Bordeaux INP
ENSC

2ème année

U.E. Sciences fondamentales

Module Génie Logiciel

Projet de Génie Logiciel

15/01/21

PASCOTTINI Canelle
VALLÉE Amandine

Année 2020 / 2021

Table des matières

I. Introduction

L'objectif de ce projet est la réalisation d'une application de gestion de collection de BD, une BDthèque. Celle-ci doit permettre à un utilisateur de consulter la liste des BD du marché et de tenir à jour une liste d'albums possédés ainsi qu'une wishlist. Elle doit aussi permettre à un administrateur de mettre les BD du marché à jour.

L'application doit utiliser la technologie WinForms et être produite en C#.

II. Exigences métier

Les exigences fonctionnelles respectées par l'application sont regroupées dans le **Tableau 1** et classées par importance métier. Les exigences métier du sujet sont les EF_01 à EF_11. En bleu sont les exigences métiers que nous avons ajoutées.

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, ou de mes possessions.
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.
EF_12	En tant que nouveau visiteur, je peux m'inscrire sous le statut d'utilisateur.

Tableau 1 : Tableau des exigences métier de l'application

III. Conception de l'IHM

Pour concevoir les formulaires WinForms nous nous sommes inspirées de ce qui se fait déjà. Nous avons fait en sorte que l'IHM soit intuitive avec le moins de choses superflues possible, et esthétiquement plaisante pour que l'utilisateur passe un moment agréable à gérer sa collection de BD. Nous avons respecté les symboliques des couleurs avec des boutons verts pour la validation et des boutons rouges pour l'annulation par exemple.

Les formulaires permettant de se connecter ou de s'inscrire à la BDThèque sont très basiques et ressemblent beaucoup à ce qui existe déjà, comme le montre la **Figure 1**.

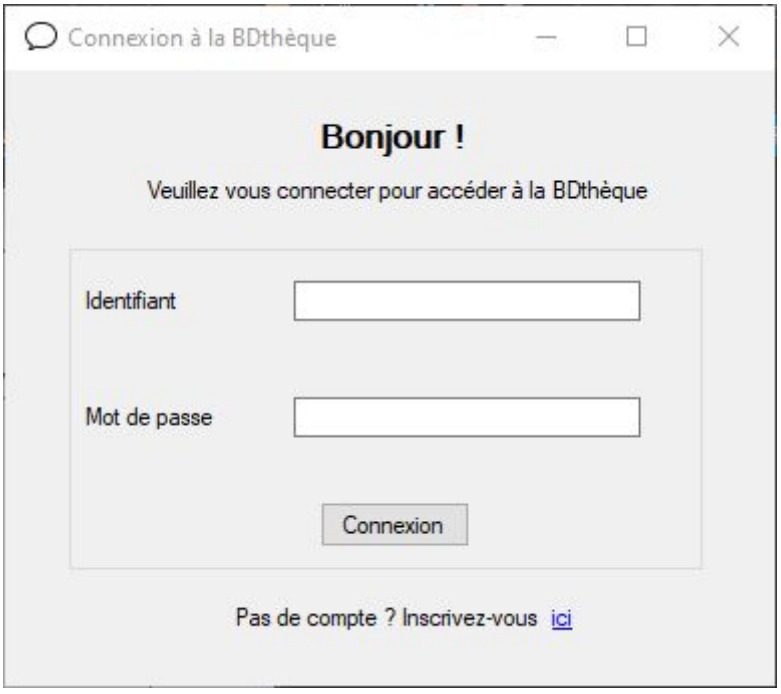
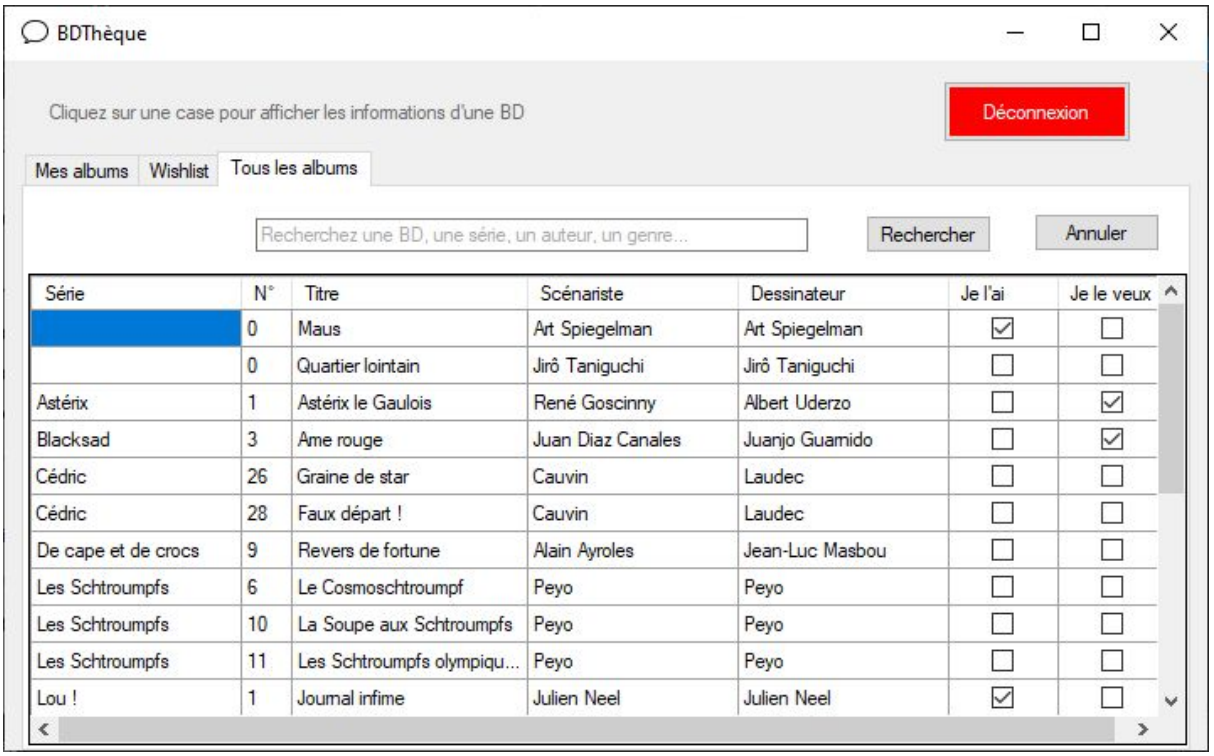


Figure 1 : Formulaire de connexion à la BDThèque

L'affichage des BD se fait dans trois onglets différents du formulaire principal (**Figure 2**) : l'un dédié aux BD possédées par l'utilisateur, l'autre pour celles de sa wishlist, et le dernier pour l'ensemble des BD du marché. Dans ce dernier onglet on peut néanmoins retrouver les BD que l'on possède ou que l'on veut grâce aux deux dernières colonnes. Cela permet à l'utilisateur de ne pas avoir à faire des aller-retours entre les différents onglets pour savoir s'il possède déjà telle ou telle BD avant de l'ajouter à sa collection depuis l'onglet du marché. Seules certaines informations clés sont utilisées ici pour décrire une BD

pour ne pas surcharger l'utilisateur d'informations. De plus, les BD sont affichées dans l'ordre alphabétique sur la série d'une BD, puis sur le numéro de la BD dans la série. Pour les BD n'appartenant pas à une série, il se fait dans l'ordre alphabétique sur le titre. Cet ordre permet à l'utilisateur de facilement retrouver une BD sans utiliser la fonction de recherche.



Série	N°	Titre	Scénariste	Dessinateur	Je l'ai	Je le veux
	0	Maus	Art Spiegelman	Art Spiegelman	☑	☐
	0	Quartier lointain	Jirô Taniguchi	Jirô Taniguchi	☐	☐
Astérix	1	Astérix le Gaulois	René Goscinny	Albert Uderzo	☐	☑
Blacksad	3	Ame rouge	Juan Diaz Canales	Juanjo Guarnido	☐	☑
Cédric	26	Graine de star	Cauvin	Laudec	☐	☐
Cédric	28	Faux départ !	Cauvin	Laudec	☐	☐
De cape et de crocs	9	Revers de fortune	Alain Ayroles	Jean-Luc Masbou	☐	☐
Les Schtroumpfs	6	Le Cosmoschtroumpf	Peyo	Peyo	☐	☐
Les Schtroumpfs	10	La Soupe aux Schtroumpfs	Peyo	Peyo	☐	☐
Les Schtroumpfs	11	Les Schtroumpfs olympiqu...	Peyo	Peyo	☐	☐
Lou !	1	Journal infime	Julien Neel	Julien Neel	☑	☐

Figure 2 : Formulaire principal

L'affichage des informations d'une BD en particulier se fait sur un formulaire à part, comme sur la **Figure 3**. Les informations sont présentées dans un ordre d'importance : d'abord le titre, la série, puis à la fin le genre, la catégorie.

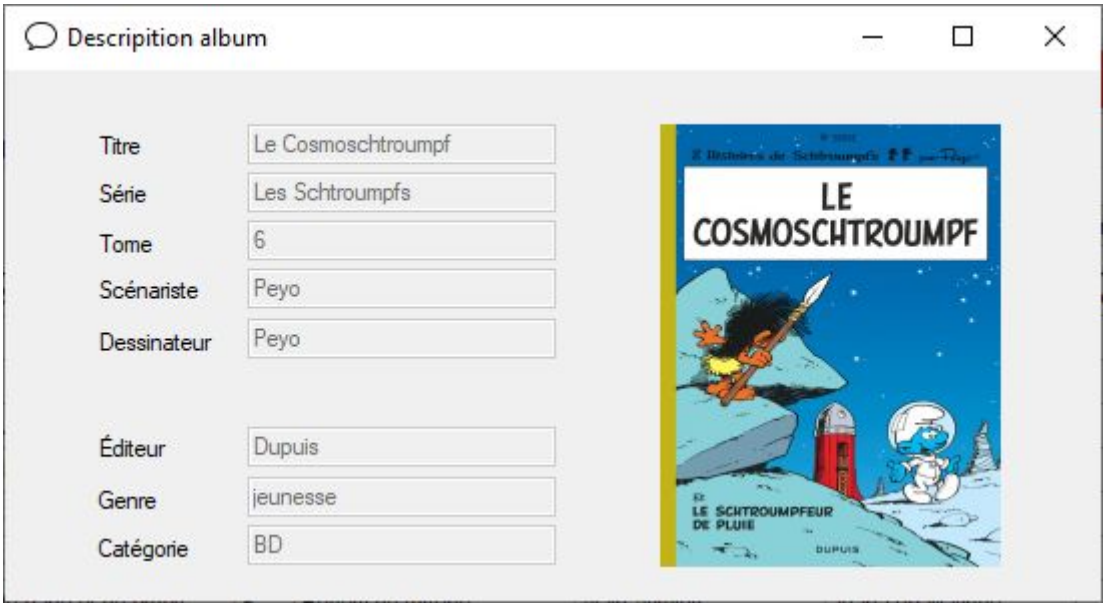


Figure 3 : Formulaire des informations détaillées d'une BD

Le formulaire d'ajout d'une BD pour un administrateur est exactement le même que le précédent à l'exception que toutes les informations sont à remplir.

IV. Modélisation des données

1. Contraintes et hypothèses

Nous avons d'abord listé plusieurs contraintes, certaines d'après le sujet et d'autres que nous nous sommes imposées.

- Il existe deux profils de visiteurs : les utilisateurs et les administrateurs.
- Les utilisateurs n'ont accès à la DB qu'en lecture.
- Les administrateurs ont accès à la DB en lecture et en écriture.
- Une personne peut posséder une ou plusieurs BD.
- Une personne peut souhaiter une ou plusieurs BD.

Nous avons également posé plusieurs hypothèses simplificatrices.

- Une BD ne peut avoir qu'un auteur.
- Une BD ne peut avoir qu'un illustrateur.
- On ne peut s'inscrire qu'en tant qu'utilisateur.

Nous sommes parties du principe que si une personne a besoin de savoir quelles BD elle possède ou souhaite, les BD n'ont pas besoin de connaître qui les possède ou les souhaite. Nous avons donc choisi une association unidirectionnelle many-to-many.

2. Modèle Conceptuel de Données (MCD)*

Suite à ces hypothèses, notre MCD est celui de la **Figure 4**.

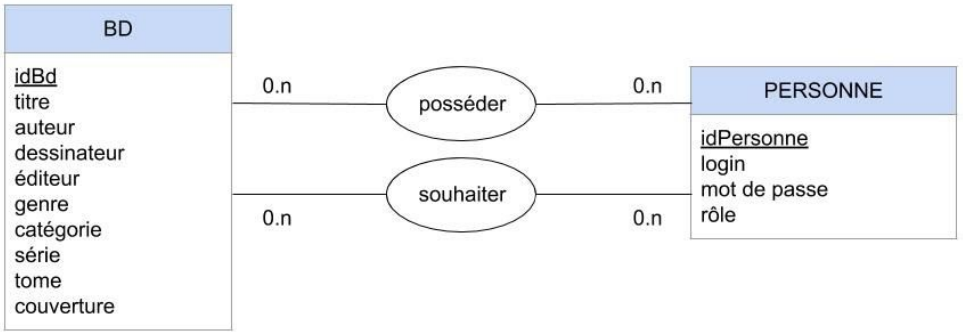


Figure 4 : Modèle Conceptuel de Données

3. Modèle Logique de Données Relationnel (MLDR)

Nous avons commencé par opter pour le MLDR suivant.

PERSONNE (pers_id, pers_login, pers_mdp, pers_role)
BD (bd_id, bd_titre, bd_auteur, bd_dessinateur, bd_editeur, bd_serie, bd_numSerie, bd_categorie, bd_genre, bd_couverture)
RELATION (#pers_id,#bd_id, rel_statut)

Cependant, celui-ci nous a posé des problèmes de mapping. En effet, puisque l'association unidirectionnelle many-to-many se mappe directement dans les classes associées, la table d'association Relation n'était pas traduite par une classe, or nous avions besoin de faire des requêtes HQL en utilisant cette table. Nous nous sommes donc tournées vers le MLDR suivant.

PERSONNE (pers_id, pers_login, pers_mdp, pers_role)
BD (bd_id, bd_titre, bd_auteur, bd_dessinateur, bd_editeur, bd_serie, bd_numSerie, bd_genre, bd_categorie, bd_couverture)
RELATION (rel_id, #pers_id,#bd_id, rel_statut)

où pers_role ∈ {'utilisateur','administrateur'} et rel_statut ∈ {'possede','veut'}

Ce dernier MLDR correspond à la **Figure 5**.

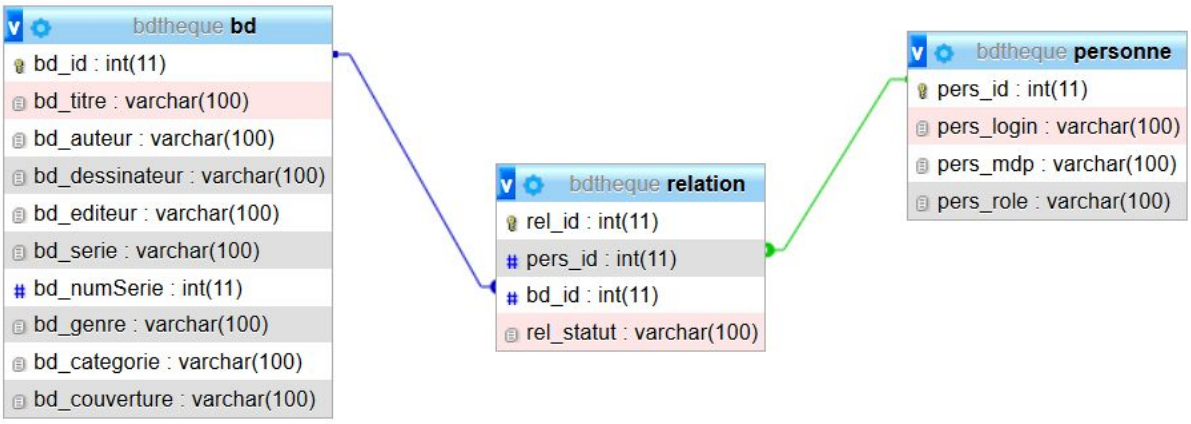


Figure 5 : Modèle Logique de Données Relationnel

4. Diagramme de classes

Afin de traduire ce MLD, nous avons réalisé trois classes métier, présentées sur le diagramme **Figure 6**.

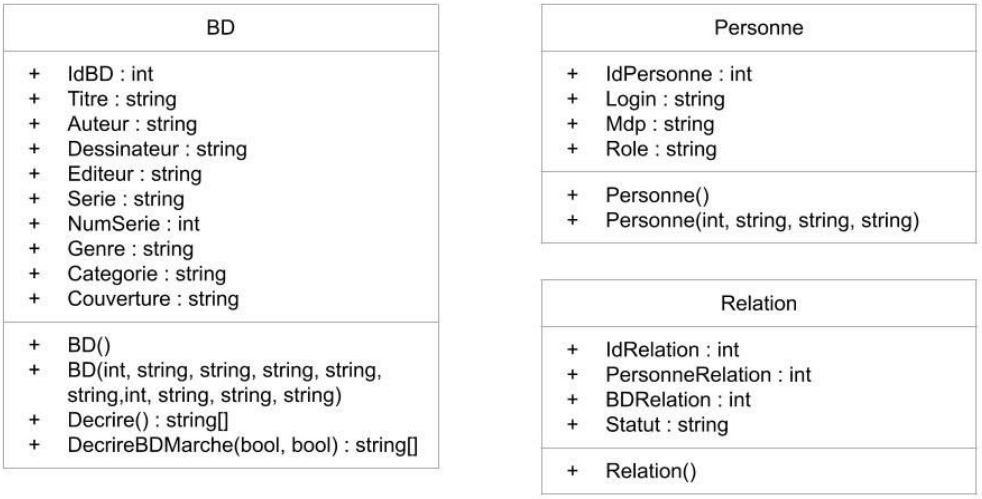


Figure 6 : Diagramme de classes réel

Ce diagramme ne présente pas de relations entre classes parce que nous avons été confrontées à un problème. En réalité, le diagramme de classes que nous aurions aimé mettre en place correspond à la **Figure 7**. Mais nous avons eu du mal à utiliser les bags dans le mapping et à faire des requêtes HQL sans classe Relation.

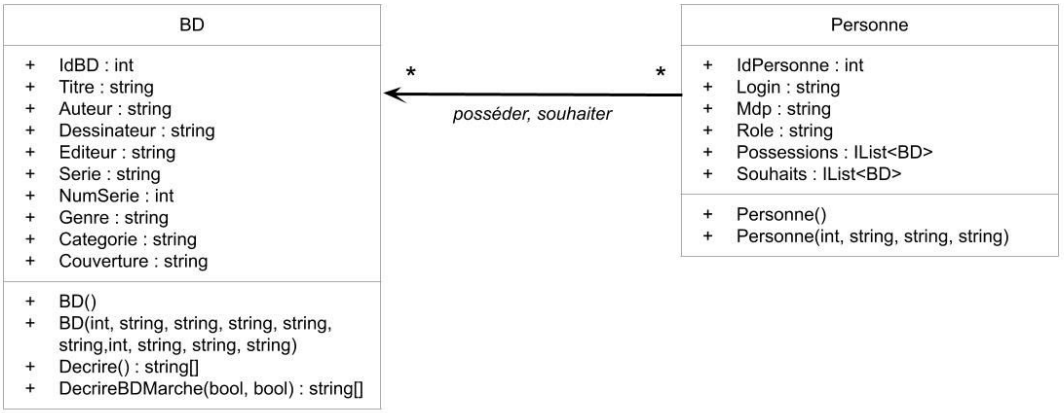


Figure 7 : Diagramme de classes théorique

V. Architecture technique

L'application utilise le Framework .NET 4.7.2. Cela peut poser problème si elle est ouverte depuis une machine qui ne supporte pas ce Framework. Elle est structurée selon une architecture en couches App/DAL/Domain.

La couche **App** est constituée de 6 formulaires :

- AjoutAlbumForm : form réservé à l'administrateur, s'il souhaite ajouter un album au marché
- AlbumForm : form qui affiche des informations détaillées sur un album

- DeconnexionForm : form permettant de choisir si l'on veut quitter l'application après s'être déconnecté ou si l'on veut s'y reconnecter
- InscriptionForm
- LoginForm
- MainForm : form qui affiche les possessions, la wishlist et le marché

La couche **DAL**, qui est la couche d'accès aux données, contient les trois repositories BDRepository, PersonneRepository et RelationRepository. Le lien avec les données persistantes stockées dans la base de données MySQL/MariaDB est effectué à l'aide de NHibernate. Les trois scripts SQL de création et remplissage de la base de données sont disponibles dans le dossier DB de DAL.
En outre, les images de couverture des BD sont stockées automatiquement dans le dossier App/Bin/Debug/couvertures pour que l'application puisse les retrouver à partir de leur nom stocké dans la base de données.

La couche **Domain** contient quant à elle les trois classes métiers décrites en **IV.4**, ainsi que le mapping de ces classes, dans le dossier Mapping.

On trouve dans les couches DAL et Domain les classes de tests associés, qui sont décrites dans la partie suivante.

VI. Tests

L'export des données de test est géré automatiquement par le RepositoryTest.

Les tests automatisés couvrent les projets DAL et Domain à travers les classes DALTests et DomainTests. Néanmoins, nous n'avons pas jugé pertinent d'étendre les tests des classes métier aux classes Personne et Relation car elles ne contiennent aucun traitement ni calcul donc seule la classe BD est couverte. DALTests contient donc 17 tests et DomainTests 2.

Les méthodes testées et de test sont décrites dans le **Tableau 2**.

Méthode	Méthode de test correspondante
Personne	
GetAll GetIdUtilisateur GetRoleUtilisateur SaveUtilisateur	TestPersRepo_GetAll TestPersRepo_GetIdUtilisateur TestPersRepo_GetRoleUtilisateur TestPersRepo_SaveUtilisateur
BD	
DAL	

GetAll	TestBDRepo_GetAll
GetBDUtilisateur	TestBDRepo_GetBDUtilisateur
GetBDRecherche	TestBDRepo_GetBDRecherche
GetBDRow	TestBDRepo_GetBDRow
GetAllCouvertures	TestBDRepo_GetAllCouvertures
SaveBD (9 arguments)	TestBDRepo_SaveBD
SaveBD (7 arguments)	TestBDRepo_SaveBD_2
Domain	
DecrireBD	DecrireTest
DecrireBDMarche	DecrireBDMarcheTest
Relation	
GetAll	TestRelationRepo_GetAll
SaveRelation	TestRelationRepo_SaveRelation
UpdateRelation	TestRelationRepo_UpdateRelation
DeleteRelation	TestRelationRepo_DeleteRelation
GetIdBD	TestRelationRepo_GetIdBD
GetIdRelation	TestRelationRepo_GetIdRelation

Tableau 2 : Méthodes testées et de test

L'ensemble de ces tests ne génère aucun échec.

La partie App du projet ne pouvant être testée par des tests automatisés, nous avons testé manuellement ses fonctionnalités en essayant de couvrir l'ensemble des cas d'utilisation possibles.

VII. Procédure d'installation

Afin d'installer l'application :

1. Cloner le dépôt GitHub.
2. Sur phpMyAdmin, exécuter le script SQL de création et remplissage de la base de données, en copiant-collant intégralement l'**Annexe 1** (ou les scripts SQL se trouvant dans le dossier DB dans DAL dans l'ordre suivant : Database, Structure, Content).
3. Lancer ProjetGL.sln.
4. Exécuter l'application.

La base de données contient 22 BD et 3 utilisateurs et la base de données de tests 6 BD et 3 utilisateurs. Attention cependant, si les tests sont exécutés, la base de données de tests, plus légère, écrasera la base de données complète. Il vaut donc mieux les exécuter avant l'étape 2 ou à la toute fin.

VIII. Gestion de projet

1. Planning

Nous nous sommes organisées de manière à avoir le plus rapidement possible une application fonctionnelle et qui répondait aux exigences les plus importantes. C'est pourquoi nous avons traité les exigences métiers dans leur ordre d'importance. Cela nous a permis de répondre à toutes les exigences du sujet du projet, et d'avoir quelques fonctionnalités supplémentaires à la fin. Voici globalement comment se sont déroulées les différentes étapes du projet :

	09/12/20	14/12/20	21/12/20	28/12/20	4/01/21	11/01/21
Définition de la structure de l'application, des classes métiers et des tables						
Rédaction des scripts SQL de création de la base de données						
Remplissage de la base de données						
Code des classes métiers						
Mapping						
Création des formulaires						
Code des repositories						
Code behind des formulaires						
Tests						
Rapport						
Finitions						

Tableau 3 : Planning

2. Organisation interne

Globalement nous avons toutes les deux traité de tous les aspects du projet. La répartition des tâches ci-dessous indique laquelle de nous deux a le plus travaillé sur chaque partie, mais en réalité nous avons chacune participé sur l'ensemble du projet. Suite à chaque tâche, la deuxième élève s'est chargée de vérifier le travail produit et d'y apporter des modifications si nécessaire.

Tâche	Elève responsable
Interface	

LoginForm	Amandine
InscriptionForm	Amandine
MainForm	Amandine
AlbumForm	Canelle
AjoutAlbumForm	Canelle
DeconnexionForm	Amandine
Accès aux données	
Mapping	Canelle
Remplissage de la DB	Canelle et Amandine
Repositories	Canelle
Tests	Canelle et Amandine
Rapport	Canelle et Amandine

Tableau 4 : Répartition des tâches

IX. Bilan et perspectives

Notre application répond à toutes les exigences métiers et techniques. Cependant nous avons identifié certaines faiblesses ou pistes d'amélioration.

Il aurait été plus pratique que la base de données se crée automatiquement à l'exécution du projet. Nous pouvions le faire, mais elle se créait à chaque exécution, perdant ainsi les BD que l'utilisateur avait ajoutées à ses possessions ou sa wishlist lors de visites précédentes. Nous avons donc préféré rester sur une installation manuelle de l'utilisateur en exécutant les scripts SQL sur phpMyAdmin avant la première exécution pour que l'application reste pertinente, mais l'idée aurait pu être creusée.

Nous ne nous sommes rendu compte que tardivement que notre architecture avec la classe Relation et son mapping n'étaient peut-être pas ce qui était attendu, mais nous n'avons pas eu le temps de le corriger. En effet, malgré cela, l'application fonctionne correctement. Nous ne pensons donc pas que ce soit un problème majeur.

Annexes

Annexe 1 : Script SQL de création et remplissage de la base de données

```
create database if not exists bdtheque character set utf8 collate utf8_unicode_ci;
use bdtheque;
grant all privileges on bdtheque.* to 'bdtheque_user'@'localhost' identified by 'asterix';

drop table if exists personne;
drop table if exists bd;
drop table if exists relation;

create table personne (
	pers_id integer not null primary key auto_increment,
	pers_login varchar(100) not null,
	pers_mdp varchar(100) not null,
	pers_role varchar(100) not null
);

create table bd (
	bd_id integer not null primary key auto_increment,
	bd_titre varchar(100) not null,
	bd_auteur varchar(100) not null,
	bd_dessinateur varchar(100) not null,
	bd_editeur varchar(100) not null,
	bd_serie varchar(100),
	bd_numSerie integer,
	bd_genre varchar(100) not null,
	bd_categorie varchar(100) not null,
	bd_couverture varchar(100) not null
);

create table relation (
	rel_id integer not null primary key auto_increment,
	pers_id integer,
	bd_id integer,
	foreign key (pers_id) references personne(pers_id),
	foreign key (bd_id) references bd(bd_id),
	rel_statut varchar(100) not null
);
```

```
insert into bd values (1,'Les bijoux de la Castafiore','Hergé','Hergé','Casterman','Tintin', 21, 'aventure','BD','lesBijouxDeLaCastafiore.jpg');
insert into bd values (2,'Ame rouge','Juan Diaz Canales','Juanjo Guarnido','Dargaud','Blacksad', 3,'policier','BD', 'ameRouge.jpg');
insert into bd values (3,'Astérix le Gaulois','René Goscinny','Albert Uderzo','Hachette','Astérix', 1,'aventure','BD', 'asterixLeGaulois.jpg');
insert into bd values (4,'Persepolis - Tome 3','Marjane Satrapi','Marjane Satrapi',"L'association",'Persepolis', 3, 'autobiographie','BD', 'persepolis3.jpg');
insert into bd values (5,'Journal infime' , 'Julien Neel' , 'Julien Neel', 'Glénat', 'Lou !' , 1,  'jeunesse' , 'BD', 'journalInfime.jpeg');
insert into bd values (6,'Mortebouse' , 'Julien Neel' , 'Julien Neel', 'Glénat', 'Lou !' , 2, 'jeunesse' , 'BD', 'mortebouse.jpg');
insert into bd values (7,'Le cimetière des autobus' , 'Julien Neel' , 'Julien Neel', 'Glénat', 'Lou !', 3, 'jeunesse' , 'BD', 'leCimetiereDesAutobus.jpg');
insert into bd values (8,'Les Schtroumpfs olympiques' , 'Peyo' , 'Peyo', 'Dupuis', 'Les Schtroumpfs' , 11, 'jeunesse', 'BD', 'lesSchtroumpfsOlympiques.jpg');
insert into bd values (9,'La Soupe aux Schtroumpfs' ,'Peyo' , 'Peyo', 'Dupuis', 'Les Schtroumpfs' , 10, 'jeunesse' , 'BD', 'laSoupeAuxSchtroumpfs.jpg');
insert into bd values (10,'Le Cosmoschtroumpf' ,'Peyo' , 'Peyo', 'Dupuis', 'Les Schtroumpfs' , 6, 'jeunesse' , 'BD', 'leCosmoschtroumpf.jpg');
insert into bd values (11,'Faux départ !' ,'Cauvin' , 'Laudec', 'Dupuis', 'Cédric' , 28, 'humour' , 'BD', 'fauxDepart.jpg');
insert into bd values (12,'Graine de star' ,'Cauvin' , 'Laudec', 'Dupuis', 'Cédric' , 26, 'humour' , 'BD', 'graineDeStar.jpg');
insert into bd values (13,'La nuit de Saint-Germain-des-Prés','Léo Malet','Emmanuel Moynot','Casterman','Nestor Burma', 5, 'policier', 'BD','laNuitDeSaintGermainDesPres.jpg');
insert into bd values (14,'Revers de fortune','Alain Ayroles','Jean-Luc Masbou','Delcourt','De cape et de crocs', 9,'aventure', 'BD','reversDeFortune.jpg');
insert into bd values (15,'Maus','Art Spiegelman','Art Spiegelman','Flammarion',null, null, 'historique', 'BD','maus.jpg');
insert into bd values (16,"La pourpre et l'or",'Jean Dufaux','Philippe Delaby','Dargaud','Murena', 1, 'historique', 'BD','laPourpreEtLOr.jpg');
insert into bd values (17,'Quartier lointain','Jirô Taniguchi','Jirô Taniguchi','Casterman', null, null, 'vie', 'BD','quartierLointain.jpg');
insert into bd values (18, 'Naruto - Tome 9', 'Masashi Kishimoto','Masashi Kishimoto','Kana Eds','Naruto', 9, 'shonen', 'Manga', 'naruto9.jpg');
insert into bd values (19, 'Naruto - Tome 6', 'Masashi Kishimoto','Masashi Kishimoto','Kana Eds','Naruto', 6, 'shonen', 'Manga', 'naruto6.jpg');
insert into bd values (20, 'À la rencontre de maître Chavipère', 'Eiichiro Oda', 'Eiichiro Oda', 'Glénat', 'One Piece', 81, 'shonen', 'Manga', 'onePiece81.jpeg');
insert into bd values (21, 'Déclaration de guerre', 'Eiichiro Oda', 'Eiichiro Oda', 'Glénat', 'One Piece', 41, 'shonen', 'Manga', 'onePiece41.jpg');
insert into bd values (22, 'La saga du Phénix Noir', 'Chris Claremont', 'John Byrne', 'Panini Comics', 'X-Men', null, 'science fiction', 'Comic', 'laSagaDuPhenixNoir.jpg');
```

```
insert into personne values (1,'fanDeBd','bonjour','utilisateur');
insert into personne values (2,'admin1','gestion','administrateur');
insert into personne values (3,'roger','vivelabd','utilisateur');

insert into relation values(1,1,1,'possede');
insert into relation values(2,1,2,'veut');
insert into relation values(3,3,1,'possede');
insert into relation values(4,3,4,'possede');
insert into relation values(5,3,3,'veut');
insert into relation values(6,1,10,'possede');
insert into relation values(7,1,14,'possede');
insert into relation values(8,1,18,'veut');
insert into relation values(9,1,15,'veut');
insert into relation values(10,3,8,'possede');
insert into relation values(11,3,22,'veut');
insert into relation values(12,3,5,'veut');
insert into relation values(13,1,21,'possede');
insert into relation values(14,1,13,'possede');
```