**Rapport JEE**

Page de garde

**Sommaire**

[I. Préambule 3](#_Toc97988719)

[II. Structuration de code, découplage et injection de dépendance 3](#_Toc97988720)

[III. Utilisation de Servlet et/ou JSP pour l’implémentation du module de service 3](#_Toc97988721)

[IV. Transformation objet-relationnel 3](#_Toc97988722)

[V. Variables sessions 4](#_Toc97988723)

[VI. Concurrence 4](#_Toc97988724)

[VII. Efficacité des requêtes d’accès à la base de données 5](#_Toc97988725)

1. Préambule

Nous avons convenus avec M. Ouziri que nous pouvions rajouter deux nouvelles méthodes dans l’interface Médiathèque.

La première méthode qui se nomme consulterDocuments() nous permet de consulter tous les documents de la bibliothèque.

La deuxième méthode qui se nomme consulterDocumentsEmprunt(String pseudo) nous permet de visualiser tous les documents qu’un utilisateur a emprunté.

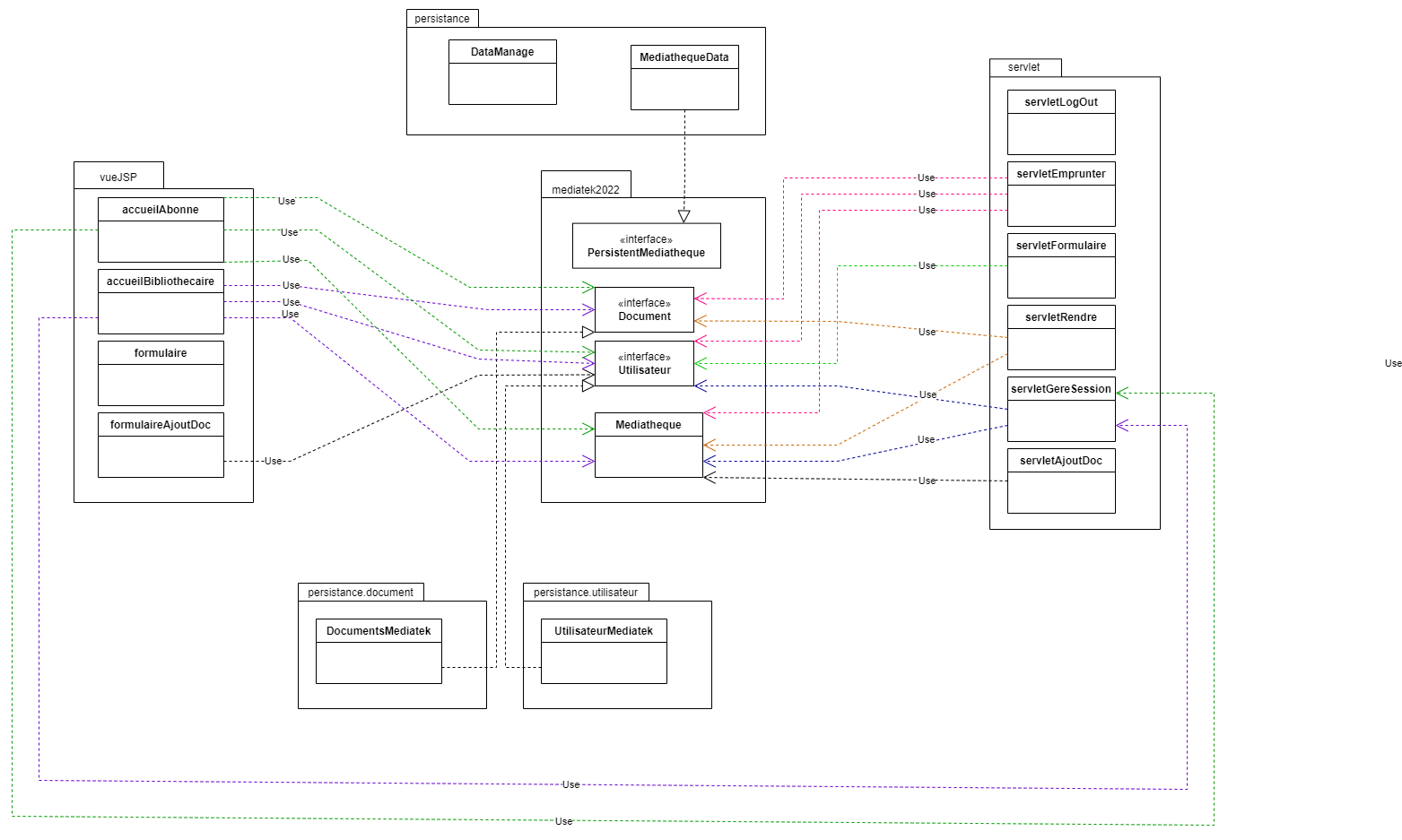
Concernant la définition de ces méthodes, nous avons suivi le même schéma que pour les méthodes précédentes : nous avons ajouter ces méthodes dans l’interface *PersistentMediatheque* et nous les avons aussi ajoutés dans la classe *Mediatheque* de la manière suivante :

Une image contenant texte

Description générée automatiquement

1. Structuration de code, découplage et injection de dépendance

Voici le diagramme d’architecture de notre projet. On peut voir les différentes liaisons entre les packages et les classes.



1. Utilisation de Servlet et/ou JSP pour l’implémentation du module de service

Nous avons choisi d’utiliser des Servlet et des JSP pour implémenter les différents services disponibles de notre application.

Nous utilisons des Servlets à la suite d’une requête HTML, par exemple, lorsqu’un utilisateur souhaite se connecter, ou bien qu’un bibliothécaire ajoute un nouveau document dans la médiathèque. Les requêtes ont généralement lieu lors d’un changement de page.

Ces servlets récupèrent les informations grâce aux méthodes GET et POST, ce qui nous permet d’effectuer les actions demandées.

Nous utilisons des JSP pour effectuer les autres actions, celles qui ne sont pas issues d’une requête HHTP.

Grâce aux JSP, nous pouvons afficher le contenu de la base de données (tous les documents d’un utilisateurs, ou de la médiathèque) dès le chargement (ou le rechargement) de la page. Cela permet d’avoir un site dynamique qui prend directement en compte un changement (un emprunt de document par exemple).

1. Transformation objet-relationnel

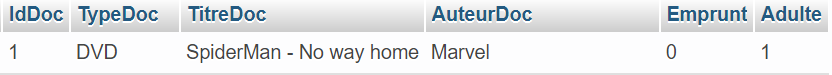
Concernant le passage d’objets de l’application Document et Utilisateur aux tables relationnelles de la base de données, nous effectuons se changement à l’aide de requêtes SQL.

Dans notre base de données, nous avons défini les documents et les utilisateurs avec différents attributs.

Exemple d’utilisateur :



Exemple de document :



Dans notre code Java, nous avons défini des classes pour les documents et les utilisateurs. Ces tables ont respectivement les mêmes attributs que ceux de la base de données.

De cette manière, lors d’une insertion ou d’une requête pour récupérer ces objets, nous avons une correspondance, ce qui permet de passer des objets à la base de données très facilement.

1. Variables sessions

Dans le cadre de notre application, nous avons mis en place des variables de sessions. Cette variable est créée. Elle continent les informations sur l’utilisateur comme son pseudo, son âge, son numéro. Le mot de passe n’est pas stocké pour des raisons de sécurité.

Les éléments stockés permettent de gérer les emprunts de documents : avec le pseudo, on peut retrouver l’utilisateur dans la base de données et avec son âge, on peut lui interdire l’emprunt de documents qui lui sont interdits.

Une session est ouverte lorsqu’un utilisateur se connecte. Lors de cette ouverture, les différentes informations nécessaires sont stockées. La session est maintenue tout le temps où l’utilisateur est connecté, ce qui nous permet de suivre l’utilisateur lorsqu’il change de page.

La session est fermée lorsque l’utilisateur se déconnecte.

1. Concurrence

Nous avons remarqué la présence de ressources critiques dans notre code : en effet, lorsque deux utilisateurs sont connectés en même temps, il peut y avoir un problème lors de l’emprunt de documents. Si deux utilisateurs empruntent le même document en même temps, quel utilisateur pourra réellement emprunter le document ?

Pour résoudre ce problème, nous avions pensé à mettre en place des sections critiques mais après réflexion, nous n’en n’avons pas mis.

En effet, nous avons appris dans nos cours de gestion de base de données que les bases tel que MySQL ou Oracle gèrent elles-mêmes la concurrence lors de l’accès à des ressources.

Vu que nos données sont stockées dans une base de données MySQL, la concurrence est directement gérée par la base de données. La base de données s’occupe d’attribuer un document à un utilisateur lorsque plusieurs utilisateurs souhaitent l’emprunter en même temps.

1. Efficacité des requêtes d’accès à la base de données

Lorsqu’on lance notre serveur, la première page qui s’affiche est celle de notre formulaire de connexion.

Une image contenant texte

Description générée automatiquement

L’utilisateur vient ensuite renseigner son pseudo et son mot de passe.

Ces informations sont récupérées et une requête vers la base de données est effectuée : s’il existe un couple <pseudo, mot de passe> qui correspond à celui qui vient d’être renseigné, alors la connexion à lieu et l’utilisateur arrive sur sa page de profil.

Si l’utilisateur n’est pas reconnu, alors il reste sur cette même page et doit à nouveau compléter les deux champs obligatoires.

En fonction du rôle de l’utilisateur, sa page d’accueil n’est pas la même car les deux rôles (abonné et bibliothécaire) n’ont pas les mêmes actions disponible.

Voici à quoi ressemble les pages d’accueil : 



*Page d’accueil d’un abonné Page d’accueil d’un bibliothécaire*

Lors de l’arrivé sur ces pages, la liste des documents est déjà chargée : l’abonné a accès à la liste des documents qu’il a emprunté et la liste des documents empruntables ; le bibliothécaire a accès à la liste des documents de la médiathèque.

L’utilisation de requêtes précompilées permet de traduire les requêtes en SQL une seule fois. Sachant que ces requêtes sont exécutées à plusieurs reprises, c’est une bonne manière de rendre le code plus « léger » lors de l’utilisation de nos services.