

# Documentation technique

Développez le nouveau système d'information  
de la bibliothèque d'une grande ville

## INFORMATIONS

---

Nom du projet	Library-IS
Type de document	Documentation technique
Date	09/02/2017
Version	1.0
Mots-clés	Architecture – Fonctionnement – Technologies – Diagrammes – Bugs – Interactions
Auteurs	Nicolas Bouème

## RÉDACTION ET MODIFICATIONS

---

Version	Date	Nom	Description
1.0	07/02/2016	Nicolas Bouème	▪ Première version

## TABLE DES MATIÈRES

---

1 – Objectif du document	4
2 – Rappel sur le fonctionnement de l'application	5
2.1 – Contexte	5
2.2 – Contraintes	6
2.3 – Travail demandé	6
2.4 – Architecture globale	7
2.5 – Diagramme de classes	8
2.6 – MPD	9
3 – Service web	10
3.1 – Technologies utilisées	10
3.2 – Architecture de l'application	10
4 – Application web	11
4.1 – Technologies utilisées	11
4.2 – Architecture de l'application	11
5 – Batch de relance des usagers	12
5.1 – Technologies utilisées	12
5.2 – Architecture de l'application	12
6 – Améliorations	13
Annexes	14
A – Diagramme de packages	14
B – Diagramme de cas d'utilisation du package "authentification"	14
C – Diagramme de cas d'utilisation du package "gestion compte utilisation"	15
D – Diagramme de cas d'utilisation du package "consultation des informations"	15
E – Diagramme de cas d'utilisation du package "gestion des prêts usagers"	16

## 1 - OBJECTIF DU DOCUMENT

---

Ce document est la documentation technique officielle de la suite applicative Library-IS. Il est divisé en quatre parties :

- La documentation technique du web service
- La documentation technique de l'application web
- La documentation technique du batch de relance des usagers
- Les améliorations envisagées dans le futur

## 2 - RAPPEL SUR LE FONCTIONNEMENT DE L'APPLICATION

---

### 2.1 – Contexte

Le service culturel d'une grande ville souhaite moderniser la gestion de ses bibliothèques. Pour cela, elle désire mettre à disposition de ses usagers, un système de suivi des prêts de leurs ouvrages.

Ce système comprendra :

- un site web (en responsive design) accessible aux usagers et permettant :
  - de rechercher des ouvrages et voir le nombre d'exemplaires disponibles
  - de suivre leurs prêts en cours. Les prêts sont pour une période de 4 semaines (durée configurable)
  - de prolonger un prêt. Le prêt d'un ouvrage n'est prolongeable qu'une seule fois. La prolongation ajoute une nouvelle période de prêt (4 semaines, durée de prolongation configurable) à la période initiale
- une application mobile iOS et Android fournissant les mêmes services que le site web
- une application spécifique pour le personnel des bibliothèques permettant, entre autres, de gérer les emprunts et les livres rendus
- un batch lancé régulièrement et qui enverra des mails de relance aux usagers n'ayant pas rendu les livres en fin de période de prêt

Ce projet va être réalisé en plusieurs itérations. Nous réaliserons que la première (se référer au paragraphe « travail demandé » pour le détail de cette itération).

Afin de centraliser les règles de gestion et généraliser les développements, nous avons adopté une démarche SOA (Service Oriented Architecture). Nous avons prévu de réaliser un web service qui portera la majeure partie de la logique métier et les applications s'appuieront sur celui-ci.

## 2.2 – Contraintes

Le déploiement du système sera assuré par le personnel de la direction des systèmes d'information de la ville. Nous devons donc leur laisser la possibilité de modifier facilement les différents paramètres de configuration (URL et identifiant/mot de passe de la base de données, URL du webservice, envoi des mails, et cetera)

## 2.2 – Travail demandé

- Une base de données PostgreSQL
- Un web service SOAP permettant :
  - l'identification des usagers via un identifiant et un mot de passe
  - de remonter les disponibilités des ouvrages
  - de gérer les prêts (nouveau prêt, prolongation, retour de prêt, état des prêts en cours / prolongés / non rendus à temps, et cetera)
- Une application web, basée sur le framework Apache Struts 2, servant d'interface pour les utilisateurs. **Elle ne doit pas se connecter à la base de données**, tout passe par le web-service
- Un batch qui envoie un mail de relance aux usagers n'ayant pas rendu les ouvrages dont la période de prêt est terminée. **Il ne doit pas se connecter à la base de données**, tout passe par le web service
- Une documentation générale (une dizaine de pages) décrivant la solution fonctionnelle et technique mise en place et sa mise en œuvre (configuration, déploiement). Ce document comprendra :
  - un diagramme de classes UML décrivant le domaine fonctionnel
  - les principales règles de gestion
  - le modèle physique de données (MPD). Si besoin, il peut faire l'objet d'un document PDF à part
  - La solution technique mise en place
  - La mise en œuvre du système (configuration, déploiement)
- Une documentation succincte expliquant comment construire les images docker depuis les sources et déployer l'application

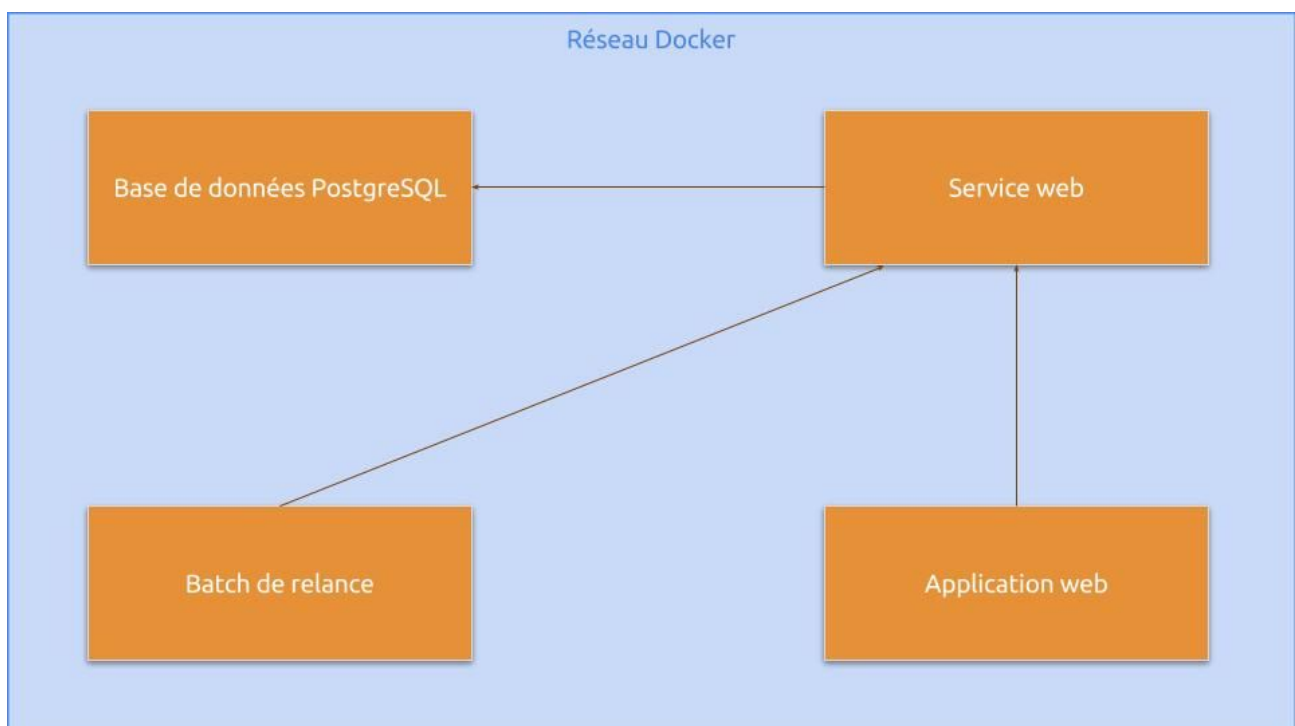
**Nous n'avons pas à développer les applications mobiles et le système des postes internes aux bibliothèques.**

Nous réaliserons le web service, l'application web et le batch en Java/JEE (JDK 8) avec les fonctionnalités décrites ci-dessus. Le web service et l'application web seront déployés sur un serveur GlassFish 5.

Les différents composants seront packagés avec Maven :

- le web service : au format WAR, et un ZIP des fichiers de configuration nécessaires
- l'application web ouverte aux usagers : au format WAR, et un ZIP des fichiers de configuration nécessaires
- le batch : au format JAR exécutable ainsi qu'un fichier ZIP contenant le JAR exécutable, les dépendances Java nécessaires, les fichiers de configuration, un script shell (sh ou bash) de lancement

### 2.3 – Architecture globale



*Illustration 1 : Architecture globale*

2.4 – Diagramme de classes

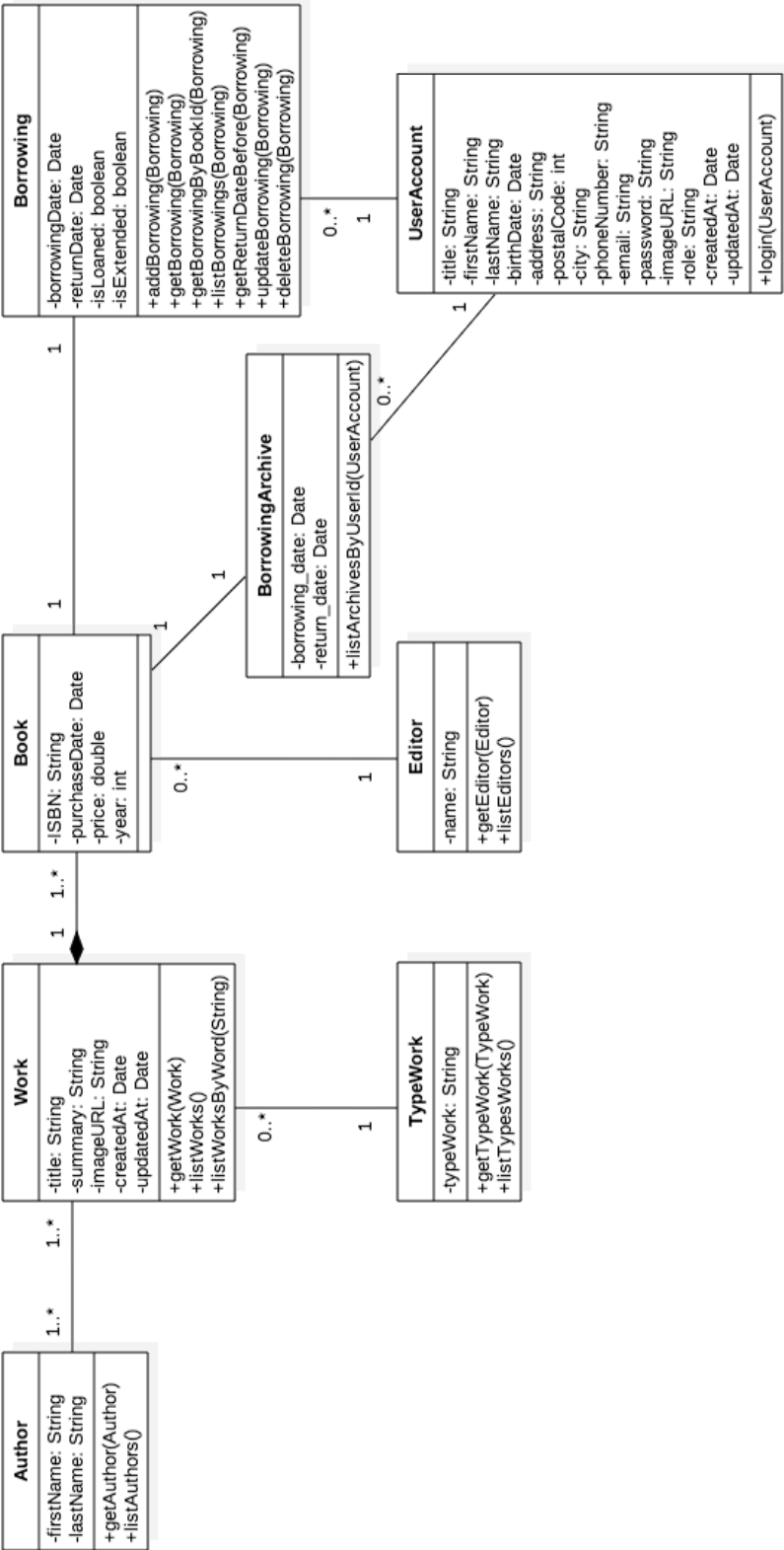


Illustration 2 : Diagramme de classes



2.5 – MPD

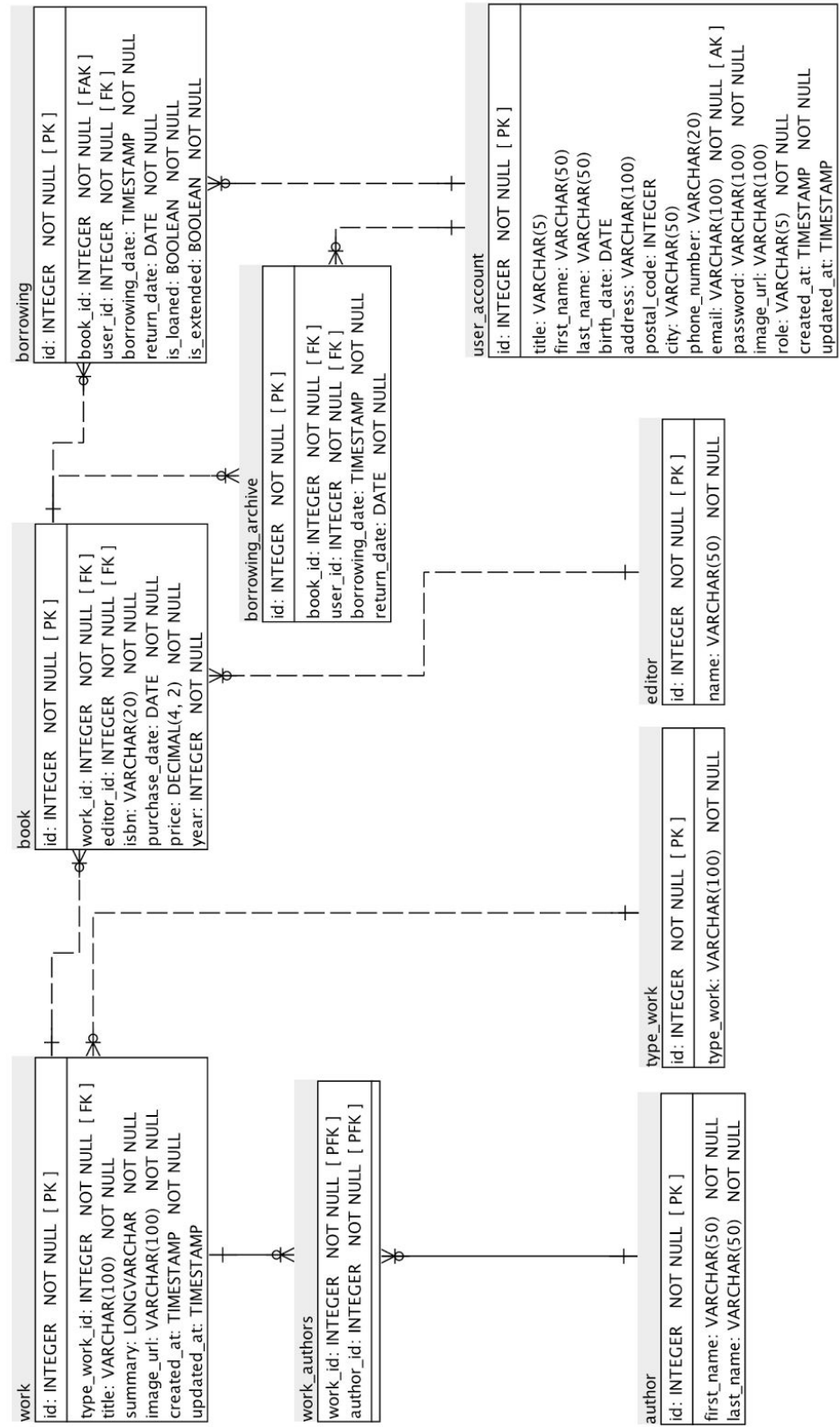


Illustration 3 : MPD

## 3 - SERVICE WEB

---

### 3.1 – Technologies utilisées

- Apache Maven 3.5.2
- Docker 17.12.0-ce-mac49
- GlassFish 5.0
- JDK8 version 162
- PostgreSQL 10.1
- Spring Framework 5.0.2.RELEASE
- Spring Data 2.0.2.RELEASE

### 3.2 – Architecture de l'application

Le service web est structuré de la façon suivante :

- library-service : module contenant le web-service
  - service : ce package contient les différents services
  - resources : contient les conteneur IoC de Spring Framework
- library-repository : module contenant la persistance et le pattern DAO
  - contract : ce package contient les contrats DAO
  - impl : ce package contient les différentes implémentations des contrats, à savoir JPA couplé avec Spring Data
  - resources : contient les conteneur IoC de Spring Framework
- library-entity : module contenant les entités de la solution
  - entity : ce package contient les différentes entités annotées avec JPA, à la racine du package nous avons leurs interfaces

## 4 - APPLICATION WEB

---

### 4.1 – Technologies utilisées

- Apache Maven 3.5.2
- Bootstrap 4.0.0
- Docker 17.12.0-ce-mac49
- GlassFish 5.0
- JDK8 version 162
- Spring Framework 5.0.2.RELEASE
- Struts2 2.5.14.1

### 4.2 – Architecture de l'application

L'application web est structuré de la façon suivante :

- library-webapp : module contenant le site web
  - action : ce package contient les différentes actions ou contrôleurs
  - interceptor : ce package contient l'intercepteur de session utilisateur
  - utility : ce package contient différentes constantes
  - resources : contient la configuration de Spring Framework et de Struts2
  - webapp : contient la partie vue du site web
- library-business : module contenant la logique métier
  - contract : ce package contient les contrats des managers
  - impl : ce package contient les différentes implémentations des contrats
  - resources : contient la configuration de Spring Framework
- library-repository : module contenant la persistance et le pattern DAO
  - contract : ce package contient les contrats DAO
  - impl : ce package contient les différentes implémentations des contrats, à savoir le service web SOAP par l'intermédiaire du module client
  - resources : contient la configuration de Spring Framework
- library-client : module contenant les entités et classes auto-générées
  - client : ce package contient les différents clients auto-générés par les fichiers WSDL

## 5 - BATCH DE RELANCE DES USAGERS

---

### 5.1 – Technologies utilisées

- Apache Maven 3.5.2
- Docker 17.12.0-ce-mac49
- JDK8 version 162
- Spring Batch 4.0.0.RELEASE
- Spring Framework 5.0.2.RELEASE

### 5.2 – Architecture de l'application

L'application web est structurée de la façon suivante :

- library-batch : module contenant le batch de relance des usagers
  - batch : ce package contient l'implémentation du batch
  - configuration : ce package contient la configuration des emails
  - service : ce package appelle le service web pour récupérer la liste des emprunts dont la date de rendu a été dépassée
  - resources : contient la configuration de Spring Framework et de Struts2
- library-business : module contenant la logique métier
  - contract : ce package contient les contrats des managers
  - impl : ce package contient les différentes implémentations des contrats
  - resources : contient la configuration de Spring Framework
- library-repository : module contenant le pattern DAO, appelle le service web
  - contract : ce package contient les contrats DAO
  - impl : ce package contient les différentes implémentations des contrats, à savoir le service web SOAP par l'intermédiaire du module client
  - resources : contient la configuration de Spring Framework
- library-client : module contenant les entités et classes auto-générées
  - client : ce package contient les différents clients auto-générés par les fichiers WSDL

## 6 - AMÉLIORATIONS

---

Différents axes d'améliorations intéressants à implémenter :

- service web
  - Créer un système de pagination pour les listes
  - Validation côté serveur
  - Créer un module pour les logs en production
  - Tests unitaires
  - Test d'intégrations
- application web
  - Ne recharger qu'une partie de la page après une requête POST, utiliser un framework JavaScript comme React serait une solution idéale
  - Persistance des objets et listes d'objets dans le cache afin de minimiser les requêtes
  - Utiliser un système de pagination pour les listes
  - Fonctionnalité de recherche avancée selon un titre de livre, un nom d'auteur, un type de livre ou un nom d'éditeur
  - Concevoir un style au site conforme à l'univers des bibliothèques et d'emprunts de livres
  - Créer un module pour les logs en production
  - Tests unitaires
  - Test d'intégrations
- batch
  - Créer un module pour les logs en production
  - Tests unitaires
  - Test d'intégrations

## ANNEXES

### A – Diagramme de packages

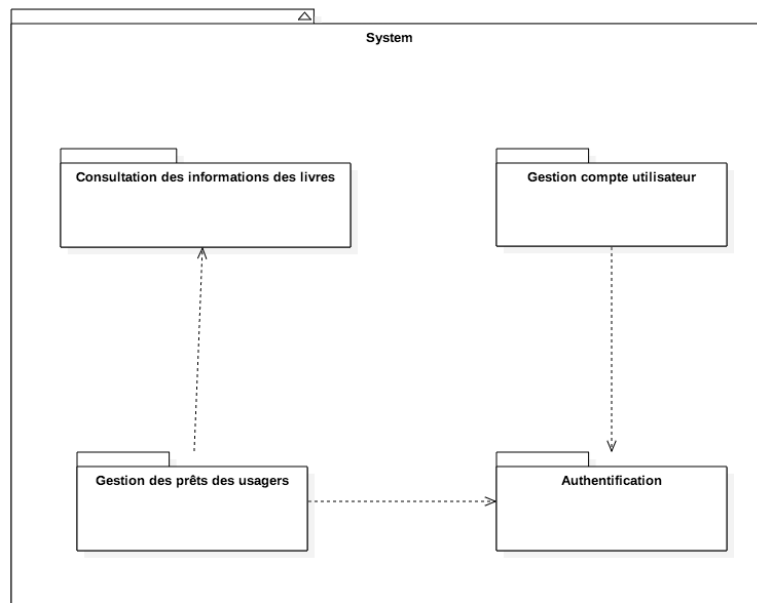


Illustration 4 : Diagramme de packages du SI

### B – Diagramme de cas d'utilisation du package "authentification"

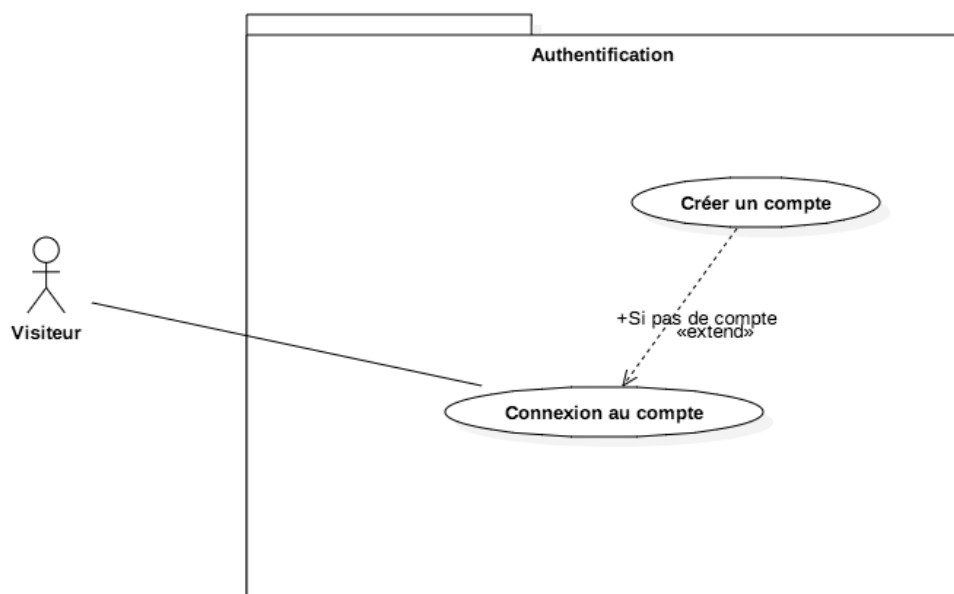


Illustration 5 : Diagramme de CU du package "authentification"

### C – Diagramme de cas d'utilisation du package "gestion compte utilisateur"

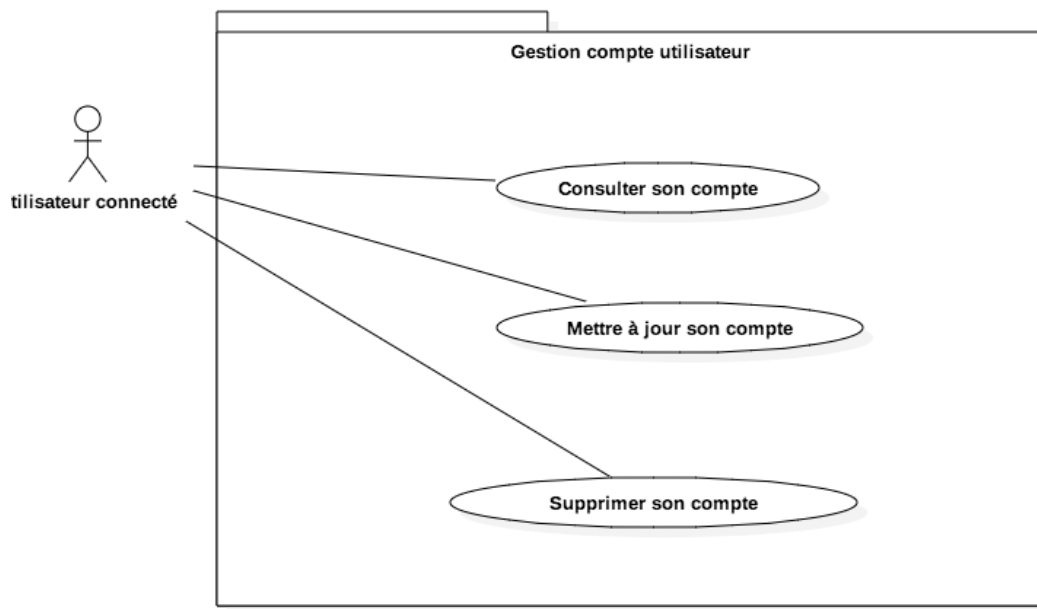


Illustration 6 : Diagramme de CU du package "gestion compte utilisateur"

### D – Diagramme de cas d'utilisation du package "consultation des informations"

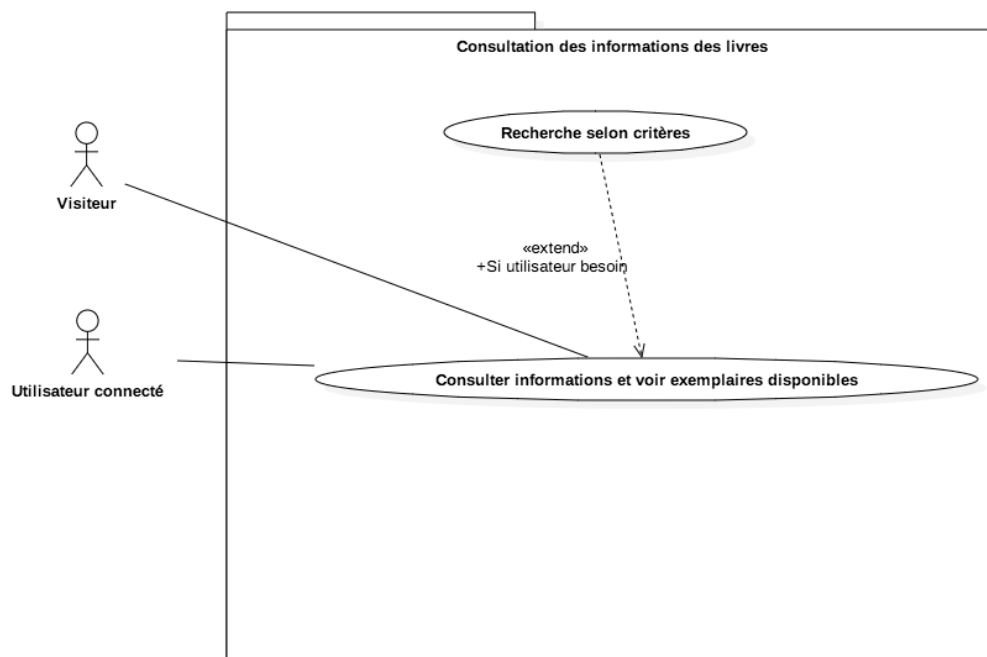


Illustration 7 : Diagramme de CU du package "consultation des informations"

## E – Diagramme de cas d'utilisation du package “gestion des prêts usagers”

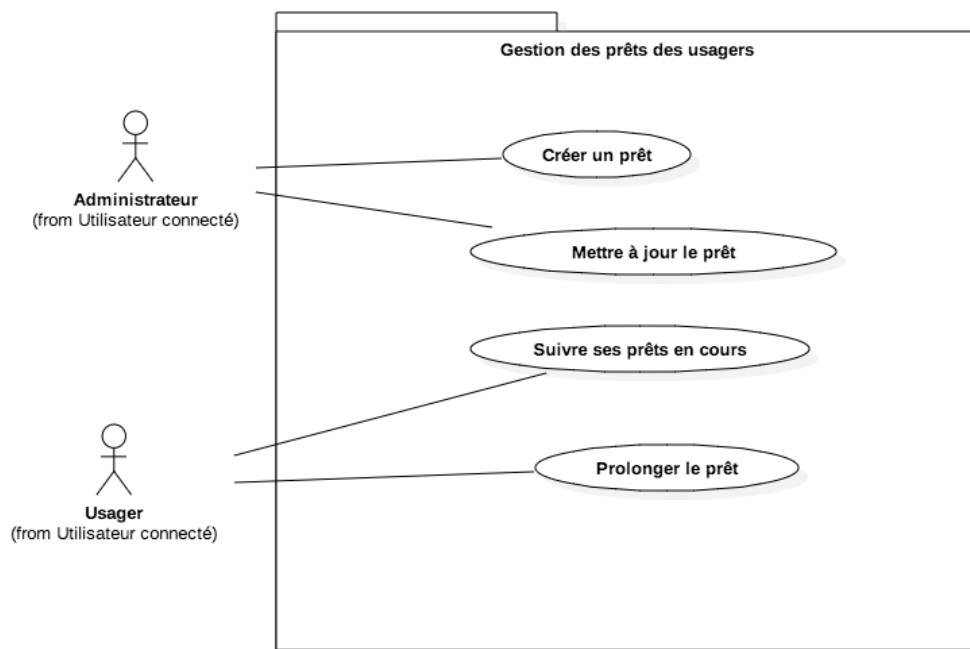


Illustration 8 : Diagramme de CU du package “gestion des prêts usagers”