15/01/2021

Génie logiciel - Projet 2020

Rapport

DEVREESE Martin

MANEUX Tangi

BORDEAUX INP

ENSC 2A – Groupe 1

SOMMAIRE

INTRODUCTION

Ce rapport a été produit dans le cadre du projet Génie Logiciel de 2ème année d'étude à l'ENSC dont le sujet se trouve en annexe. Pour résumer, il était question de réaliser une application de gestion de collection de BD grâce à la technologie WinForms et à l'aide d'une base de données relationnelle MySQL. Le présent rapport détaille tout le travail réalisé par notre binôme ainsi que les choix de conception qui ont été faits.

TRAVAIL EFFECTUÉ

Procédure d'installation détaillée

1. Installer un logiciel permettant la mise en place d'un serveur web local, tel que Xampp.
2. Installer Visual Studio 2019
3. Copier le lien de notre GitHub:

   https://github.com/ensc-glog/projet-2020-projetgl_2020_martin_tangi.git

4. Cloner le dépôt Git de notre projet sur votre machine.
5. Se rendre à l'emplacement du code dans votre chemin local et lancer la solution ProjetGL.sln.
6. Lancer votre logiciel de serveur web local (type XAMPP) et activer le module MySQL.
7. Dans la solution Visual Studio, ouvrir le dossier DB du projet nommé "DAL", puis importer sur votre serveur web local les fichiers SQL dans l'ordre suivant :
   a. Database.sql
   b. Structure.sql
   c. Content.sql

Attention : Chaque fois que vous activez un ou plusieurs **tests** de la solution, la base de données est vidée puis se voit insérée de nouvelles données de test. Si des tests ont été effectués, il vous faut donc réinsérer les données en suivant les étapes présentées ci-dessus.

8. Démarrer l'application.
9. Vous êtes dans l'application BDThèque !
10. Les identifiants à utiliser sont stockés dans le tableau ci-dessous, veuillez noter que pour lancer l'application sans avoir à s'identifier et gagner du

temps, il suffit de commenter et dé-commenter quelques lignes (comme indiqué) dans ProjetGL -> App -> Program.cs.

Adresse Mail	Mot de Passe
Pour être utilisateur (parcours de la base de données, pas d'insertion de nouvel album) :	
toto@lambada.com	mdp
Pour être administrateur (parcours de la base de données + insertion de nouveaux albums) :	
mdevreese@ensc.fr	mdp
tmaneux@ensc.fr	mdp

Liste des exigences métier respectées par l'application

Le tableau suivant contient toutes les exigences métier primaires (fond clair) et secondaires (fond foncé). Les exigences respectées sont sur fond vert, les non-respectées sur fond rouge.

Code	Description
EF_01	En tant qu'utilisateur, je peux me connecter à l'application grâce à mes identifiants (login/mot de passe).
EF_02	En tant qu'utilisateur, je peux consulter la liste de mes albums.
EF_03	En tant qu'utilisateur, je peux afficher des informations détaillées sur un album : image de couverture, nom, série, auteur(s), catégorie (BD/manga/comic/…), genre (fantasy/polar/jeunesse/…), éditeur.
EF_04	En tant qu'utilisateur, je peux effectuer une recherche dans la liste des albums du marché. Cette recherche peut être basée sur les critères suivants : nom (ou partie du nom), série, auteur, genre.
EF_05	En tant qu'utilisateur, je peux ajouter un ou plusieurs album(s) du marché à la liste de mes albums.

EF_06	En tant qu'utilisateur, je peux ajouter des albums du marché à ma liste de souhaits. Cette liste est mise à jour en cas d'achat d'un album.
EF_07	En tant qu'utilisateur, je peux consulter la liste de mes souhaits.
EF_08	En tant qu'utilisateur, je peux retirer un ou plusieurs album(s) de la liste de mes souhaits.
EF_09	En tant qu'utilisateur, je peux me déconnecter de l'application pour revenir à l'écran d'accueil permettant de s'y connecter.
EF_10	En tant qu'administrateur, je peux me connecter à l'application grâce à des identifiants spécifiques (login/mot de passe).
EF_11	En tant qu'administrateur, je peux ajouter un album à la liste des albums du marché.
BONUS EF_12	En tant qu'utilisateur, je peux ajouter et supprimer un album que je possède dans la liste de mes favoris.

Liste des exigences techniques respectées par l'application

Code	Description
ET_01	L'application est réalisée sous Windows à l'aide de la technologie WinForms.
ET_02	Les données persistantes sont stockées dans une base de données relationnelle MySQL/MariaDB.
ET_03	L'application est structurée soit selon une architecture en couches (App/DAL/Domain), soit selon une architecture MVP.
ET_04	Le lien entre la BD et les objets de l'application est fait à l'aide de l'outil NHibernate.
ET_05	L'application respecte autant que possible les grands principes de conception étudiés en cours : séparation des responsabilités, limitation de la duplication de code, KISS, YAGNI, etc.
ET_06	L'ensemble du code source respecte la convention camelCase.

ET_07	Les noms des classes, propriétés, méthodes, paramètres et variables sont choisis avec soin pour refléter leur rôle.
ET_08	L'application dispose de tests automatisés couvrant les parties "accès aux données" (DAL) et "classes métier" (DAL). L'ensemble de ces tests ne doit générer aucun échec.

Liste et description des procédures de test automatisé

Conformément à l'exigence technique ET_08, l'application dispose de tests automatisés couvrant les parties "accès aux données" (*DAL*) et "classes métier" (*Domain*), l'ensemble de ces tests ne générant aucun échec tels quels. L'ensemble du code des classes métier ne contenant que des constructeurs et des méthodes *ToString*, il n'a pas été jugé utile de créer des tests automatisés pour le projet *Domain*. En ce qui concerne le projet *DAL*, des tests ont été mis en place pour contrôler le bon fonctionnement des classes *UserRepository* et *AlbumRepository*. Tous les tests sont initialisés de la même manière en faisant appel aux méthodes de la classe *TestRepository* qui vident la base de données et la remplissent de données de test.

Pour la classe *UserRepository*, les méthodes testées sont les suivantes:

- *GetAll* qui doit renvoyer tous les *User* contenus dans la base de données. On vérifie dans son test que cette méthode renvoie bien les 3 *User* des données de test;
- *GetUser* qui doit renvoyer le *User* unique contenu dans la base de données dont l'adresse et le mot de passe sont passés en paramètre. On vérifie dans son test que cette méthode renvoie bien "Martin Devreese", le *User* des données de test demandé;
- *Save* qui doit enregistrer le *User* passé en paramètre dans la base de données. On vérifie dans ce test que le *User* passé en paramètre a bien été ajouté à la base de données en utilisant la fonction *GetUser* testée plus tôt;
-

Pour la classe *AlbumRepository,* les méthodes testées sont les suivantes:

- *GetByTitle* qui doit renvoyer la liste des *Album* contenus dans la base de données dont le titre contient la chaîne de caractères passée en paramètre. On vérifie dans son test que cette méthode renvoie bien le seul *Album* parmi les données de test dont le titre contient "Tome 1 - Planète Dakoï";

- *GetByAuthor* qui doit renvoyer la liste des *Album* contenus dans la base de données dont l'auteur porte le nom passé en paramètre. On vérifie dans son test que cette méthode renvoie bien le seul *Album* parmi les données de test dont l'auteur est "Crisse";
- *GetBySeries* qui doit renvoyer la liste des *Album* contenus dans la base de données appartenant à la *Series* dont le nom est passé en paramètre. On vérifie dans son test que cette méthode renvoie bien le seul *Album* parmi les données de test appartenant à la *Series* intitulée "Kookaburra";
- *GetByGenre* qui doit renvoyer la liste des *Album* contenus dans la base de données appartenant au *Genre* dont le nom est passé en paramètre. On vérifie dans son test que cette méthode renvoie bien le seul *Album* parmi les données de test appartenant au *Genre* "Science Fiction";
- *GetAll* qui doit renvoyer tous les *Album* contenus dans la base de données. On vérifie dans son test que cette méthode renvoie bien les 3 *Album* des données de test;
- Save qui doit enregistrer le *Album* passé en paramètre dans la base de données. On vérifie dans ce test que le *Album* passé en paramètre a bien été ajouté à la base de données en utilisant la fonction *GetAll* testée plus tôt;

MODÉLISATION DES DONNÉES ET ARCHITECTURE

Diagramme des classes métier

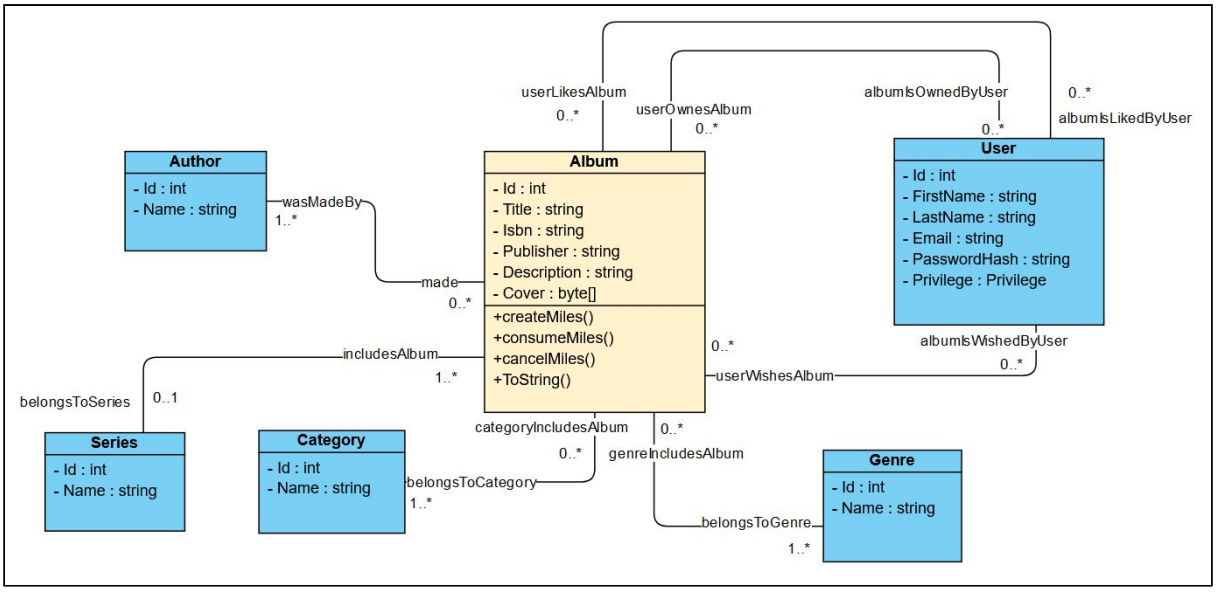
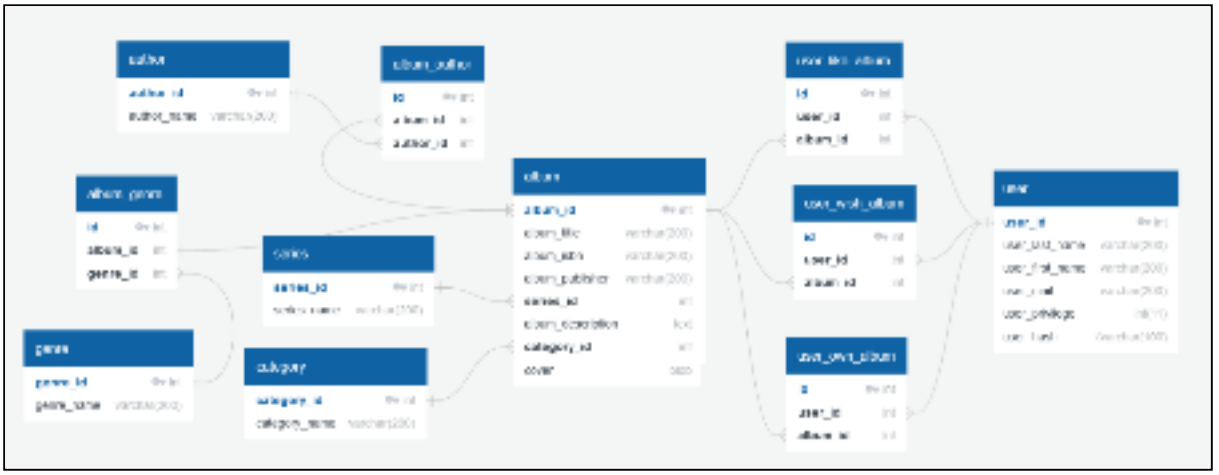


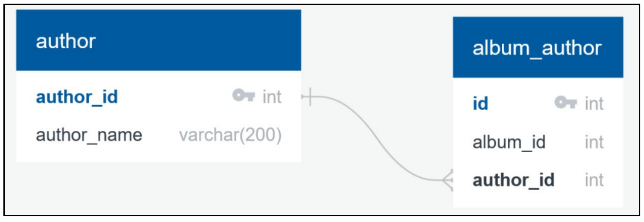
Diagramme des classes métier du projet *Domain* de la solution *ProjetGL*

Modèle relationnel



Modèle relationnel de notre base de données “albums”

Légende:



La relation ci-contre signifie qu’une ligne de la table *album_author* ne peut avoir la référence que d’une ligne de la table *author*, tandis qu’une ligne de la table *author* peut être référencée par plusieurs lignes de la table *album_author*. Les propriétés *album_id* et *author_id* de la table *album_author* sont des clés étrangères des tables *author* et *album*.

Description de l’architecture technique de l’application

L'architecture de l’application est conforme à l’exigence technique ET_03, nous avons fait le choix d’une architecture composée de trois couches: *App*, *DAL* et *Domain*. La couche *App* est la couche chargée de la partie visible de l’application avec les *WinForms*. La couche *DAL* est la Data Access Layer c’est-à-dire la couche chargée de l’accès à la base de données. La couche *Domain* est chargée de la définition de la structure des données, avec la définition des classes métier et des fichiers de mapping relationnel. Pour communiquer avec la base de données, la couche *DAL* utilise la version 5.3.5 de l’ORM *NHibernate*, et toute la solution utilise la version 4.7.2 du framework *.NET*.

Démarche de conception de l’IHM

Notre démarche de conception de l’interface Homme-Machine a d’abord été basée sur le fonctionnel et le fonctionnel uniquement. Lorsque le fonctionnel a été assuré nous nous sommes attelés à l’aspect visuel. Nous avons donc tout

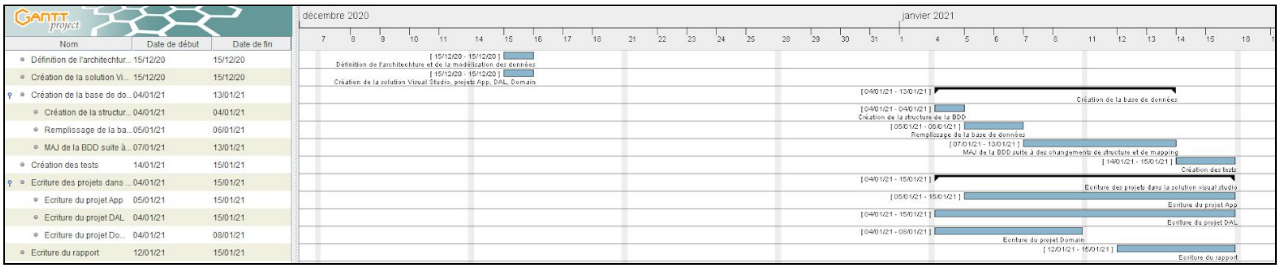
d'abord créé des *WinForms* très simples reliés entre eux puis nous avons testé l'affichage des données sous différents formats d'affichage. Nous avons ensuite convenu d'une interface utilisateur respectant les points suivants:

1. L'interface doit s'articuler autour de 3 *Forms* spécialisés dans des fonctions différentes pour suivre le principe de séparation des responsabilités:
    1.1. Le premier est le *LoginForm*, celui sur lequel on s'identifie en premier lieu. Si les informations de connexion sont exactes, il renvoie sur le *MainForm*.
    1.2. Le second est le *MainForm*, il est le *Form* principal depuis lequel on peut explorer et manipuler la collection de BD.
    1.3. Le troisième est l'*AlbumForm*, il est le *Form* permettant d'afficher des informations détaillées sur un album bien précis sélectionné depuis le *MainForm*.
2. Le LoginForm doit être simple, il ne doit contenir qu'une consigne d'identification, une *TextBox* pour l'identifiant, une *TextBox* pour le mot de passe, un *Button* pour lancer la connexion et un message d'erreur potentiel.
3. Le *MainForm* doit permettre d'accéder à la liste des albums du marché, la liste des albums possédés par l'utilisateur et la liste des souhaits de l'utilisateur. Nous avons donc décidé de diviser ce *Form* en 3 onglets, "Albums possédés", "Albums sur le marché", "Mes souhaits".
4. Le *MainForm* doit afficher le minimum des informations de l'utilisateur actuellement connecté, nous avons choisi d'afficher son nom et prénom en haut à droite comme sur de nombreuses autres applications pour que cet emplacement soit naturel pour l'utilisateur.
5. Chaque onglet doit permettre d'afficher la couverture des albums concernés puisque c'est l'information permettant la meilleure discrimination entre eux. Leur titre arrive en second puisque c'est la première information recherchée par l'utilisateur lorsque la couverture lui plaît. Enfin nous avons choisi d'afficher également le nom de l'auteur pour apporter une information supplémentaire, mais en plus petit car cette information est moins essentielle.
6. L'affichage des albums doit donc se faire par tableaux, les tableaux doivent être fluides, un utilisateur doit pouvoir "scroller" pour faire défiler les albums.
7. La recherche d'un album se fait grâce à un bandeau situé au-dessus des onglets du *MainForm.*
8. La création et ajout d'un nouvel album dans la base de données se fait au travers d'une page uniquement accessible à l'administrateur depuis le *MainForm*.

L'affichage des albums sous forme de grille a été réalisé à l'aide du *User Control*, un élément d'interface customizable, qui a pu ensuite être répété à la chaîne dans un *FlowLayoutPanel*.

ORGANISATION

Planning



Planning final du projet de Génie Logiciel

Pour décrire rapidement notre planning, nous avons fait une légère reconnaissance et initialisation du projet en décembre, puis nous avons repris le travail le 4 Janvier. Notre choix a été de faire une coupure complète sur ce projet pendant les vacances de Noël pour prioriser le travail de révision (et le repos - en majeure partie).

Répartition des tâches

Notre répartition des tâches a été faite suivant la méthode agile et de façon évolutive. Voici un court résumé des tâches accomplies.
Conjointement : Diagramme des classes métier, Modèle relationnel de la base de données, Mapping relationnel, Résolution des problèmes de mapping, Choix d'architecture;

Travail réalisé par Martin : LoginForm et connexion, population de la base de données, affichage d'un album (*AlbumQuickView*), affichage dynamique des pages d'albums, refonte du MainForm, options de recherches d'album, gestion des favoris et des souhaits, ajout d'un album par un compte administrateur;

Travail réalisé par Tangi : Écriture des classes métier, première version du MainForm, création des Repositories, création de l'AlbumForm, écriture des tests, rédaction du rapport, retour à la page d'authentification.

BILAN ET PERSPECTIVES

Le produit de notre travail est une application de gestion de collection de BD répondant aux exigences techniques et métier présentées plus tôt dans le rapport. Nous sommes globalement satisfaits de notre travail puisque nous avons su, tout au long de ce projet, résoudre tous les problèmes qui se sont posés et nous familiariser avec les outils WinForm, SQL, les tests unitaires, l'ORM NHibernate et le mapping relationnel de façon plus générale. Nous pensons avoir grandement gagné en maîtrise de ces outils, comme en atteste l'application que nous avons produite. Nous sommes également satisfaits de notre gestion du projet puisque nous avions anticipé dès le départ l'architecture de l'application et du mapping, comme en attestent nos schémas, et nous n'avons subi que peu de changement en cours de route. Pour finir, nous sommes satisfaits du visuel de notre application puisqu'il est conforme aux limites de WinForms tout en étant pas la préoccupation première.
Le projet ne reste pas sans perspectives puisque nous avons prévu dans notre structure de BDD et mapping un lien entre un album et un utilisateur, appelé "favoris". Il ne sert pour l'instant qu'à marquer d'une étoile un album mis dans sa liste de favoris mais il pourrait être intéressant de développer un partage des favoris ou de liste à thèmes à un autre utilisateur via une liste d'amis par exemple. La décomposition du code en éléments à responsabilité unique nous permettrait d'ajouter facilement une liste d'amis aux liens qui unissent des utilisateurs.

Nous souhaitons un bon voyage à nos utilisateurs dans le monde de la BD !

Annexes

Annexe 1 : Sujet du projet de génie logiciel 2020

Annexe 2 : Lien GitHub du projet