

PROJET JAVAEE : MEDIATEK

Rapport de développement d'une application
web JAVAEE à destination d'une médiathèque

DUT 2ÈME ANNÉE - MARS 2022

Développeurs

CLÉMENT PROST

CLÉMENT MAUPERON



OBJECTIFS

L'objectif de ce projet était de développer une application Java destinée à une médiathèque et basée sur un package stable **mediatek2022** fournis au format *JAR*. L'application devait:

- intégrer une interface sous la forme d'un site web grâce à aux servlets **Java EE** ;
- gérer la persistance des données de la médiathèque sur une base de données, et via un connecteur **JDBC**.

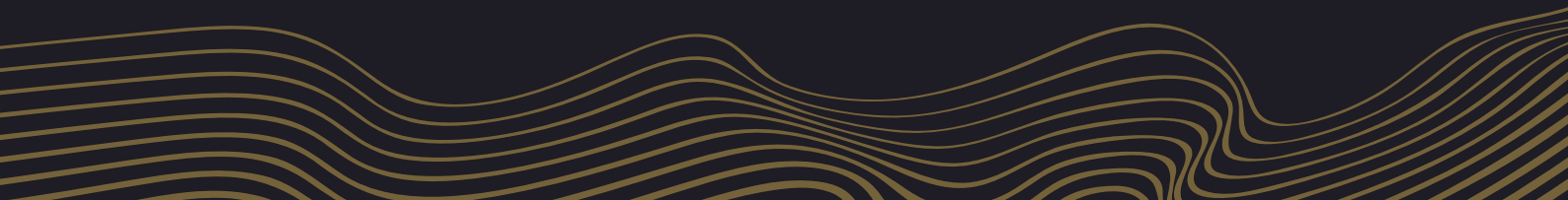
*Ces deux parties de l'application devaient être **indépendantes** l'une de l'autre et utiliser uniquement les interfaces et la classe Médiatheque de la bibliothèque **mediatek2022**.*


DONNÉES PERSISTANTES

Les **documents** sont les principales données que l'application doit manipuler et faire persister. Ils sont stockés dans la table *documents* et identifiés par un numéro unique (*id*). L'**emprunt** d'un document se traduit par l'enregistrement du nom de l'utilisateur dans la colonne *emprunteur* (*clé étrangère*). Une valeur nulle dans cette colonne signifie que le document est disponible.

Nous avons implémenté deux types de documents : les **DVD** et les **BluRay**. Ces documents possèdent trois champs d'informations supplémentaires : le titre du film, le réalisateur et un indicateur spécifiant s'il s'agit d'un film pour adulte. Pour anticiper l'ajout d'autres types de documents (*comme des livres*), le champs du réalisateur est stocké dans la colonne *auteur* de la table *SQL*.

L'implémentation des documents permet une **sérialisation des données du document** en un objet String, via la méthode *toString*, ce qui permet aux services de lire ces données. Ce système permet de contourner les limitations de l'interface originale tout en respectant la **contrainte de découplage** (cf. *services.api.APIDoc*).





L'application devait également gérer les **utilisateurs** de la médiathèque, et leur authentification via un **nom d'utilisateur** et un **mot de passe**. Ils sont divisés en deux catégories :

- les **bibliothécaires** qui peuvent ajouter de nouveaux documents, consulter la liste des documents et leurs disponibilités ;
- les **abonnés** qui peuvent consulter les documents disponibles, les emprunter, et retourner les documents qu'ils ont empruntés.

Les utilisateurs sont rassemblés dans la table *utilisateurs* et leur type est défini par un booléen (colonne *bibliothecaire*). L'*âge* et le *statut d'abonnement* sont également enregistré pour les abonnés.

TECHNOLOGIES

Au cours du développement de l'application, nous avons utilisé la version d'**Apache Tomcat 9** fournie sur Moodle. Nous avons opté pour une combinaison entre les **servlets**, pour les opérations de l'application (*rôle de contrôleur*), et les **JSP**, pour l'affichage dynamique de l'interface (*rôle de vue*). Nos fichiers JSP utilisent également la librairie **JSTL** pour faciliter les opérations d'affichage. Pour la base de données, nous avons utilisé **MariaDB**, via **XAMPP**.

STRUCTURE DU CODE

Notre application est donc divisée en deux packages Java que sont **persistance**, qui s'occupe de la persistance des données, et **services** qui contient les services de l'application (*servlet*).

Le premier contient la classe **MediathequeData**, qui implémente l'interface **PersistentMediatheque** et s'auto-déclare comme instance de gestion des données à l'unique instance de **Mediatheque**. Il contient également les implémentations des interfaces **Document** et **Utilisateur** dans les packages respectifs **documents** et **utilisateurs**.

La connexion à la base de donnée est effectuée à l'instanciation de la classe **MediathequeData** (démarrage de l'application) et sa validité est vérifiée avant chaque requête (cf. méthode **connexion_valide**). Chaque opération (*getUser, getDocument, etc...*) correspond à une **requête SQL précompilée** afin d'être optimisée.

Les scripts SQL de création et de population de la base de données sont situés dans le dossier sql.

Le deuxième package contient les **services HTTP** de l'application. Ils sont tous basés sur les quatre classes abstraites du package **services.base** :

- **Service**, qui étend la classe abstraite **HttpServlet** et initialise le service, la **session HTTP** et la **délégation au fichier JSP** ;
- **ServiceAuthentification**, qui étend la classe **Service** et gère l'authentification de l'utilisateur avant le chargement de la page ;
- **ServiceAbonne**, qui étend la classe **ServiceAuthentification** et accepte les connexions des utilisateurs de type **Abonné**. Il peut être configuré pour accepter les utilisateurs de type **Bibliothécaire**.
- **ServiceBibliothécaire**, qui étend la classe **ServiceAuthentification** et n'accepte que les utilisateurs de type **Bibliothécaire**.

Un service étendant une des trois dernières classes redirigera les utilisateurs anonymes vers le service de connexion.

Les fichiers **JSP** sont situés dans le dossier **jsp** (lui-même situé dans le dossier **WEB-INF** de l'application). Un service peut également définir une **procédure POST** qui sera exécutée avant le chargement de la page.

L'authentification de l'utilisateur est enregistrée dans la **session HTTP** à sa connexion depuis le service **services.Connexion**, et est effacée par le service **services.Deconnexion**. La session **HTTP** est également initialisée (*récupération ou création si besoin*) pour les services ne nécessitant pas d'authentification.

*Le service d'authentification n'utilise pas de **cookie** car, l'application étant destinée à être utilisées sur des postes locaux et en accès libre pour les abonnés, nous avons conclu que cela n'était pas utile.*

DÉCOUPLAGE & APIDOC

La **contrainte de découplage** imposant une non-dépendance mutuelle des deux packages et les interfaces définies dans *mediatek2022* proposant peu de méthode de récupération des informations, nous avons décidé d'utiliser un **système symétrique de sérialisation des métadonnées** (ex : *identifiant du document, titre, auteur, etc.*) :

- dans le package de persistance des données, les implémentations de l'interface document sont **sérialisables** grâce à la méthode *toString*. Les métadonnées du document sont concaténées et **séparées par une chaîne constante** (`//`) ;
- dans la package des services, la classe **APIDoc** du package *api* implémentent des méthodes utilisant cette méthode de sérialisation (*meta*, *estEmprunteur*), ainsi que d'autres méthodes de contournement (*tousLesDocuments*, *emprunts*).

ACCÈS CONCURRENTIEL

L'instance unique de *mediatek2022.Mediatheque*, auxquels accèdent tous les services grâce à la constante *Service.MEDIATHEQUE*, est manifestement **la ressource critique de cette application**. De ce fait, toutes les opérations relatives à la médiathèque sont effectuées dans des **sections synchronized** (sur le singleton de *Mediatheque*).

AMÉLIORATIONS

Notre application présente plusieurs axes d'améliorations :

- un **hashage des mots de passes** enregistrés ;
- une intégration de la **vérification de l'adresse IP** du client, qui doit appartenir au réseau local de la médiathèque ;
- la possibilité d'**enregistrer un cookie de connexion** spécifique à un poste précis pour les bibliothécaire ;
- une page de **consultation et gestion des abonnements** dans l'espace bibliothécaire.