Ticket management

|  |  |
| --- | --- |
| **Project** | OpenClassRooms Développeur d'application – Java  Projet 10 Améliorez le système d’information de la bibliothèque |
| **Git repository** | <https://github.com/FredLeroux/P10_OC_FLE> |
| **Ticket number** | 1 |
| **Issue number** | 5 |
| **Type** | Feature |

Table des matières

[Ticket details 1](#_Toc67484314)

[Title 1](#_Toc67484315)

[Applicant 1](#_Toc67484316)

[Request 1](#_Toc67484317)

[Definition 1](#_Toc67484318)

[Management rules 1](#_Toc67484319)

[Management rules on web application 2](#_Toc67484320)

[Implementation 2](#_Toc67484321)

[Database update 2](#_Toc67484322)

[Tables 2](#_Toc67484323)

# Ticket details

## Title

Ajoutez un système de réservation d’ouvrages

## Applicant

The customer

## Request

### Definition

Nous souhaiterions mettre en place un système de réservation de livres pour les usagers.

Le principe est le suivant :

L’usager peut vouloir emprunter un livre dont aucun exemplaire n’est actuellement disponible (ils sont tous en cours de prêt). Il a donc la possibilité de réserver l’ouvrage. Le premier à avoir réservé aura la priorité de la réservation dès le retour d’un exemplaire. Chaque ouvrage (titre, pas exemplaire) aura donc une liste d’attente de réservation. Dès qu’un exemplaire est de retour à la bibliothèque, un mail est envoyé au premier usager de la liste et il dispose de 48h pour venir le chercher. Passé ce délai, sa réservation est annulée et un mail est envoyé au suivant, etc.

### Management rules

* Tous les types d’ouvrages peuvent être réservés.
* La liste de réservation ne peut comporter qu’un maximum de personnes correspondant à 2x le nombre d’exemplaires de l’ouvrage.
* Il n’est pas possible pour un usager de réserver un ouvrage qu’il a déjà en cours d’emprunt
* Quand un usager emprunte le livre réservé ou si le délai de 48h après la notification est dépassé, sa réservation est supprimée de la liste et le processus recommence pour l’usage suivant dans la liste

### Management rules on web application

* Lors de la recherche d’ouvrage, pour ceux indisponibles, il doit y avoir la date de retour prévue la plus proche et le nombre de personnes ayant réservé l’ouvrage. Si la liste d’attente de l’ouvrage n’est pas complète, il doit pouvoir demander une réservation. Il sera alors ajouté à la liste d’attente
* L’usager doit pouvoir avoir une liste des réservations qu’il a en cours avec pour chaque ouvrage la prochaine date de retour prévue et sa position dans la liste d’attente
* L’usager doit pouvoir annuler une reservation

# Implementation

## Database update

### Tables

#### library\_book\_reservation

Add library\_book\_reservation on LIBRARY database.

##### Table aim

Stock book reservation information, manage book reservation, used to display reservation information, used to send notification.

##### Table definition

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Constrains** |
| id | Integer | NotNull / Unique |
| notification\_date | String | N/A |
| canceled\_status | Boolean | NotNull |
| priority | Integer | NotNull |
| library\_book\_fk | Integer | NotNull |
| library\_customer\_fk | Integer | NotNull |

##### Request cover

notification\_date = 48 Hours delay

canceled\_status = book loaned or out of delay, customer reservation canceling

priority = loop on next customer (if there is one), position on waiting list

library\_book\_fk = all related book information, all books can be reserved

library library\_customer\_fk = all related customer information and restriction, customer reservation canceling

##### Annex

SQL power architect :

* \P7\_DOC\mpd \Library\_mpd\_v1.2.architect
* \P7\_DOC\mpd \Library\_mpd\_v1.2.pdf

SQl file :

* \P7\_DB\Ticket\_1\_Update\_DB\_v1.2.sql

#### ~~library\_book\_reservation\_number~~

##### ~~Table aim~~

~~Manage reservation limit, stock number of reservation ongoing by book.~~

##### ~~Table definition~~

|  |  |  |
| --- | --- | --- |
| **~~Column Name~~** | **~~Type~~** | **~~Constrains~~** |
| ~~id~~ | ~~Integer~~ | ~~NotNull / Unique~~ |
| ~~Number\_of\_reservation~~ | ~~String~~ | ~~NotNull~~ |
| ~~library\_book\_fk~~ | ~~Integer~~ | ~~NotNull~~ |

##### ~~Request cover~~

~~No more 2 reservation by exemplary~~

##### Annex

~~SQL power architect :~~

* ~~\P7\_DOC\mpd \Library\_mpd\_v1.1.architect~~
* ~~\P7\_DOC\mpd \Library\_mpd\_v1.1.pdf~~

~~SQl file :~~

* ~~\P7\_DB\Ticket\_1\_Update\_DB\_v1.1.sql~~

Approach modified :

-by adding a new table containing the number of reservations involves a call to the database at each book reservation availability verification in the called book list.

Moreover, as reservations number is a specific book information, the better way is to add a column reservation number to the library\_books table.

#### Library\_books upadte

##### Update

Addition of column :

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Constrains** |
| Number\_of\_reservations | String | NotNull |

##### Request cover

No more 2 reservation by exemplary

##### Annex

SQL power architect :

* \P7\_DOC\mpd \Library\_mpd\_v1.2.architect
* \P7\_DOC\mpd \Library\_mpd\_v1.2.pdf

SQl file :

* \P7\_DB\Ticket\_1\_Update\_DB\_v1.2.sql

## New micro-service

### Name

libraryReservations

### Aim

In charge of managing the book reservation, and related to the new database tables created above

## Book list behavior modification

### Aim



### Flowchart

