Rapport

UE Génie Logiciel

Projet 2016-2017

Étudiants : LESUR Antoine, VERON Yann

Introduction

L'objectif de ce mini-projet de Génie Logiciel était de mettre en application les différents concepts vus en cours concernant la conception et la maintenance d'un projet.
Ces concepts sont les suivants :

- Mise en place/Utilisation d'une plateforme d'intégration continue.

- Mise en place de tests unitaires et d'intégration

- Application des concepts de programmation tels que l'héritage, les interfaces, la création et la levée d'exception ou l'implémentation de design patterns.

Pour ce faire, nous avons choisi d'utiliser la plateforme GitHub en terme de plateforme d'intégration continue, les packages Maven et JUnit pour les test unitaires et d'intégration, le tout sous l'IDE Eclipse.

Cahier des charges

L'objectif de l'application commandée sera de faciliter l'insertion d'informations sur des musique dans une base de données, et de faire la transition d'un ancien système de gestion de données à l'utilisation d'une base de données.

Ces informations étaient en effet stockées sous forme de fichier texte de la forme suivante :

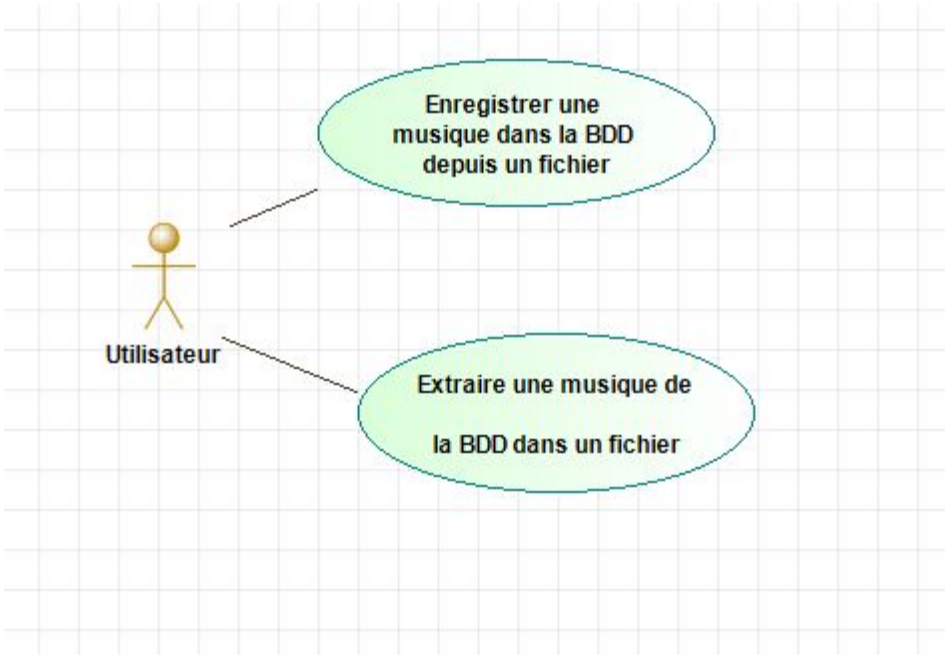
titre_musique1,durée_musique1,artiste_musique1,titre_musique2,durée_musique2,artiste_musique2 .

Ces données devront être stockées dans une base de données Oracle-Relationnel.

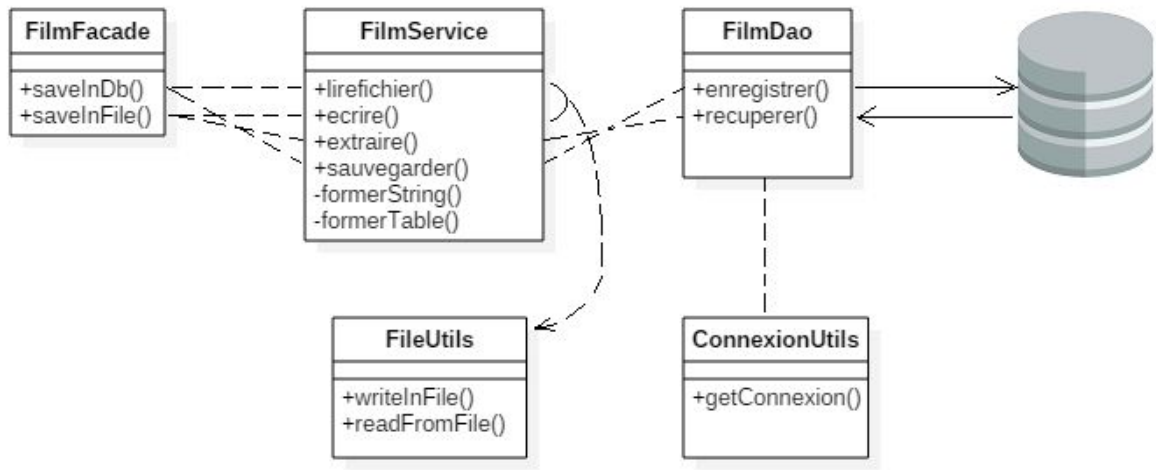
Toutefois, il est nécessaire pour faciliter le traitement et la manipulation de ces informations de pouvoir les extraires de la BDD dans un fichier texte sous le même format que celui décrit précédemment.

Dossier d’analyse

D’après le cahier des charges décrit précédemment, nous avons pu en déduire le diagramme de cas d’utilisation suivant :



En effet, un utilisateur peut enregistrer les informations stockées sur une musique depuis un fichier dans la base de données.
Un utilisateur doit également extraire les informations d’une musique vers un fichier pour en faciliter le traitement.



Dossier de programmation

Architecture

Pour atteindre le but voulu, nous avons séparé l'application en trois couches différentes : une couche Façade, une couche Service et une couche DAO, pour les accès à la base de données.

Package Utils

ConnexionUtils

Cette classe nous permet d'établir une connexion à la base de données renseignée, que l'utilisateur utilise l'application depuis l'université, ou l'extérieur (les URL sont différentes). L'usage d'un singleton permet de ne pas ouvrir une nouvelle connexion chaque fois que l'on en a besoin.

FileUtils

Cette classe nous permet permet de manipuler les fichiers csv. une méthode writeInFile() nous permet d'écrire une chaîne de caractères dans un fichier dont on renseigne le chemin, tandis qu'une autre méthode readFromFile() permet de lire un fichier pour retourner le contenu sous forme de chaîne de caractères.

Facade

La classe MusiqueFacadeImpl contient deux méthodes. La première, saveInDb() appelle les différents services permettant de sauvegarder le contenu d'un fichier dont on renseigne le chemin, dans la base de données. L'autre, saveInFile(), appelle les différents services permettant d'extraire de contenu de la table de la base de données et de l'enregistrer dans un fichier dont on renseigne le chemin.

Service

Ce package contient deux classes MusiqueService, déclarant seulement les méthodes publiques pour permettre leur appel, et MusiqueServiceImpl, qui implémente ces méthodes et en définit d'autres, privées, dont l'accès depuis l'extérieur sera impossible :

- (public) lirefichier() appelle la méthode readFromFile de FileUtils pour récupérer dans une chaîne le contenu d'un fichier
- (public) ecrire() appelle la méthode writeInFile de Fileutils pour écrire le contenu d'une chaîne dans un fichier.
- (private) formerString() permet, à partir d'un tableau de chaînes de caractères, de construire la chaîne à insérer dans le fichier, en séparant les différents champs récupérés par une virgule
- (private) formerTable() permet, à partir d'une chaîne de caractères, d'extraire les différents champs séparés par des virgules afin de les insérer ensuite dans la table de la base de données.
- (public) extraire() appelle une méthode de MusiqueDao pour récupérer les attributs d'un tuple dont l'id est connu, les insère dans un tableau de chaînes de caractères, et appelle formerString() pour préparer la chaîne à ensuite écrire dans un fichier.
- (public) sauvegarder() appelle formerTable() pour récupérer dans un tableau de chaînes de caractère les champs récupérés depuis un fichier, et appelle une méthode de MusiqueDao pour les insérer dans la table de la base de données.

DAO

La classe MusiqueDao contient deux méthodes, accédant à la base de données via la classe ConnexionUtils. L'une, enregistrer(), insère le contenu d'un tableau de chaînes de caractères dans la table de la bases de données. La seconde, recuperer() récupère dans un ResultSet les attributs d'un tuple dont l'id est renseigné.

Exceptions

Nous avons également créé 3 exceptions permettant d'assurer le bon fonctionnement de l'application :

- FileException est levée si l'on tente d'accéder à un fichier dont le chemin est null, ou si on essaie de le remplir avec un contenu null,
- NoRowSelected est levée si le résultat d'une requête ne retourne aucune colonne,
- WrongFormatException est levée si un fichier ou un tableau n'est pas formé correctement (mauvais nombre d'attributs..).

Tests unitaires

Les test unitaires sur les classes vues précédemment permettent de vérifier les choses suivantes :

- Vérification du format d'insertion des données
- Test des valeurs nulles
- Vérification de la bonne création des fichiers après extraction.
- Vérification de la bonne insertion des données extraites depuis un fichier.
- Vérification de la correspondance entre fichier et données insérées.

Ces tests unitaires permettent également de vérifier la bonne intégration des fonctions crées au fur et à mesure, ces dernières étant utilisées dans d'autres fonctions à l'intérieur de l'application, qui sont elles aussi testées.

Conclusion

Il est toujours intéressant d'avoir des informations et retours sur ce qui est fait dans le cadre de projets d'entreprise, de pouvoir le comparer à notre expérience, et de pouvoir appréhender et mettre en place ces concepts.
Ce mini-projet nous a donc permis de combler quelques lacunes en Programmation Orientée Objet, de découvrir les concepts de tests unitaires et d'intégration ainsi que d'appliquer les concepts liés à l'intégration continue.
L'intégration continue nous a été très utile de part la sauvegarde sur un serveur, et de part la division des tâches facilitée par la création de branches sur le système de versionning.

Nous avons rencontré peu de difficultés si ce n'est le fait de bien réaliser des tests exhaustifs et complets.