

# 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	06/06/2018
Version	1.2
Mots-clés	Architecture – Fonctionnement – Technologies – Diagrammes – Bugs – Interactions
Auteurs	Nicolas Bouème

## RÉDACTION ET MODIFICATIONS

---

Version	Date	Nom	Description
1.2	06/06/2018	Nicolas Bouème	Mise à jour des diagrammes et du travail demandé pour la version 1.2.1 du projet
1.1	13/02/2018	Nicolas Bouème	Ajout section "Déploiement"
1.0	09/02/2018	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 pour la version 1.0.0	6
2.4 – Travail demandé pour la version 1.2.1	7
2.4.1 – Ajout d'un système de réservation d'ouvrage	7
2.4.2 – Bug dans la gestion des prolongations de prêt	8
2.4.3 – Créer un batch de rappel des prêts arrivant bientôt à expiration	8
2.5 – Architecture globale	9
2.6 – Diagramme de classes	10
2.7 – MPD	11
3 – Service web	12
3.1 – Technologies utilisées	12
3.2 – Architecture de l'application	12
4 – Application web	13
4.1 – Technologies utilisées	13
4.2 – Architecture de l'application	13
5 – Batch	14
5.1 – Technologies utilisées	14
5.2 – Architecture de l'application	14
6 – Déploiement	15
7 – Améliorations	17
Annexes	18
A – Diagramme de packages	18
B – Diagramme de cas d'utilisation du package "authentification"	18
C – Diagramme de cas d'utilisation du package "gestion compte utilisation"	19
D – Diagramme de cas d'utilisation du package "consultation des informations"	19
E – Diagramme de cas d'utilisation du package "gestion des prêts usagers"	20
F – Diagramme de cas d'utilisation du package "gestion des réservations"	20

## 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)

## 2.3 – Travail demandé pour la version 1.0.0

- 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

## **2.4 – Travail demandé pour la version 1.2.1**

### *2.4.1 – Ajout d'un système de réservation d'ouvrage*

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

Le principe est le suivant :

L'utilisateur 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.

- 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

- Sur l'application web :
  - 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'utilisateur 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'utilisateur doit pouvoir annuler une réservation

Avertissez-nous avant de livrer cette évolution afin que nous nous organisions. De plus cette demande n'est pas urgente et il est possible que nous vous demandions d'autres évolutions à embarquer en même temps dans la nouvelle version du système.

#### *2.4.2 – Bug dans la gestion des prolongations de prêt*

Nous avons découvert un bug de la gestion des prolongations de prêt. En effet, un utilisateur peut prolonger un prêt après la date butoir. Il ne doit pas être possible pour l'utilisateur de prolonger un prêt si la date de fin de prêt est dépassée.

La correction de cette anomalie est urgente car elle nous pose de gros problèmes de gestion. Actuellement beaucoup trop de personnes attendent le mail de rappel pour prolonger leur prêt. Cela entraîne la plainte de certains utilisateurs à qui nous avons communiqué une date de retour et qui se déplacent en bibliothèque pour rien car le prêt a été prolongé a posteriori.

Je vous demande de livrer une nouvelle version de l'application au plus vite afin de corriger ce bug mais sans inclure nos autres demandes en cours. Nous souhaiterions pouvoir mettre en production rapidement sans avoir à faire de recette sur les autres évolutions demandées.

#### *2.4.3 – Créer un batch de rappel des prêts arrivant bientôt à expiration*

Nous aimerions, en parallèle de la correction de la gestion des prolongations de prêt, ajouter un mécanisme de rappel aux utilisateurs des prêts arrivant bientôt à expiration. Les utilisateurs qui le

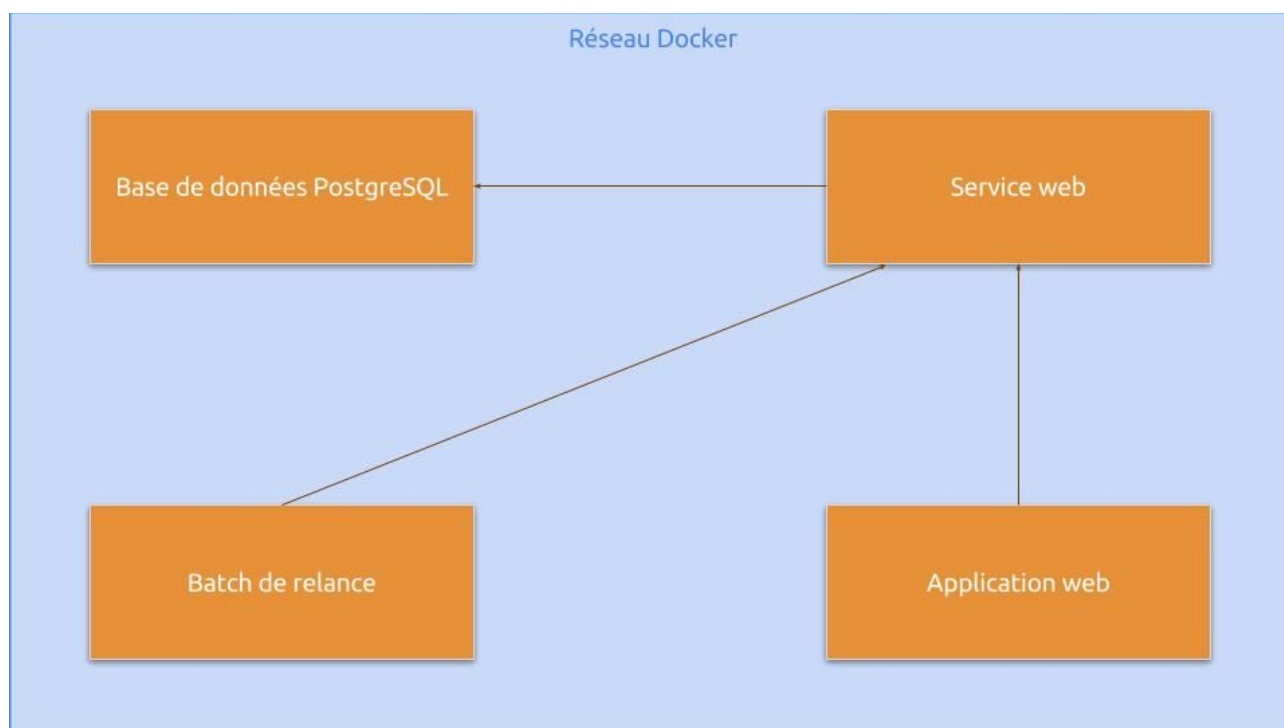


souhaitent pourront activer une option (dans leur profil sur l'application web) qui leur enverra un mail de rappel 5 jours calendaires avant la fin de leur prêt. Le mail contiendra la liste de leurs prêts en cours qui arrivent à expiration dans 5 jours ou moins avec la date d'expiration associée à chaque prêt.

Cette option sera à activer par défaut pour tous les utilisateurs lors de la mise en production.

Cette demande n'est pas urgente, vous livrerez cette évolution en même temps que celle demandé dans le ticket de réservation d'ouvrage.

## 2.5 – Architecture globale



*Illustration 1 : Architecture globale*

## 2.6 – Diagramme de classes

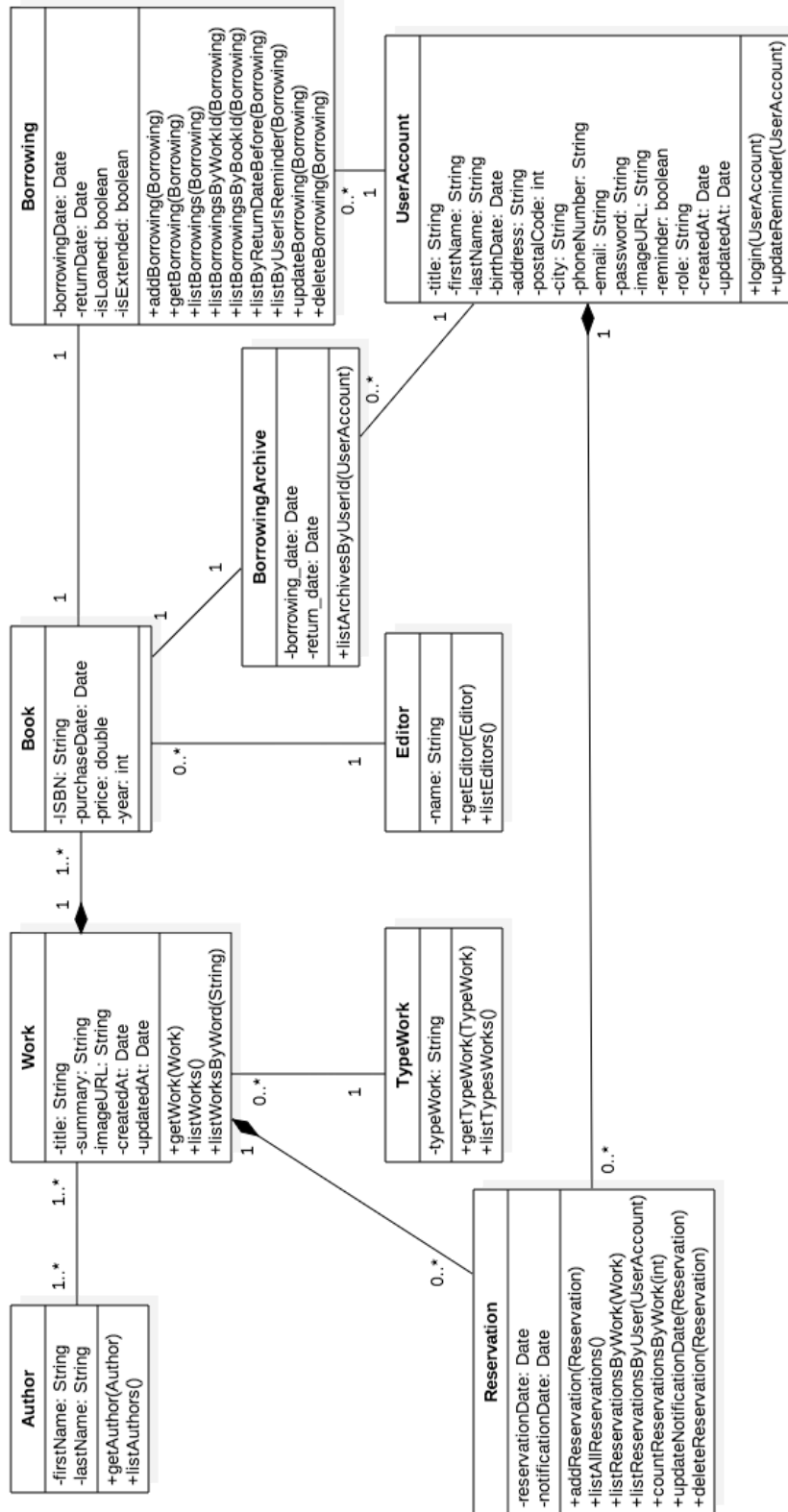


Illustration 2 : Diagramme de classes

2.7 – MPD

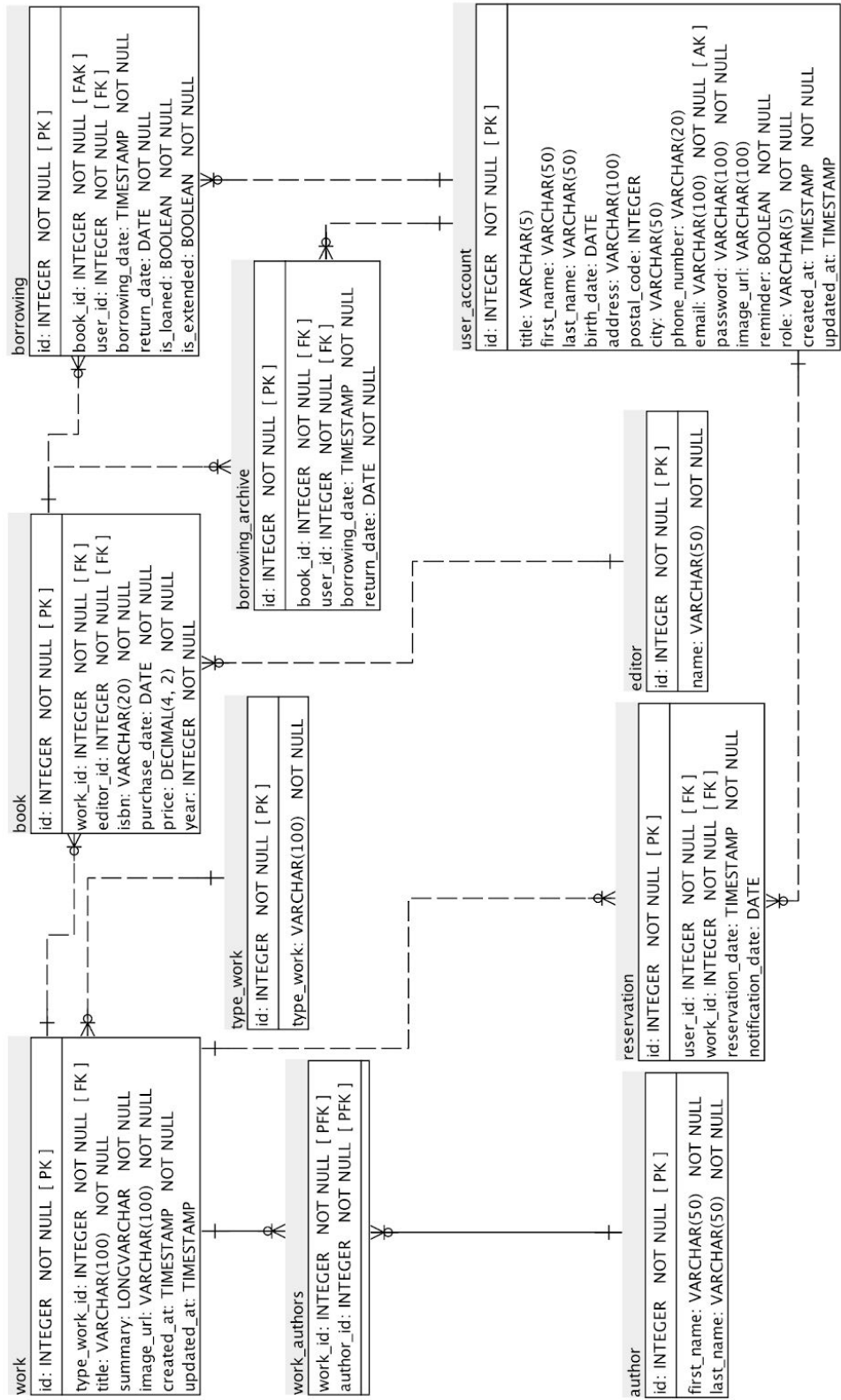


Illustration 3 : MPD

## 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 structurée 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

---

### 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 les batch de relance des usagers lorsque que ceux-ci souhaitent avoir un rappel 5 jours calendaires avant la date de retour d'un de leurs prêts, de relance des usagers lorsque la date de retour est arrivée à terme, de notification de réservation
  - batch : ce package contient l'implémentation des 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 - DÉPLOIEMENT

---

Si jamais vous souhaitez compiler vous même l'application web ou le batch, à la machine hôte, celle qui va héberger les conteneurs Docker du projet :

- attribuer une adresse IP LAN fixe de votre choix
- créer un alias `web-service` dans le fichier `/etc/hosts` à cet IP

Dans le dépôt, créer un fichier `datasource.properties` dans le répertoire `docker/web-service/app/` avec les paramètres suivants :

```
library.datasource.driver=org.postgresql.Driver  
library.datasource.url=jdbc:postgresql://database:5432/DB_NAME  
library.datasource.username=USER_NAME  
library.datasource.password=USER_PASSWORD
```

Puis, créer un fichier `batch.properties` dans le répertoire `docker/batch/app/` avec les paramètres suivants :

```
mail.host=smtp.gmail.com  
mail.port=587  
mail.username=USER_NAME  
mail.password=USER_PASSWORD  
batch.cron=CRON_VALUE
```

Adapter dans le fichier `docker-compose.yml` du dossier `docker` les variables d'environnements de création de la base de données :

```
environment:  
- POSTGRES_DB=DB_NAME  
- POSTGRES_USER=USER_NAME  
- POSTGRES_PASSWORD=USER_PASSWORD
```

Se placer dans le dossier `docker` du dépôt du projet et exécuter les commandes suivantes :

- `docker-compose -p library up -d database` pour déployer la base de données
- `docker-compose -p library up -d web-service` pour déployer le service web
- `docker-compose -p library up -d web-application` pour déployer l'application web
- `docker build -t nboueme/library-batch` pour créer l'image de déploiement du batch
- `docker-compose -p library up -d batch` pour déployer le batch

Les identifiants d'accès aux serveurs GlassFish sont `admin` avec pour mot de passe `glassfish`.

L'application web est désormais disponible via l'URL suivante : <http://localhost/>.



## 7 - 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

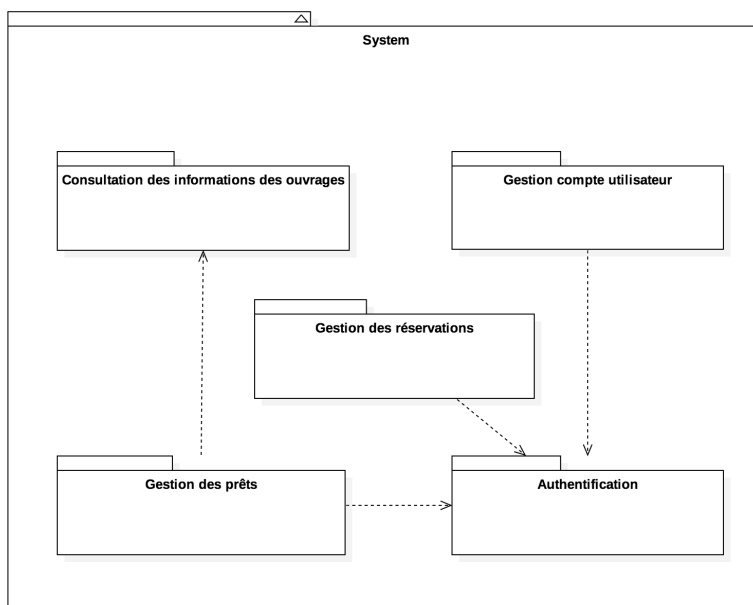


Illustration 4 : Diagramme de packages du SI

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

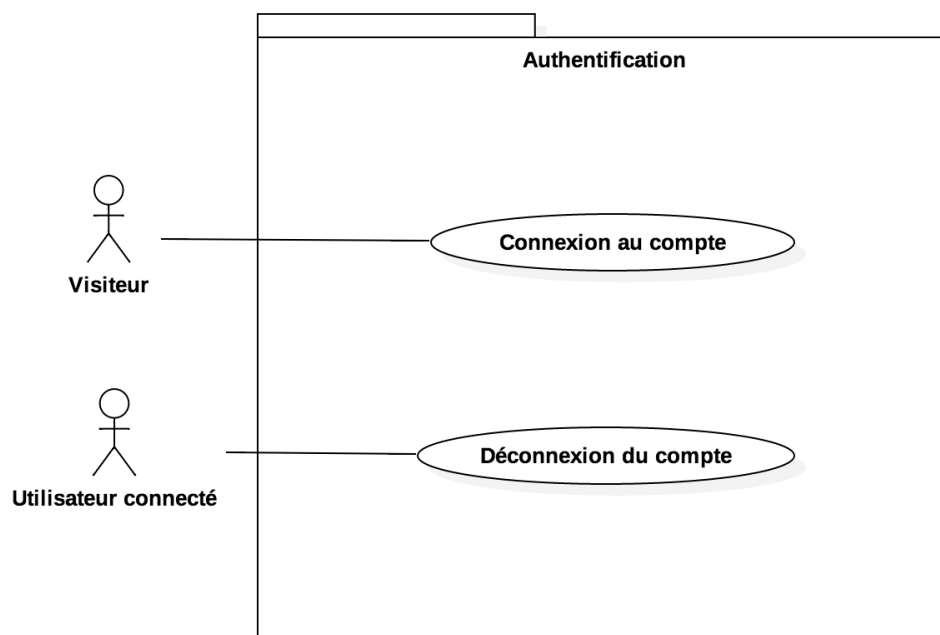


Illustration 5 : Diagramme de CU du package "authentification"

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

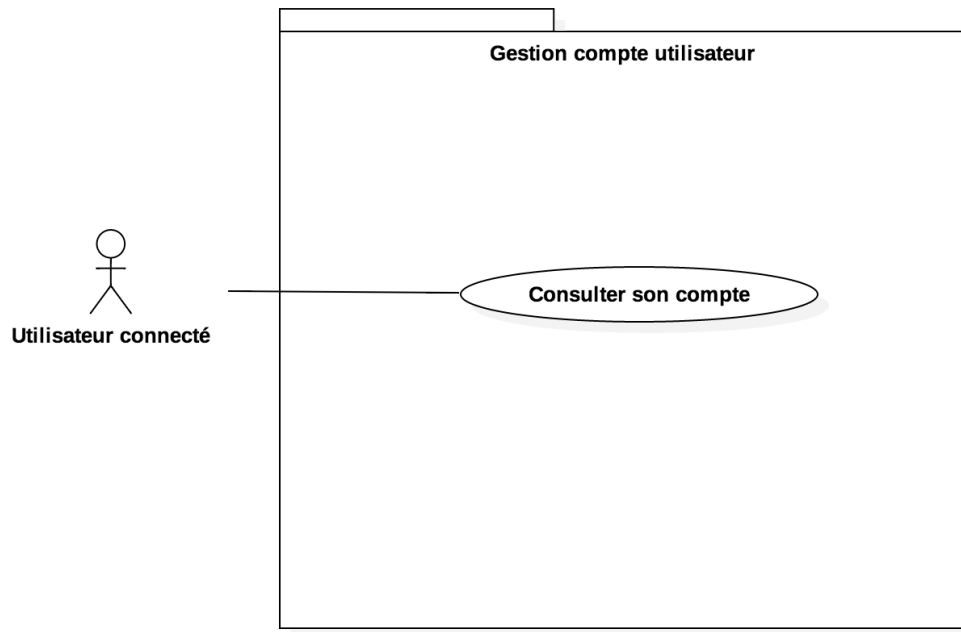


Illustration 6 : Diagramme de CU du package “gestion compte utilisateur”

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

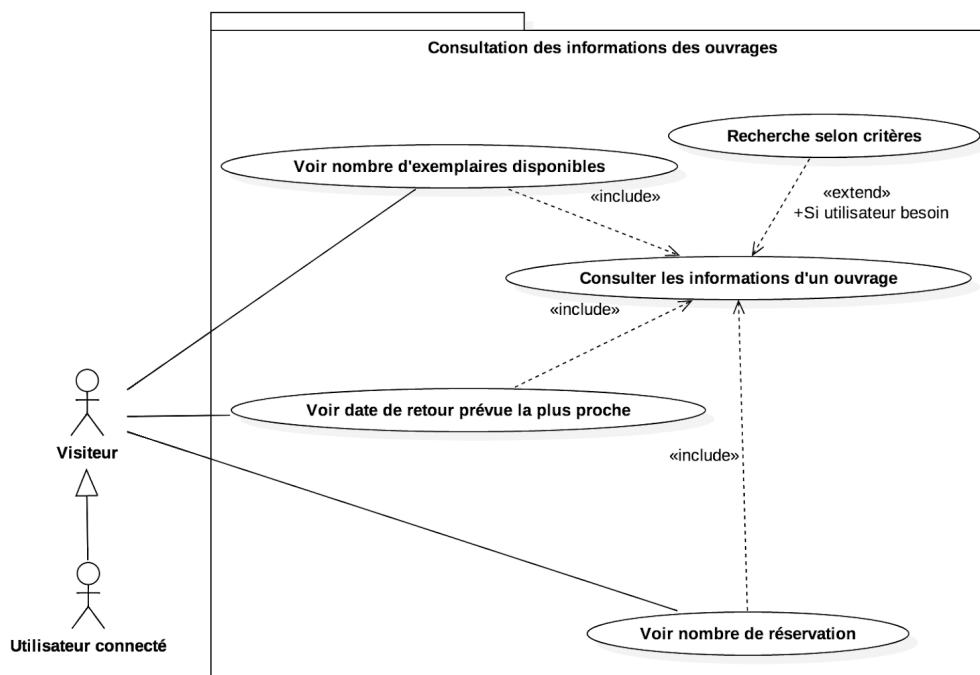


Illustration 7 : Diagramme de CU du package “consultation des informations”

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

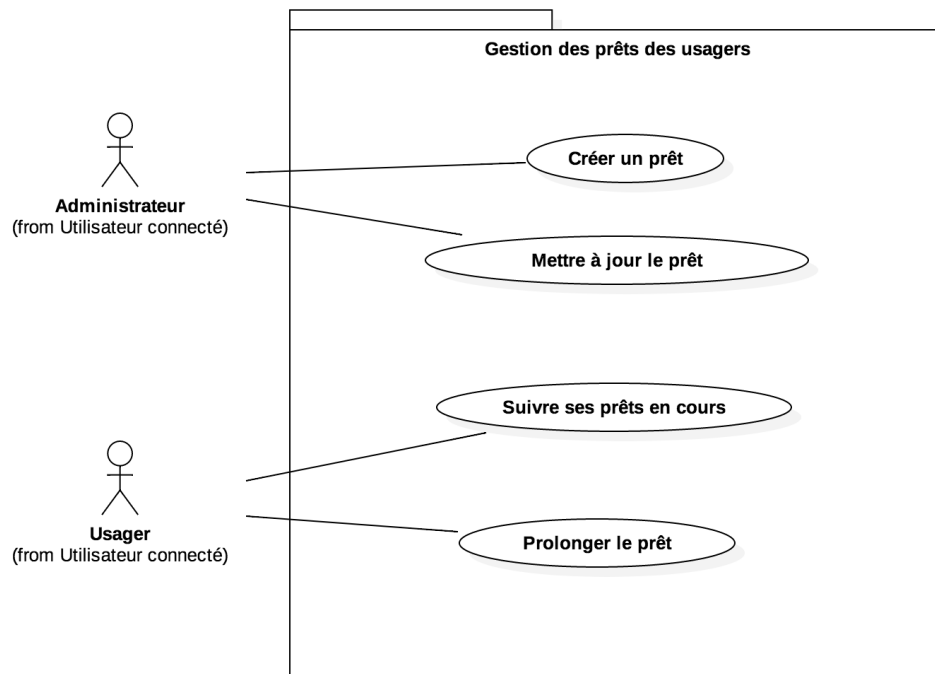


Illustration 8 : Diagramme de CU du package “gestion des prêts usagers”

## F – Diagramme de cas d'utilisation du package “gestion des réservations”

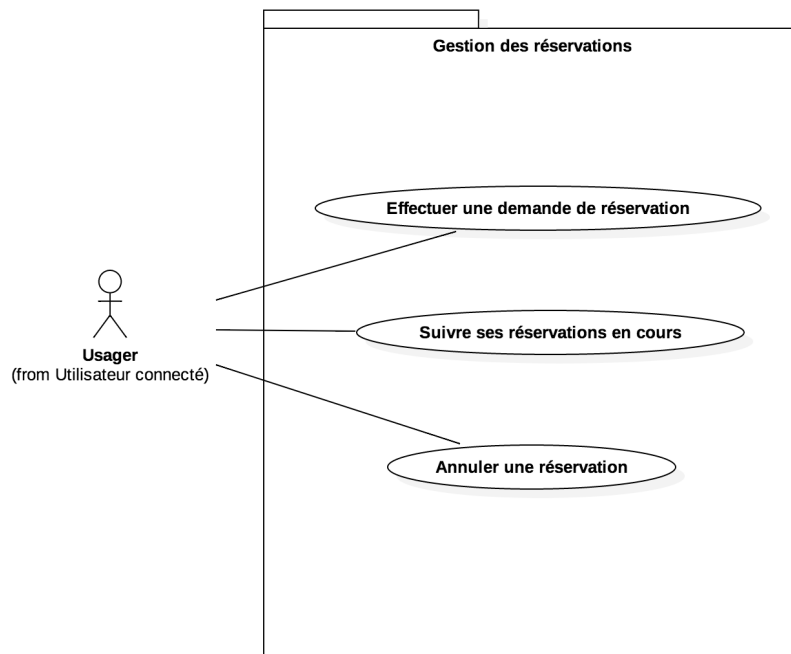


Illustration 9 : Diagramme de CU du package “gestion des réservations”