

MÉDIATHÈQUE

projet de fin de cursus
formation Développeur Fullstack Java

Camélia **De Sousa**
Agathe **Ghafari**



Camélia De Sousa

<https://github.com/Miambox/Mediatheque>

Agathe Ghafari

<https://github.com/Sanghxa/ProjetFormationFullstack2022>



Plan



///// Spécifications fonctionnelles

----- Repository

"""" Service

}}}}} Prochaines étapes



Spécifications fonctionnelles



Use case

Login via

- email
- mot de passe

Catalogue du stock

- voir TOUT
- voir par type
(*dvd, cd, livre*)
- voir items
- disponibles
- voir nouveautés

Emprunts

- voir emprunts
- rendre emprunts
- faire emprunt



Règles Métier

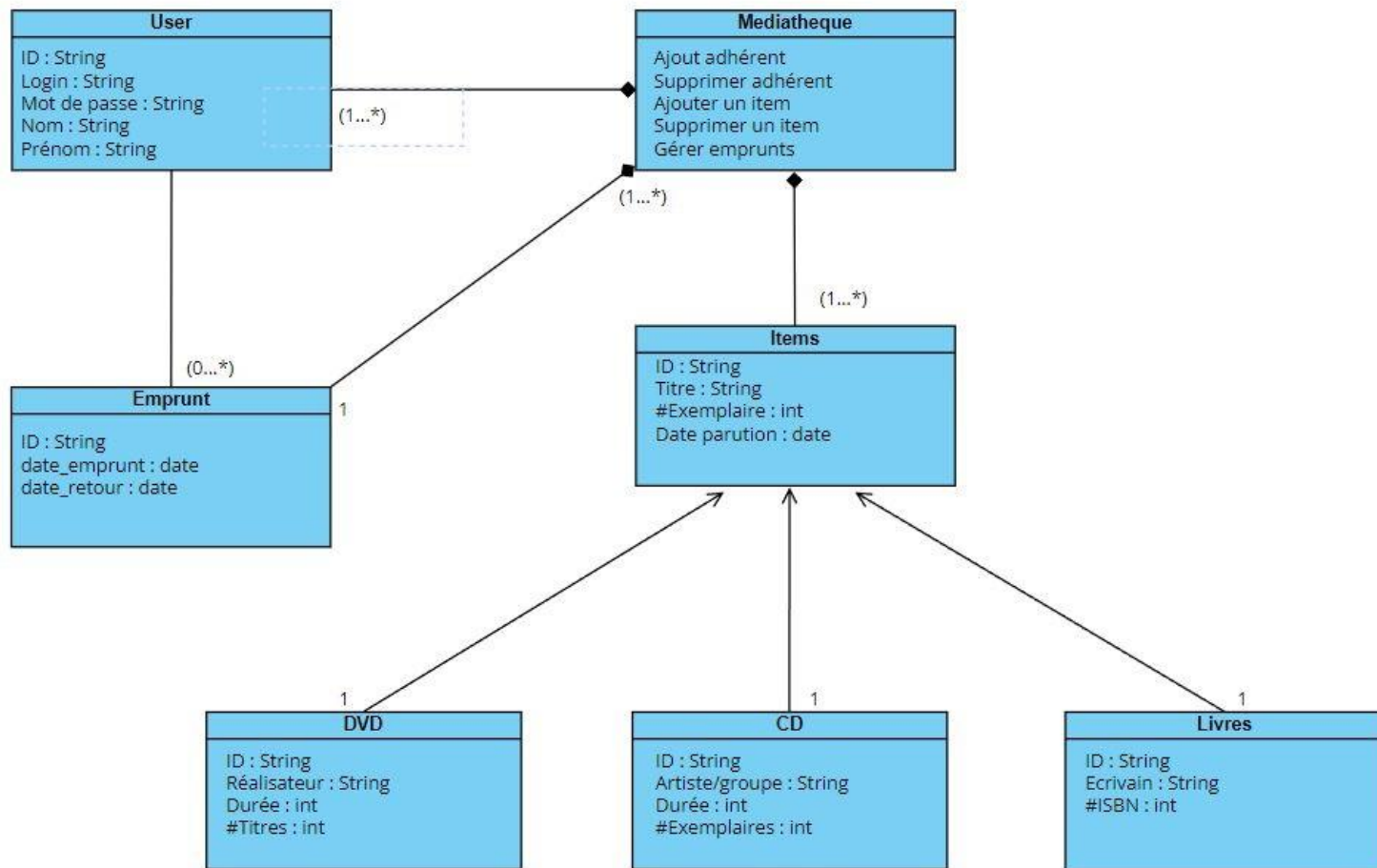
Contraintes :

- pas plus de 3 items empruntés
simultanément par utilisateur
- items doivent être rendus sous 7 jours



Structure

uml





Outils

JUnit 

Outils :

- SpringBoot : développement
- JPA : organisation / gestion des relations
- JUnit : tests unitaires
- Hibernate : simulation de la BD

-
- Swagger : test requêtes HTTP
 - API RESTful



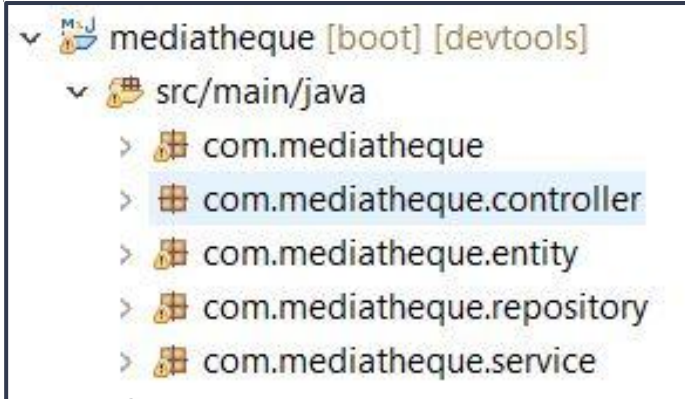
HIBERNATE





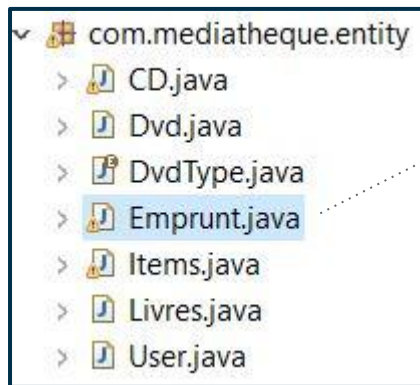
Structure de l'application

Couche	Objectif
controller	Réceptionner la requête et fournir la réponse
service	Exécuter les traitements métiers
repository	Communiquer avec la source de données
model	Contenir les objets métiers





Exemple de code d'Entity



```
@Entity
@Table(name="emprunt")
public class Emprunt {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "emprunt_sequence_generator")
    @SequenceGenerator(name = "emprunt_sequence_generator", allocationSize = 1)
    private Long id;

    @Column(name= "date_emprunt", nullable=false)
    private LocalDateTime date_emprunt;

    @Column(name= "date_retour", nullable=false)
    private LocalDateTime date_retour;

    @ManyToOne
    @JoinColumn(name="utilisateur_id", referencedColumnName = "id")
    private User user;

    @ManyToMany
    @JoinColumn(name="items_id")
    private List<Items> items;





}
```



REPOSITORY



Exemple de code de Repository

▼  com.mediatheque.repository
 >  EmpruntRepository.java
 >  ItemsRepository.java
 >  UserRepository.java

```
package com.mediatheque.repository;

import java.util.*;
import org.springframework.*
import com.mediatheque.entity.Emprunt;
import com.mediatheque.entity.User;

@Repository
public interface EmpruntRepository extends JpaRepository<Emprunt, Long> {
    //Avoir un user avec son ID + tous les emprunts associés
    @Query("SELECT e FROM Emprunt e WHERE e.user= :user")
    List<Emprunt> findEmpruntById(User user);
}
```

Démonstration tests

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.



SERVICE



Emprunt

Service

FAIRE UN EMPRUNT

- Vérifications
 - utilisateur existe ?
 - item existe ? disponible ?
 - items déjà empruntés + items à emprunter < 3
- Validation Emprunt
 - décrémentation nombre exemplaires dans table items
 - création emprunts dans table emprunts

RENDRE UN EMPRUNT

- Vérifications
 - utilisateur existe ?
 - emprunt existe ? déjà rendu ?
- Validation retour
 - création date retour (*table emprunt*)
 - incrémentation nombre exemplaires (*table items*)



PROCHAINES ÉTAPES



Back-end

- 1) Définition de l'API Rest
- 2) Mise en place de la sécurité
- 3) Test



Front-end

- 1) Home
- 2) Login
- 3) Catalogue
- 4) Panier
- 5) Emprunts



MERCI DE VOTRE ATTENTION



ANNEXES

AIDES



Repository c'est quoi

. Après création entité → il faut implémenter le code qui déclenche les actions pour communiquer avec la BDD.

. code fait requête à la bdd, puis le résultat nous est retourné sous forme d'instance d'objet: le code à implémenter est une INTERFACE

/!\ une interface ne contient pas de code
⇒ @Repository (annotation sur l'interface)

- @Repository : indique que la classe est un bean et que son rôle est de communiquer avec une source de données
- bean :
<https://www.axopen.com/blog/2019/02/java-spring-les-beans/>

(la classe fournie doit être annotée @Entity)



Mise en place interface Repository (exemple)

```
@Repository  
public interface UtilisateurRepository  
extends JpaRepository<Utilisateur, Long> {  
  
}
```

(Long parce que c'est le type de notre clé
primaire)

<https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/repository/CrudRepository.html>



INFOS REPOSITORY

<https://openclassrooms.com/fr/courses/6900101-creez-une-application-java-avec-spring-boot/7078015-creez-un-controlleur-rest-pour-gerer-vos-donnees>

JPA-JPQL

<https://isolution.pro/fr/t/jpa/jpa-jpql/jpa-jpql>



Test Repository

<https://reflectoring.io/spring-boot-data-jpa-test/>



Ajouter swagger ui dans les dépendances

```
<dependency>
```

```
<groupId>org.springdoc</groupId>
```

```
<artifactId>springdoc-openapi-ui</artifactId>
```

```
<version>1.6.12</version>
```

```
</dependency>
```

- ajouter:

donne accès à Swagger UI via
`localhost:8080/swagger-ui/index.html`



Application properties

```
spring.datasource.url=jdbc:h2:mem:testdb
```

```
spring.h2.console.enabled=true
```

```
spring.jpa.hibernate.ddl-auto=create-drop
```

```
spring.jpa.show-sql=true
```