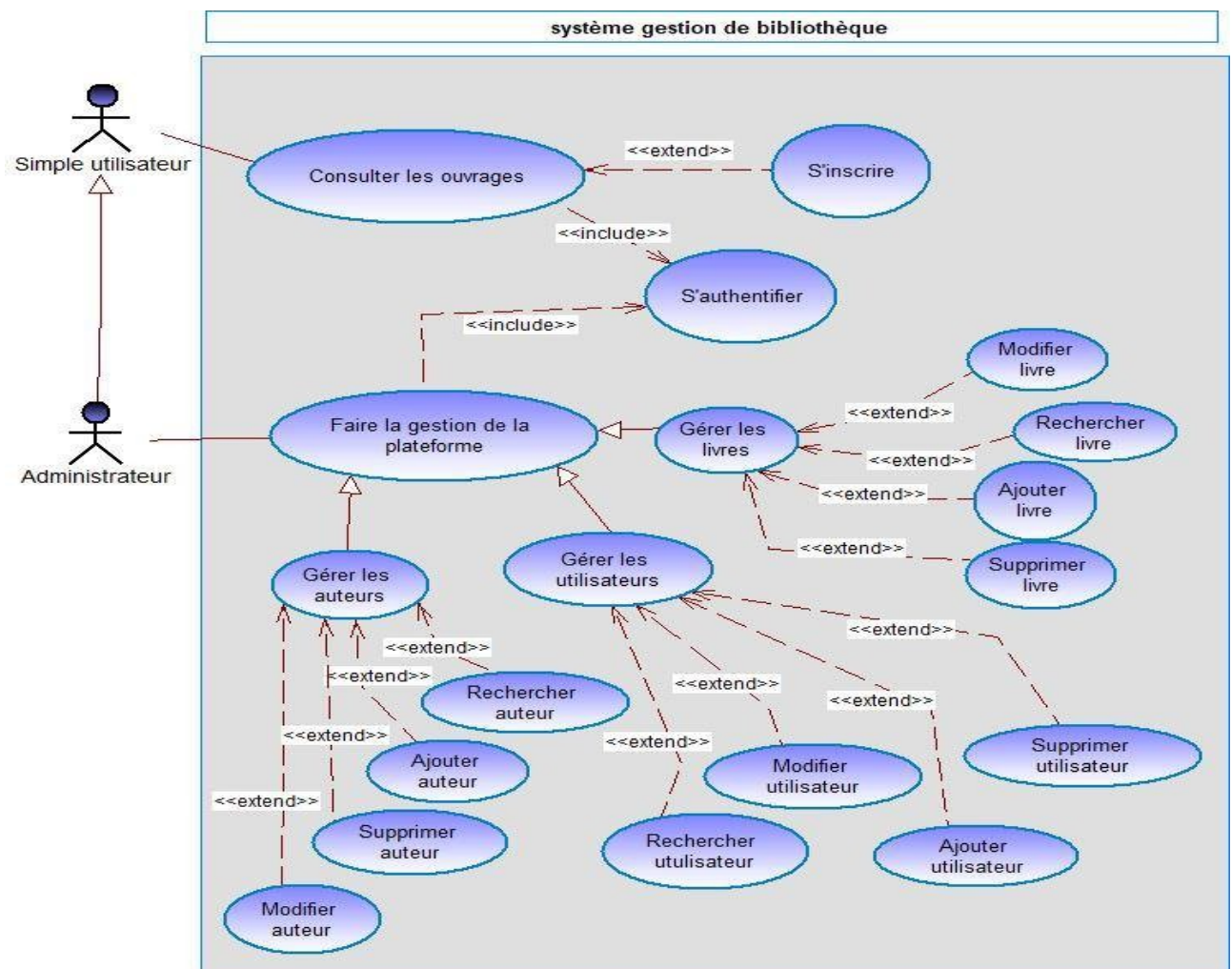


Illustration 1: Diagramme de cas d'utilisation "Gestion de la bibliothèque"

Description textuelle du cas d'utilisation « rechercher un utilisateur »

- **Titre:** rechercher un livre
- **But :** afficher la liste des livres correspondant à la recherche
- **Acteur :**
 - Admin.
 - Simple utilisateur.
- **Pré-Condition :** La page de recherche est affichée.
- **Post-Condition :**
 - Affichage des résultats de la recherche.
 - L'utilisateur consulte les livres recherchés.
- **Enchaînement nominaux :**
 - L'utilisateur Choisit le type de recherche qu'il veut faire, par
 - Titre.
 - Code.
 - Auteur.
 - Domaine.
 - L'utilisateur Remplit le champ d'insertion.
 - L'utilisateur clique sur le bouton Search.
 - La page de recherche est affichée avec les résultats.
- **Enchaînement alternatifs :**
 - Le livre recherché n'existe pas.

3. Conception du système :

Afin de réaliser notre application, nous avons établis des règles de gestion, fait un diagramme de classe ainsi qu'un MLD

1. Règles de gestion

Afin de réaliser notre plateforme web, nous avons utilisé ces différentes règles de gestion :

- L'auteur qui peut écrire un ou plusieurs livres.
- Le livre qui peut être écrit par un ou plusieurs auteurs.
- L'utilisateur, qui peut être un admin ou un simple utilisateur.
- Le simple utilisateur peut consulter un ou plusieurs livres et le livre peut être consulté par plusieurs utilisateurs.
- L'admin peut gérer les livres, les simples utilisateurs ainsi que les auteurs, ils peuvent tous être gérés par plusieurs admins.

Qui nous ont permis d'obtenir le diagramme de classes suivant

2. Diagramme de classe

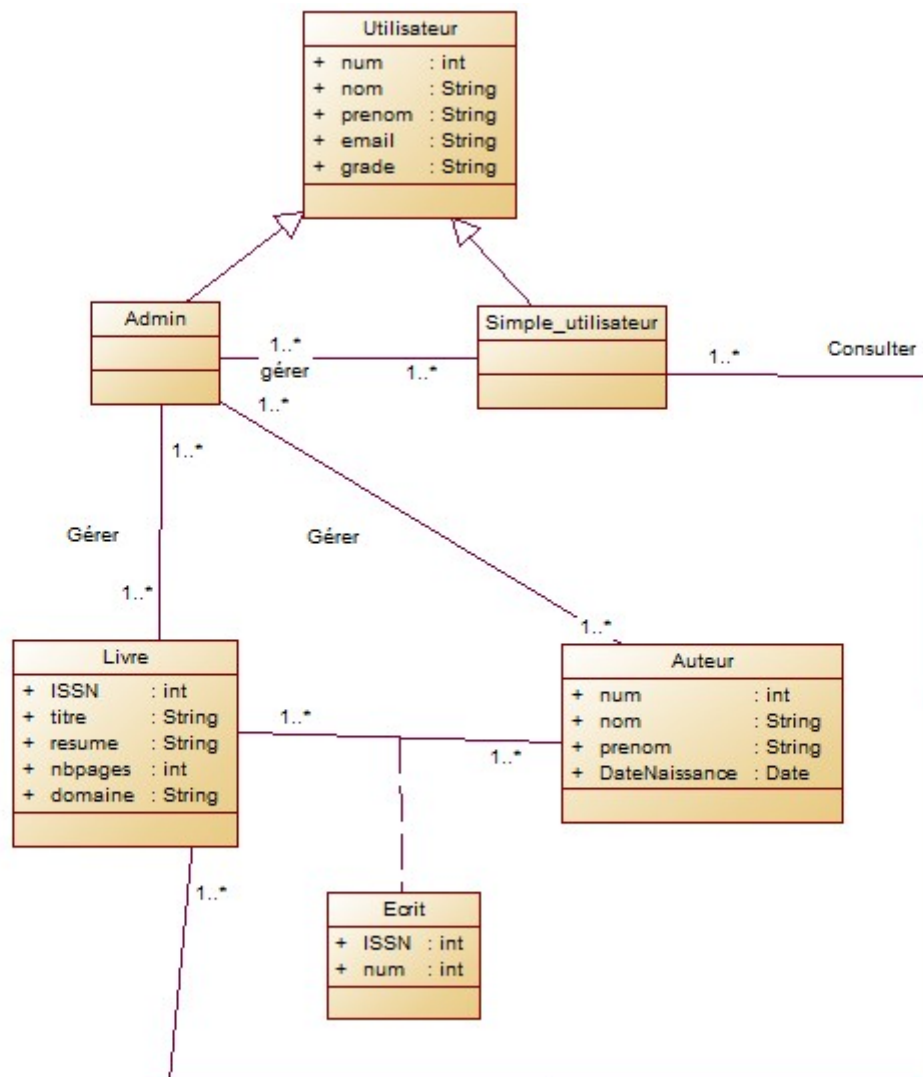


Illustration 2: Diagramme de classes

3. MLD :

- Auteur(Num,Nom,Prenom,DateDeNaissance)
- Utilisateur(id,nom,prenom,mail,passWord,grade)
- Livre(ISSN,Titre,Auteur,Resume,NbPages, Domaine)

4. Implémentation :

1. Architecture MVC :

Ayant suivi le modèle MVC, nous avons partagé notre application en 3 parties :

- La partie Modèle qui encapsule la logique métier ainsi que l'accès aux données.
- La partie Vue qui s'occupe des interactions avec l'utilisateur, qui sont les différents fichiers Jsp.
- La partie Contrôleur qui gère la dynamique de l'application, qui fait le lien entre l'utilisateur et le reste de l'application et qui s'occupe du traitement des données et des insertions des utilisateurs.

Les différentes interactions entre le modèle, la vue et le contrôleur peuvent se résumer par le schéma suivant :

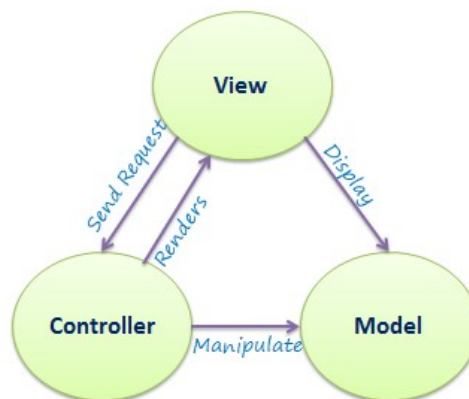


Illustration 3: MVC

2. Outils de développement :

1. Mysql :

Ayant besoin d'un SGBD, nous avons opté pour MYSQL, qui est un serveur de base de données relationnelles Open Source, qui stocke les données dans des tables séparées, les tables sont reliées par des relations définies, qui permettent la combinaison des données entre les différentes requêtes.

2. NetBeans :

Afin de développer notre application, nous avons opté pour l'IDE Netbeans, qui est Open Source également, qui nous a permis d'avoir un environnement de développement prêt au préalable, c'est à dire que nous n'avons pas à configurer notre serveur.

3. Présentation de l'application :

1. Partie Modèle

Elle correspond à la structure de notre base de données.

Notre base de données a pour nom « cawa », et dispose de 3 différentes tables :

1. **Table Auteur** : Représente l'entité auteur de notre application.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/> 1	Num	int(3)			Non	Aucun(e)		AUTO_INCREMENT
<input type="checkbox"/> 2	Nom	varchar(20)	utf8_general_ci		Non	Aucun(e)		
<input type="checkbox"/> 3	Prenom	varchar(20)	utf8_general_ci		Non	Aucun(e)		
<input type="checkbox"/> 4	DateDeNaissance	date			Non	Aucun(e)		

Illustration 4: Table Auteur

2. **Table Utilisateur** : Représente les utilisateurs.

Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
id	int(10)			Non	Aucun(e)		AUTO_INCREMENT
nom	varchar(20)	utf8_general_ci		Non	Aucun(e)		
prenom	varchar(20)	utf8_general_ci		Non	Aucun(e)		
email	varchar(80)	utf8_general_ci		Non	Aucun(e)		
passWord	varchar(20)	utf8_general_ci		Non	Aucun(e)		
grade	tinyint(1)			Non	Aucun(e)		

Illustration 5: Table Utilisateur

3. **Table livre** : Représente les ouvrages présents dans notre base de données.

Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
ISSN	int(10)			Non	Aucun(e)		AUTO_INCREMENT
Titre	varchar(100)	utf8_general_ci		Non	Aucun(e)		
Auteur	varchar(40)	utf8_general_ci		Non	Aucun(e)		
Resume	varchar(2000)	utf8_general_ci		Non	Aucun(e)		
NbPage	int(3)			Non	Aucun(e)		
Domaine	varchar(20)	utf8_general_ci		Non	Aucun(e)		

Illustration 6: Table Livre

2. Partie Vue :

Qui représente nos différentes pages JSP.

Nous avons désigné notre application en utilisant les classes du Framework css Bulma.

La première page étant la page d'accueil, où l'utilisateur doit insérer ses identifiants ou bien s'inscrire afin d'accéder à la rubrique utilisateur.

Illustration 7: Page Index.jsp


Après s'être authentifié, nous distinguons deux différents cas :

- Si l'utilisateur est un simple utilisateur, il sera redirigé vers la page HomeUser.jsp, où l'utilisateur pourra voir tous les livres disponibles dans notre bibliothèque, il pourra faire une recherche par livre/Code/auteur/Domaine et le résultat sera affiché dans la page HomeUserResearch.jsp,

Illustration 8: Page HomeUser.jsp

Votre recherche

Vous avez fait une recherche par Domaine ayant pour nom Fantastique



- **Titre :** Harry Potter and the Philosopher
- **Ecrit par :** J. K. Rowling
- **Domaine :** Fantastique
- **Nb Pages :** 330
- **Resumé :** The first novel in the Harry Potter series and Rowling's debut novel, it follows Harry Potter, a young wizard who discovers his magical heritage on his eleventh birthday, when he receives a letter of acceptance to Hogwarts School of Witchcraft and Wizardry. Harry makes close friends and a few enemies during his first year at the school, and with the help of his friends, Harry faces an attempted comeback by the dark wizard Lord Voldemort, who killed Harry's parents, but failed to kill Harry when he was just 15 months old

Illustration 9: Page HomeUserResearch.jsp

- Si l'utilisateur est un administrateur, il sera redirigé vers la page HomeAdmin.jsp, où il pourra choisir quoi faire, soit faire la gestion des utilisateurs, des livres ou bien des auteurs.


BOUQUINI YACINE.FRHAT@GMAIL.COM

USERS BOOKS WRITERS

DISCONNECT

Home


As an admin, you can manage the whole system



Users

Here, u can access to the users of the system, you will be able to consult their informations, maybe modify some and delete some.


Move on



Books

Here, u can access to the books stored in the system, you will be able to consult their informations, add, delete and modify.

Move on



Writers

Here, you can access to the writers, you will be able to consult their informations, add writers, modify or delete some.

Move on

Illustration 10: Page HomeAdmin.jsp

Les 3 différentes parties sont identiques, il peut faire une recherche par différents critères, ajouter, supprimer, ou bien modifier, prenons l'exemple de la page de gestion des livres.

BOUQUINI YACINE.FR.HAT@GMAIL.COM

USERS BOOKS WRITERS

DISCONNECT

Code

▼

Type!

Search

add a book

Code	Name	Auteur	Domain	Nb pages	Resume	Actions
1	Venus en feu, Mars de glace	John Gray	Psychologie	318	Livre psychologique qui parle de la différence entre les hommes et les femmes	<div>delete</div> <div>modify</div>
2	Escale	Franck Bonnet	Littérature	104	Au coeur de chaque nouvelle, il y a une rencontre réel ou mytique	<div>delete</div> <div>modify</div>
3	A la rencontre des animaux	Camille Brunel	Encyclopédie	300	Présentation des différents animaux sauvages et domestiques	<div>delete</div> <div>modify</div>
4	Questions et réponses, notre monde	Pr Smith	Culture G	175	Traites différentes thématiques en rapport avec la Terre.	<div>delete</div> <div>modify</div>
5	Les classiques de la cuisine méditerranéenne	Samira Chibane	Cuisine	64	Recueil de recettes d'origine méditerranéenne.	<div>delete</div> <div>modify</div>

Illustration 11: Page AdminBook.jsp

BOUQUINI YACINE.FR.HAT@GMAIL.COM

DISCONNECT

Code

Type!

Search

add a book

Code	Name	A
1	Venus en feu, Mars de glace	Jo
2	Escale	Fr
3	A la rencontre des animaux	Ca
4	Questions et réponses, notre monde	Pr
5	Les classiques de la cuisine méditerranéenne	Sa

Add a book

Title

Apprendre à coder

Book's Writer

Apprendre à coder

Book's domain

Coding

Book's Resume

e.g made for begginer, to learn how to code !

Nb Pages

520

Submit

Reset

Illustration 12: Modal InsertBook

Modify!

Title

📄 apprendre à coder

Book's writer

✍️ Graitikart

Book's domain

🔑 coding

Book's Resume

u will learn how to code

Nb Pages

102

Submit

Cancel

Illustration 13: Page ModifyBook.jsp

2. Partie Contrôleur:

Les différents traitements de notre application seront principalement exécuté par la méthode DoGet qui inclut les méthodes des bibliothèques HttpServlet et Sql et HttpSession.

1. Servlet Index :

Dans cette partie, nous faisons le premier traitement de notre application, où on analysera les identifiants de l'utilisateur, si les identifiants sont corrects, il sera redirigé vers la page adéquate, soit HomeAdmin.jsp ou bien HomeUser.jsp, si les identifiants sont incorrects, la page sera réactualisée avec un message d'erreur.

2. Servlet RegisterUser :

Cette Servlet sera appelée si le visiteur n'a pas de compte, il devra remplir le Modal présent dans la page Index.jsp et les informations seront traitées dans cette page, son compte sera créé et il sera redirigé vers la page HomeUser.jsp, si des erreurs occurred, un message d'erreur sera afficher via une Alert.

3. Validate

Class dédiée à l'authentification, elle s'occupe de la récupération des identifiants depuis notre base de données, elle sera appelé lors de l'exécution de la Servlet Index.

4. Servlet d'ajout

Nous avons créé 3 différentes servlets pour ajouter des données dans notre base de données

1. **AddBook**, qui s'occupe de l'ajout des livres dans notre BDD.
2. **AddUser** qui s'occupe de la création des utilisateurs par un des admins.
3. **AddWriters** qui s'occupe de l'ajout des auteurs par un des admins.

Le traitement est standard, à chaque fois, nous récupérons les données insérées par l'utilisateur dans les JSP avec des « request.getParameter("exemple") », faisons la connexion à notre base de donnée grâce à

- Class.forName("com.mysql.jdbc.Driver");
- Connection c= (Connection);
- DriverManager.getConnection("jdbc:mysql://localhost:3306/Cawa","root","");
- Statement st = c.createStatement();

Afin de faciliter le traitement, l'ISSN est en autoIncrement, donc nous récupérons à chaque fois le max ISSN + 1 afin d'affecter le dernier ISSN + 1 présent dans notre base de données.

Nous exécutons la requête grâce à un objet ResultSet et le l'objet PreparedStatement et la méthode executeQuery() .

Puis nous insérons les données dans notre table grâce à une autre requête, avec un objet de type Statement et la méthode executeUpdate() ;

Pour finir par renvoyer l'administrateur vers la page JSP d'avant, soit

- AdminBooks
- AdminsUsers.
- AdminsWriters.

5. Servlet de suppression :

Nous avons créé 3 différents servlets pour supprimer des données dans notre base de données

- **DeleteBook**, qui s'occupe de la suppression des livres dans notre BDD.
- **DeleteUser** qui s'occupe de la suppression des utilisateurs par un des admins.
- **DeleteWriters** qui s'occupe de la suppression des auteurs par un des admins.

Les différents servlets récupèrent l'identifiant du champ du bouton cliqué grâce à un «request.getParameter("id")»,

Nous faisons notre connexion à la base de données. Grâce à un objet de type Statement et à la méthode executeUpdate, suivis de la requête Delete from ... where ... de SQL, nous supprimons l'id, et nous affichons de nouveau une des pages JSP

- AdminBooks
- AdminsUsers.
- AdminsWriters.

6. Servlet d'edit :

Nous avons créé 3 différents servlets pour modifier les données dans notre base de données

- **EditBook**, qui s'occupe de la modification des livres dans notre BDD.
- **EditUser** qui s'occupe de la modification des utilisateurs par un des admins.
- **EditWriter** qui s'occupe de la modification des auteurs par un des admins.

Les différents servlets récupèrent les nouveaux champs saisis par l'utilisateur dans les pages JSP grâce à des «request.getParameter("id") »

Nous faisons notre connexion à la base de données, grâce à un objet de type Statement et à la méthode executeUpdate, suivis de la requête Update TableName Set ColumnName ... where Id=id de SQL.

Nous exécutons la méthode executeUpdate grâce à un objet de type PreparedStatement et nous renvoyons l'admin vers une des JSP

- AdminBooks
- AdminsUsers.
- AdminsWriters.

Grâce à un response.SendRedirect pour lui afficher le nouveau contenu

5. Conclusion

Pour conclure, nous avons pu travailler sur un projet JEE en utilisant l'architecture MVC. Nous avons pu automatiser le traitement d'une bibliothèque, créé plusieurs utilisateurs, faire les différents traitements sur nos données, néanmoins nous aurions voulu améliorer notre projet en rajouter des images spécifiques pour chaque livre et en rajoutant un rafraichissement automatique aux pages après les différentes insertions.