

Gestion de bibliothèque

10.02.2022

—
TI 02

Boudraa Mohamed Ouadjih
Neggazi Mohamed Lamine



Table des matières

Introduction	2
Présentation du projet	2
Objectifs du projet	2
Conception de système	2
Présentation d'UML	2
Diagramme de cas d'utilisation :	3
Identification des acteurs :	3
Administrateur :	3
Utilisateur :	3
Domaine.	3
Titre.	3
Auteur.	3
Diagramme de cas d'utilisation générale :	3
Description textuelle de certain cas d'utilisation	4
Diagramme de classes	6
Schéma relationnel	6
Implémentation de système	7
Java 2 Enterprise Edition (J2EE) :	7
Java Server Pages Standard Tag Library (JSTL) :	7
Explication des page JSTL :	7
Captures des interfaces avec explications :	11
INDEX.JSP :	11
ADMIN.JSP :	11
AjouterLivre.jsp :	12
ListLivres.jsp :	13
RechercherLivre.jsp :	13
Captures de code mysql :	14
Conclusion:	15

Introduction

Nous vivons dans un monde qui devient de plus en plus relié aux outils informatiques qui sont devenus incontournables voir nécessaires et indispensables dans certains domaines.

En effet, aucun domaine n'est resté étranger à l'informatisation, une stratégie qui offre tant de services notamment les établissements supérieurs et aussi pour l'administration.

I. Présentation du projet

La bibliothèque est un moyen d'accès aux informations et œuvres. Pour cela et pour faciliter la tâche aux lecteurs nous mettons en place une solution web pour la recherche via la technologie JEE.

II. Objectifs du projet

L'objectif de ce rapport est d'implémenter en JEE une application web de la gestion d'une bibliothèque pour faciliter la recherche des livres aux lecteurs par différents critères (domaine, titre, auteur) et d'offrir à l'administrateur la possibilité d'enrichir la bibliothèque tout en sécurité. Et pour cela on a utilisé la plateforme JEE avec le model MVC.

Conception de système

I. Présentation d'UML

Pour atteindre notre objectif et répondre aux besoins des utilisateurs, il est nécessaire de choisir une bonne méthode qui facilite la description et l'analyse du système.

Afin de faire face à la complexité croissante du système informatique, nous avons choisi le langage de modélisation UML ». Ce modèle est défini comme une boîte à outil permettant d'améliorer progressivement la démarche de conception et facilite la compréhension de la solution.

UML (Unified Modelling Langage) se définit comme un langage de modélisation objet. C'est un support de communication des divers aspects (formels et visuels) d'un système d'information.



Pour notre application, nous on a opté pour le diagramme de cas d'utilisation et de classe qui sont les mieux placés pour montrer les fonctionnalités principales et la structure de notre système.

II. Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation est un modèle qui représente les fonctions du système de point de vue de l'utilisateur en décrivant ses propres besoins.

III. Identification des acteurs :

Plusieurs acteurs peuvent participer à la gestion de notre application selon les besoins. Dans notre système on coopère avec :

- Administrateur :

Celui qui gère l'application, il possède des privilèges d'administration du système tel que la gestion des utilisateurs, gestion des auteurs, gestion des livres, afficher la liste des livres et peut jouer le rôle d'utilisateur et ensuite rechercher un livre.

- Utilisateur :

L'utilisateur peut rechercher un livre par :

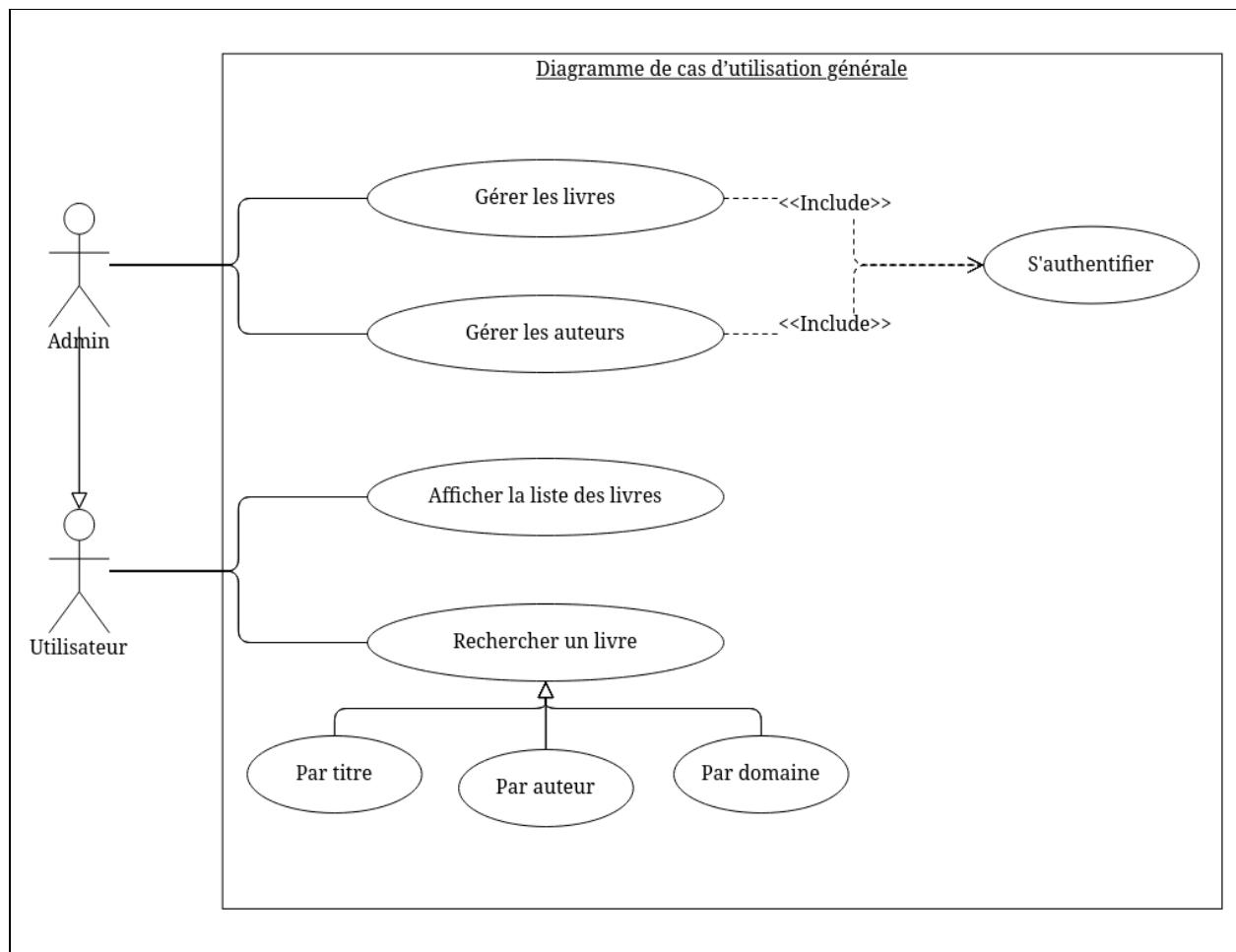
- Domaine.

- Titre.

- Auteur.

IV. Diagramme de cas d'utilisation générale :

Le diagramme suivant illustre une vue globale de notre application et les différentes fonctionnalités des acteurs.



V. Description textuelle de certain cas d'utilisation

Nom de cas	Ajouter un auteur.
Acteur	Administrateur.
Date	12/02/2021.
Responsable	Etudiant.
Version	1.0
Pré-condition	L'utilisateur doit s'authentifier.
Déclenchement	Le cas d'utilisation commence lorsque l'acteur choisit d'ajouter un auteur.



Enchaînement nominal	<ol style="list-style-type: none"> 1. Le système affiche le formulaire d'ajout. 2. L'acteur remplit le formulaire avec les renseignements du l'auteur et il valide. 3. Le système vérifie les champs. 4. Le système enregistre les informations correspond à cet auteur.
Enchaînement alternatif	Si le remplissage des informations n'est pas complet, le système demande de le compléter.
Post-condition	Un nouvel auteur est ajouté.

Table 1 - Description textuelle de cas « Ajouter un auteur ».

Nom de cas	Rechercher un livre.
Acteur	Administrateur, Utilisateur
Date	12/02/2021.
Responsable	Etudiant.
Version	1.0
Pré-condition	L'utilisateur doit s'authentifier.
Déclenchement	Le cas d'utilisation commence lorsque l'acteur choisit rechercher un livre.
Enchaînement nominal	<ol style="list-style-type: none"> 1. Le système affiche le formulaire de recherche soit par domaine, auteur ou bien par titre. 2. L'acteur remplit le formulaire avec les données du livre qu'il veut recherché et il valide. 3. Le système vérifie les champs. 4. Le système affiche les résultats.
Post-condition	La recherche est effectuée.

Table 2 - Description textuelle de cas « Rechercher un livre ».



VI. Diagramme de classes

Les diagrammes de classes jouent un rôle central dans l'analyse et le design orientés objet. Ils présentent un ensemble d'éléments de modèle statiques, leur contenu (structure interne) et leurs relations aux autres éléments. Les principaux éléments représentés dans un diagramme de classes sont les classes, les packages, les associations, les héritages et les dépendances.

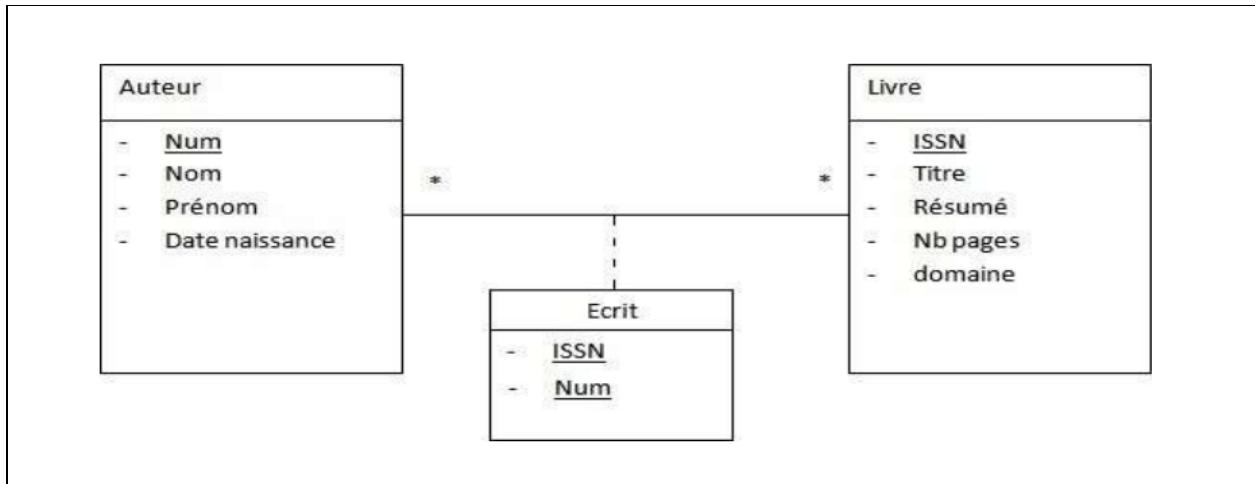


Figure 1 - Diagramme de classe.

Une classe est la description formelle d'un ensemble d'objets ayant une sémantique et des propriétés communes.

Un objet est une instance d'une classe. C'est une entité discrète dotée d'une identité, d'un état et d'un comportement que l'on peut invoquer. Les objets sont des éléments individuels d'un système en cours d'exécution.

VII. Schéma relationnel

- **Auteur** (Num, Nom, Prenom, Date_de_naissance).
- **Livre** (ISSN, #num, Titre, Résumé, Nbpage, Domaine).
- **Ecrit** (#ISSN, #num).



Implémentation de système

III. Java 2 Enterprise Edition (J2EE) :

Le terme Java EE » signifie Java Enterprise Edition, et était anciennement raccourci en « J2EE ». Il fait quant à lui référence à une extension de la plate-forme standard. Autrement dit, la plate-forme Java EE est construite sur le langage Java et la plateforme Java SE, et elle y ajoute un grand nombre de bibliothèques remplies tout un tas de fonctionnalités que la plate-forme standard ne remplit pas d'origine. L'objectif majeur de Java EE est de faciliter le développement d'applications web robustes et distribuées, déployées et exécutées sur un serveur d'applications.

IV. Java Server Pages Standard Tag Library (JSTL) :

La JavaServer Pages Standard Tag Library est un composant de la plate-forme JEE de développement. Elle étend la spécification JSP en ajoutant une bibliothèque de balises pour les tâches courantes, comme le travail sur des fichiers XML.

V. Explication des page JSTL :

```
<c:if test="${param.user != null}">
    <c:set var="user" value="${param.user}" scope="request" />
    <c:set var="pass" value="${param.pass}" scope="request" />
    <c:if test="${user == 'admin' && pass == 'admin'}">
        <c:set var="session" value="admin" scope="session" />
        <c:redirect url="Admin.jsp"></c:redirect>
    </c:if>
    <c:if test="${user != 'admin' || pass != 'admin'}">
        <c:remove var="session"/>
        <c:redirect url="index.jsp"></c:redirect>
    </c:if>
</c:if>
```

Figure 3 - Authentification de l'administrateur.



Ce code permet de se connecter à l'application à partir d'un formulaire d'authentification, dans lequel l'administrateur doit entrer l'utilisateur et le mot de passe correcte.

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<sql:setDataSource var="DS" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/bibliotheque" user="root" password="" />
<sql:update dataSource="${DS}" var="new">
    | INSERT INTO `auteur`(`nom`, `prenom`, `dateNaissance`) VALUES (?,?,?)
    | <sql:param value="${param.nom}" />
    | <sql:param value="${param.prenom}" />
    | <sql:param value="${param.dateNaissance}" />
</sql:update>
<c:redirect url ="Listateurs.jsp"/>
```

Figure 4 - Ajouter un auteur.

Ce code permet d'ajouter un auteur dans la base des données bibliothèque. en utilisant une requête préparée pour garantir la sécurité des inputs.

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<sql:setDataSource var="DS" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/bibliotheque" user="root" password="" />
<sql:update dataSource="${DS}" var="new">
    | INSERT INTO `livre`(`titre`, `resume`, `nbPage`, `domaine`, `num`)VALUES(?,?,?,?,?,?)
    | <sql:param value="${param.titre}" />
    | <sql:param value="${param.resume}" />
    | <sql:param value="${param.nbPage}" />
    | <sql:param value="${param.domaine}" />
    | <sql:param value="${param.num}" />
</sql:update>
<c:redirect url ="Listlivres.jsp"/>
```

Figure 5 - Ajouter un livre.

Ce code permet d'ajouter un livre dans la base des données bibliothèque. En utilisant une requête préparée pour garantir la sécurité des différents inputs, On utilise aussi l'attribut num qui est un clé étrangère (foreign key) pour assurer la cohérence entre les tables auteur et livre .

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>

<c:if test="${session == null }">
|   <c:redirect url="index.jsp"></c:redirect>
</c:if>

<sql:setDataSource var="DS" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/bibliotheque" user="root" password="" />
<sql:query dataSource="${DS}" var="result">
|   select * from auteur;
</sql:query>
```

```
<legend>liste des Auteurs</legend>
<br><br>
<table class="table table-bordered table-responsive " style="width: 40em; overflow: hidden; background-color: #rgba(160, 164, 165,>
    <thead>
        <tr>
            <th scope="col">Num</th>
            <th scope="col">Nom</th>
            <th scope="col">Prenom</th>
            <th scope="col">Date Naissance</th>
        </tr>
    </thead>
    <tbody>
        <c:choose>
            <c:when test="${result != null}">
                <c:forEach var="row" items="${result.rows}">
                    <tr>
                        <td>${row.num}</td>
                        <td>${row.nom}</td>
                        <td>${row.prenom}</td>
                        <td>${row.dateNaissance}</td>
                    </tr>
                </c:forEach>
            </c:when>
            <c:otherwise>
                <c:out value="Pas d'auteur disponible !"/>
            </c:otherwise>
        </c:choose>
    </tbody>
</table>
```

Figure 6 - Liste des livres.

Ce code permet d'afficher la liste des livres dans la page listeLivre.jsp.

Chaque page contient un bloc de code qui vérifie la session d'utilisateur.

On récupère la liste des auteurs pour afficher pour chaque livre le nom de son auteur.

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<c:set var="query" value="${param.q}" scope="session" />
<c:set var="type" value="${param.type}" scope="session" />
<sql:setDataSource var="DS" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/bibliotheque" user="root" password="" />
<c:choose>
    <c:when test="${type == 'Auteur'}">
        <sql:query dataSource="${DS}" var="resultat">
            | SELECT * FROM livre where num = '${query}';
        </sql:query>
    </c:when>
    <c:when test="${type == 'Livre'}">
        <sql:query dataSource="${DS}" var="resultat">
            | SELECT * FROM livre where titre = '${query}';
        </sql:query>
    </c:when>
    <c:when test="${type == 'Domaine'}">
        <sql:query dataSource="${DS}" var="resultat">
            | SELECT * FROM livre where domaine = '${query}';
        </sql:query>
    </c:when>
    <c:otherwise>
        <c:choose>
            <c:when test="${type == null}">
                <sql:query dataSource="${DS}" var="resultat">
                    | SELECT * FROM livre;
                </sql:query>
            </c:when>
        </c:choose>
    </c:otherwise>
</c:choose>

```

```

    <c:otherwise>
        <sql:query dataSource="${DS}" var="resultat">
            | SELECT * FROM livre where titre = '${query}';
        </sql:query>
    </c:otherwise>
</c:choose>
</c:otherwise>
</c:choose>

```

Figure 7 - Rechercher un livre.

Ce code permet de faire la recherche des livres par (auteur , titre ,domain) puis il affiche le résultat dans la page resultatRecherche.jsp

```

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:remove var="session"/>
<c:redirect url="index.jsp"></c:redirect>

```

Figure 8 - Logout.

Ce code permet à l'administrateur de se déconnecter en supprimant la session et la rediriger vers la page index.jsp.

Captures des interfaces avec explications :

INDEX.JSP :

l'interface qui permet les visiteurs de faire la recherche (par auteur,titre (défaut) ou domaine grâce au drop down) sans authentifier, si le visiteur est un admin, il peut authentifier avec son login et mot de passe pour faire d'autres tâches qui se trouve au niveau de page admin.jsp.

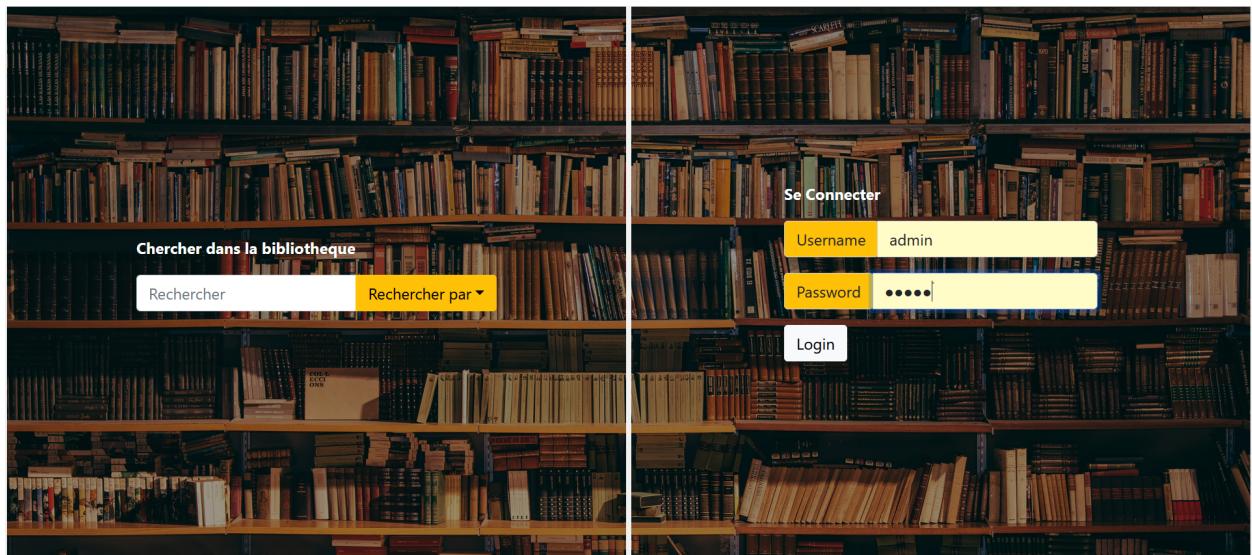


Figure 9 - Page index.jsp

ADMIN.JSP :

Admin.jsp : cette page permet à l'administrateur de créer un auteur ou ajouter un livre dans la bibliothèque à travers des formulaire qui se trouve au niveau des page : ajouterLivre.jsp, ajouterAuteur.jsp et elle permet aussi de consulter la liste des livre dans la bibliothèque et la liste des auteurs et Chercher un livre à travers la barre de recherche en haut.

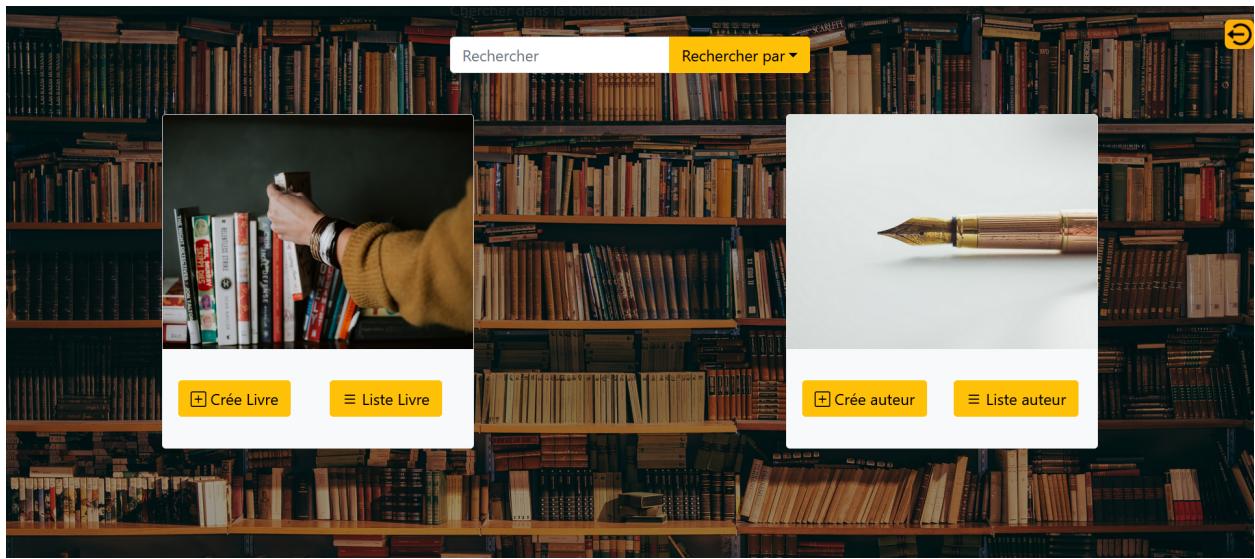


Figure 10 - Admin.jsp

AjouterLivre.jsp :

Cette page permet à l'administrateur d'ajouter un nouveau livre à la bibliothèque à travers un formulaire qui contient les champs des informations des livres.

Titre	J2EE
Nombre de pages	120
Domaine	Technologie
Resume du livre	Ce livre contient des informations et des exercices sur le framework J2EE
Auteur de livre	amine neggazi

Figure 11 - AjouterLivre.jsp

ListLivres.jsp :

Cette page permet à l'administrateur de consulter tous les livres de la bibliothèque.

The screenshot shows a table titled "Liste des livres" displaying two book entries. The columns are labeled: ISSN, titre, résumé, nbpage, domaine, and Auteur. The first book has ISSN 5, title "Ce que le jour doit à la nuit", summary about a boy named Younès, 437 pages, Histoire domain, and author auteurName. The second book has ISSN 7, title "développement web html css js", summary about a practical and theoretical part, 40 pages, Technologie domain, and author auteurName.

ISSN	titre	résumé	nbpage	domaine	Auteur
5	Ce que le jour doit à la nuit	Algérie, années 1930. Venu de la campagne, Younès, 9 ans, emménage avec sa famille à Oran, après que la terre de son père a été incendiée, puis saisie par un créancier. Mohamed, son oncle pharmacien	437	Histoire	auteurName
7	développement web html css js	Ce livre contient une partie pratique et une partie théorique	40	Technologie	auteurName

Figure 12 - ListLivres.jsp

RechercherLivre.jsp :

Cette page permet à tous les utilisateurs de l'application de faire une recherche des livres dans la bibliothèque soit par auteur ou bien domaine ou bien titre et bien sur afficher le résultat dans la page resultatRecherche.jsp..

The screenshot shows a search interface with a dropdown menu for "Rechercher par" set to "Domaine". The search term "technologie" is entered. Below the search bar, the results are displayed in a table titled "Résultat de recherche" with columns: ISSN, Titre, Nombre de pages, and Domaine. Three books are listed: one with ISSN 7, one with ISSN 8, and one with ISSN 9, all categorized under the Technologie domain.

ISSN	Titre	Nombre de pages	Domaine
7	développement web html css js	40	Technologie
8	JSTL	12	Technologie
9	J2EE	120	Technologie

Figure 13 - ResultatRecherche.jsp

Captures de code mysql :

```
--  
-- Table structure for table `auteur`  
  
--  
  
DROP TABLE IF EXISTS `auteur`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
CREATE TABLE `auteur` (  
  `num` int(11) NOT NULL AUTO_INCREMENT,  
  `nom` varchar(30) NOT NULL,  
  `prenom` varchar(30) NOT NULL,  
  `dateNaissance` varchar(20) NOT NULL,  
  PRIMARY KEY (`num`)  
) ENGINE=InnoDB AUTO_INCREMENT=23 DEFAULT CHARSET=latin1;  
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Table structure for table `ecrit`  
  
--  
  
DROP TABLE IF EXISTS `ecrit`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
CREATE TABLE `ecrit` (  
  `ISSN` int(4),  
  `num` int(4),  
  FOREIGN KEY (`ISSN`) REFERENCES livre(`ISSN`),  
  FOREIGN KEY (`num`) REFERENCES auteur(`num`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;  
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--  
-- Table structure for table `livre`  
  
--  
  
DROP TABLE IF EXISTS `livre`;  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
CREATE TABLE `livre` (  
  `ISSN` int(11) NOT NULL AUTO_INCREMENT,  
  `num` int(11) NOT NULL,  
  `titre` varchar(30) NOT NULL,  
  `resume` varchar(200) NOT NULL,  
  `nbPage` int(4) NOT NULL,  
  `domaine` varchar(50) NOT NULL,  
  PRIMARY KEY (`ISSN`),  
  FOREIGN KEY (`num`) REFERENCES auteur(`num`)  
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=latin1;  
/*!40101 SET character_set_client = @saved_cs_client */;
```



Conclusion:

Après la réalisation de ce projet l'an passé qui nous a permis d'apprendre et de comprendre les étapes de conception et de développement d'une application de gestion d'une bibliothèque basée sur la technologie J2EE en utilisant des servlets.

Cette année on a réalisé le même projet mais avec une nouvelle technologie JSTL avec des tag library (core , sql...) qui permet d'utiliser les différentes fonctionnalités et tout ça pour faciliter et sécuriser le développement de cette application.