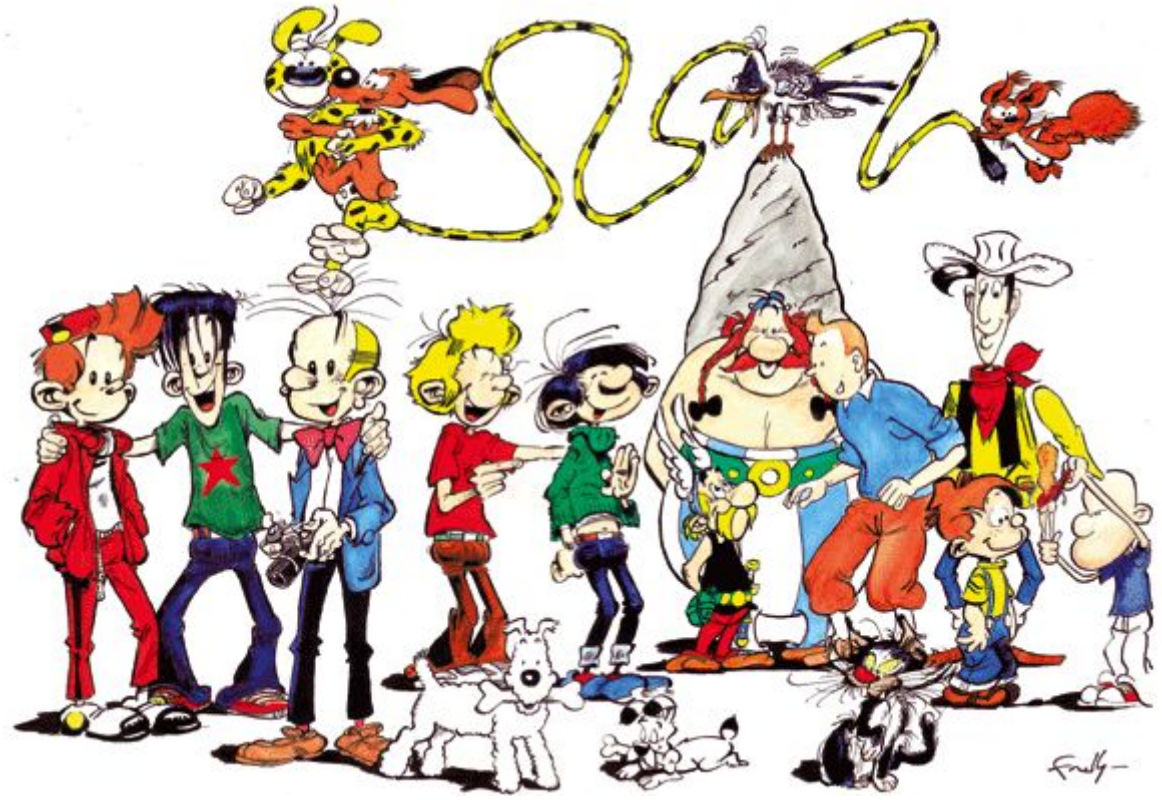
RAPPORT BD'THEQUE



BANNAY Lucie CARACATSANIS Oriane

TABLE DES MATIÈRES

I. Introduction **3**

II. Spécification **3**
II.1 Analyse fonctionnelle 3
II.2 Spécification des classes 4
II.3. Architecture de la solution 6
II.4. Choix de conception 7
II.5. Maquettage 8
II.6.Gestion du projet 9

III. Résultats et tests **11**
III.1. Exigence métiers respectées 11
III.2. Interfaces 12
Page de connexion 12
BDThèque 12
InfosAlbum 12
III.3. Protocoles de tests 12
III.4. Résultats 12

IV. Procédure d'installation **13**
IV.1. Dézipper le fichier et l'ouvrir sur Visual Studio 13
IV.2. Créer la base de données 13
IV.3. Lancer l'application 16

V. Bilan et perspectives **16**
III. Résultats et tests 16

Annexes **18**
Zoning 18
Maquettes 20

I. Introduction

Ce projet consiste en la réalisation d'un outil de gestion de BD. L'application est faite sous WinForm et utilise de la programmation orientée objet en C#. De plus, elle est reliée à une base de données qui permet de sauvegarder les informations entre plusieurs utilisations de l'application.
Un des objectifs de ce projet est également de suivre une démarche de génie logiciel, avec une analyse des exigences poussées menant à des schémas UML ainsi que le respect de principes de conception.

II. Spécification

II.1 Analyse fonctionnelle

Plusieurs exigences métier nous étaient fournies avec des ordres de priorité différents. Le produit minimum viable correspond à celles en orange ci-dessous. Nous les avons analysées pour les traduire en fonctionnalités, voici celles que nous avons retenu :

En tant qu'utilisateur :
- S'identifier/Se déconnecter
- Consulter sa liste d'albums
- Consulter sa liste de souhaits
- Afficher les informations d'un album
- Rechercher un album à partir d'un paramètre
- Ajouter un album à sa liste d'albums
- Ajouter/retirer un album à sa liste de souhaits

En tant qu'administrateur :
- S'identifier/Se déconnecter
- Ajouter un album à la liste des albums

En plus de cela, une chose importante qui n'apparaît pas dans ces fonctionnalités est la connexion à une base de données. Nous avons respecté la majorité d'entre elles puisque seulement deux ne le sont pas. L'ensemble des fonctionnalités du produit minimum viable ont été respectées. Nous avons en plus réalisé la majorité des fonctionnalités de la partie utilisateur puisque seule une fonctionnalité n'est pas entièrement respectée.

En plus des fonctionnalités déduites des exigences métiers, nous en avons ajouté qui nous semblent ergonomiques. Les voici :

Supprimer un album de sa liste d'albums
Modifier ses informations personnelles
Modifier ses informations de connexion

II.2 Spécification des classes

Ce qui ressort dans la liste des fonctionnalités est qu'il y a six types d'éléments : des utilisateurs, des albums, des catégories, des séries, des éditeurs, des auteurs et des genres. C'est donc les classes que nous avons créées. Nous avons choisi de séparer les informations d'un album en plusieurs classes afin que la recherche par genre, série, etc, soit plus simple à réaliser. Nous avons gardé dans la classe Album les informations spécifiques à un seul album, soit le titre et l'image de couverture.
On considère qu'un utilisateur a aucun ou plusieurs albums dans liste d'albums et qu'inversement un album peut appartenir à aucun ou plusieurs utilisateurs. Un album possède une seule catégorie mais une catégorie contient aucun ou plusieurs albums. Une série est quant à elle composée d'au moins deux albums. Il est à noter cependant que dans notre base de données certains albums sont seuls dans la série car on est conscient qu'ils appartiennent à une série mais un seul album est entré dans la base de données, par exemple Lucky Luck. Un album quant à lui appartient à aucune ou une seule série. Un éditeur est composé d'aucun ou plusieurs albums et un album n'est édité que par un seul éditeur. Enfin, un auteur peut avoir écrit un ou plusieurs albums (un minimum sinon on ne le considère pas comme un auteur, il n'y a pas d'album sur le marché sans auteur) et un album est la création d'un ou plusieurs auteurs. En effet, on considère comme auteur d'un album à la fois celui qui écrit l'histoire mais aussi le dessinateur.
On obtient le diagramme de classe suivant :

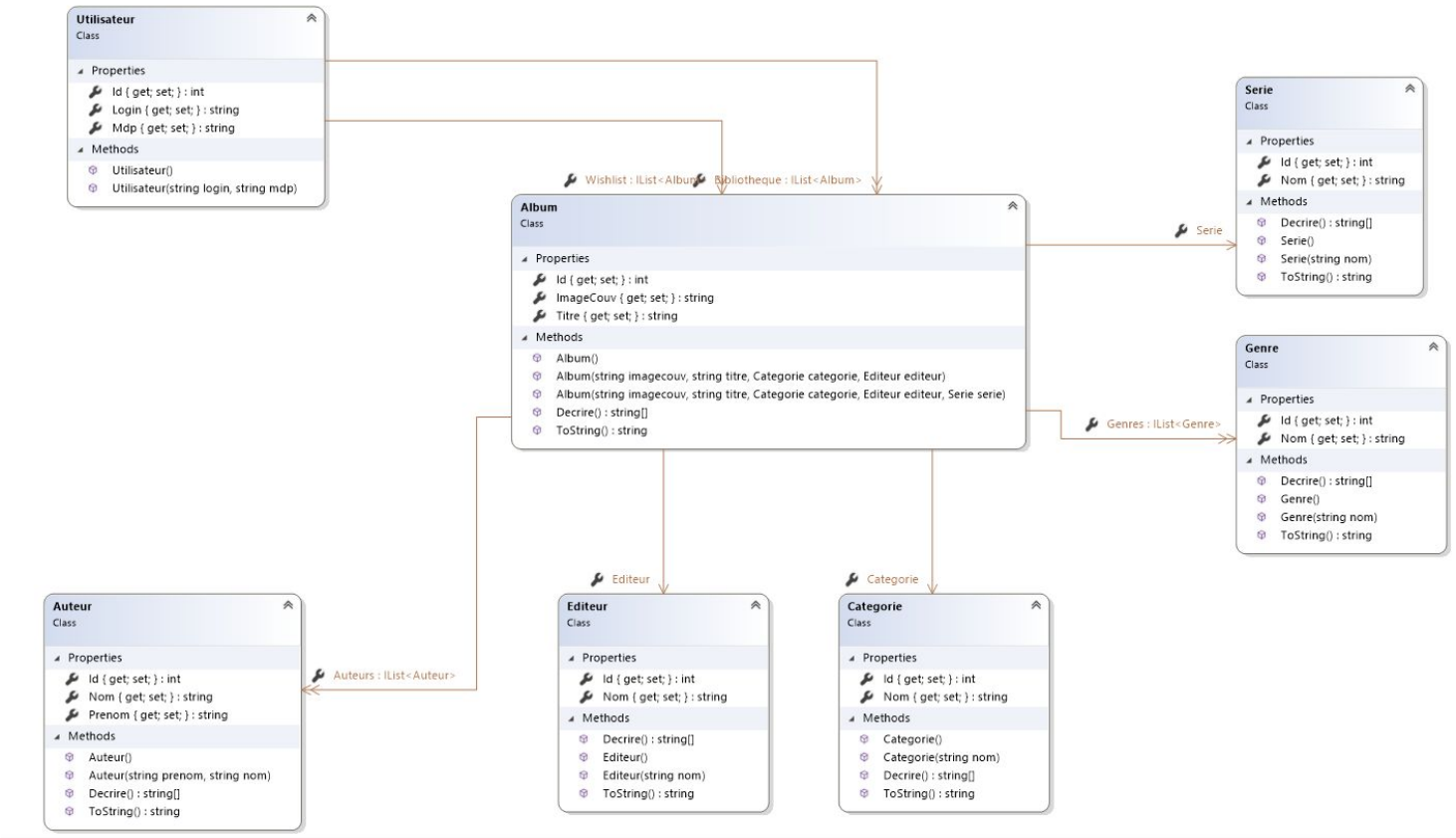


Figure 1 : Diagramme de classe

De plus, nous avons fait un diagramme de séquences pour scénariser une utilisation de l'application et mieux représenter les liens entre les classes.

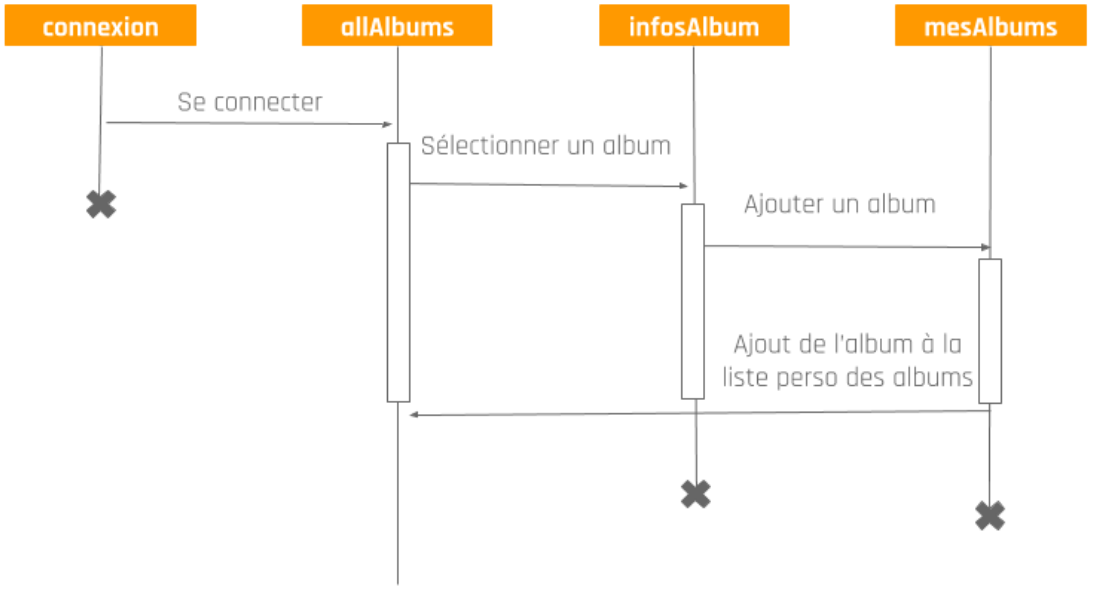


Figure 2 : Diagramme de séquence

Grâce à l'injection qui est réalisée dans le DAL, les tables associées au classe sont directement créées (si elles n'existent pas déjà) dans la base de données "bdtheque", voilà la structure que l'on obtient :

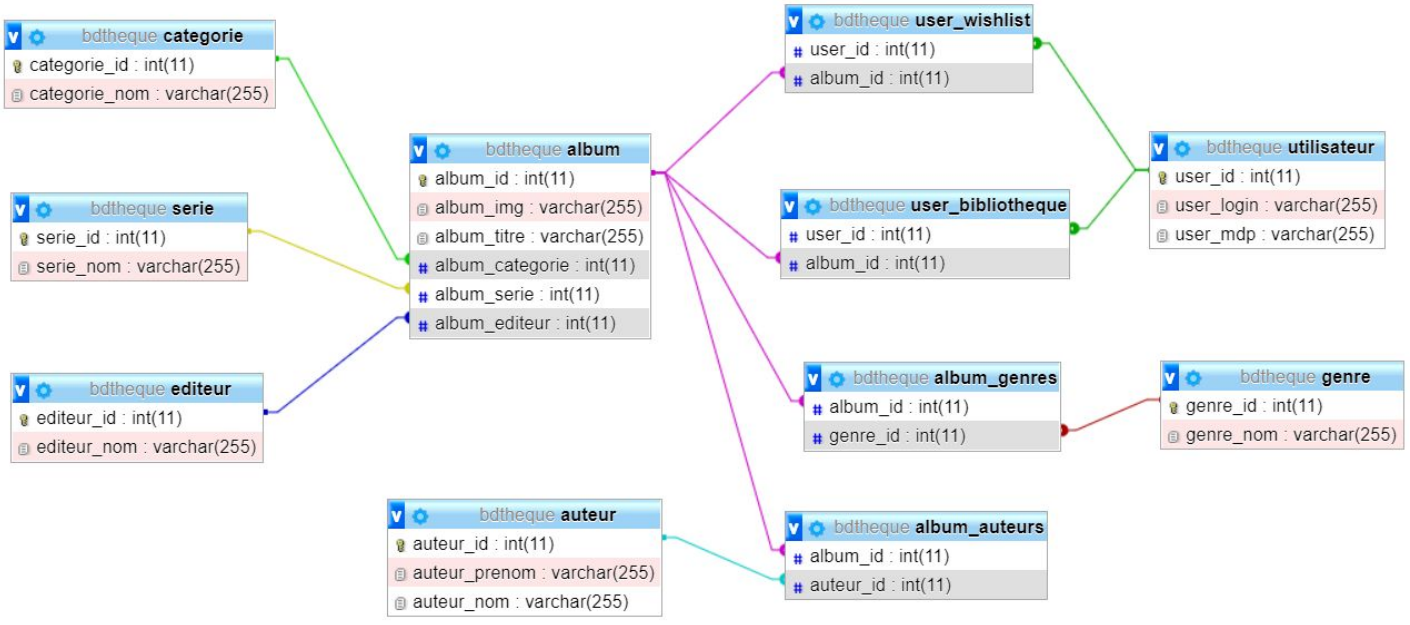


Figure 3 : Structure de la base de données

En plus de ces classes, nous avons créé des repositories qui leur sont associées. Ce sont à des répertoires de données correspondants à chacune des classes de l'application. Elles permettent de récupérer et d'enregistrer les données sur une base de données.

II.3. Architecture de la solution

Nous avons utilisé l'architecture en couches suivante lors du développement de notre application :

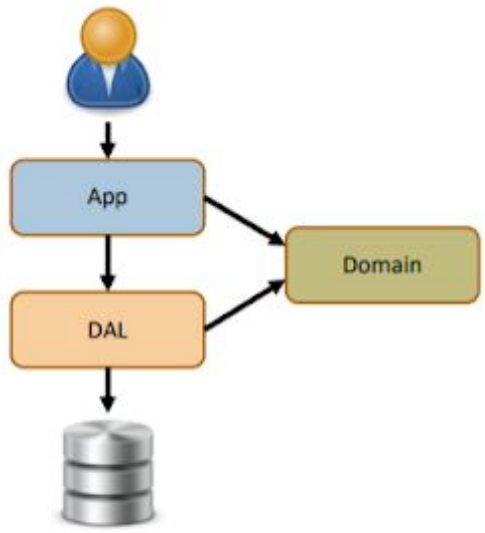


Figure 4 : Architecture en couches - DAL/Domain/App

Cette architecture nous a semblé être la plus adaptée à notre projet. Celle-ci correspond en effet parfaitement au problème traité. De plus, elle a été étudiée en cours magistraux et lors de travaux pratiques ce qui nous permet de mettre en pratique les connaissances vu en cours.

La solution est constituée de trois WinForm différents. Nous avons réalisé un WinForm principal permettant d'accéder à la majorité des fonctionnalités et deux WinForm de plus afin de gérer la connexion et l'affichage des informations spécifiques d'un album et son ajout. Ainsi on a :

Nom	Accès par	Fonctionnalités
Connexion	Première fenêtre	S'identifier

BDTheque	Connexion	Accéder à la liste des albums du marché Accéder à sa liste d'albums Accéder à sa wishlist Rechercher un album Afficher les informations d'un album
InfosAlbum	BDTheque	Ajouter un album à sa liste d'albums Ajouter un album à sa wishlist

Et voici le lien entre les différentes fenêtres :

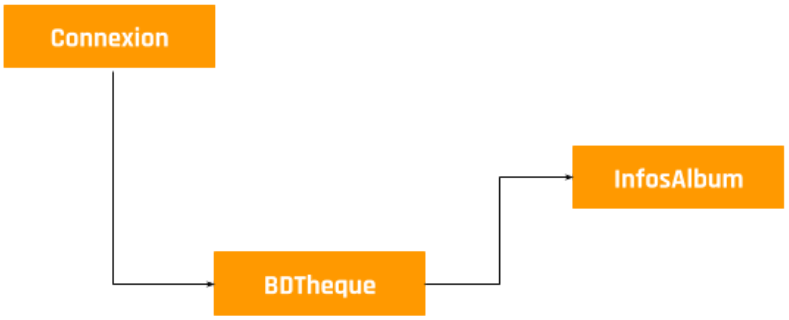


Figure 5 : Représentation des liens WinForm

II.4. Choix de conception

Nous avons choisi d'orienter la conception vers la simplicité au vu du peu de temps disponible pour la réalisation du projet. Nous avons ainsi suivi la méthode KISS (Keep It Simple, Stupid). Cela peut se voir dans les liens entre les tables puisque les associations sont seulement de 1 à plusieurs alors qu'un modèle de plusieurs à plusieurs auraient pu être cohérents par moment. Notre modèle était plus simple à implémenter.
De plus, nous n'avons seulement gardé que des fonctionnalités qui sont nécessaires comme le précise la méthode YAGNI.

II.5. Maquettage

Nous avons réalisé un zoning et une maquette dynamique (voir Annexes) afin d'avoir une idée générale de l'interface de l'application. Comme on peut le voir sur les exemples suivants, nous avons essayé de rester sur un positionnement des éléments relativement simple qui nous a permis de respecter les principaux principes d'ergonomie et, comme précisé dans la partie précédente, de respecter les contraintes de temps. Dans le but de valider le design de notre application nous avons fait tester à quelques personnes notre maquette dynamique. Leurs retours étant bons, nous avons eu une première validation de nos choix ergonomiques, confirmée par la suite lors des tests réalisés sur l'application finale.

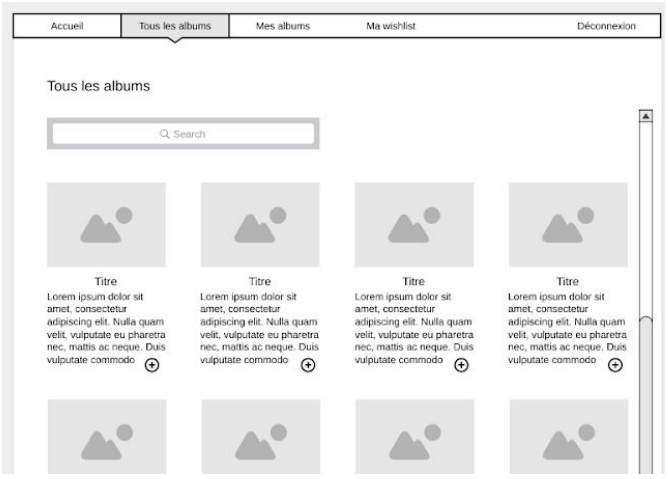


Figure 6 : Zoning tous les albums

Figure 7 : Maquette tous les albums

Figure 8 : Zoning album sélectionné

Figure 9 : Maquette album sélectionné

II.6.Gestion du projet

La gestion de ce projet et la répartition des tâches ont été marquées par deux périodes distinctes. La première phase a duré 3 semaines et a été réalisée en distanciel. Durant celle-ci nous nous sommes répartis les tâches de la façon suivante : l'architecture et les spécifications des classes ont été pensées en groupe, puis Oriane s'est chargée de les mettre au propre, le zoning et le maquettage ont été réalisé par Oriane pendant que Lucie s'occupait de sélectionner les données à insérer dans la base de données et de configurer NHibernate.
La deuxième phase a duré environ 2 semaines. Il s'agissait de celle en présentiel. Lors de celle-ci nous nous sommes concentrées sur le développement de la solution et avons pour cela travaillé principalement en pair coding. Cette méthode de travail nous a permis de retrouver une dynamique de groupe et des interactions sociales qui nous avaient beaucoup manqué dans le contexte sanitaire actuel.

Le planning originel que nous avions prévu est le suivant :

Figure 10 : Résumé du planning

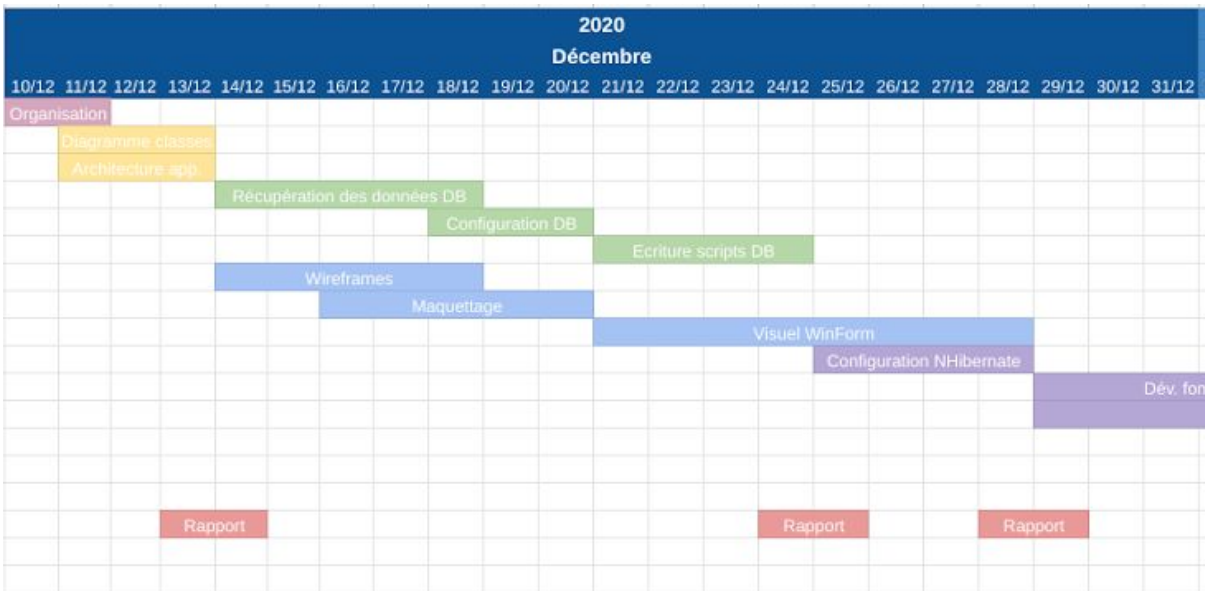


Figure 11 : Planning de décembre 2020

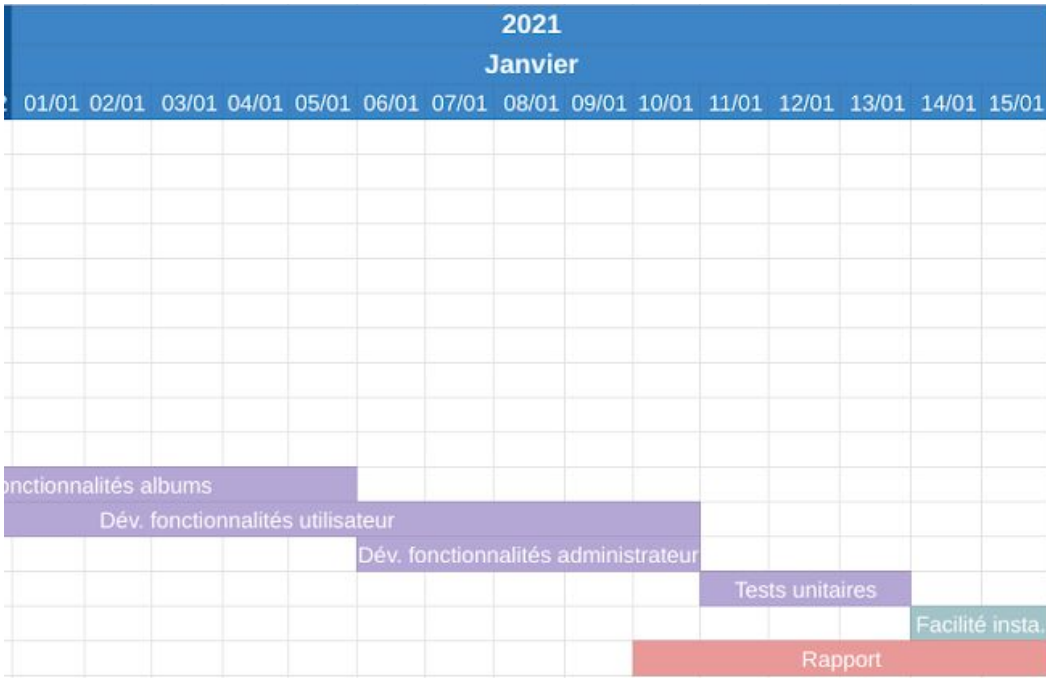


Figure 12 : Planning de janvier 2021

Nous avons cependant sous-estimé la charge de travail de décembre et l'impact du travail en distanciel sur nous. Nous avons ainsi pris du retard sur le début du projet, retard que nous avons rattrapé durant le mois de janvier en travaillant notamment en pair coding.

III. Résultats et tests

III.1. Exigence métiers respectées

Code	Exigence	Respectée
EF_01	En tant qu'utilisateur, je peux me connecter à l'application grâce à mes identifiants (login/mot de passe).	Oui
EF_02	En tant qu'utilisateur, je peux consulter la liste de mes albums.	Oui
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.	Oui
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.	Oui
EF_05	En tant qu'utilisateur, je peux ajouter un ou plusieurs album(s) du marché à la liste de mes albums.	Oui
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.	Oui
EF_07	En tant qu'utilisateur, je peux consulter la liste de mes souhaits.	Oui
EF_08	En tant qu'utilisateur, je peux retirer un ou plusieurs album(s) de la liste de mes souhaits.	Non
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.	Oui
EF_10	En tant qu'administrateur, je peux me connecter à l'application grâce à des identifiants spécifiques (login/mot de passe).	Non
EF_11	En tant qu'administrateur, je peux ajouter un album à la liste des albums du marché	Non

Page de connexion

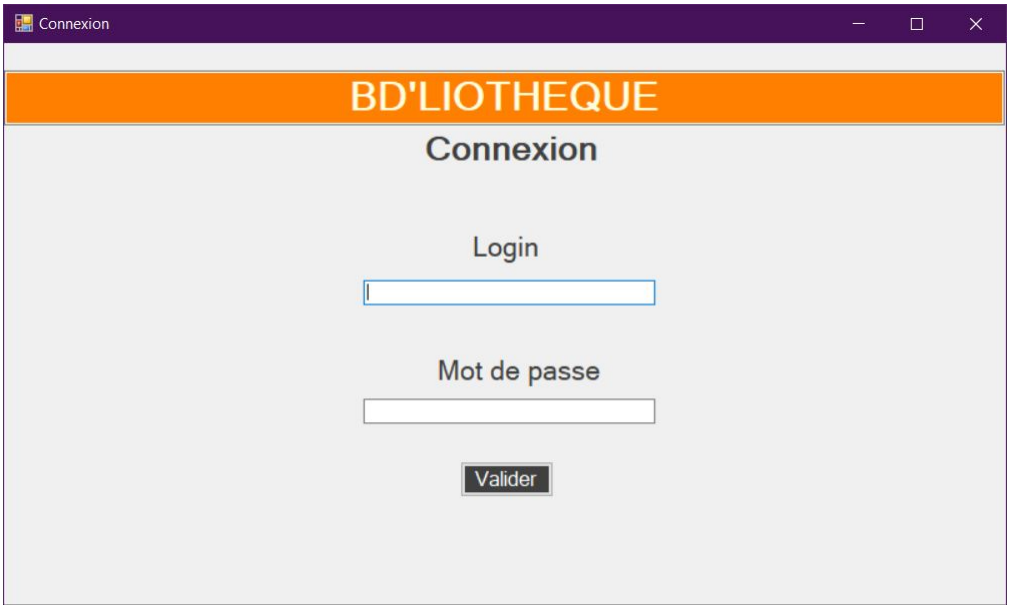


Figure 13 : Capture d'écran de la page de connection de l'application

BDThèque

L'utilisateur peut voir la liste des albums du marché et en sélectionner un pour avoir plus d'informations dessus.

Figure 14 : Capture d'écran de la page principale de l'application

InfosAlbum

L'utilisateur peut voir les informations détaillées d'un album et l'ajouter à sa liste d'albums ou à sa wishlist.

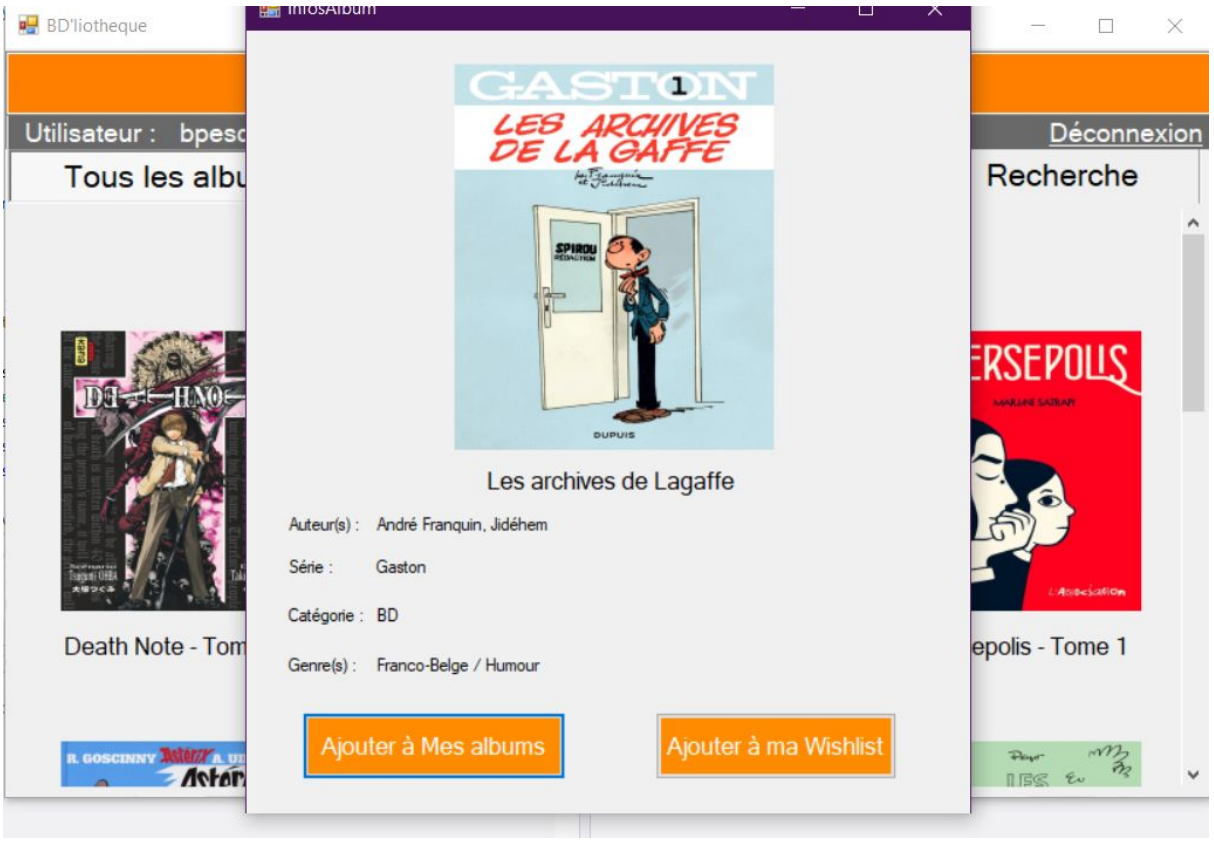


Figure 15 : Capture d'écran de la page d'affichage des informations d'un album

III.3. Protocoles de tests

Nous avons réalisé quelques tests unitaires pendant la conception de l'application cependant ils ne fonctionnent pas. En effet, nous n'avons pas réussi à faire marcher celui permettant de recréer la base de données et dont découlent les autres tests. Ainsi, nous n'avons pas pu tester notre code avec les tests unitaires. Afin de compenser un minimum ce manque, nous avons demandé à quelques personnes de tester notre application. Toutes les fonctionnalités implémentées semblent fonctionner parfaitement.

III.4. Résultats

Suite à ces tests, nous pouvons conclure que, sur les exigences métiers réalisées, notre application est entièrement fonctionnelle. Les retours des personnes ayant testé l'application étaient bons et ils ont pu utiliser la solution sans avoir de difficultés particulières. Ainsi nous confirmons l'impression que nous avions eu lors des tests utilisateurs par rapport aux exigences ergonomiques, à savoir qu'elles nous semblent respectées.

IV. Procédure d'installation

IV.1. Dézipper le fichier et l'ouvrir sur Visual Studio

1. Téléchargez le zip sur Github et enregistrez-le dans vos documents.

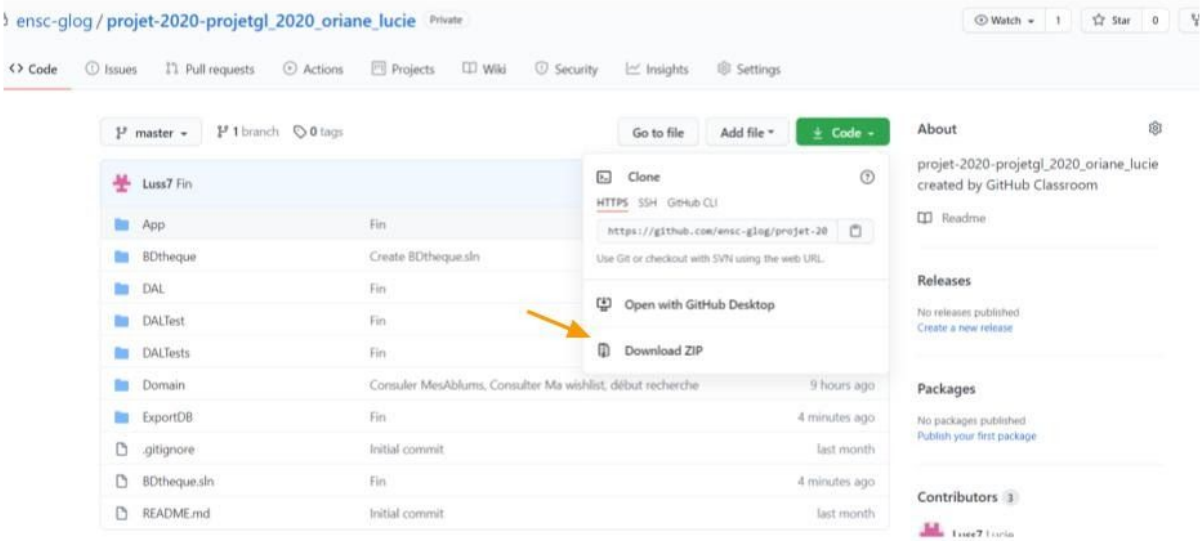


Figure 16 : Téléchargement du zip

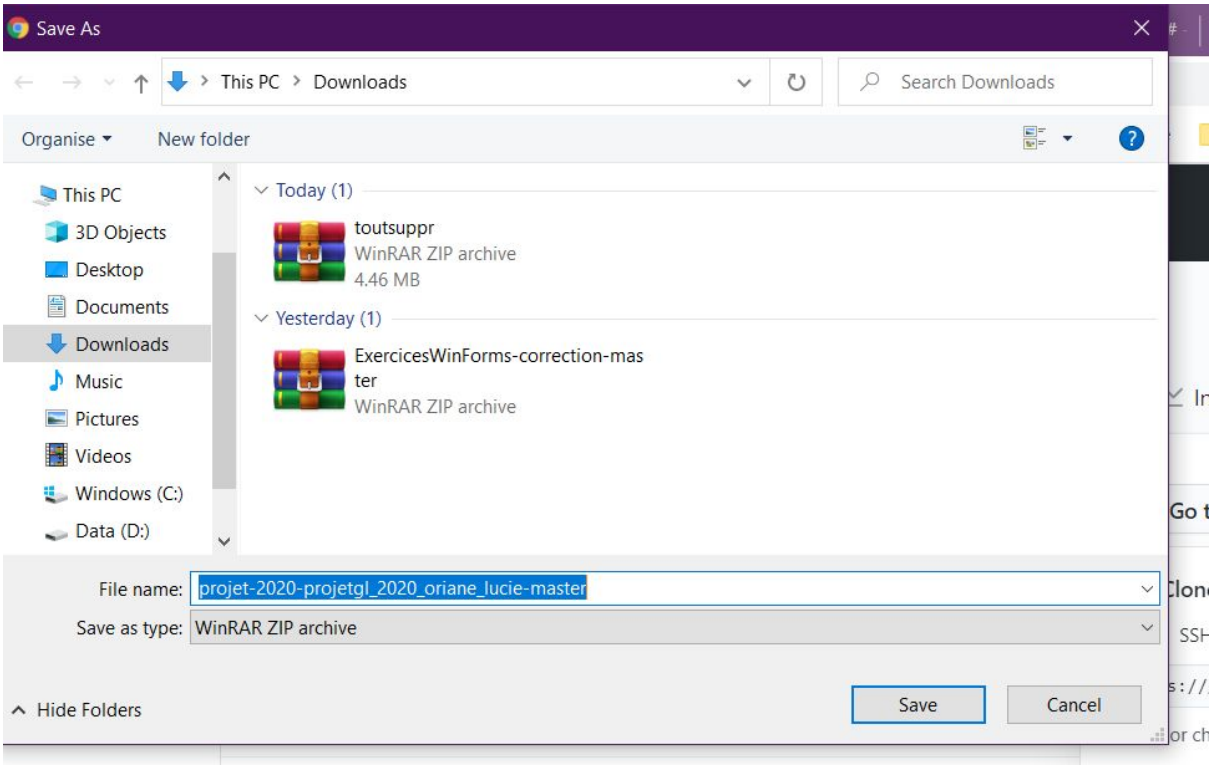


Figure 17 : Enregistrement du zip

2. Ouvrez le zip. Vous devez obtenir le résultat suivant :

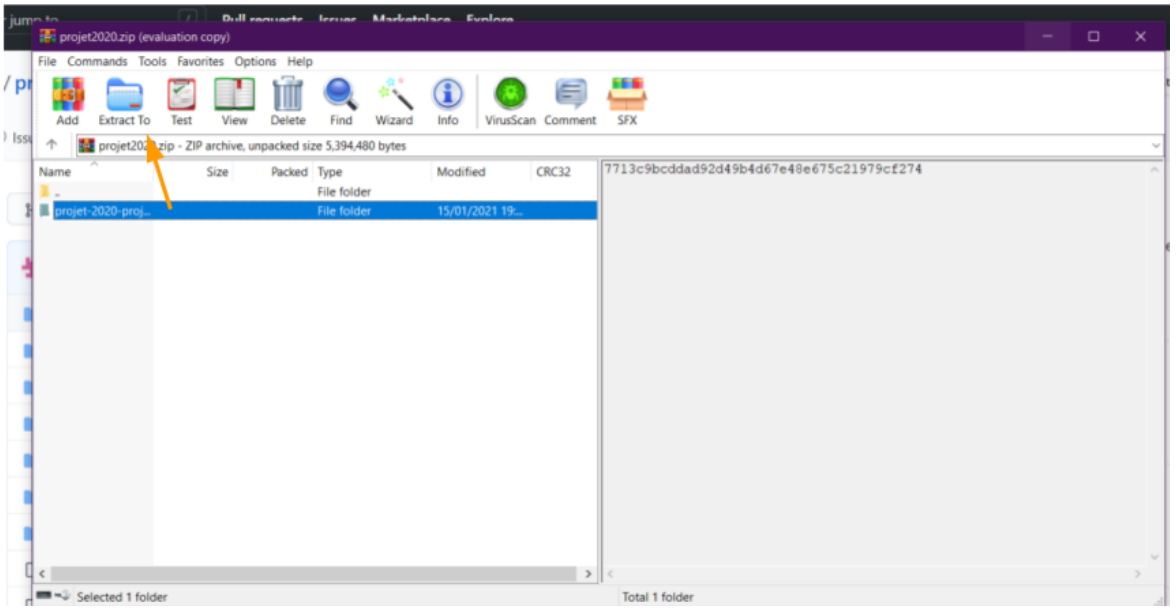


Figure 18 : Extraction du dossier

Cliquez sur “Extraire” afin de dézipper la solution dans vos documents.

3. Ouvrez le dossier et ouvrez le fichier BDtheque.sln dans Visual Studio.

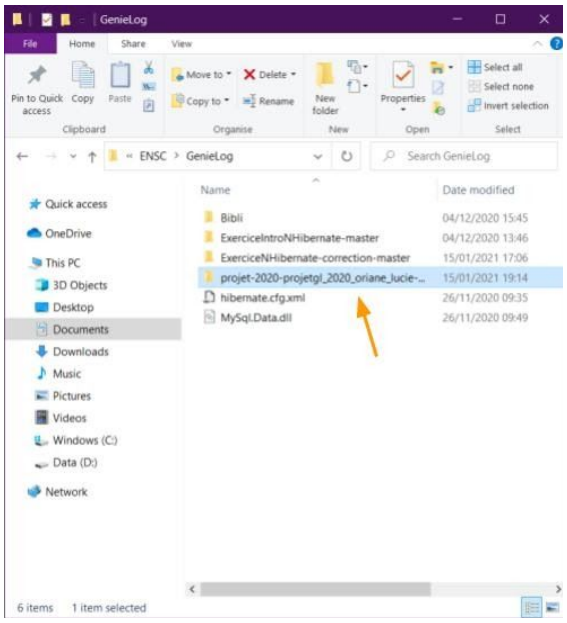


Figure 19 : Ouverture dossier

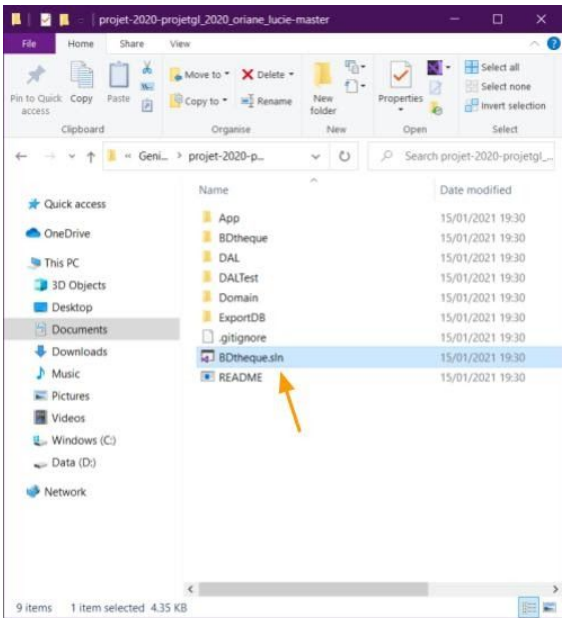


Figure 20 : Lancement de la solution

IV.2. Créer la base de données

1. Lancez Xampp sur votre ordinateur et cliquez sur “Start” pour les lignes “Apache” et “MySQL”. Vous devez obtenir le résultat suivant :

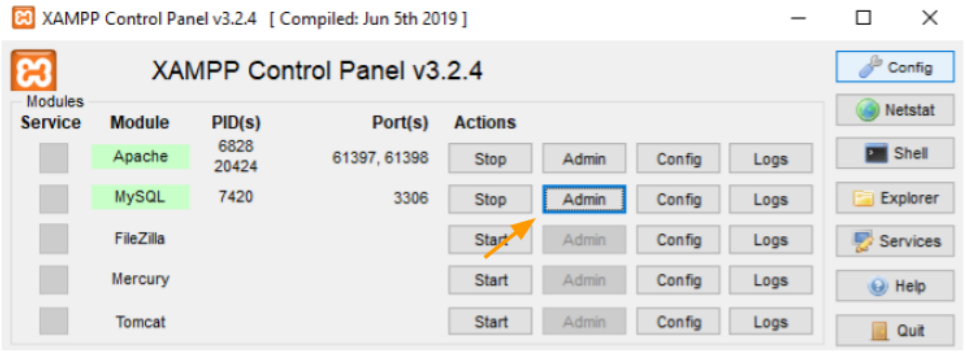


Figure 21 : Lancement de PhpMyAdmin

Cliquez sur “Admin” au niveau de “MySQL”. PhpMyAdmin s’ouvre alors dans votre navigateur web.

2. Dans le code Visual Studio, ouvrez le fichier Database.sql. Copiez alors l’ensemble des lignes du fichier (Ctrl+C).

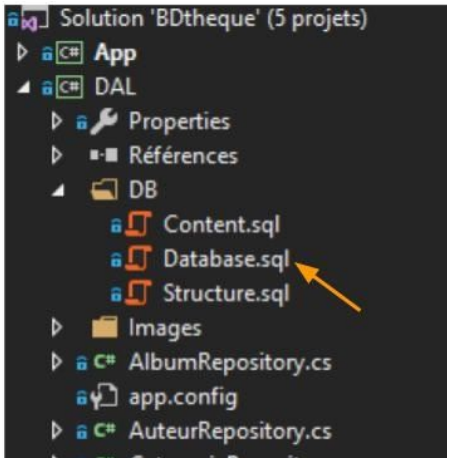


Figure 22 : Fichier Database.sql

```
Database.sql
1  create database if not exists bdtheque character set utf8 collate utf8_unicode_ci;
2  use bdtheque;
3
4  grant all privileges on bdtheque.* to 'bdtheque_user'@'localhost' identified by 'secret';
```

Figure 23 : Code à copier

3. Allez sur la page PhpMyAdmin qui s'est ouverte précédemment dans votre navigateur et cliquez sur "SQL" en haut à gauche (flèche orange sur la figure X ci-contre).

Figure 24 : SQL dans PhpMyAdmin

4. Copiez le code que vous avez collé à l'étape 2 et cliquez sur "Exécuter" en bas à droite (flèche orange).

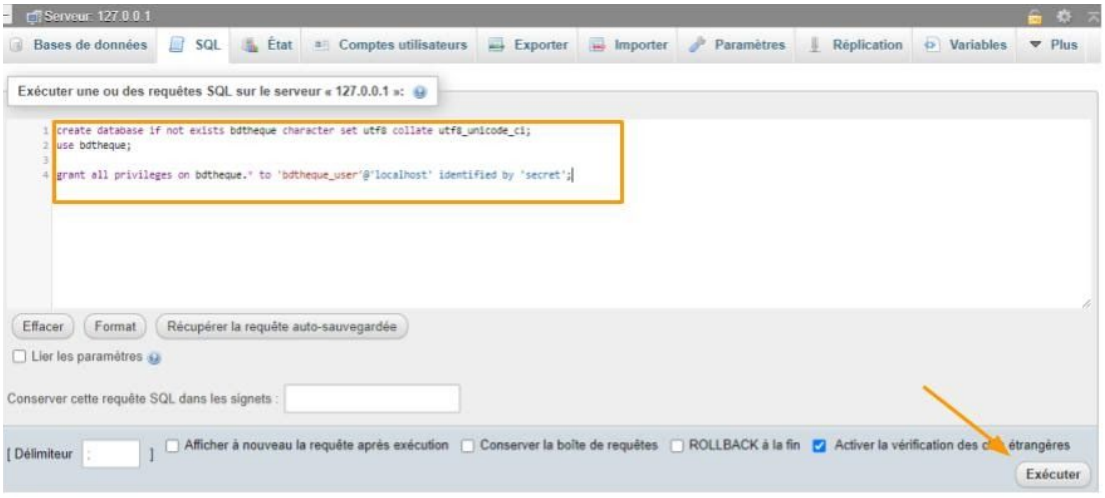


Figure 25 : Exécution de la requête

Vous devez obtenir le résultat suivant :

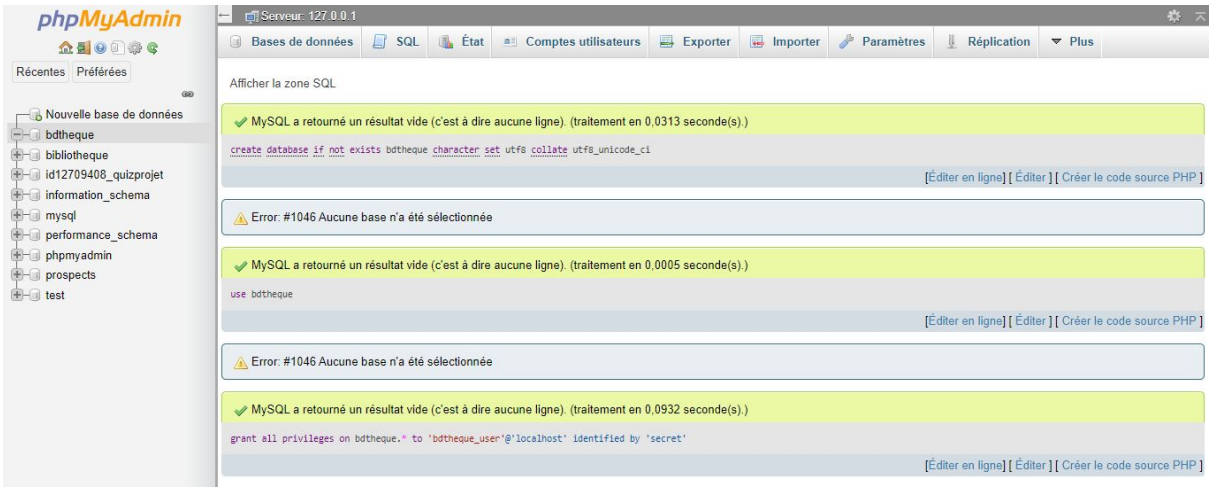


Figure 26 : Résultat après exécution de la requête

5. De retour sur Visual Studio sélectionnez “ExportDB” dans la flèche déroulante présente en haut de la fenêtre, puis cliquez sur “Démarrer” (flèche verte de lancement). La création de la base de données et son remplissage se lancent alors.

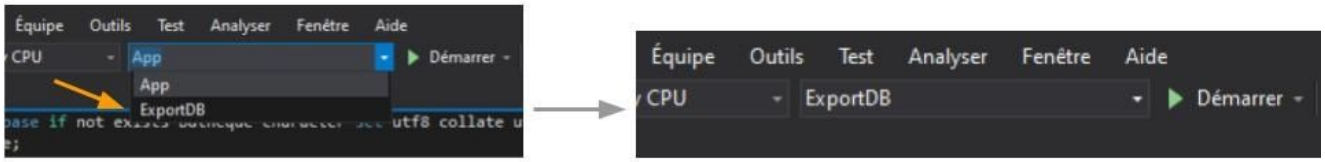


Figure 27 : Passage de App à ExportDB

/!\ Cette étape peut prendre un certain temps. Une fois celle-ci correctement réalisée vous devez obtenir le résultat suivant :

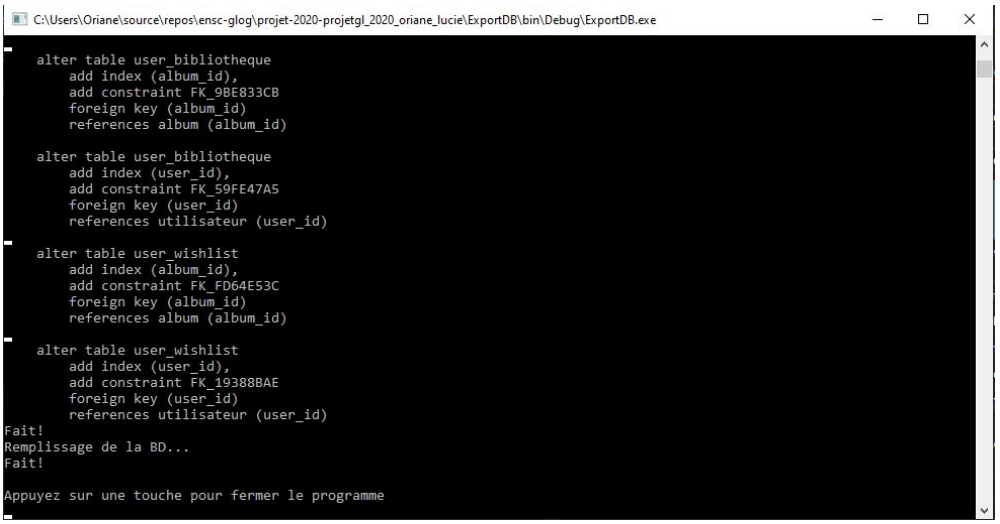


Figure 28 : Résultat création de la DB - Panneau de commandes

Vous pouvez alors fermer cette fenêtre. Votre base de données a bien été créée. Vous pouvez la retrouver dans PhpMyAdmin en actualisant la page (figure X ci-dessous).

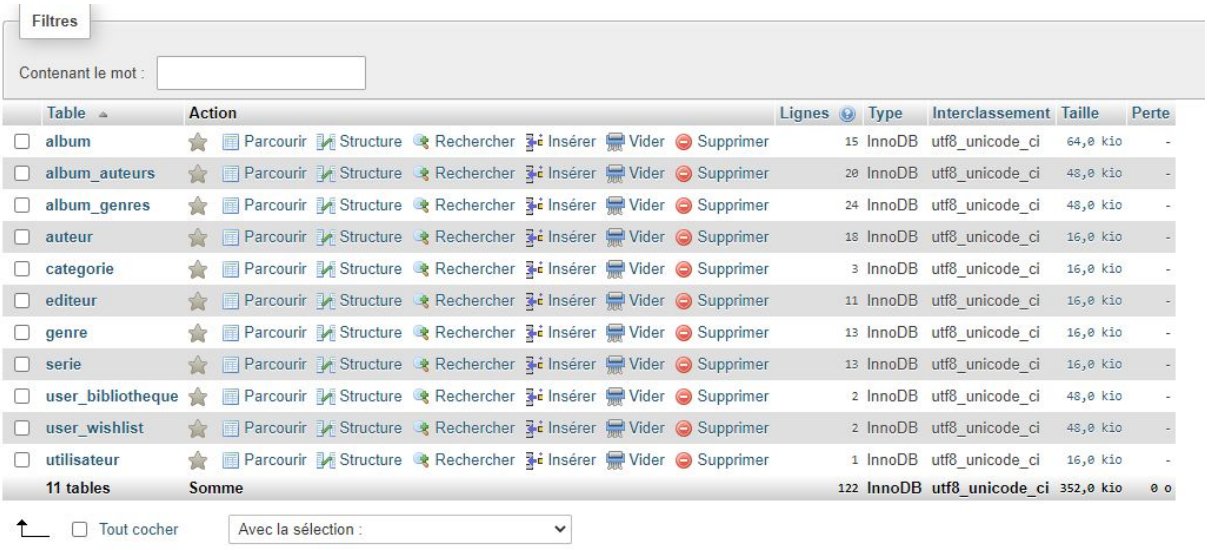


Figure 29 : Résultat création de la DB - PhpMyAdmin

IV.3. Lancer l'application

Afin de lancer l'application, retournez sur Visual Studio. Ici, resélectionnez "App" dans la barre déroulante en haut du logiciel (cf étape 5 du IV.2. Créer la base de données). Une fois ceci fait, vous pouvez lancer l'application en cliquant sur le bouton Démarrer.

V. Bilan et perspectives

La majorité des exigences métiers sont respectées puisque toutes les fonctionnalités nécessaires ont bien été précisées. Les exigences métiers du produit minimum viable sont toutes fonctionnelles et quelques exigences supplémentaires ont été implantées. Il aurait pu être intéressant de traiter la partie administrateur de l'application ou bien la suppression d'un album des WinForm "Ma liste d'albums" et "Ma wishlist". Dans la même dynamique, la fonctionnalité permettant l'automatisation de la mise à jour de la wishlist lorsqu'un album souhaité est acheté aurait pu être mise en place. Ces exigences n'ont pas pu être réalisées à cause de contraintes de temps.

De façon générale, l'application est esthétique et simple à prendre en main, notamment grâce au fait que chaque fonctionnalité soit liée à un bouton qui lui est propre. Les tests nous ont également permis de vérifier que la solution fonctionne bien et que le lien avec la base de données est bon. Ils nous ont également permis de relever quelques points à améliorer, par exemple le fait qu'on puisse ajouter deux fois le même album dans "Ma liste d'albums" et "Ma whislist".

Nous avons choisi de rester proches de ce qui était demandé pour finir le projet dans les temps et ainsi les possibilités de l'utilisateur sont plutôt limitées. Dans une utilisation réelle il aurait été intéressant d'avoir des informations ou fonctionnalités complémentaires, par exemple un résumé pour chaque album.

Au niveau personnel, ce projet nous a permis de mettre en pratique les notions vues en cours, par exemple l'utilisation de NHibernate, les WinForm ou les patterns. Nous avons ainsi développé nos compétences en génie logiciel. Ce projet nous a également demandé beaucoup d'adaptation au vu de la situation sanitaire actuelle. Nous avons donc dû faire preuve d'une bonne communication et d'organisation. Enfin, le pair coding nous a permis de retrouver une bonne dynamique de groupe, des interactions sociales et des échanges d'idées qui nous avaient manqué durant la première partie en distanciel.

Annexes

Zoning

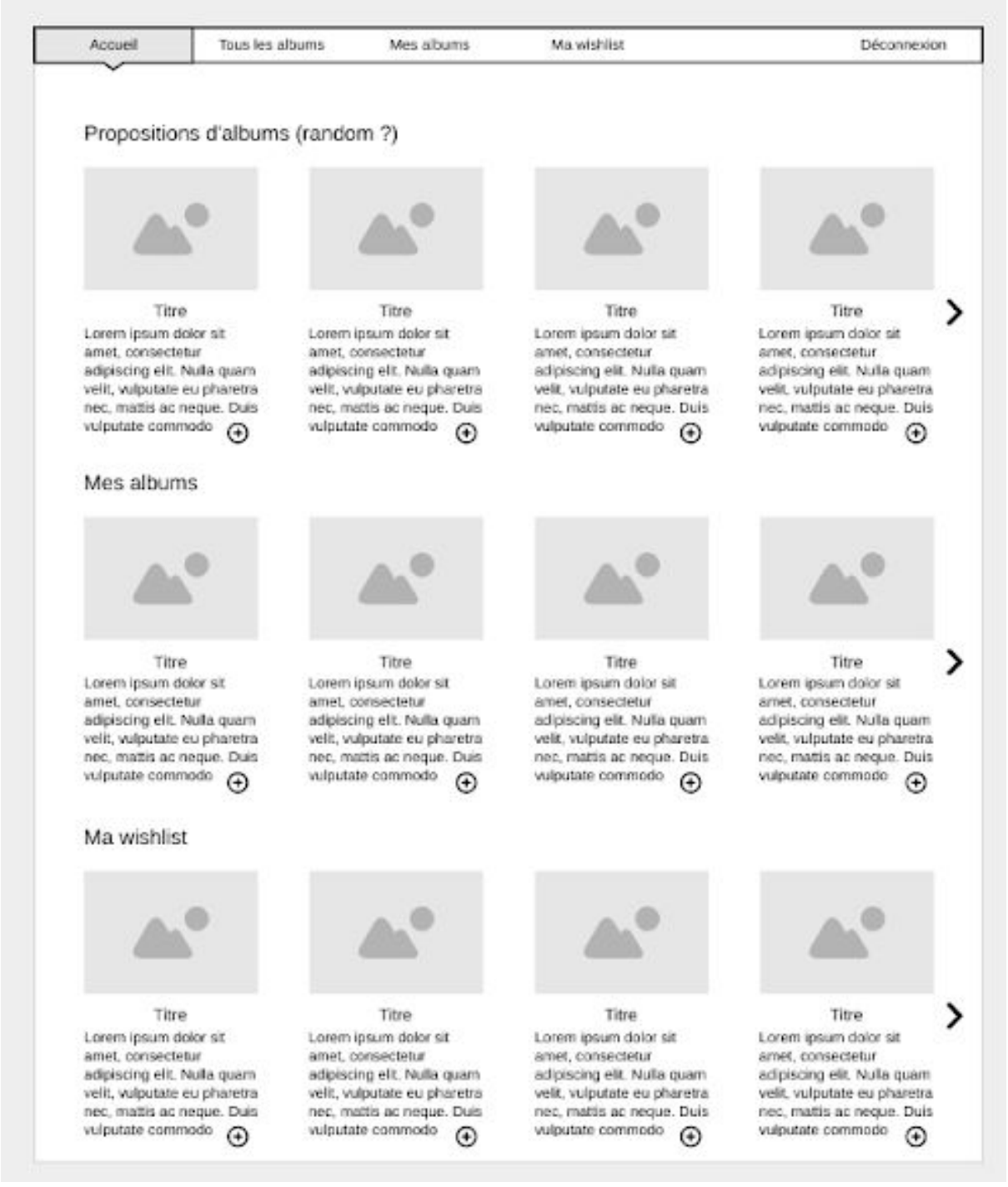


Figure 30 : Zoning de la page d'accueil initialement prévue

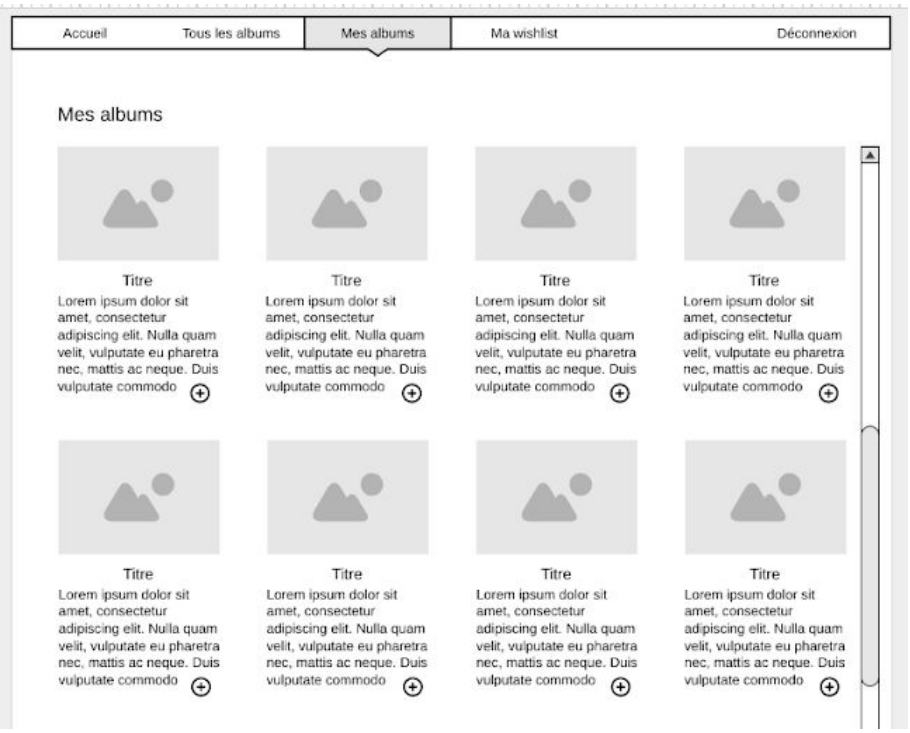


Figure 31 : Zoning de la page "Mes albums"

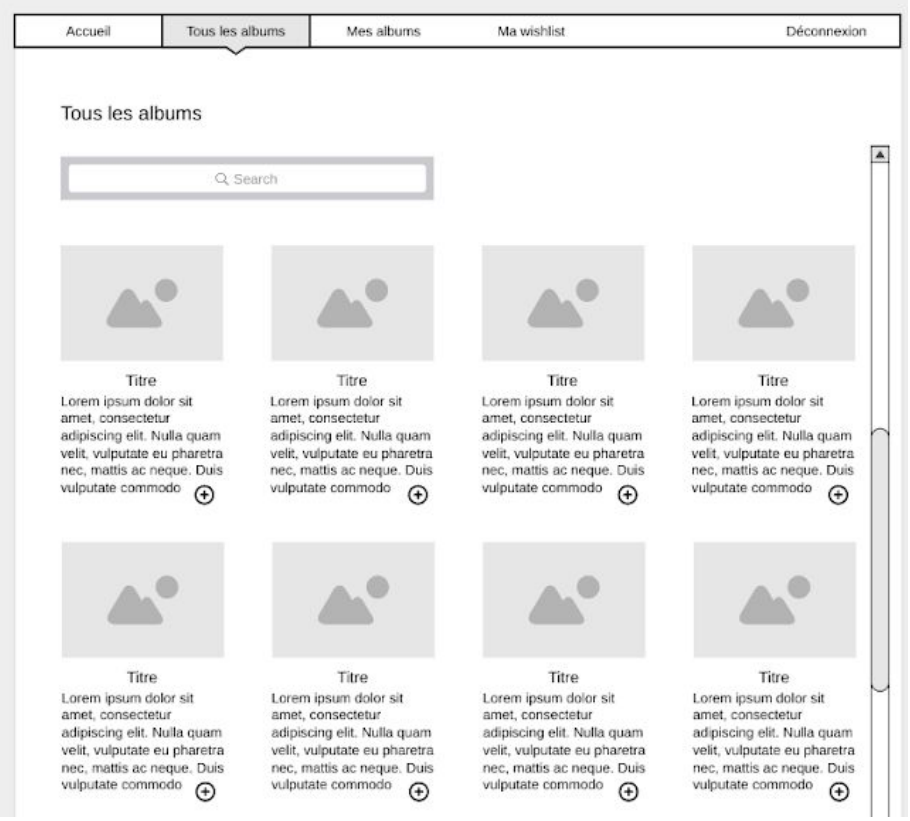


Figure 32 : Zoning de la page "Tous les albums"

Figure 33 : Zoning de la page d'un album sélectionné

Maquettes

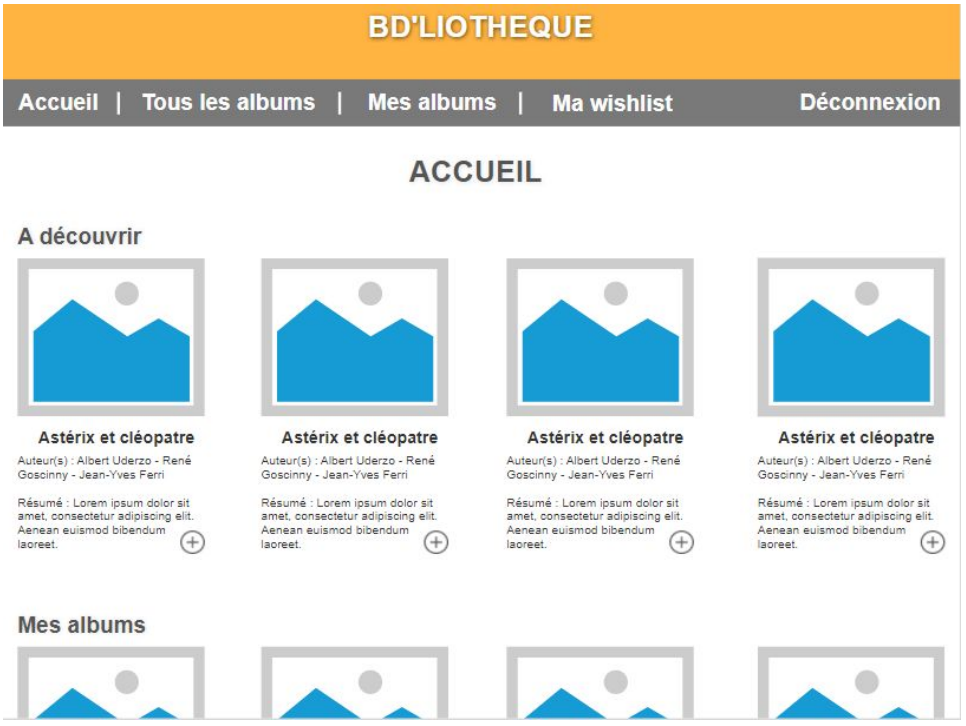


Figure 34 : Maquette de la page d'accueil initialement prévue

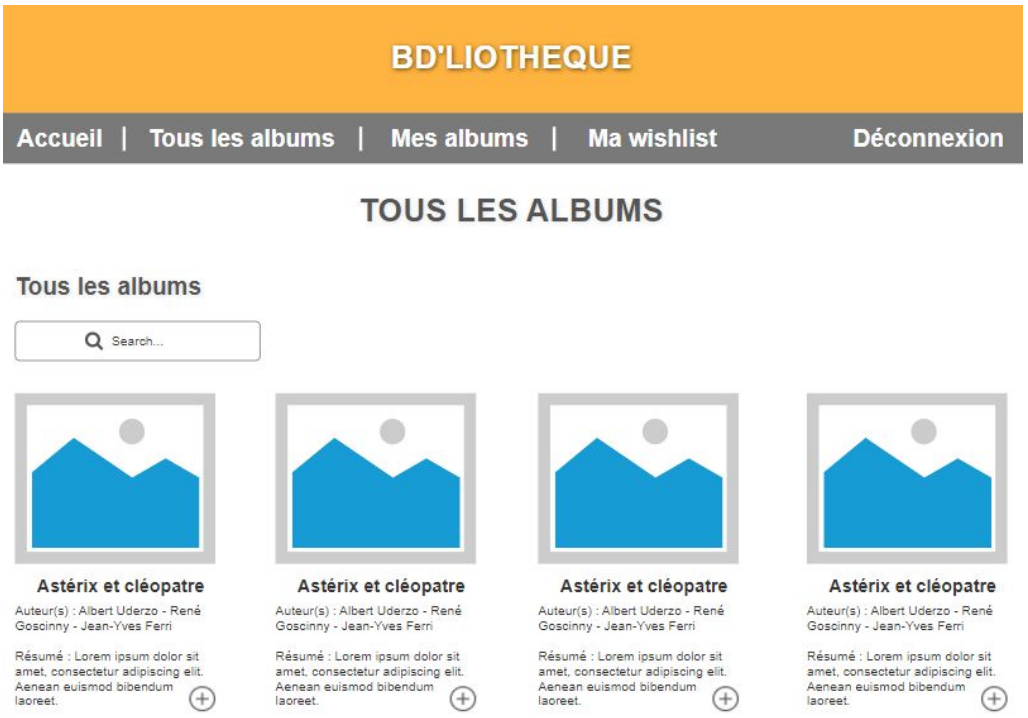


Figure 35 : Maquette de la page "Tous mes albums"

Figure 36 : Maquette de la page "Mes albums"

Figure 37 : Maquette de la page d'un album sélectionné

Figure 38 : Maquette de la page "Ma whishlist"