Université Blaise Pascal
UFR Sciences et Technologies
Master Informatique Décisionnelle et Logicielle

Rapport du groupe Serveur pour le projet

VIRLIBRARY

GESTION D'UNE BIBLIOTHÈQUE VIRTUELLE

Réalisé par groupe serveur :
Xia WU
Mohamed Arezki MAREK
Bilal KIFAHI
Ndiaye SERIGNE

Encadrés par :
Mr. Mamadou KANTÉ

23 Mars 2014

Abstraction

Pour le compte de l'UE Étude de cas décisionnel/intégration, il nous a été proposé de mettre en place un système de prêt de livres en ligne.

Notre tâche consistait à mettre en place un serveur web où sera déployée notre application ainsi que l'ensemble de nos web services qui communiquent avec une base de données pour assurer l'ensemble des opérations sur les livres, les clients, etc. Le serveur devra être accessible par différents clients (Web, Mobile etc…).

Ce travail, prévu pour trois mois, consistait à travailler avec différents groupes chargés de piloter ce projet de la phase étude à la phase réalisation.

Ce rôle exige un planning assez conséquent que l'on peut difficilement tenir en trois mois. A travers différentes documentations à différents stades d'avancements, on a pu balayer les différentes étapes permettant de mener à bien ce travail.

Remerciements

Au début, nous souhaiterons remercier Mr Mamadou Kanté pour l’opportunité qu’il nous a donnée, travailler sur un projet telle que virlibrary est une expérience qui a été assez bénéfique pour tous les participants. Sur le niveau organisationnel comme technique. C’était aussi une opportunité qui nous a permis d’avoir une vue assez consistante de l’élaboration d’un projet, et ceci depuis la conception jusqu'à la réalisation.

On aimerait aussi remercier les autres groupes du projet, ainsi que leurs représentants pour leur support et leur implication dans l’avancement synchrone des tâches.

Et au final, on aimerait faire parvenir nos remerciements au corps enseignant et administratif de l’université Blaise Pascal, qui n’ont pas manqué à nous venir en aide chaque fois que l’occasion se présentait.

Table des matières

1. Introduction

Comme convenu chaque année, les étudiants de la deuxième année en Master sont chargés de réaliser un projet afin qu'ils se préparent au monde de l'entreprise, et qu'ils partent sur de bonnes bases pour réaliser leurs stages. Cette année nous fîmes chargé de réaliser une application de gestion de prêts de livres. A vrai dire une vraie bibliothèque virtuelle. nommée virlibrary.

Ce projet réunissait quatre groupe, chacun se chargeant d'une tâche définie au préalable. La notre consistait a réaliser la partie serveur de l'application. En gros tout ce qui est gestion et contrôle d'accès au données. Pour cela on a travaillé main à main avec les autres groupes, afin de mettre à leur disposition les données de la manière la plus adéquate possible, ceci dit ce n'était pas une simple tâche, puisque chaque personne avait une vision différente de la problématique. mais grâce aux discussions et réunions on a pu comparer les différents points de vus et choisir les plus pertinents pour l'avancement de notre projet.

Au tout début notre travail consistait à tester les différentes technologies susceptibles d'êtres utilisées. En parallèle un travail de compréhension et de modélisation s'était automatiquement mis en route, afin de bien cerner le périmètre de nos futurs travaux.

Notre solution pour la partie serveur est basé sur une architecture orienté service, et qui dit web service dit interopérabilité et possibilité d'adaptation future a un environnement en constant changement.

2. Analyse

2.1 Spécifications Conceptuelles

Avant la réalisation de chaque projet, il faut toujours procéder à une analyse des besoins et des contraintes, ainsi que les différentes solutions aidant à la construction de solution efficaces et durables pour de telles problématiques. Notre tâche alors, était de cerner les limites des besoins des autres groupes en termes de données, et de performances.

Nos différentes réunions avec les autres groupes avaient pour but l'élaboration d'un modèle de données et d'une conception, qui soit robuste simple et qui répondrait à tous nos besoins.
Pour l'aspect technique on établissait des réunions avec des propositions de chaque élément du groupe. Qu'on mettait à l'épreuve pour au final ne garder que les éléments qui sauront répondre aux exigences instaurées par le projet.

Notre solution finale était de réaliser un ensemble de services web qui travailleront en coordination pour répondre aux besoins demandés. Ceci a pour but d'optimiser la manière d'accéder aux données ainsi que le temps de réponse. Puisqu'un des grands problèmes qu'on avait à résoudre était de minimiser au maximum le temps de réponse du serveur. Ce qu'on a réussi à faire avec notre architecture distribuée de web services.
L'architecture comme déjà cité se présente comme un ensemble de web services. Avec chaque web service qui fait une tâche spécifique (voir figure 1).

Liste des web services hébergés sur le serveur :

Gestion des Livres : pour tout ce qui est ajout, suppression… Etc. de livres.
Système des Recommandation : pour recommander des livres à un utilisateur suivant ses goûts.
Gestion des Amis : pour gérer les relations entre les utilisateurs.
Gestion des Utilisateurs : pour gérer les utilisateurs (droits d'accès, authentification …).
Gestion de la messagerie : gestion de la messagerie.
Gestion des emprunts : gérer la fonction d'emprunts de livres.
Systèmes des Transaction : gérer la transaction entre les clients.

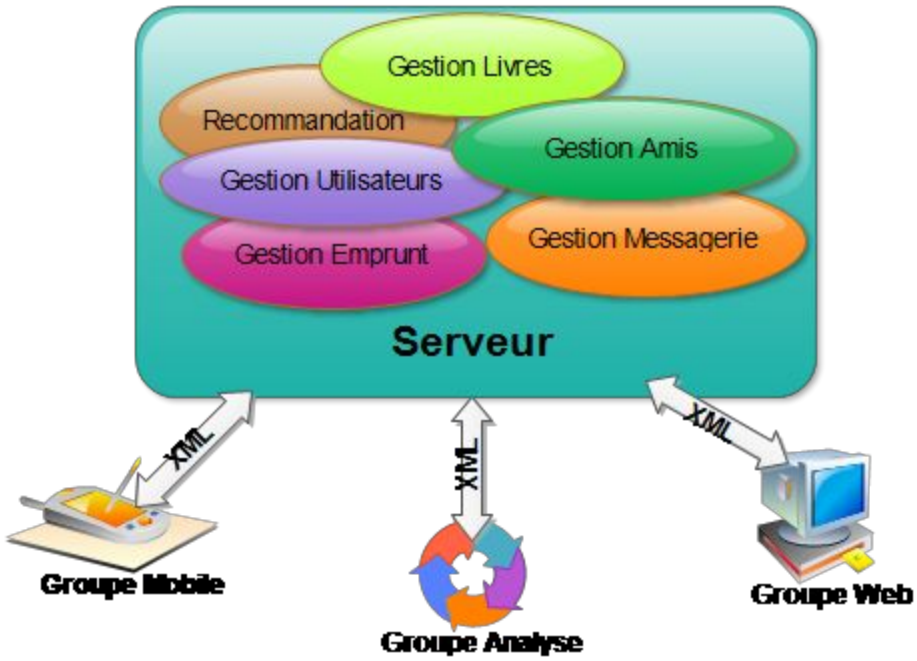

figure 1

Pourquoi avoir choisi une telle architecture ? :

Au début nous avons envisagé de mettre en œuvre une mono architecture, à service web unique. Mais dans une telle situation des questions de performances et de temps de réponse se posaient. Un autre problème était présent : si une fonctionnalité tombe en panne il faut arrêter tout le serveur pour procéder à une réparation. On y perd en interopérabilité. Puisque tous les services seront présents sur un seul serveur. Par contre l'architecture modulaire permet de distribuer les services à sa guise sur différentes machines. Limitant l'impact des pannes et facilitant la reprise après panne. Ceci dis le chapitre suivant expliquera au mieux les mécanismes qu'on a utilisée pour prévenir les pannes et les sinistres.

2.2 Spécification Technique

En parallèle avec la conception, il fallait aussi définir les éléments et les outils à utiliser. Ceci dit il fallait faire des tests pour choisir les technologies adéquates à notre cas d'étude. Au début il fallait définir le type de base de données pour stocker les données, le type de serveur d'application, ainsi que la technologie web service à utiliser.

Au niveau de la base de données on a utilisé le SGBD MySql pour sa légèreté ainsi que sa facilité d'utilisation. MySql permet aussi de mettre en place des stratégies de reprise après panne, comme la mise en miroir, la réplication et le transfert de logs.

Ainsi au niveau de la journalisation on à décider d'implémenter une résistance au pannes à base de fichiers de logs, ou ce qu'on appelle communément « log shipping » ou bien transfert de log. L'avantage de cette solution par rapport aux autres solutions c'est le faite qu'elle ne consomme pas beaucoup de ressources en termes de

temps d'exécution, puisqu'elle n'est pas une solution en temps réelle, elle ne garde que l'état de la base de données à un instant donné.

Pour le serveur d'application on a lancé des tests sur l'outil appelé SoapUI pour tester les différents services web, sous différents serveurs d'applications. On avait le choix entre JBoss et GlassFish. Des tests de surcharge en étaient effectués sur les deux serveurs, et la conclusion était le choix de JBoss en tant que serveur d'application.

Et au final il nous fallait savoir le type de technologie web services qu'on va utiliser. On a dû faire des tests pour savoir qui est le plus approprié dans notre cas : SOAP ou bien REST. Surtout On a finis par choisir la technologie REST : plus légère que SOAP. Puisque SOAP a besoin de fabriquer des enveloppes SOAP en format XML pour chaque réponse. Ceci a pour conséquence de bloquer les communications surtout que le nombre d'objet qui transitera augmentera d'une manière drastique, et il faut alors compter une enveloppe SOAP en format XML pour chaque message envoyé. En plus on n'a pas besoin de garder l'état des communications, alors la solution adéquate était d'opter pour la technologie REST avec configuration Stateless.

Une petite synthèse est présentée dans la figure qui suit (voir figure 2)

figure 2

3. Base de données

3.1 Modèle Conceptuel de Données (MCD)

Plusieurs Modèles ont été proposés que se soit par les membres de notre groupe ou par les autres groupes qui participent au projet. Après plusieurs réunions et discussions entre les différents groupes, le modèle suivant a été adopté.

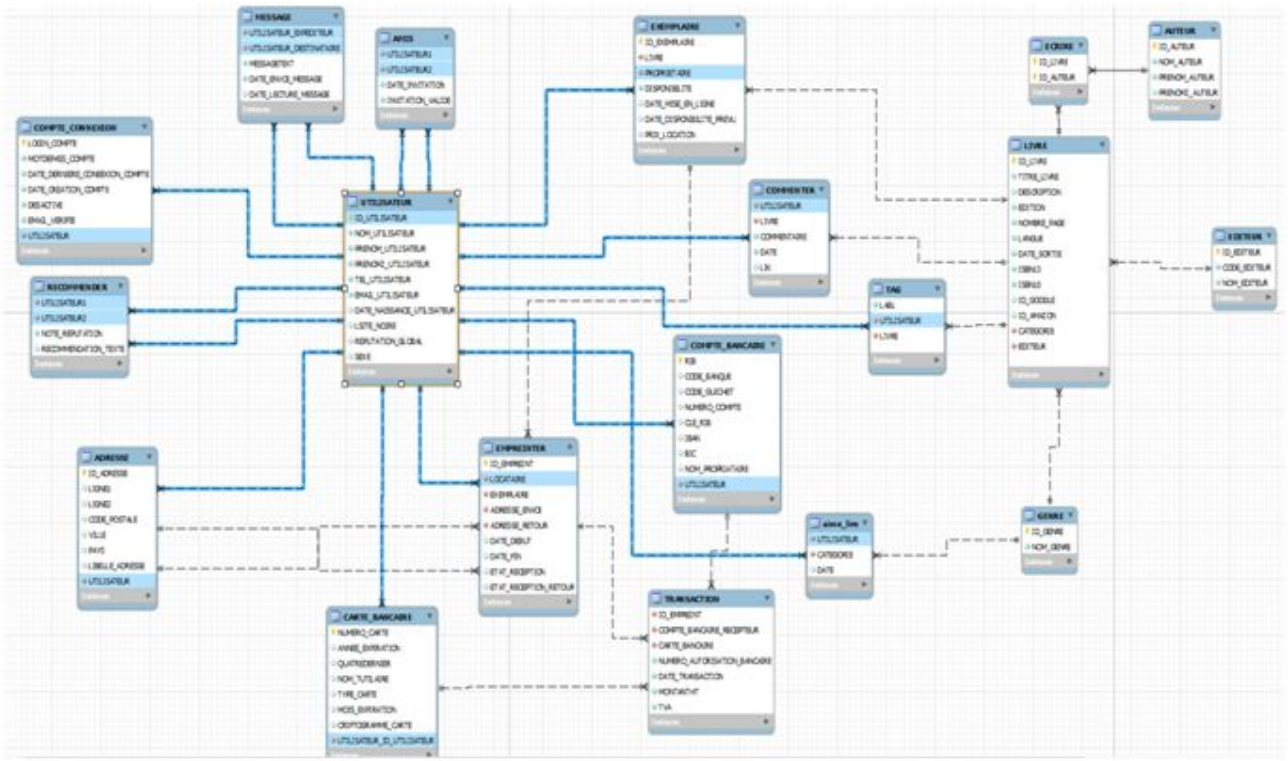

figure 3

3.2 Validation et Vérification du MCD

Pour vérifier et valider le MCD, nous avons mis en place et exécuté plusieurs scénarios de tests qui ont concerné l'ensemble des fonctionnalités qui devaient être implémentées au sein de notre application. Voici quelques uns des scénarios :

Scénario 1 : (Ajout d'un utilisateur, d'une adresse, d'un compte de connexion, d'un compte bancaire et d'une carte bancaire)

L'utilisateur Blaise Pascal s'inscrit au site afin d'emprunter des livres et de mettre à disposition des autres utilisateurs ses propres livres.

Voici les requêtes SQL qui permettent d'implémenter les différentes fonctionnalités :

```
INSERT INTO `utilisateur`
VALUES (1,' Pascal ','Blaise','' ,'0123456789','1623-06-29',0,4,'M');

INSERT INTO `adresse`
VALUES (1,' 34 Avenue Carnot ','',63000 ,'Clermont-Ferrand','France','ADRESSE PRINCIPALE',1);

INSERT INTO `compte_connexion`
VALUES ('blaise.pascal@gmail.com','bl@isep@sc@lp@ssword',' 2014-04-06 00:22:15','2014-04-05 01:00:12',1, 1,1);

INSERT INTO `carte_bancaire`
VALUES (01234567890123,'2017','4567','Pascal' ,'MasterCard','Septembre','890',1);

INSERT INTO `compte_bancaire`
VALUES ('0123456789012345','67890','12345', '6789012345A', '67', 'FR0067890123456789012345A12', 'ABCDEFGH', 'Pascal', 1);
```

Scénario 2 : (Ajout d'un livre, d'un genre de livre, d'un éditeur, d'un auteur et de la fonctionnalité écrire un livre).
Ajouter un nouveau livre « Mémé » en supposant qu'il appartient à un nouveau genre, il est édité par un éditeur non encore renseigné et écrit par un auteur non présent dans la base.
Voici les requêtes SQL qui permettent d'implémenter les différentes fonctionnalités :

```
INSERT INTO `genre`
VALUES (1,' Littérature française ');

INSERT INTO `editeur`
VALUES (1,1,' ICONOCLASTE EDITIONS ');

INSERT INTO `livre`
VALUES (1,'Mémé','«Un bijou de tendresse»','Première' ,162,'Français','2014-01-16','978-2913366619', '2913366619',0,0,1,1,'', 10.99, 1, 0,0);

INSERT INTO `auteur`
VALUES (1,' Phillipe Torreton');

INSERT INTO `ecrire`
VALUES (1,1);
```

Scénario 3 : (Ajout d'un exemplaire d'un livre et d'un tag)

L'utilisateur Pascal propose un exemplaire du livre « mémé » avec les informations suivantes :

- **Date de mise en ligne :** 01/02/2014
- **Date de disponibilité :** 02/04/2014
- **Prix de location :** 3.5 Euro

Et taggue son exemplaire comme étant un roman.
Voici les requêtes SQL qui permettent d'implémenter les fonctionnalités ci-dessus :

```
INSERT INTO `exemplaire`
VALUES (1,1,1, 1, '2014-02-01', '2014-02-02',3.5);

INSERT INTO `tag`
VALUES ('roman',1,1);
```

Scénario 4 : (Empreint et transaction)

L'utilisateur Seguin décide d'emprunter un exemplaire du livre « mémé » le 06/02/2014 pour le rendre le 13/02/2014.
Ce qui va se traduire par les requêtes SQL suivantes :

```
INSERT INTO `empreinter`
VALUES (1,2,1, 2,1, '2014-02-06', '2014-02-13','bon','')
```

Et qui va déclencher une transaction pour débiter le compte de Seguin et créditer le compte de Pascal de 3.5 Euro.
Ce qui va se traduire par les requêtes SQL suivantes :

```
INSERT INTO `transaction`
VALUES (1,'0123456789012345','45678901234567', '4698','2014-02-06', 2.91, 0.59);
```

Scénario 5: (Envoyer un message)

L'utilisateur Pascal envoie un message à Seguin pour savoir s'il a reçu le livre qu'il lui a envoyé.
Ce qui va se traduire par les requêtes SQL suivantes :

```
INSERT INTO `message`
VALUES (1,2,' Bonjour, Mr Seguin, vous avez reçu le livre que je vous ai envoyé? ','2014-04-08 14:20:00','2014-04-09 01:14:25')
```

Scénario 6: (Commenter et noter un livre)

L'utilisateur Seguin a commenté et noté le livre « mémé » après l'avoir lu.
Ce qui va se traduire par les requêtes SQL suivantes :

```
INSERT INTO `commenter`
VALUES (2,1,' Roman très passionant\, je le recommande vivement','2014-04-11',5)
```

Scénario 7: (Recommander un utilisateur)

L'utilisateur Blaise Pascal recommande Marc Seguin et lui donne la note de 5 après lui avoir rendu le livre qu'il lui a lui prêté en bon état et dans le délai prévu.
Ce qui va se traduire par les requêtes SQL suivantes :

```
INSERT INTO `recommender`
VALUES (1,2, 5,'Je lui ai loué un livre et il me l\'a rendu à la date prévue et dans un très bon état, je le recommande vivement!!!')
```

3.3 Sauvegarde et journalisation

Afin de ne pas trop surcharger le rapport, nous avons décidé de ne pas inclure les explications concernant les mécanismes de sauvegardes et de journalisation de MySQL. L'explication est fournie dans les documents suivants:
Journalisation: http://stackoverflow.com/questions/9950246/mysql-transaction-journaling
Sauvegarde: http://dev.mysql.com/doc/refman/5.0/fr/backup.html
Pour la génération des données de manière aléatoire, nous avons utilisé l'outil en ligne "generatordata" http://www.generatedata.com/#generator

4. Serveur

4.1 Tests statistiques et comparaisons

L'objectif des tests statistiques est de justifier le choix du type du serveur d'applications à utiliser. Et les deux indices essentiels à l'évaluation d'un serveur d'application sont : le temps de réponse (rapidité), la stabilité et robustesse avec lesquels ces derniers répondent à un nombre important de processus exécutés en parallèle.

4.1.1 Comparaison entre JBoss EAP 6.0 et Glassfish 4.0

Les deux serveurs d'applications que nous avons décidé d'évaluer sont JBoss EAP 6.0 et Glassfish 4.0.
Voici les tests que nous avons effectué sur ces deux derniers: P100/ T5000, P500/ T25000, P1000/ T50000 et P5000/T250000[1]. Où P représente le nombre de processus exécutés en parallèle et T le nombre de transactions maximum lancées par chaque processus par seconde.

Voici les résultats que nous obtenu après avoir exécuté les tests ci-dessus:

- Les performances de JBoss EAP[2] 6.0 sont relativement stables. Par contre, les performances de Glassfish 4.0 sont beaucoup moins stables surtout lorsque le nombre de transactions par seconde est élevé (>250000).
- Par contre la rapidité et le temps de réponse du serveur Glassfish 4sont meilleurs que ceux du serveur JBoss EAP 6..
- Pour conclure, nous pouvons dire que chacun de ces deux serveurs d'application présente des avantages et des inconvénients. L'avantage de JBoss EAP 6 est sa stabiliité et l'avantage de Glassfish est sa rapidité.

De part la nature de notre application, bibliothèque virtuelle, nous avons décidé de choisir le serveur d'application JBoss EAP 6.0 parce que la stabilité et la robustesse sont beaucoup plus important que la rapidité dans ce type d'applications.

4.1.2 La comparaison entre WSDL et REST sur le serveur JBoss EAP 6.0

La deuxième série de tests que nous avons effectué est la comparaison entre WSDL et REST sur le serveur JBoss EAP 6.0 et les critères pris en compte sont la rapidité et la stabilité.
Après avoir effectué les tests indiqués ci-dessus, nous avons constaté que les services web de type WSDL sont plus rapides que les services web de type REST. Par contre, les services web de type REST sont plus stables que les services web de type WSDL surtout lorsque le nombre de transaction par seconde est élevé (>250000).
En conclusion, les deux types de services web ont leurs avantages et inconvénients, d'un point de vue rapidité WSDL est meilleur mais d'un point de vue stabilité REST est meilleur.

Pour notre application, le choix s'est porté sur REST pour les raisons évoquées ci-dessus et il s'est fait aussi en concertation avec les autres groupes.

4.1.3 Tests bursting des demandes massives avec les vrais services web

La prochaine étape des tests s'est faite en utilisant les services web qui sont réellement implémentés par notre application.
L'objectif de ce type de test est de vérifier la robustesse de nos web services implémentés en REST sur JBoss EAP 6.
Les résultats obtenus confirment ce qui a été dit précédemment.

Nous avons constaté que les services web REST sont robustes et répondent bien aux caractéristiques de la bibliothèque virtuelle qui sera utilisée par un nombre important d'utilisateurs mais dont les traitements sont plutôt légers.

Donc pour conclure sur les choix effectués que ce soit pour le serveur d'application ou sur le type de services web nous pouvons dire que:

- Le choix de JBOSS EAP 6 est dû à sa stabilité et sa robustesse par rapport à Glassfish 4 même si il est moins rapide que ce dernier.
- L'implémentation des services web en WSDL se fait à la fois côté serveur et côté client. Donc, on doit avoir une bonne capacité de traitement à la fois côté serveur et côté client. Et vu que la capacité de traitement des clients mobile iOS ou Android est assez limitée, ça ce serait plus judicieux de choisir REST au lieu de WSDL. Et ce même si les services web de type REST sont moins fiables et moins rapides que ceux de type WSDL.
- Le choix de REST est aussi dû au fait que ce dernier est plus stable et robuste que WSDL sur des données massives.

4.2 Installation du serveur

Pour le déploiement de notre projet une machine virtuelle a été créé et qui tourne sur Linux Debian 7[3] dont les caractéristiques sont : CPU 2.5 GHz, RAM 8 Go et Disque dur 25 Go pour /home et 60 Go pour /données. Voici les logiciels installés sur le serveur : MySQL Community Server 5.6.16, MySQL Workbench, Httpd, Apache, JBoss EAP 6, Eclipse, SoapUI, LoadUI, Git.

Les étapes essentielles pour démarrer la machine virtuelle Serveur en Linux Debian 7:

- La configuration de SSH et Xmine sur Debian[4]
- L'installation, la configuration et le démarrage du serveur JBoss EAP 6
- La configuration et la gestion du Mysql community serveur et du Mysql workbench[5]
- L'installation et la configuration de l'Apache Httpd serveur pour la partie WEB
- L'installation et la configuration du serveur SVN et Git
- L'installation et la configuration de l'Eclipse Kepler
- L'installation et la configuration du SoapUI[6] et du LoadUI[7]
- L'installation et la configuration du Subversion en Debian
- Mise à jour du JRE 1.6 au JSE 1.7

4.3 Déploiement, configuration et maintenance des services web

Le déploiement du projet ne peut pas se faire directement sur JBoss EAP 6 à partir de l'IDE Eclipse Kepler. Pour déployer ce dernier, nous devons d'abord l'exporter dans un répertoire. Ensuite, importer le projet via la console d'administration de JBoss EAP 6 et puis déployer ce dernier. Une fois le déploiement effectué,les services seront mises en ligne et seront accessibles aux utilisateurs.

La configuration du projet a nécessité la mise à jour de Java 6 vers Java 7 ainsi que la mise à jour de Mysql connector Java. Les paramètres d'accès au serveur MySQL tel que l'url du serveur, le login et le mot de passe sont spécifiés dans un fichier de configuration .properties[8].

La maintenance des services web est faite manuellement. Ce qui fait qu'à chaque fois qu'une mise à jour est effectuée sur un service web, le projet doit être redéployé.

5. Services Web

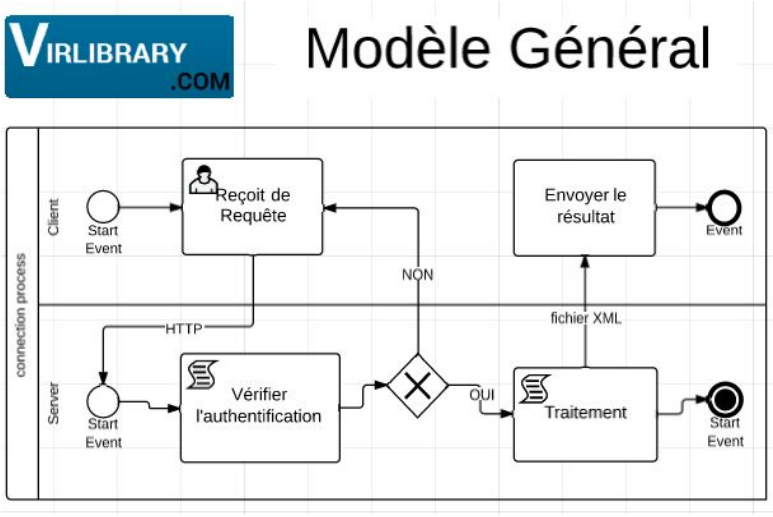

figure 4

Pour l'implémentation des services web, nous avons utilisé l'API JAX-RS pour concevoir et mettre en œuvre nos services coté serveur. Le développement des services web est basé sur des POJO (Plain Old Java Project) en utilisant des annotations spécifiques à JAX-RS.

Quelque soit le type de clients utilisé, web ou mobile, L'accès au service web se fait en envoyant le nom du service web, le login et le mot de passe de l'utilisateur ainsi que les paramètres du web service. Ensuite, le service web vérifie que l'utilisateur est bien authentifié. Si l'authentification du client est correcte, alors le service web sera exécuté et renverra le résultat au client au format XML. Sinon, le service web retournera *"<Authentification>false</Authentification>"*.

5.1 Services web généraux

Authentification d'utilisateur

Authentification c'est l'un des services web les plus importants, il est évoqué à chaque appel de n'importe quel autre service web. Il permet de vérifier qu'un utilisateur est bien enregistré dans la base de données et que son mot de passe est correct. Le service web prend en paramètre le login et le mot de passe d'un utilisateur et renvoi l'identifiant de l'utilisateur en cas de succès ou 0 dans le cas contraire. La requête SQL exécutée est indiquée en annexe[9] .

Existence d'un login

IsLoginExist c'est un autre service web basique permet de vérifier l'existence d'un login donné par un utilisateur. Le service web prend en paramètre un login et renvoi "true" en cas de succès ou "false" dans le cas contraire. La requête SQL exécutée par le web service est indiquée en annexe[10] .

Existence d'un utilisateur

IsUserExist c'est un autre service web très important, il est évoqué à chaque appel à la plupart des services web relatifs à la gestion des utilisateurs. Il permet de vérifier qu'un utilisateur est bien enregistré en base de données. Le service web prend en paramètre un identifiant et renvoi "true" en cas de succès ou "false" dans le cas contraire. La requête SQL exécutée par le web service est indiquée en annexe[11] .

Existence d'un livre

IsLivreExist c'est un autre service web très important, il est évoqué à chaque appel à la plupart des services web relatifs à la gestion des livres. Il permet de vérifier qu'un livre est bien enregistré en base de données. Le service web prend en paramètre un identifiant et renvoi 1 en cas de succès ou 0 dans le cas contraire. La requête SQL exécutée par le web service est indiquée en annexe[12] .

Existence d'un exemplaire d'un livre

IsExemplaireExist c'est un service web aussi important que les précédents, il est évoqué à chaque appel à la plupart des services web relatifs à la gestion des exemplaires de livres. Il permet de vérifier qu'un exemplaire de livre est bien enregistré en base de données. Le service web prend en paramètre un identifiant et renvoi 1 au cas ou l'exemplaire existe ou 0 dans le cas contraire. La requête SQL exécutée par le web service est indiquée en annexe[13] .

5.2 Gestion des amis

Cette partie regroupe l'ensemble des web services relatifs à la gestion des amis, c'est à dire supprimer un ami, ajouter un ami, recommander un livre à un ami, au sein de l'application.

Ajouter un ami

Ce web service est intitulé "**AjouterAmi**", il permet de faire une demande d'ajout comme ami à un membre du site et de l'enregistrer en base de données. Et ceci après avoir vérifié l'existence des utilisateurs concernés en en exécutant le web service *IsUserExsit.* Le web service prend comme paramètres les éléments suivants, identifiant des deux utilisateurs concernés. Et il renvoie comme résultat 1 en cas de succès de l'enregistrement ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[14] .

Supprimer un ami

Ce web service est intitulé "**SupprimerAmi**", il permet de supprimer la relation d'amitié entre deux utilisateurs. Et ceci après avoir vérifié l'existence des utilisateurs concernés en en exécutant le web service *IsUserExsit.* Le web service prend comme paramètres les éléments suivants, identifiant des deux utilisateurs concernés. Et il renvoie comme résultat un fichier XML contenant la nouvelle liste des amis de l'utilisateur en cas de succès de la suppression ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[15] .

Recommander un livre à un ami

Ce web service est intitulé "**CommenterLivreAmi** ", il permet un utilisateur de recommander un livre à un de ses amis. Et ceci après avoir vérifié l'existence du livre en question en exécutant le web service *IsLivreExsit* et après avoir aussi vérifié l'existence de l'utilisateur pour lequel on veut recommander un livre en exécutant le web service *IsUserExsit.* Le web service prend comme paramètres les éléments suivants, identifiant des deux utilisateurs concernés et l'identifiant du livre. Et il renvoie comme résultat 1 en cas de succès de la requête ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[16] .

Modifier l'état d'une invitation

Ce web service est intitulé "**ModifierInvitation** ", il permet à un utilisateur de modifier l'état d'une invitation à devenir ami. Et ceci après avoir vérifié l'existence de l'invitation en question en exécutant le web service *IsInvitationExsit.* Le web service prend comme paramètres l'identifiant de l'invitation en questi. Et il renvoie

comme résultat 1 en cas de succès de la requête ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[17] .

5.3 Gestion des comptes utilisateur

Cette partie regroupe l'ensemble des web services relatifs à la gestion des comptes des utilisateurs. Elle contient notamment les web services qui permettent de désactiver un utilisateur, ajouter un utilisateur, modifier les préférences d'un utilisateur.

Déactiver un utilisateur

Ce web service est intitulé "**DeactiverUser** ", il permet de désactiver le compte d'un utilisateur sans le supprimer. Et ceci après avoir vérifié l'existence de l'utilisateur en question en exécutant le web service *IsUserExsit.* Le web service prend comme paramètres l'identifiant de l'utilisateur concerné. Et il renvoie comme résultat 1 en cas de succès de la désactivation ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[18] .

Ajouter un utilisateur

Ce web service est intitulé "**AjouterUser**", il permet d'ajouter un nouvel utilisateur après avoir vérifié que login renseigné par ce dernier n'est pas déjà utilisé et ceci en exécutant le web service *IsLoginUsed.* Le web service prend comme paramètres l'identifiant de l'utilisateur concerné. Et il renvoie comme résultat 1 en cas de succès de l'ajout ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[19] .

Consulter un utilisateur

Ce web service est intitulé "**ConsulterUser**", il permet de consulter les informations d'un utilisateur donné après avoir vérifié que l'utilisateur en question existe dans la base de données et ceci en exécutant le web service *IsUserExsit.* Le web service prend comme paramètres l'identifiant de l'utilisateur concerné. Et il renvoie comme résultat un fichier XML contenant les informations de l'utilisateur en cas de succès de la requête ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[20] .

Modifier un utilisateur

Ce web service est intitulé "**ModifierUser**", il permet de modifier les informations d'un utilisateur. D'abord, le service crée une requête en SQL qui vérifie l'existence de l'utilisateur. Si l'utilisateur n'existe pas, le service web retournera immédiatement le résultat "*<IsUserUsed>" +false+ "</IsUserUsed>*". Sinon, le service crée une requête en SQL[21] qui modifie les information de l'utilisateur. Enfin, il crée d'abord une script SQL qui retournera les informations d'utilisateur dans un fichier XML.

Modifier la cofiguration d'utilisateur

Ce web service est intitulé "**ModifierConfigUser**", il permet de modifier les préférences d'un utilisateur. D'abord, le service crée une requête en SQL qui vérifie l'existence de l'utilisateur. Si l'utilisateur n'existe pas, le service web retournera immédiatement le résultat "*<IsUserUsed>" +false+ "</IsUserUsed>*". Sinon, le service crée une requête en SQL[22] qui modifie la configuration de l'utilisateur. Enfin, il crée d'abord une script SQL qui retournera la configuration d'utilisateur dans un fichier XML.

5.4 Gestion de la messagerie

Cette partie regroupe l'ensemble des services web relatifs à la gestion de la messagerie interne des utilisateurs. Il contient notamment les services web qui permettent d'enregistrer un message envoyé à un utilisateur, de

modifier l'état d'un message (de non lu à lu), de consulter l'ensemble des messages d'un utilisateur, et de consulter le contenu et les informations d'un message en particulier.

Envoyer un message:

Ce web service est intitulé "**EnvoyerMessage**", il permet d'enregistrer un nouveau message en base de données. Il prend comme paramètres les éléments suivants, identifiant de l'émetteur, l'identifiant du récepteur, ainsi que le contenu du message. Et il renvoie comme résultat 1 en cas de succès de l'envoi ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[23] .

Modifier l'état d'un message:

Ce web service est intitulé "**ModifierMessage**", il est exécuté lorsque le destinataire lit son message. Il permet de faire passer l'état du message de non lu à lu et d'insérer la date de lecture du message. Il prend comme paramètres l'identifiant du message. Et il renvoie comme résultat 1 en cas de succès de la modification ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[24] .

Consulter les messages d'un utilisateur:

Ce web service est intitulé "**ConsulterListeMessage"**, il est exécuté lorsqu'un utilisateur veut voir la liste de ses messages. Il prend comme paramètres l'identifiant d'un utilisateur. Et il renvoie comme résultat un fichier XML contenant l'ensemble des messages d'un utilisateur en cas de succès de la requête ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[25] .

Consulter le détail d'un message:

Ce web service est intitulé "**ConsulterInfoMessage"**, il est exécuté lorsqu'un utilisateur veut voir le détail d'un message en particulier. Il prend comme paramètre l'identifiant d'un message. Et il renvoie comme résultat un fichier XML contenant les informations qui concernent le message tel que l'identité de l'émetteur, le contenu du message et la date d'envoi du message en cas de succès de la requête ou 0 dans le cas contraire. La requête SQL exécutée par le web service est indiquée en annexe[26].

5.5 Gestion des emprunts

Cette partie regroupe l'ensemble des services web relatifs à la gestion des emprunts effectués par les différents utilisateurs de l'application. Il contient notamment les services web qui permettent d'emprunter un livre, de consulter les emprunts en cours d'un utilisateur, de consulter l'historique de tout ses emprunts, d'annuler un emprunt ou de modifier l'état d'un emprunt (Indiquer l'état de réception d'un livre et son état de renvoi).

Emprunter un livre:

Ce web service est intitulé "**EmpreinterLivre**", il permet d'ajouter un nouvel empreint dans la base de données. Il prend les éléments suivants comme paramètres, identifiant de l'émetteur, l'identifiant du récepteur du livre à emprunter, l'adresse où sera envoyé le livre, l'adresse à laquelle, on doit le renvoyer ainsi que la date de début et de fin de l'empreint. Et il renvoie comme résultat 1 en cas de succès de l'empreint ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[27] .

Consulter les emprunts en cours:

Ce web service est intitulé "**ConsulterEmpreintEnCours**", il permet de consulter les emprunts qui sont toujours en cours pour un utilisateur donné. Il prend en paramètre l'identifiant d'un utilisateur et renvoie

l'ensemble des informations concernant les livres actuellement empruntées. La requête SQL exécutée par le web service est indiquée en annexe[28] .

Annuler un emprunt:

Ce web service est intitulé "**AnnulerEmpreint**", il permet d'annuler un prêt qu'un utilisateur avait prévu ainsi que la transaction qui va avec . Il prend en paramètre l'identifiant de l'emprunt et renvoie 1 si l'emprunt a été annulé ou 0 dans le cas contraire. La requête SQL exécutée par le web service est indiquée en annexe[29] .

Consulter l'historique des emprunts:

Ce web service est intitulé "**ConsulterEmpreint**", il permet de consulter l'ensemble des emprunts effectués par un utilisateur. Il prend en paramètre l'identifiant d'un utilisateur et renvoie l'ensemble des livres empruntées par ce dernier depuis son inscription. Ce web service peut être utilisé pour recommander un livre à l'utilisateur en fonction de cette liste. La requête SQL exécutée par le web service est indiquée en annexe[30] .

5.6 Gestion des transactions

Cette partie regroupe l'ensemble des services web relatifs à la gestion des transactions effectuées lors de l'emprunt de livres. Il contient notamment les services web qui permettent d'ajouter une transaction, et d'annuler une transaction.

Ajouter une transaction:

Ce web service est intitulé "**AjouterTransaction**", il permet d'ajouter une nouvelle transaction lorsqu'un utilisateur effectue un prêt de livre et ceci afin de débiter le compte du locataire et créditer le compte du propriétaire du livre. Il prend les éléments suivants comme paramètres, l'identifiant de l'emprunt, le numéro de compte bancaire récepteur, le numéro de la carte bancaire du locataire, le numéro de l'autorisation bancaire, le montant HT de la transaction et la TVA de la transaction. Et il renvoie comme résultat 1 en cas de succès de l'enregistrement de la transaction ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[31] .

Annuler une transaction:

Ce web service est intitulé "**AnnulerTransaction**", il permet d'annuler une transaction notamment lorsqu'un utilisateur décide d'annuler le prêt d'un livre. Il prend en paramètres, l'identifiant de la transaction. Et il renvoie comme résultat 1 en cas de succès de l'annulation de la transaction ou 0 sinon. La requête SQL exécutée par le web service est indiquée en annexe[32] .

5.7 Gestion des livres

Cette partie consiste à l'implémentation l'ensemble des services web qui permet d'assurer la gestion des livres, c'est à dire ajouter un exemplaire d'une livre, récupérer la liste des livres, récupérer la nombre des exemplaires disponible d'un livre, consulter les informations d'un exemplaire etc. Ils sont déploies au sein de l'application.

Ajouter un exemplaire d'une livre

Le service web **AjouterExempleLivre** permet d'ajouter un nouvel exemplaire d'un livre. Après avoir vérifié que l'utilisateur est bien **authentifié.** Le service web vérifie que le livre existe bien en base de données en exécutant le service web **IsLivreExist.** Si le livre n'existe pas, le service web retournera le résultat "<I*sLivreExsit>" +false+ "</IsLivreExsit>*". Sinon ce dernier rajoutera l'exemplaire dans la base de données et renverra comme résultat la nouvelle liste des exemplaires du livre dans un fichier XML. Le service web prend en paramètres l'identifiant du propriétaire de l'exemplaire, l'identifiant du livre concerné ainsi que les

informations qui correspondent à l'exemplaire. La requête SQL exécutée par le web service est indiquée en annexe[33] .

Supprimer un exemplaire d'une livre

Le service web **SupprimerExempleLivre** permet de supprimer un exemplaire d'un livre**.** Le service web vérifie que l'exemplaire du livre existe bien en base de données en exécutant le service web **IsExemplaireExist.** Si l'exemplaire n'existe pas, le service web retournera le résultat "<I*sExemplaireExsit>" +false+ "</IsExemplaireExsit>*". Sinon ce dernier supprimera l'exemplaire dans la base de données et renverra comme résultat la nouvelle liste des exemplaires du livre dans un fichier XML. Le service web prend en paramètres l'identifiant de l'exemplaire, et le titre du livre. La requête SQL exécutée par le web service est indiquée en annexe[34] .

Modifier la disponibilité d'un exemplaire

Le service web **ModifierDisponibiltéExemp** permet de modifier la disponibilité d'un exemplaire d'un livre, il est exécuté lorsqu'un exemplaire d'un livre est loué ou bien rendu par son locataire**.** Le service web vérifie que l'exemplaire du livre existe bien en base de données en exécutant le service web **IsExemplaireExist.** Si l'exemplaire n'existe pas, le service web retournera le résultat "<I*sExemplaireExsit>" +false+ "</IsExemplaireExsit>*". Sinon ce dernier changera la disponibilité de l'exemplaire dans la base de données et renverra comme résultat les informations de l'exemplaire dans un fichier XML. Le service web prend en paramètres l'identifiant de l'exemplaire. La requête SQL exécutée par le web service est indiquée en annexe[35] .

Récupérer la liste des livres par catégorie

Le service web **RecupererListeLivreParCateg** permet de récupérer l'ensemble des livres d'une catégorie donnée**.** Le service web vérifie que la catégorie demandée existe bien en base de données en exécutant le service web **IsGenreExsit.** Si la catégorie n'existe pas, le service web retournera le résultat "<IsGenreExsit>"+false+ "</IsGenreExsit>". Sinon ce dernier renverra comme résultat l'ensemble des les livres de la catégorie dans un fichier XML. Le service web prend en paramètres l'identifiant d'une catégorie. La requête SQL exécutée par le web service est indiquée en annexe[36] .

Récupérer la liste des exemplaires d'un livre

Le service web **RecupererListeExemplaires** permet de récupérer l'ensemble des exemplaire d'un livre donnée**.** Le service web vérifie que le livre demandé existe bien en base de données en exécutant le service web **IsLivreExsit.** Si le livre n'existe pas, le service web retournera le résultat "*<IsLivreExsit>"+false+"</IsLivreExsit>*". Sinon ce dernier renverra comme résultat l'ensemble des exemplaires du livre et les informations de chaque exemplaire dans un fichier XML. Le service web prend en paramètres l'identifiant du livre souhaité. La requête SQL exécutée par le web service est indiquée en annexe[37] .

Récupérer l'auteur d'une livre

Le service web **RecupererAuteurLivre** permet de récupérer l'auteur ou les auteurs d'un livre donné**.** Le service web vérifie que le livre demandé existe bien en base de données en exécutant le service web **IsLivreExsit.** Si le livre n'existe pas, le service web retournera le résultat "*<IsLivreExsit>"+false+"</IsLivreExsit>*". Sinon ce dernier renverra comme résultat l'auteur ou les auteurs du livre en question dans un fichier XML. Le service web prend en paramètres l'identifiant du livre souhaité. La requête SQL exécutée par le web service est indiquée en annexe[38] .

Récupérer la liste des livres par un mot clé

Le service web **RecupererListelivreParMotCle** permet de récupérer l'ensemble des livres contenant un mot clé saisi par l'utilisateur**.** Le service web inspecte le nom et la description de chaque livre contenu dans la base de données et renvoie comme résultat la liste des livres qui contiennent le mot clé saisi dans un fichier XML ou bien 0 si aucun livre ne contient le mot clé saisi. La requête SQL exécutée par le web service est indiquée en annexe[39] .

Récupérer la liste des livres par un critique

Le service web **RecupererListeLivresParCrit** sert à récupérer la liste des livres par un critique donné par l'utilisateur. D'abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête SQL[40] contient un critique qui permet de vérifier si le critique est inclus dans un commentaire de quelque exemplaire d'un livre. Enfin, le serveur retournera une liste des livres dans un fichier XML.

Récupérer la liste des livres par un état

RecupererListeLivresParEtat est un service qui permet de récupérer la liste des livres par un état donné par l'utilisateur. D'abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête SQL[41] contient un état d'exemplaire de livre qui permet de chercher des livres. Enfin, le serveur retournera une liste des livres dans un fichier XML.

Consulter la liste des livres par un tag

RecupererListeLivreParTag est un service qui permet de récupérer la liste des livres par un critique donné par l'utilisateur. D'abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête SQL[42] contient un état d'exemplaire de livre qui permet de chercher des livres. Enfin, le serveur retournera une liste des livres dans un fichier XML.

Récupérer la liste des commentaires d'un livre

cet service web **RecupererCommLivre** permet de récupérer des commentaires d'un livre. D'abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête en SQL qui vérifie l'existe du livre par le numéro d'identité et la titre du livre. Si le livre n'existe pas, le service web retournera immédiatement le resultat "*<IsLivreExsit>" +false+ "</IsLivreExsit>*". De plus, il crée une script SQL[43] qui utilise le numéro d'identité du livre pour trouver une liste des commentaires du livre. Enfin, le service retournera un résultat qui contient une liste des commentaires du livre dans un fichier XML.

Récupérer la résume d'une livre

cet service web **RecupererResumeLivre** permet de récupérer la résume d'un livre. D'abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête en SQL qui vérifie l'existe du livre par le numéro d'identité et la titre du livre. Si le livre n'existe pas, le service web retournera immédiatement le resultat "*<IsLivreExsit>" +false+ "</IsLivreExsit>*". De plus, il crée une script SQL[44] qui utilise le numéro d'identité du livre pour récupérer sa résume. Enfin, le service retournera un résultat qui contient la résume d'un livre dans un fichier XML.

Récupérer la catégorie d'une livre

cet service web **RecupererCategLivre** permet de récupérer la catégorie d'un livre. D'abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête en SQL[45] qui vérifie l'existe du livre par le numéro d'identité et la titre du livre. Si le livre n'existe pas, le service web.

Récupérer l’éditeur d’une livre

cet service web **RecupererEditeurLivre** permet de récupérer **l’éditeur** d’un livre. D’abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête en SQL qui vérifie l’existe du livre par le numéro d’identité et la titre du livre. Si le livre n’existe pas, le service web retournera immédiatement le resultat "*<IsLivreExsit>" +false+ "</IsLivreExsit>*". De plus, il crée une script SQL[46] qui utilise le numéro d’identité du livre pour récupérer son éditeur. Enfin, le service retournera un résultat qui contient l’éditeur d’un livre dans un fichier XML.

Récupérer la nombre des exemplaires disponible d’une livre

cet service web **RecupererNbrExpDispo** permet de récupérer le nombre des exemplaires disponibles d’un livre. D’abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête en SQL qui vérifie l’existe du livre par le numéro d’identité et la titre du livre. Si le livre n’existe pas, le service web retournera immédiatement le resultat "*<IsLivreExsit>" +false+ "</IsLivreExsit>*". De plus, il crée une script SQL[47] qui utilise le numéro d’identité du livre pour récupérer son éditeur. Enfin, le service retournera un résultat qui contient le nombre des exemplaires disponibles d’un livre dans un fichier XML.

Ajouter une résume pour une livre

cet service web **AjouterResume** permet d’ajouter une résume d’un livre. D’abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête en SQL qui vérifie l’existe du livre par le numéro d’identité et la titre du livre. Si le livre n’existe pas, le service web retournera immédiatement le resultat "<IsLivreExsit>" +false+ "</IsLivreExsit>". De plus, il crée une script SQL[48] qui utilise le numéro d’identité du livre pour récupérer son éditeur. Enfin, le service retournera un résultat qui contient des résumes du livre dans un fichier XML.

Ajouter un commentaire

cet service web **AjouterCommentaire** permet d’ajouter un commentaires sur un exemplaire d’un livre. D’abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête en SQL qui vérifie l’existe du livre par le numéro d’identité et la titre du livre. Si le livre n’existe pas, le service web retournera immédiatement le resultat *"<IsLivreExsit>" +false+ "</IsLivreExsit>*". De plus, il crée une script SQL[49] qui utilise le numéro d’identité du livre pour récupérer son éditeur. Enfin, le service retournera un résultat qui contient une liste des commentaires de tous les exemplaires du livre dans un fichier XML.

Supprimer un commentaire

cet service web **SupprimerCommentaire** permet de supprimer un commentaires d’un livre. D’abord, cet service web reçoit le login et le mot de passe afin de vérifier l'authentification de l'utilisateur. Ensuite, il crée une requête en SQL qui vérifie l’existe du livre par le numéro d’identité et la titre du livre. Si le livre n’existe pas, le service web retournera immédiatement le resultat "<IsLivreExsit>" +false+ "</IsLivreExsit>". De plus, il crée une script SQL[50] qui utilise le numéro d’identité du livre pour récupérer son éditeur. Enfin, le service retournera un résultat qui contient une liste des commentaires de tous les exemplaires du livre un fichier XML.

5.8 Système de recommandation

Le système de recommandation du groupe analyse permettra de présenter à l'utilisateur les livres qui sont les plus proches à ses inclinaisons en question de lecture. Mais pour cela ils doivent avoir accès à des données qui vont leur permettre d'agir de la sorte.
Leur système de recommandation se base sur les données, qui sont analysées selon différents approches, ils solliciteront autres systèmes tels que google et amazon, tandis que le serveur va leur donner les données présentes sur la base de données. Les résultats obtenus depuis les systèmes permettront de tirer des recommandations qui vont être présentés à l'utilisateur final.

Méthodes offertes par le web service :

Récupérer les livres par catégorie :

Ce service s'intitule « **getBooksByCateg**» il permet de retourner la catégorie de livre spécifié, il prend en paramètre l'id du livre, et retourne une liste de livre de la même catégorie. Si une telle catégorie existe le web service renvoi une collection de livres, sinon il renvoi une réponse « Aucun livre».

Récupérer les livres lus par d'autres utilisateurs :

Ce service s'appelle « getReadBooks» il permet de retourner une collection de livres. Les livres qui ont été lu par les utilisateurs qui ont lu un livre de référence. Ex : les lecteurs qui ont lu ce livre ont aussi lu … le paramètre de cette méthode est l'id du livre. Au cas où il n y'a aucun livre relatif au livre de référence, le service renvoi un message : « Aucun livre relatif», sinon c'est une collection de livre qu'il envoi.

Récupérer les livres lus par les amis de l'utilisateur :

Ce service s'appelle «**getReadFriendBooks** », il retourne l'ensemble de livres lu par les amis de l'utilisateur, le service prend en paramètre l'id de l'utilisateur ainsi que l'id du livre. Il retourne une liste de livres sinon c'est un message d'erreur qui signale l'absence de livres relatives au livre référence. Un message d'erreur est aussi retourné au cas où l'utilisateur n'a pas d'amis.

Récupérer les dernières nouveautés :

Le service se nomme : « **getNewBooks** », il permet de retourner l'ensemble des nouveaux livres ajoutés. Aucun paramètre n'est requis.

Récupérer les livres populaires :

Le service est « **getPopularBooks**», retourne l'ensemble des livres populaires, retourne une liste de livres. Aucun paramètre n'est requis.

Récupérer les livres les plus consultés :

Le service « **getMostSeenBooks**», permet de retourner une liste de livres les plus consultés par les utilisateurs. Retourne une liste de livres. Aucun paramètre n'est requis.

Récupérer les livres les mieux notés :

Le service « **getMostRatedBooks**» permet de retourner les livres les mieux notés. Retourne une liste de livres. Aucun paramètre n'est requis.

Récupérer les livres les plus empruntés :

Le service « **getMostLandedBooks**» permet de récupérer les livres les plus empruntés par les utilisateurs et les amis de l'utilisateur. Retourne une liste de livres. Aucun paramètre n'est requis.

Récupérer les livres les mieux notés par les amis :

Le service « **getMostRatedFriendsBooks**» permet de récupérer les livres les mieux notés par les amis de l'utilisateur, le web service retourne une collection de livres. Il prend en paramètres l'id de l'utilisateur. Au cas où aucun livre n'est disponible il retourne un message d'erreur.

Récupérer les livres du mois :

Le service « **getBooksOfTheMonth**» permet de récupérer l'ensemble des livres les mieux notés pour un mois. Le service prend en paramètre l'id de l'utilisateur. Il retourne une collection de livres.

Récupérer les livres de l'année:

Similaire au service précédents, ce service intitulé «**getBooksOfTheYear** », permet de récupérer l'ensemble des livres les mieux notés pour une année. Le service prend en paramètre l'id de l'utilisateur. Il retourne une collection de livres. En cas d'erreur il envoi des messages d'erreur.

6. Conclusion et discussion

6.1 Conclusion

Ce projet est un complément indispensable à notre formation théorique et aux autres modules que nous avons étudié tout au long de notre cursus universitaire, particulièrement en Master. Il nous a permis d'avoir un véritable aperçu du monde de l'entreprise et de la manière dont se déroulent les projets informatiques. Et aussi de saisir l'importance de la communication et de la collaboration dans la réussite d'un projet.

Ce projet nous a aussi permis d'approfondir nos connaissances et compétences dans plusieurs domaines notamment l'analyse et la collecte des besoins utilisateurs, la conception d'application web, la conception de MCD, la conception d'architecture orientées services, l'administration de serveur de type linux, l'administaration de SGBD de type MySQL ainsi que le langage Java et les web services de type SOAP et REST….

Nous sommes convaincus que ce projet nous servira fortement dans nos futurs stages et dans nos futures expériences professionnelles.

6.2 Discussion

Selon l'implémentation des services web et les tests statistiques aux fonctionnalités de la bibliothèque virtuelle. Les tâches plus difficiles sont la conception de BDD, les modèles BPMN des service web et les tests robustesses des vrais services web sur quelque serveur d'application.

La BDD est sur la base de données relationnelles MySQL. elle est bien optimisée pour recevoir des massives demandes des services. Mais elle ne donne pas des belles performances pour traiter des échanges en fichier XML. Donc, nous pensons qu'on pourrait utiliser une base de données non-relationnelle, les bases de données NoSQL et une base de données XML afin d'accélérer les traitements à la base de données.

Le modèle BPMN nous indique bien les workflow entre des différentes tâches des services web. Mais, il n'est pas très bon pour les système qui utilise souvent des services web en temps réel. Donc, nous pensons plutôt de dessiner le modèles avec des notes pour les condition du temps.

Au niveau des tests statistiques aux fonctionnalités, nous avons utilisé SoapUI qui peut seulement tester les services web un par un sans scénario. Mais il existe LoadUI qui permet de créer des scénarios avec le modèle BPMN.

Liste de figures

Annexe

[1] 50 transactions maximum par processus (P1/ T50)
[2] Entrepris Application Plateforme
[3] Debian (/de.bjan/) est un système d'exploItation libre en GNU/Linux.
[4] La configuration et la gestion du Mysql community serveur et du Mysql workbench

- Configuration du Mysql.

```
Database changed
mysql> update user set password=PASSWORD("**************") where User='root';
Query OK, 4 rows affected (0.05 sec)
Rows matched: 4  Changed: 4  Warnings: 0
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
mysql> quit
Bye
etudiant@projet2:~$ sudo /etc/init.d/mysql start
[ ok ] Starting MySQL database server: mysqld already running.
etudiant@projet2:~$ mysql -u root -p
Enter password:
mysql> quit
Bye
etudiant@projet2:~$
```

- Configuration du Mysql workbench

```
etudiant@projet2:~/Programme/MySQL$ sudo dpkg -i
mysql-workbench-community-6.0.9-1ubu1310-i386.deb
[sudo] password for etudiant:
etudiant@projet2:~/Programme/MySQL$ /usr/bin/mysql-workbench --version &
[3] 18358
etudiant@projet2:~/Programme/MySQL$ /usr/bin/mysql-workbench --version
** Message: Gnome keyring daemon seems to not be available. Stored passwords
will be lost once quit
MySQL Workbench CE 5.2.40 8790
etudiant@projet2:~/Programme/MySQL$ /usr/bin/mysql-workbench --versioon &
```

[5] SoapUI est un logiciel libre multiplate-forme de tests fonctionnels.
[6] LoadUI est un logiciel libre multiplate-forme pour tests robusting du système.
[7] .properties est un fichier de configuration pour le projet web dynamique en Java

[8] La configuration pour SSH et Xmine vis le Debian

- [Reconfigurer le XDMCP

  Debian squeeze et debian wheezy. Il faut distinguer la machine "serveur" et "client" : dans ce cas pour faire simple, le serveur est la machine distante, le client celui sur lequel tu accèdes au clavier.
  Sur le serveur : dans `/etc/gdm3/daemon.conf`

```
[XDMCP]
Enable=TRUE
```

  puis relancer gdm3, pour prendre en compte le réglage : `$sudo /etc/init.d/gdm restart`
  Sur le client :

```
X -query IP.serveur :1
```

[9] **Authentification:** SELECT UTILISATEUR FROM compte_connexion WHERE login_compte = ?AND motdepass_compte = ?

[10] **IsLivreExist**: DELETE FROM exemplaire WHERE PROPRIETAIRE = ? AND LIVRE = ? AND ID_EXEMPLAIRE = ?

[11] **IsExemplaireExist:** SELECT * FROM exemplaire WHERE LIVRE = ? AND PROPRIETAIRE = ? AND ID_EXEMPLAIRE = ?

[12] **AjouterExempleLivre:** INSERT INTO exemplaire (LIVRE, PROPRIETAIRE, DISPONIBILITE,DATE_MISE_EN_LIGNE, DATE_DISPONIBILITE_PREVU, PRIX_LOCATION) VALUES (?, ?, ?, ?, ?, ?)

[13] **SupprimerExempleLivre:** SELECT L.ID_LIVRE, L.TITRE_LIVRE, L.DESCRIPTION, L.EDITION, L.NOMBRE_PAGE, L.LANGUE, L.DATE_SORTIE, L.ISBN13, L.ISBN10, L.ID_GOOGLE, L.ID_AMAZON, L.CATEGORIE, L.EDITEUR, L.URLIMAGE, L.PRIX, L.GRATUIT, L.averageRating, L.ratingsCount FROM livre L LEFT JOIN GENRE G ON (L.CATEGORIE = G.ID_GENRE) WHERE G.NOM_GENRE = ?

[14] **ModifierDisponibiltéExemp:** UPDATE INTO exemplaire SET DISPONIBILITE = ? WHERE ID_EXEMPLAIRE = ?AND PROPRIETAIRE = ?AND LIVRE = ?

[15] **RecupererListeLivreParCateg:** SELECT L.ID_LIVRE, L.TITRE_LIVRE, L.DESCRIPTION, L.EDITION, L.NOMBRE_PAGE, L.LANGUE, L.DATE_SORTIE, L.ISBN13, L.ISBN10, L.ID_GOOGLE, L.ID_AMAZON, L.CATEGORIE, L.EDITEUR, L.URLIMAGE, L.PRIX, L.GRATUIT, L.averageRating, L.ratingsCount FROM livre L LEFT JOIN GENRE G ON (L.CATEGORIE = G.ID_GENRE) WHERE G.NOM_GENRE = ?

[16] **RecupererListeExemplaires:** SELECT Examp.ID_EXEMPLAIRE, Examp.LIVRE, Examp.PROPRIETAIRE, Examp.DISPONIBILITE, Examp.DATE_MISE_EN_LIGNE, Examp.DATE_DISPONIBILITE_PREVU, Examp.PRIX_LOCATION FROM exemplaire Examp LEFT JOIN livre L ON (Examp.LIVRE = L.ID_LIVRE) WHERE L.ID_LIVRE = ?AND L.TITRE_LIVRE = ?

[17] **RecupererAuteurLivre:** SELECT AUT.ID_AUTEUR, AUT.NOM_AUTEUR FROM livre L NATURAL JOIN ecrire ECR NATURAL JOIN auteur AUT WHERE L.ID_LIVRE = ? AND L.TITRE_LIVRE = ?

[18] **RecupererListelivreParMotCle:** SELECT * FROM livre WHERE TITRE_LIVRE Like CONCAT('%',? , '%') OR DESCRIPTION LIKE CONCAT('%', ? , '%')

[19] **RecupererListeLivresParCrit:** SELECT L.ID_LIVRE, L.TITRE_LIVRE, L.DESCRIPTION, L.EDITION, L.NOMBRE_PAGE, L.LANGUE, L.DATE_SORTIE, L.ISBN13, L.ISBN10, L.ID_GOOGLE, L.ID_AMAZON, L.CATEGORIE, L.EDITEUR, L.URLIMAGE, L.PRIX, L.GRATUIT, L.averageRating, L.ratingsCount FROM livre L, exemplaire EXEM, tag T WHERE L.ID_LIVRE = EXEM.LIVRE AND EXEM.ID_EXEMPLAIRE = T.EXEMPLAIRE AND T.LABL = ? GROUP BY L.ID_LIVRE

[20] **RecupererListeLivresParEtat:** SELECT L.ID_LIVRE, L.TITRE_LIVRE, L.DESCRIPTION, L.EDITION, L.NOMBRE_PAGE, L.LANGUE, L.DATE_SORTIE, L.ISBN13, L.ISBN10, L.ID_GOOGLE, L.ID_AMAZON, L.CATEGORIE, L.EDITEUR, L.URLIMAGE, L.PRIX, L.GRATUIT, L.averageRating, L.ratingsCount, EXEM.DISPONIBILITE, EXEM.DATE_DISPONIBILITE_PREVU, EXEM.PRIX_LOCATION FROM livre L, exemplaire

EXEM, empreinter EMP WHERE L.ID_LIVRE = EXEM.LIVRE AND EXEM.ID_EXEMPLAIRE = EMP.EXEMPLAIRE AND EMP.ETAT_RECEPTION = ? GROUP BY L.ID_LIVRE

[21] **RecupererListeLivreParTag:** SELECT L.ID_LIVRE, L.TITRE_LIVRE, L.DESCRIPTION, L.EDITION, L.NOMBRE_PAGE, L.LANGUE, L.DATE_SORTIE, L.ISBN13, L.ISBN10, L.ID_GOOGLE, L.ID_AMAZON, L.CATEGORIE, L.EDITEUR, L.URLIMAGE, L.PRIX, L.GRATUIT, L.averageRating, L.ratingsCount FROM livre L, exemplaire EXEM, tag T WHERE L.ID_LIVRE = EXEM.LIVRE AND EXEM.ID_EXEMPLAIRE = T.EXEMPLAIRE AND T.LABL = ? GROUP BY L.ID_LIVRE

[22] **RecupererCommLivre:** SELECT C.UTILISATEUR, C.COMMENTAIRE, C.DATE, C.LIK FROM livre L LEFT JOIN commenter c ON (L.ID_LIVRE = C.LIVRE) WHERE L.ID_LIVRE = ? AND L.TITRE_LIVRE = ?

[23] **RecupererResumeLivre:** SELECT ID_LIVRE, TITRE_LIVRE, DESCRIPTION FROM livre WHERE ID_LIVRE = ? AND TITRE_LIVRE = ?

[24] **RecupererCategLivre:** SELECT GEN.ID_GENRE, GEN.NOM_GENRE FROM genre GEN, livre L WHERE L.CATEGORIE = GEN.ID_GENRE AND L.ID_LIVRE = ? AND L.TITRE_LIVRE = ?

[25] **RecupererEditeurLivre:** SELECT EDI.ID_EDITEUR, EDI.CODE_EDITEUR, EDI.NOM_EDITEUR FROM editeur EDI JOIN livre L ON (L.EDITEUR = EDI.ID_EDITEUR) WHERE L.ID_LIVRE = ? AND L.TITRE_LIVRE = ?

[26] **RecupererNbrExpDispo:** SELECT * FROM livre L, exemplaire E WHERE L.ID_LIVRE = E.LIVRE AND E.LIVRE = ? AND E.DISPONIBILITE = 1

[27] **AjouterResume:** UPDATE livre SET DESCRIPTION = ? WHERE ID_LIVRE = ? AND TITRE_LIVRE = ?

[28] **AjouterCommentaire:** INSERT INTO commenter (UTILISATEUR, LIVRE, COMMENTAIRE, DATE, LIK) VALUES (?, ?, ?, ?, ?)

[29] **SupprimerCommentaire:** DELETE FROM commenter WHERE UTILISATEUR = ? AND LIVRE = ?

[30] **EmpreinterLivre:** INSERT INTO empreinter (LOCATAIRE, EXEMPLAIRE, ADRESSE_ENVOI, ADRESSE_RETOUR,DATE_DEBUT, DATE_FIN,ETAT_RECEPTION,ETAT_RECEPTION_RETOUR) VALUES (? , ? , ? ,? , ? ,?,NULL,NULL);

[31] **ConsulterEmpreintEnCours:** select em.locataire, l.titre_livre from empreinter em, exemplaire ex, livre l where em.exemplaire = ex.id_exemplaire and l.id_livre = ex.livre and date_fin>now() and em.locataire = ?;

[32] **ConsulterEmpreint:** select em.locataire, l.titre_livre from empreinter em, exemplaire ex, livre l where em.exemplaire = ex.id_exemplaire and l.id_livre = ex.livre and em.locataire = ?;

[33] **AnnulerEmpreint :** delete from empreinter where id_empreint = ?;

[34] **AjouterTransaction :** INSERT INTO `transaction` (ID_EMPREINT, COMPTE_BANCAIRE_RECEPTEUR, CARTE_BANCAIRE, NUMERO_AUTORISATION_BANCAIRE,DATE_TRANSACTION,MONTANTHT,TVA) VALUES (?,?,?, ?,?,?,?);

[35] **AnnulerTransaction:** delete from transaction where id_empreint = ?;

[36**] EnvoyerMessage:** INSERT INTO `message` (UTILISATEUR_EXPEDITEUR, UTILISATEUR_DESTINATAIRE ,MESSAGETEXT, DATE_ENVOI_MESSAGE, DATE_LECTURE_MESSAGE, etat_message)
VALUES (?,?,?,now(),null, 0,);

[37**] ModifierMessage:** update message set etat_message =? where id_message =?;

[38] **ConsulterListeMessage:** select id_message, MESSAGETEXT from message where UTILISATEUR_DESTINATAIRE = ?;

[39] **ConsulterInfoMessage** select * from message where id_message = ?;

[40] **IsLoginExist:** SELECT 1 FROM compte_connexion WHERE LOGIN_COMPTE = ?

[41] **IsUserExist:** SELECT 1 FROM utilisateur WHERE ID_UTILISATEUR = ? AND PRENOM_UTILISATEUR = ? AND NOM_UTILISATEUR = ?

[42] **AjouterAmi:** INSERT INTO amis VALUES (?, ?, ?, ?);

[43**] SupprimerAmi:** DELETE FROM amis WHERE UTILISATEUR1= ? AND UTILISATEUR2 = ?

[44**] CommenterLivreAmi:** INSERT INTO commenter VALUES (?, ?, ?, ?)

[45**] ModifierInvitation:** UPDATE INTO amis SET INVITATION_VALIDE = ? WHERE UTILISATEUR1= ? AND UTILISATEUR2 = ?

[46**] DeactiverUser:** UPDATE INTO compte_connexion SET DEACTIVE = ?
WHERE LOGIN= ? AND UTILISATEUR = ?

[47**] AjouterUser:** INSERT INTO utilisateur VALUES (?, ?, ?, ? , ?, ?, ?, ?, ?);

[48**] ConsulterUser:** SELECT * FROM utilisateur WHERE ID_UTILISATEUR = ?

[49**] ModifierUser:** UPDATE INTO utilisateur SET `ID_UTILISATEUR = ?,
NOM_UTILISATEUR = ?, PRENOM_UTILISATEUR = ?, PRENOM2_UTILISATEUR = ?,
TEL_UTILISATEUR = ?, DATE_NAISSANCE_UTILISATEUR = ?, LSITE_NOIRE = ?,
REPUTATION_GLOBAL = ?, SEXE = ?
WHERE ID_UTILISATEUR = ?

[50**] ModifierConfigUser:** UPDATE INTO preference SET PREFERENCE = ?,VALEUR = ? WHERE UTILISATEUR=?

ESSENTIELLES DU STAGE

Durée du stage : 24 Semaines du 31 Mars 2014 au 28 Septembre 2014

Intitulé du stage : Développement d'un serveur de Notifications pour applications mobiles

Problématiques techniques abordées :

- Paramétrage de serveur
- Langage PPH, Mysql + WebServices de liaison entrée le serveur et les applications
- Langage Objective-C (IOS) et Java (Android) pour la création de SDK d'intégration et paramétrage plus la gestion des pushs
- Connaissance des données internes aux devices mobiles pour les remonter dans les statistiques
- Finalisation du cahier des charges avant lancement des développements (en liaison direct avec moi)

Descriptif de la mission : Création d'un outil type BackOffice multi profiles et multi utilisateurs permettant d'expédier et suivre les statistiques d'envoi de pushs sur application IOS et Android (Windows Phone en V2). Dans cet outil, nous trouverons les zones suivantes :

- Zone notifications
    - Envoi de push avec option Dev, Prod, gestion des Tokens, envoi par programmation de date, intégration de photo, vidéo, …
    - Historique des envois
- Zone statistiques
    - Nombre de téléchargement
    - Vision temps réel des statistique d'utilisation
    - Géolocalisation du trafic
    - Affichage des pages vues (nom des pages, volume, …)
- Zone Administration
    - Gestion de l'application
    - Gestion des utilisateurs
    - Gestion des devices
    - Gestion des paquets d'expédition
    - Téléchargement des SDK pour intégration de la solution dans les applications
    - Zone API et WebServices pour application sur mesure avec expédition direct sans passer par le BO

Rémunération : 2,875 € par heure de stage, soit 436,05 € pour un temps complet correspondant à 35 heures hebdomadaires (temps légal du travail). données de Service Public : http://vosdroits.service-public.fr/professionnels-entreprises/F20559.xhtml

Tuteur entreprise : Damien Ruiz au poste de directeur technique

Environnement de travail : Travail en Open Space sur poste Mac Mini intégrer dans le pôle développement Web et Applications

Horaire de travail : 9h-12h30 et 13h30-17h du Lundi au Vendredi

Adresse de travail : 8 route de Frugères les mines, 43410 Lempdes sur Allagnon

Téléphones :

- De la société : 04 82 82 60 50
- De Damien Ruiz : 06 98 90 74 94

Mails :

- De la société : contact@dtweb.fr
- De Damien Ruiz : damien@dtweb.fr
-