LIBRARY MANAGER

DOCUMENTATION

Version 1.0 / Janvier 2019

Le projet **LibraryManager** est un système d'information basé sur un web service SOAP délivrant des services dédiés à la gestion d'une base de données de prêts de livre pour une bibliothèque.

TABLE DES MATIERES

1. COMPOSITION DU PROJET

1.1. WEB SERVICE (LIBRARYMANAGER-WS)

Le web service fournit à l'ensemble du système plusieurs opérations permettant à la fois la gestion des prêts et également la gestion d'autres informations de la base de données tels que les membres, les livres ou les catégories de livres. Le WS utilise Apache CXF pour générer les classes Java à partir des fichiers WSDL décrivant chaque service. La connexion à la base de données est gérée à l'aide Hibernate. Les modèles créés implémentent des annotations permettant la création automatique des tables dans la base de données (voir la section Configuration).

1.2. APPLICATION WEB (LIBRARYMANAGER-WEBAPP)

L'application Web permet aux utilisateurs d'accéder à une interface graphique pour gérer les données. Elle créée à l'aide de Spring MVC et n'accède aux informations qu'à travers le WS qui lui fournit les fichiers WSDL pour créer un client.

1.3. BATCH DE RELANCE PAR EMAIL (LIBRARYMANAGER-BATCH)

Le batch est utilisée pour envoyer un mail aux membres n'ayant pas rendu leurs livres empruntés à temps. Il récupère les prêts concernés et les utilise afin d'agréger les données nécessaires à l'envoi des mails de relance. Le batch ne se connecte à la base de données qu'à travers le WS.

2. TECHNOLOGIES

2.1. APACHE MAVEN

La gestion des dépendances est gérée entièrement par Maven dans sa version ***3.6.0***. La gestion de la compilation et du packaging est également déléguée à Maven via ses plugins.

2.2. SPRING FRAMEWORK

Le projet utilise la version ***5.1.3.RELEASE*** de Sring Framework. Principalement utilisé pour l'injection des dépendances, le module Spring MVC est également utilisé pour l'application web afin d'intégrer le modèle MVC et gérer les Controllers/Views via les requêtes http sur les URLs/namespaces définis par le projet.

L'ensemble des modules de chaque projet dispose d'un fichier ***applicationContext-*.xml*** afin de gérer la configuration Spring.

Spring Batch, dans sa version ***4.1.0***, est également implémenté dans le module Batch pour exécuter les différents *Jobs* nécessaires à l'envoi des emails de relance.

2.3. POSTGRESQL

Le système de gestion de base de données relationnelle utilisé est PostgreSQL dans sa version ***10.6***. L'ensemble de la gestion des tables et des transactions de données est délégué à *Hibernate*.

2.4. HIBERNATE

Le Framework Hibernate est utilisé pour gérer l'intégralité des données des applications, de la création des tables à la persistance des données. Utilisé dans sa version ***5.4.0***, Hibernate fournit une implémentation de JPA et donc un *EntityManager* permettant de gérer l'ensemble du contexte de persistance et des requêtes SQL envoyées à la base de données *PostgreSQL*.

2.5. APACHE CXF

La gestion des web services SOAP se fait à l'aide d'***Apache CXF***, fournissant une implémentation de JAX-WS. Via une configuration XML, l'ensemble des classes et interfaces, destinés aux services, est généré via Apache CXF depuis les fichiers ***WSDL*** construits.

Au déploiement des web services, Apache CXF fournit une API utilisable par d'autres applications afin de faire appel aux opérations disponibles dans chaque service.

3. UTILISER LIBRARYMANAGER

3.1. INSTALLATION

Le projet peut être déployé à travers ***Docker***. Un fichier ***docker-compose.yml*** est proposé pour publier automatiquement l'ensemble des composants dans des *containers* de Docker. Une fois les éléments compilés (voir *README*), ouvrez un Terminal et lancez les commandes :

```
$ docker-compose build --no-cache
$ docker-compose up
```

L'application sera disponible à l'adresse, en local, sur http://localhost:8888 (si aucune modification des configurations n'a été faite).

3.2. IMPLEMENTATION D'UN CLIENT

L'ensemble des contrats *WSDL* des services de l'application sont diffusés par Apache CXF au déploiement du projet ***LibraryManager-WS***.

Une page permettra d'avoir une récapitulatif des URLs vers lesquels joindre les fichiers des services SOAP. Chaque fichier permettra de créer un client pour un service donné.

4. DOMAINE FONCTIONNEL

4.1. MODELE CONCEPTUEL DE DONNEES

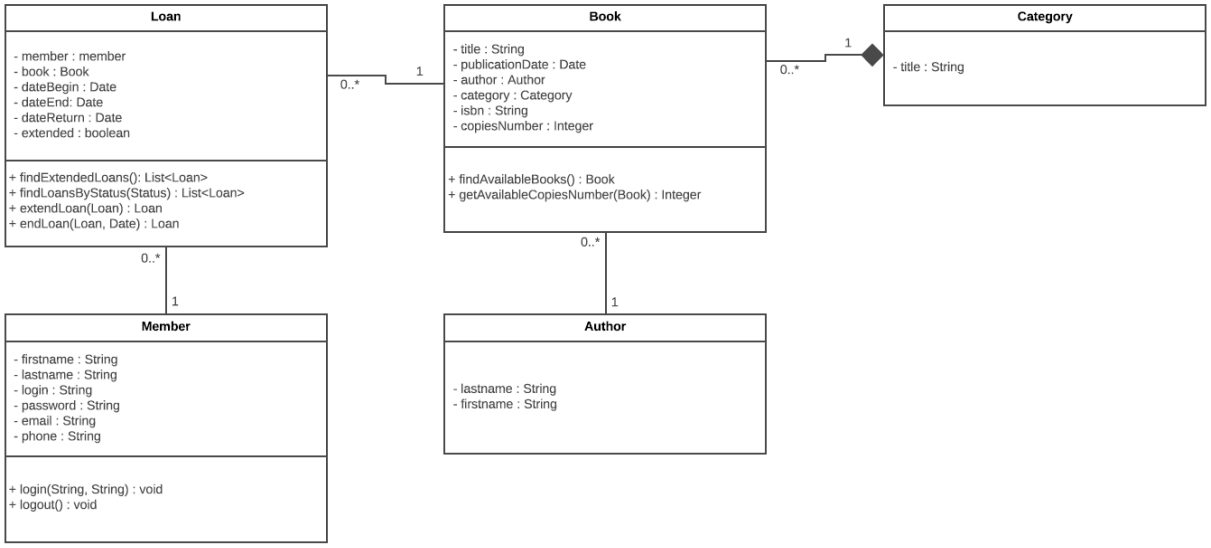

Figure 1 : Modèles conceptuel de données / Diagrammes de classes

4.2. MODELE PHYSIQUE DE DONNEES

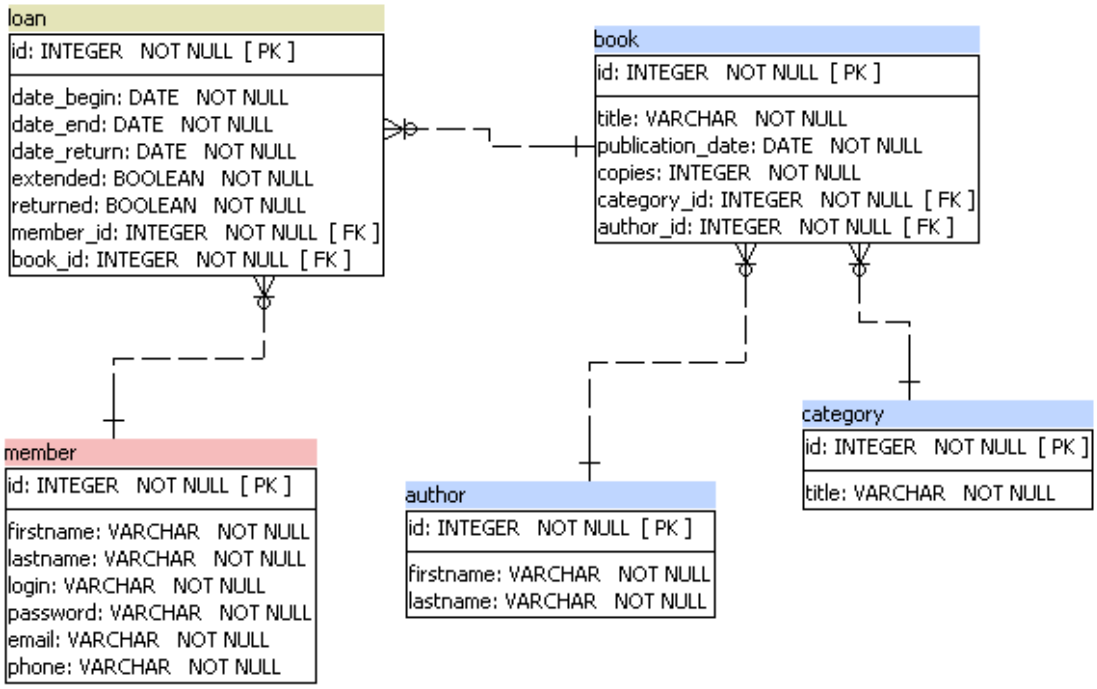

Figure 2 : Modèles physique de données

5. CONCEPTION

5.1. CONTEXTE

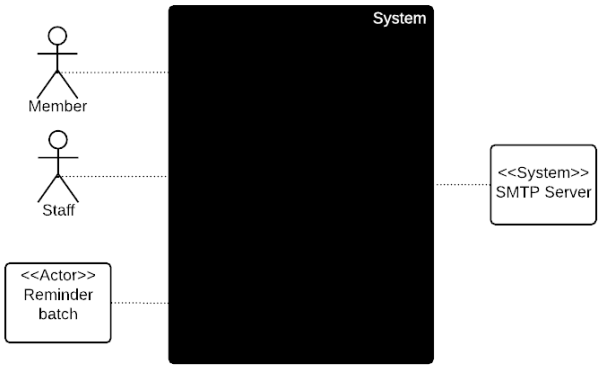

Figure 3 : Diagramme de contexte

5.2. FONCTIONNALITES

1.1.1. DIAGRAMME DE PACKAGES

Les différentes fonctionnalités se regroupent en 3 principaux packages : **Relancer les utilisateurs**, **Authentification**, **Gestion de prêts**.

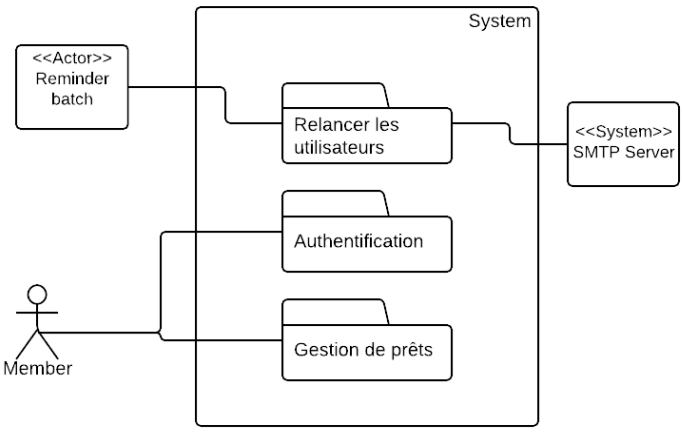

Figure 4 : Diagramme de packages

1.1.2. FONCTIONNALITES

- RELANCER LES UTILISATEURS

Cette fonctionnalité est utilisée par le Batch afin d'envoyer un mail aux utilisateurs dont le rendu prêt n'a pas été validée et dont la date de rendu est antérieure à la date du jour.

- AUTHENTIFICATION

Cette fonctionnalité est utilisée pour authentifier l'utilisateur qui souhaite se connecter à l'application afin de vérifier son identité.

- GESTION DE PRETS

Cet ensemble de fonctionnalités regroupe toutes les opérations utilisées pour afficher, étendre ou terminer un prêt.

5.3. CAS D'UTILISATION

- GERER UN COMPTE UTILISATEUR

Voir document 03 – Cas d'utilisations

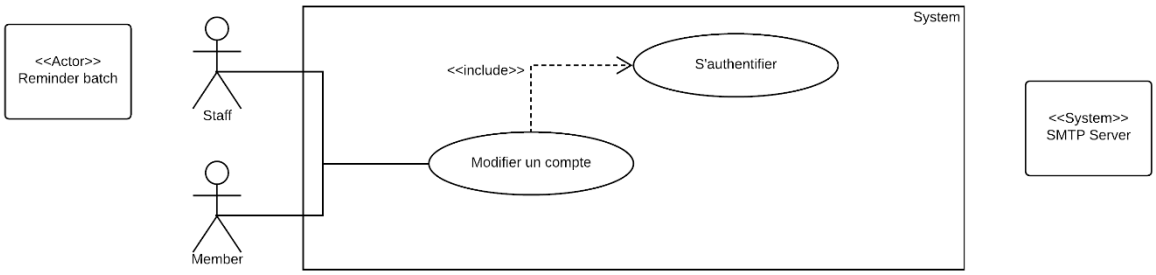

Figure 5 : Diagramme de cas d'utilisation « Gérer un compte utilisateur »

- GERER UN PRET

Voir document 03 – Cas d'utilisations

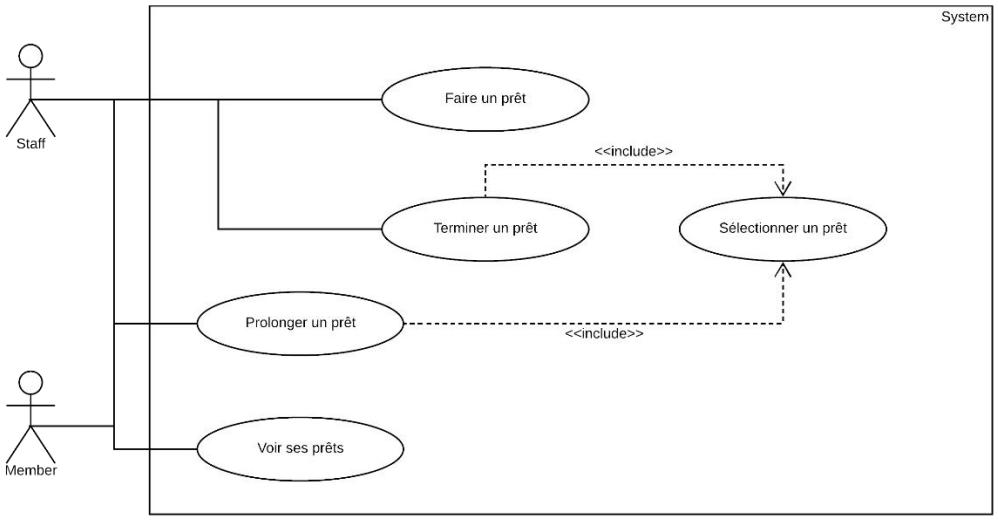

Figure 6 : Diagramme de cas d'utilisation « Gérer un prêt »

- RELANCER LES UTILISATEURS

Voir document 03 – Cas d'utilisations

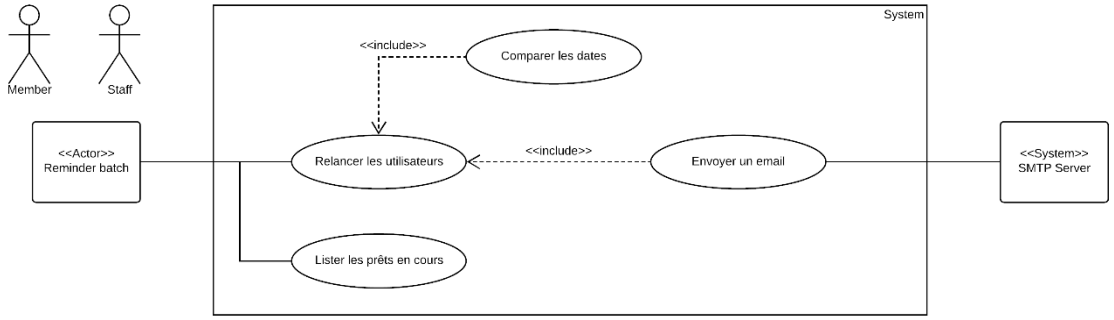

Figure 7 : Diagramme de cas d'utilisation « Relancer un utilisateur »

5.4. ARCHITECTURE

1.1.3. WEB SERVICE

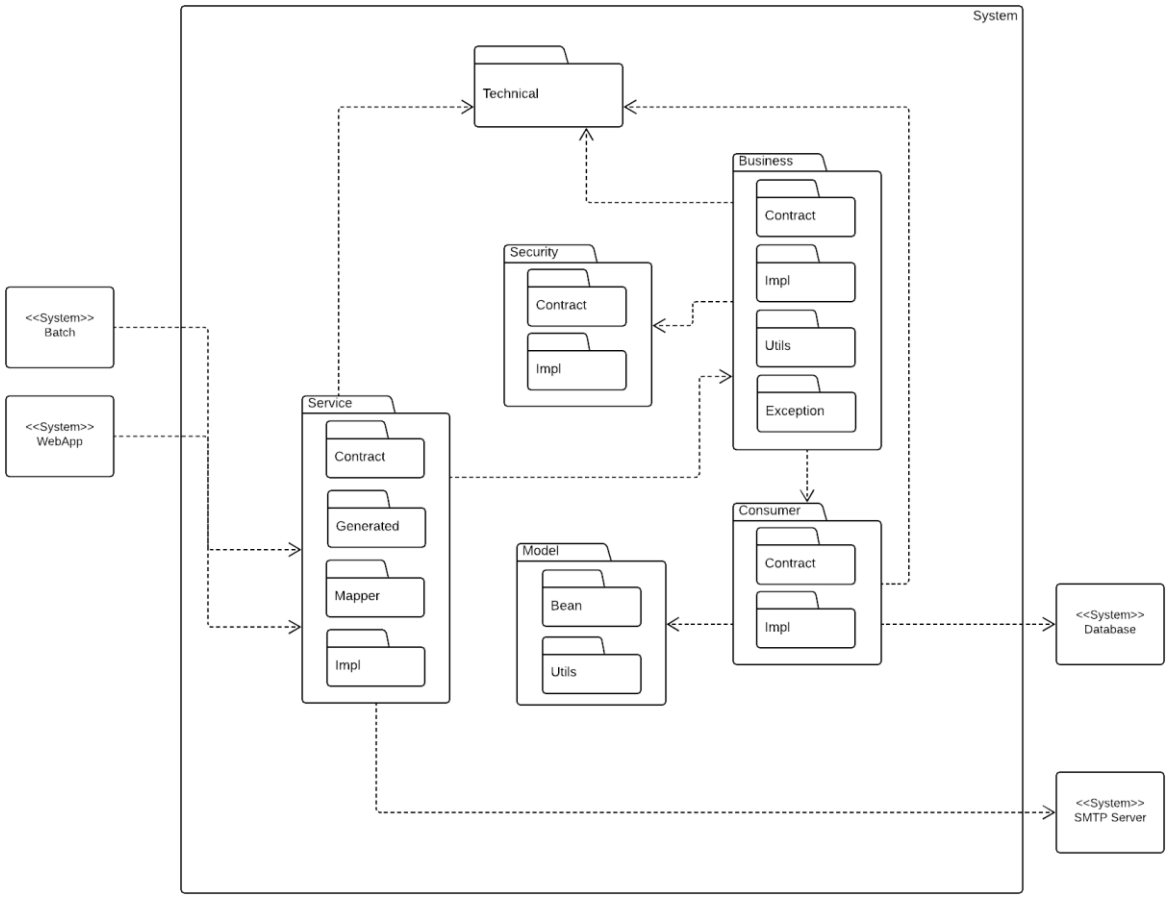

Figure 8 : Architecture et relations des modules du Web Service

1.1.4. WEB APPLICATION

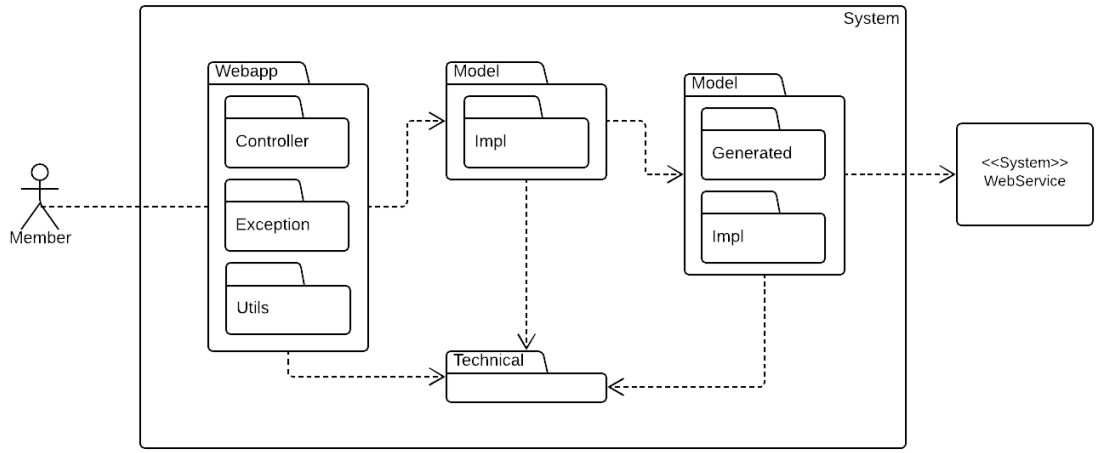

Figure 9 : Architecture et relations des modules de la Web Application

1.1.5. BATCH

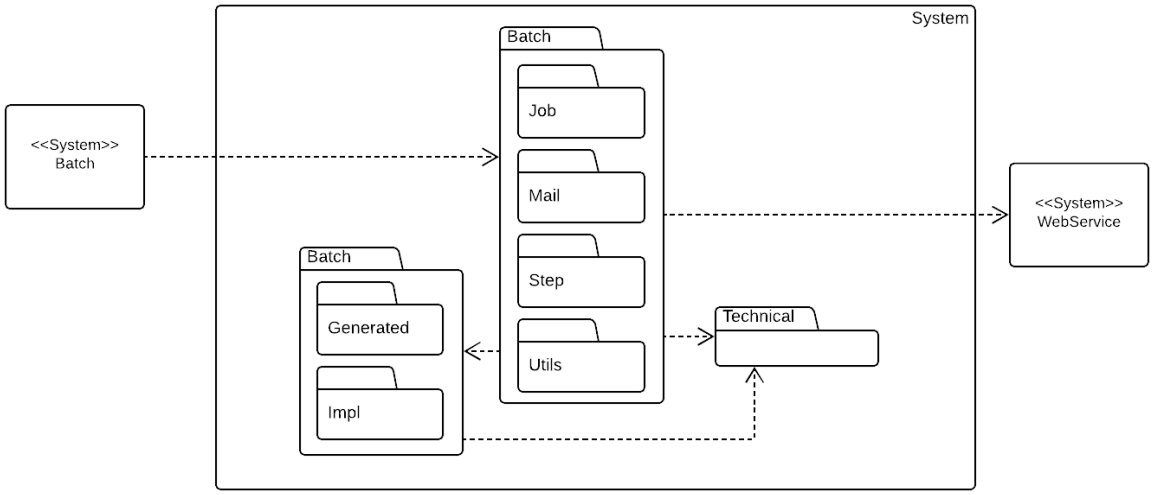

Figure 10 : Architecture et relations des modules du Batch

6. CONTACT

Auteur : Anthony Tazzari
Email : anthony.tazzari@gmail.com

Version : 1.0
Date : Janvier 2019

Repository GitHub : https://github.com/getantazri/LibraryManager