

ECOLE NATIONALE SUPÉRIEURE DE COGNITIQUE

2ème année

U.E GÉNIE LOGICIEL

PROJET GÉNIE LOGICIEL



Présenté par :

Brissaud Cloé

Lavenseau Louis

Année universitaire 2020 - 2021

Table des matières

I- INTRODUCTION	3
II-Présentation du projet	3
III- Démarche de la conception IHM	5
IV Modélisation des données	15
a - Diagramme des classes métiers	15
b- Modèle relationnel	16
V-Description de l'architecture technique	16
a - Projet Model	16
b - Projet DatabaseAccess	17
c - Projet Presentation	17
VI- Procédure d'installation	18
VII- Liste des procédures de test	18
a - Projet ModelTests	18
b - Projet DatabaseAccessTests	18
VIII- Répartition des tâches et planning détaillé	19
IX- Bilan et perspectives du projet	21

I- INTRODUCTION

Chaque année des millions de albums sont achetés. En ces temps de confinement, les français ont davantage lu . Il n'est pas toujours facile de se rendre dans une librairie, c'est pour cette raison que notre plateforme s'adresse à l'ensemble des lecteurs voulant commander un album.

Le but de notre projet est de créer une application permettant d'explorer les albums sur le marché, de les ajouter à nos souhaits ou bien de les ajouter à nos albums personnels.

Afin de faciliter la consultation de tous les albums du marché, une barre de recherche est disponible permettant de rechercher un album selon des mots clés, comme le titre de l'album, son auteur, son genre, etc.

Pour lancer la solution, un document pdf en annexe permettra à n'importe quel utilisateur de d'installer l'application facilement.

II-Présentation du projet

Pook est une application WinForms avec données persistantes, permettant de découvrir une large gamme d' albums sur le marché tels que des romans, des essais, des BD, etc.. Son nom provient du mélange entre Puke = album en Haïtien et Book = album en anglais.

Afin de mener à bien le projet, des exigences préalables ont été données. Elles sont répertoriés dans le tableau ci-après. Les exigences en bleu correspondent aux exigences obligatoires, et celles en gris clair aux exigences secondaires.

Code	Description	Présente dans l'application	Présentation de l'exigence dans l'application
EF_01	En tant qu'utilisateur, je peux me connecter à l'application grâce à mes identifiants (login/mot de passe).	✓	Lorsque je lance la solution, une fenêtre apparaît permettant de me connecter avec mon identifiant personnel ainsi que mon mot de passe. Il est aussi possible de créer un compte.
EF_02	En tant qu'utilisateur, je peux consulter la liste de mes albums.	✓	Sur l'application, un bouton dans la left-bar me permet d'accéder à la liste de mes albums en cliquant dessus.
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.	✓	Dans chaque onglet de l'application, les détails de chaque albums apparaissent, permettant d'obtenir toutes les informations relatives à ceux-ci.
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.	✓	Sur l'application, il est possible grâce à la barre de recherche d'effectuer une recherche dans les 3 onglets parmi n'importe quelle information relative aux albums.
EF_05	En tant qu'utilisateur, je peux ajouter un ou plusieurs album(s) du marché à la liste de mes albums.	✓	Dans l'onglet "Albums du marché" se trouve à côté de chaque album un bouton permettant d'ajouter les albums désirés à ses albums. Chaque album ajouté à ses albums peut être retrouvé dans l'onglets "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.	✓	Dans l'onglet "Albums du marché" se trouve à côté de chaque album un autre bouton permettant d'ajouter les albums désirés à la liste de ses souhaits. Chaque album ajouté à ses souhaits peut être retrouvé dans l'onglets "Mes souhaits".

EF_07	En tant qu'utilisateur, je peux consulter la liste de mes souhaits.	✓	Dans la left bar, un onglet "Mes souhaits" permet d'accéder à la liste de ses souhaits.
EF_08	En tant qu'utilisateur, je peux retirer un ou plusieurs album(s) de la liste de mes souhaits.	✓	Lorsque je suis sur l'onglet "Mes souhaits", il m'est possible de retirer un ou plusieurs albums de la liste des souhaits en cliquant sur le bouton présent à côté de chaque album.
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.	✓	Il est possible de se déconnecter en cliquant sur le dernier bouton de la left bar.
EF_10	En tant qu'administrateur, je peux me connecter à l'application grâce à des identifiants spécifiques (login/mot de passe).	✓	Des identifiants présentés dans la notice d'utilisation permettent d'accéder à un compte administrateur.
EF_11	En tant qu'administrateur, je peux ajouter un album à la liste des albums du marché.	✓	Une fois que j'ai pu accéder à mon compte administrateur, un form permet de renseigner et d'ajouter de nouveaux albums qui seront ajoutées à la liste des albums du marché.

Figure n°1 : Tableau des exigences de l'application

III- Démarche de la conception IHM

Dans le développement d'une application, il est très important de porter attention à la conception de l'interface-homme machine. L'application doit répondre à un bon nombre de critères. En effet, elle doit être la plus intuitive possible, l'utilisateur doit pouvoir apprécier le visuel de l'interface, tout en pouvant accéder avec facilité aux diverses fonctions de l'application.

Afin de mettre en place la structure et le visuel de l'interface, nous avons dans un premier temps réalisé un benchmark. Nous avons comparé les stratégies préexistantes sur le marché afin de nous en inspirer et d'en retenir le meilleur pour la mise en place de notre application Pook. Nous en avons retenu une structure commune, que nous avons décidé de maquetter (*cf figure n°2*)



Figure n°2 : Maquette de l'interface du menu principal de l'application

S'appuyer sur des structures connues par de nombreux utilisateurs leur permet de réduire leur effort cognitif et de ne pas chercher pendant trop de temps les fonctionnalités auxquels ils souhaitent accéder. Afin de confirmer le choix de la structure de notre application, et de choisir une charte graphique qui sera propre à Pook, nous avons décidé de faire passer quelques tests utilisateurs auprès de notre entourage (crise sanitaire oblige).

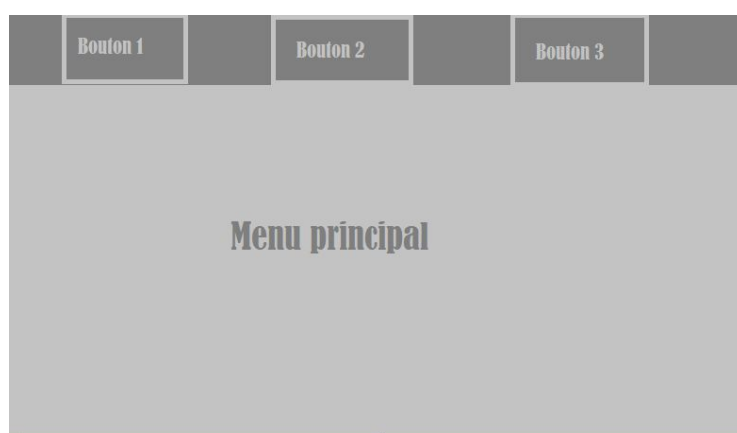
Nous avons présenté plusieurs maquettes de l'application, présentées ci-dessous. L'une avec les fonctionnalités (bouton "Album du marché", "Mes albums" et "Mes souhaits") présentes sur la left-bar (celle qui est la plus retrouvée dans les applications du marché), une avec une right bar, et la dernière avec une top bar. Les sujets devaient dire quelle maquette ils préféraient.



Maquette avec la left bar



Maquette avec la right bar



Maquette avec la top bar

Figure n°3 : Maquette des différentes interfaces testées

Sur les 5 personnes interrogées, 4 ont préféré la maquette avec la left bar et 1 avec la top bar. Nous préférons nous aussi la left bar. Nous avons donc retenu la structure avec la **left bar**. Ce choix est sans doute dû à une habitude inconsciente qui fait que nous préférons une left bar. Notre regard se porte généralement sur la gauche d'une application lorsque l'on recherche une fonction sur l'application.

Ensuite nous avons sondé les personnes sur leur préférences d'interface en terme de couleur, et puis nous leur avons proposé plusieurs associations. La première question portait sur le type d'interface qu'il préférerait : coloré (plus de 3 couleurs), épuré (avec peu de couleurs différentes et peu surchargé), et avec peu ou beaucoup d'information visuel. Parmi les 5 sujets, à l'unanimité, ils préféraient un site épuré avec peu de couleurs (moins de 3). Ils l'ont justifié par le fait qu'une trop grande quantité d'informations pouvaient nuire à la fonction première de l'application/ou du site et rendre plus difficile le fait de trouver les

fonctionnalités auxquels on veut accéder. Les préférences visuelles avec peu de couleurs ont aussi été préférées à l'unanimité.

Afin de déterminer la répartition des couleurs, nous leur avons proposé deux types de maquettes, afin d'évaluer quel type de contraste était favori.



Figure n°5: Maquette de l'interface de l'application

Parmi les 5 sujets, 4 ont préféré la left bar et la top bar plus foncées et le menu au centre plus clair, contre seulement 1 personne pour l'autre possibilité. A nouveau, nous avons une préférence pour la maquette ayant reçu le plus d'avis favorables. Nous avons donc conservé le contraste de la maquette de droite.

Nous leur avons ensuite proposé plusieurs gammes de couleurs. L'évaluation des couleurs est plus subjective et personnelle que le reste des autres tests effectués. Mais nous voulions recueillir leurs avis sur les différentes couleurs possibles de l'interface. Nous leur avons proposé cinq couleurs : bleu, vert gris, rouge, jaune et orange. Elles ont été mises sur la left bar et la bottom bar. Nous avons mis le menu dans ces exemples le menu en gris afin de ne pas risquer une mauvaise association de couleurs et donc une mauvaise évaluation de la couleur.

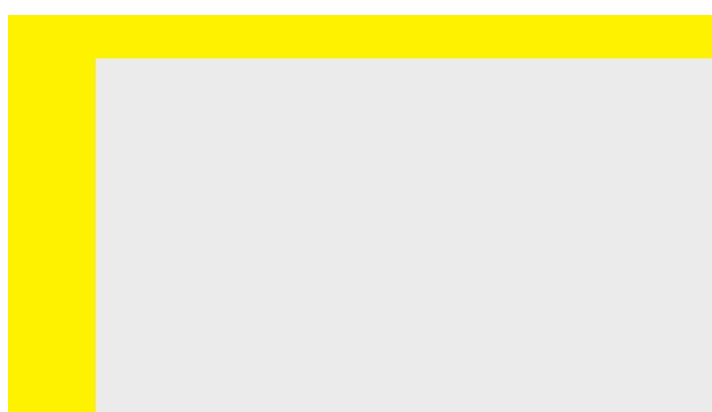
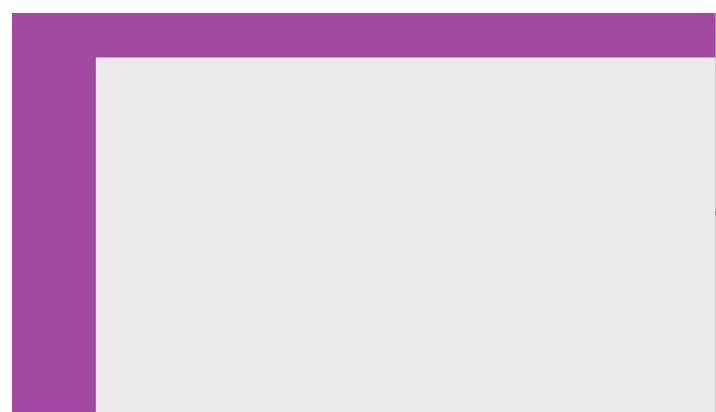
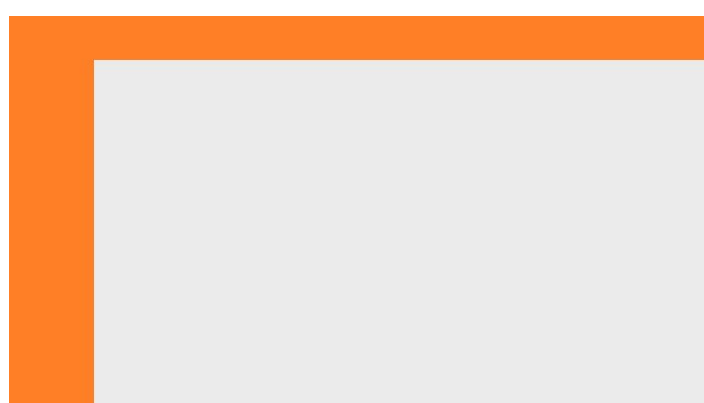
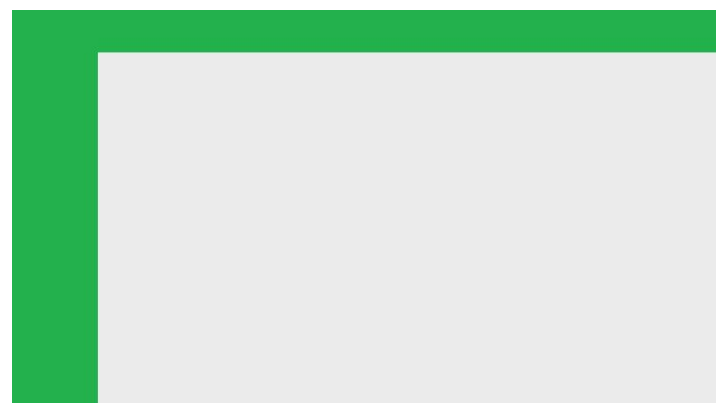
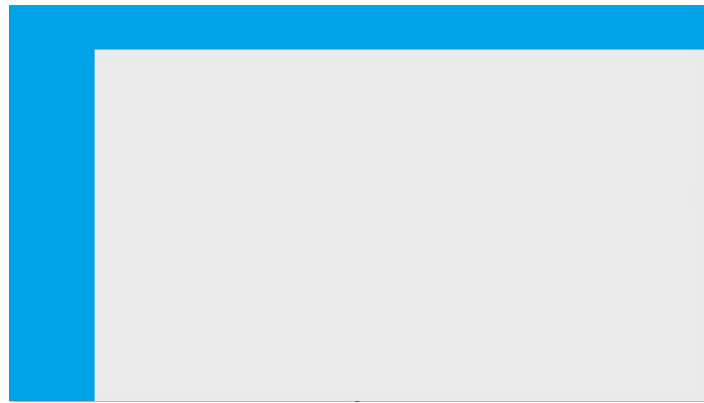
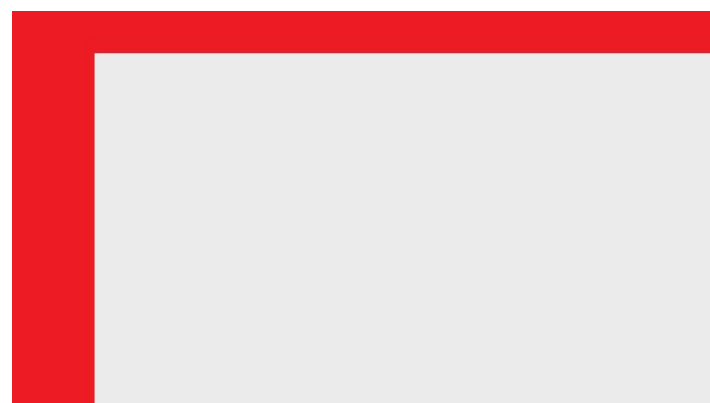


Figure n°6: Maquette de l'interface de l'application avec différente couleur

Cette évaluation subjective des couleurs, a effectivement partagé les sujets. En effet, 1 a préféré la couleur verte, 1 la couleur orange, 2 la couleur bleue et 1 la couleur rouge. Cette évaluation n'a pas eu d'impact sur notre choix des couleurs puisque, comme l'a confirmé le test, elle est proche à chaque personne.

Enfin, afin de déterminer quel type de police était préférentiellement adapté à une interface d'une application, nous avons proposé 5 types de polices différentes, au maximum différentes les unes des autres. Toutes les polices ont été mises en taille 28.

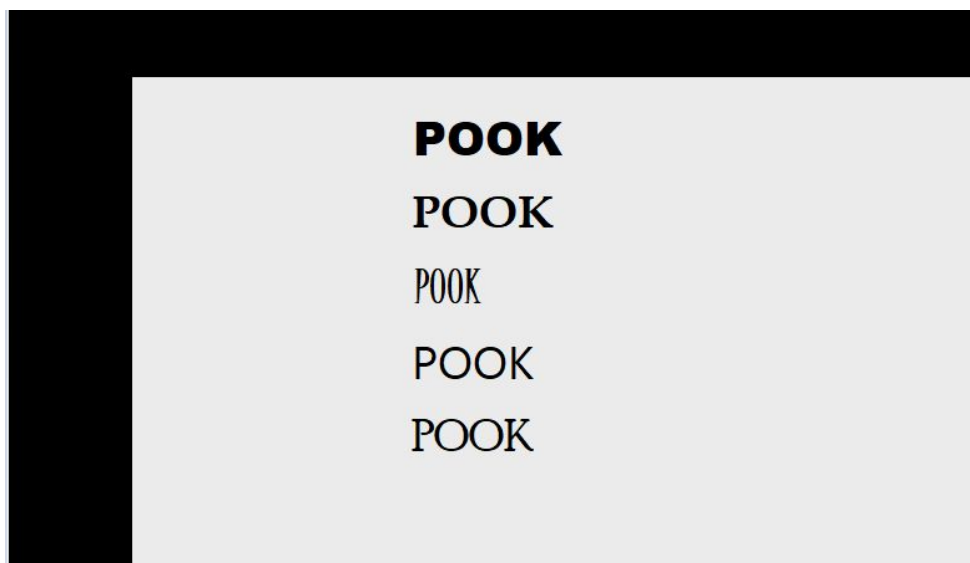


Figure n°7 : Maquette de différents types de polices

Les sujets ont préféré pour 3 d'entre eux la 1ère police, pour 1 d'entre eux la deuxième police, et un dernier a préféré la 4ème. De manière générale, les sujets ont préféré les polices qui n'étaient pas en italique, et plutôt en gras. Ils l'ont justifié par le fait qu'il pouvaient mieux distinguer le texte. Ils ont aussi déconseillé l'usage de polices originales.

Pour conclure, d'après les tests passés aux différents sujets, il en est ressorti que les sujets ont une préférence pour une application épurée avec peu de couleurs (moins de 3) et avec seulement les informations essentielles. Ils aiment que les fonctionnalités soient situées sur la left bar, que la couleur de la left et de la top bar soient plus foncées que le celle du menu central, et recommandent une police classique, plutôt en gras, permettant de bien distinguer ce qui écrit sur l'application.

Pour la question des couleurs, les sujets sont très divergents et cela reste un choix personnel à effectuer.

Grâce aux benchmark et aux différents tests effectués, nous avons pu faire ressortir la structure qu'aurait notre application dans le tableau ci-dessous.

	Structure/Fonctionnalités	Description
Left bar	<ul style="list-style-type: none"> - Information sur son profil - Bouton “Albums du marché” - Bouton “Mes albums” - Bouton “ Mes souhaits” - Bouton déconnexion 	La left bar permet d’accéder à l’ensemble des fonctionnalités proposées par l’application.
Profil en haut à gauche de la left bar	<ul style="list-style-type: none"> - Photo de profil - Login 	Les informations relatives à votre profil sont situées en haut à gauche. Il y a le login que nous avons rentré lors de votre inscription, notre photo de profil, et enfin le bouton déconnexion permettant de se déconnecter de l’application.
Bouton “Les albums du marché”	<ul style="list-style-type: none"> - Permet d’accéder à l’ensemble des albums du marchés 	L’ensemble des albums présents sur notre application sont accessibles au centre du menu principal. L’utilisateur peut faire défiler l’ensemble des albums du marché. Il peut en ajouter à la liste de ses souhaits ou de ses albums.
Bouton “Mes albums”	<ul style="list-style-type: none"> - Permet d’accéder à l’ensemble de mes albums personnels. 	L’ensemble des albums que j’ai pu ajouter à partir des albums du marché sont répertoriés dans cet onglet. Je peux consulter la liste de mes albums personnels, ainsi que la description de celle-ci (titre, auteur, image de couverture, genre, etc.).
Bouton “Mes souhaits”	<ul style="list-style-type: none"> - Permet d’accéder à l’ensemble de mes souhaits. 	L’ensemble des albums que j’ai pu ajouter à partir des albums du marché sont répertoriés dans cet onglet. Si je souhaite supprimer un album de mes souhaits, je peux le retirer en cliquant sur le bouton associé. Il est aussi possible d’ajouter l’album à mes albums en cliquant sur l’autre bouton.
Barre de recherche	<ul style="list-style-type: none"> - Permet de faire une recherche 	Sur chaque onglet, les albums du marché, mes albums ou mes souhaits, il est possible de faire une recherche en écrivant les mots clés correspondants (auteur, titre, genre etc.)

Figure n°8 : Tableau de la structure de l’application

La charte graphique est l'un des piliers de la conception d'une application. Elle permet d'avoir une identité visuelle et d'avoir une cohérence au sein de l'application. Notre charte a pris en compte les opinions lors des tests.





Charte graphique	
Couleurs thème de notre application	Leftbar (51,52,78)  Top bar (0,134,138)  Menu central : (241,239,241)  Profil en haut à gauche (39,38,60)
Logo	
Choix de la police	Nirmala UI

Figure n°9 : Tableau de la structure de l'application

Le choix des couleurs permet de donner le ton de l'application. Les couleurs sont représentatives de notre univers. Nous avons choisi des couleurs sobres et foncées pour la leftbar, ce qui donne un effet épuré à la plateforme, qui est en contraste avec la couleur clair du menu central. Pour ce qui est de la couleur de la topbar, nous avons choisi un vert d'eau. Cette couleur est alliée à l'apaisement. En effet, lorsqu'un utilisateur souhaite acheter un album, c'est pour avoir un moment de tranquillité et d'apaisement. Le logo est donc composé de la couleur la plus colorée et la plus représentative de notre application, avec un album pour rappeler le thème de notre application, qui est la consultation et l'achat d'albums.

Le choix de la police a été fait aussi selon les opinions des tests. La police est claire, et plutôt en gras afin de pouvoir lire avec clarté les fonctionnalités ou les informations présentes sur notre application.



Figure n°10 :Interface finale de notre application

IV Modélisation des données

a - Diagramme des classes métiers

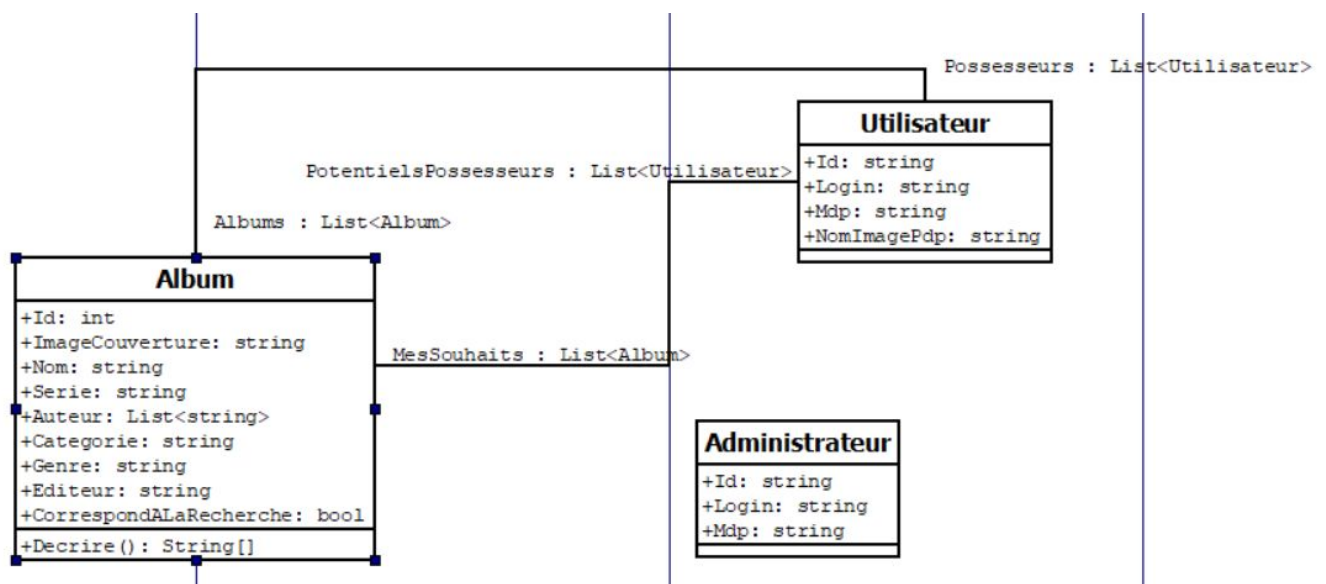


Figure n°11 : diagramme des classes métier

b- Modèle relationnel

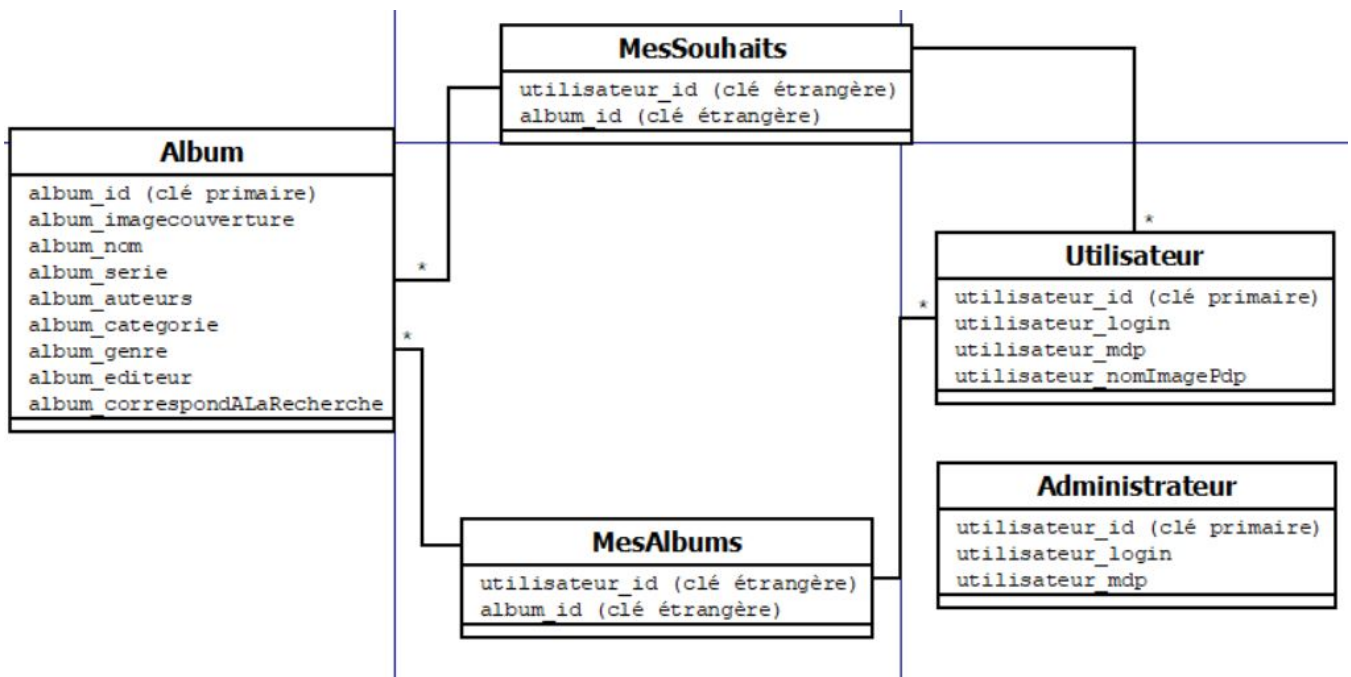


Figure n°12 : modèle relationnel

V-Description de l'architecture technique

L'application se découpe en trois projets, *Presentation*, *Model*, et *DatabaseAccess* et deux projets de test, *ModelTests* et *DatabaseAccessTests*. Chacun de ces projets sont sous la version 4.6.1 du framework .NET. Elle utilise la bibliothèque *NHibernate*.

L'architecture choisie est une architecture Model/View/Presenter. La couche *Model* correspond au projet du même nom, tandis que les couches *Presenter* et *View* sont des dossiers du projet *Presentation*. Le troisième projet *DatabaseAccess* permet la connexion et le partage de données avec la base de données.

a - Projet Model

Le projet *Model* gère la partie domaine métier. Il contient les classes métier *Administrateur*, *Utilisateur*, et *Album* décrites dans le diagramme de classe de la partie “Modélisation des données”, et le dossier *Mapping*. Ce dossier comprend les fichiers de mapping *.hbm.xml des classes métier.

b - Projet DatabaseAccess

Le projet *DatabaseAccess* fait le lien entre les données métier du projet *Model* et la base de données. Il est composé des fichiers *MySQL.Data.dll* et *hibernate.cfg.xml*, du dossier *DB* comprenant les fichiers *.sql de création de la base de données MySQL, des tables et de leur contenu nécessaires à l'accès à la base de données, et des classes *Repository* :

- *Repository* : classe de création de la *ISession* et de la *Configuration* avec un pattern de lazy-loading.
- *AlbumRepository* : permet de récupérer de la base de données : l'ensemble des albums / les albums d'un utilisateur / les albums de la liste de souhaits d'un utilisateur. Cette classe permet aussi d'ajouter à la base de données : un album aux albums du marché / à la liste de souhaits d'un utilisateur / aux albums d'un utilisateur. Enfin, elle permet de supprimer un album de la liste des souhaits d'un utilisateur.
- *UtilisateurRepository* : permet de récupérer la liste des utilisateurs de la base de données et d'en ajouter un nouveau.
- *AdministrateurRepository* : permet de récupérer la liste des administrateurs de la base de données.

Les trois classes décrites juste avant dérivent de la classe *Repository* et implémentent les interfaces du même nom : *IAlbumRepository*, *IUtilisateurRepository*, et *IAdministrateurRepository*. Les méthodes *SaveMesSouhaits* et *SaveMesAlbums* utilisent des requêtes SQL et non HQL car elles permettent de sauvegarder des objets dans des tables (*MesAlbums* et *MesSouhaits*) qui voient leur correspondance dans les classes C# à travers des attributs (respectivement les listes *MesSouhaits/Possesseurs* et *MesSouhaits/PotentielsPossesseurs*) et non des classes.

c - Projet Presentation

Le projet *Presentation* gère la partie UI et traitement des services de l'application, en lien avec le projet *Model*, respectivement avec les dossiers :

- *View* : contient les interfaces *IView* qui fournissent les services pour chaque form, les form correspondant (*FormConnexion*, *FormAccueil*, *FormAccueilAdmin*, *FormAjouterUnAlbum*, *FormCreerUnCompte*), l'interface *IView* qui fournit les services pour les *userControls* du form *formAccueil*, et les *userControls* (*UserControlAccueil*, *UserControlAlbumsDuMarche*, *UserControlMesAlbums*, *UserControlMesSouhaits*).
- *Presenter* : contient les classes *Presenter* qui réalisent les traitements liés aux services de chaque form et des *userControls*.

VI- Procédure d'installation

Une notice d'utilisation est présente en annexe. Elle décrit la procédure d'installation et le fonctionnement de l'application Pook.

VII- Liste des procédures de test

a - Projet ModelTests

La seule méthode du projet *Model* est *Decrire*. Elle permet de renvoyer un tableau avec les informations des albums à afficher à l'utilisateur. Nous choisissons un album (*Kirikou et la Sorcière*) et écrivons en dur ses attributs dans des variables string. Nous les rentrons ensuite dans un tableau, et comparons ce tableau au tableau que ressort la méthode *Decrire* appelée sur l'album.

b - Projet DatabaseAccessTests

TestRepository

Nous enlevons le contenu des tables de la base de données avec la méthode *ResetSchema*. La méthode *Session.Clear* appelée à l'intérieur ne supprimant pas la table *album*, nous avons été obligé d'exécuter un script SQL "drop table". La méthode *InitDB* exécute 15 scripts d'ajout de lignes dans l'ensemble des tables de la base de données. Nous utilisons ces nouvelles données pour tester les méthodes des classes *Repository* du projet *DatabaseAccess*.

AlbumRepository

- *GetAll* : nous rentrons la liste des albums de la base de données dans une variable *album*. Nous récupérons les noms de chacun d'eux et les comparons à une liste des noms rentrés des albums de la classe *TestRepository* en dur.
- *GetMesAlbums* : nous faisons la même chose que pour la méthode *GetAll*. Cependant, comme le script HQL de la méthode *GetMesAlbums* renvoie une liste d'*Objet* et non d'*Album*, nous ajoutons une étape de parcours de la liste d'objets obtenus et forçons leur conversion en type *Album*.
- *GetMesSouhaits* : nous utilisons la même démarche que pour *GetMesAlbums* avec la méthode *GetMesSouhaits*.
- *Save* : nous créons l'album "Les Fourberies de Scapin" en dur et l'ajoutons à la base de données avec la méthode *Save*. Nous récupérons les albums de la base de données avec la méthode *GetAll* et sauvons dans une variable les albums récupérés qui ont le nom "Les Fourberies de Scapin". Nous vérifions que nous n'en ayons qu'un (cet album n'existait pas dans la base de données avant son ajout avec *Save*) et comparons chacun des attributs de l'album récupéré avec la valeur des attributs attendues, rentrées en dur.
- *SaveMesAlbums* : nous faisons la même chose qu'avec la méthode *Save* avec la méthode *SaveMesAlbums* en ajoutant l'étape de conversion des types *Objet* récupérés en type *Album*, comme décrit plus haut.
- *SaveMesSouhaits* : idem que pour *SaveMesAlbums* avec la méthode *SaveMesSouhaits*.
- *SupprimerMesSouhaits* : nous supprimons l'album "Martine" de la liste des souhaits de l'utilisateur "louloulala" rentrés dans la base de données avec la classe *TestRepository*. Nous récupérons ensuite les albums de la liste des souhaits pour ce même utilisateur, les convertons du type *Objet* en type *Album*, et gardons seulement ceux qui ont comme nom "Martine". Nous vérifions que la liste obtenue est de taille zéro (l'utilisateur "louloulala" n'avait qu'une fois l'album "Martine" dans sa liste de souhaits).

VIII- Répartition des tâches et planning détaillé

Nous avons pu réaliser une projet partie du projet en "présentiel", nous permettant de confronter nos idées et de facilement se répartir les tâches à réaliser. Lorsque nous étions en présentiel, nous travaillions régulièrement sur un seul ordinateur pour éviter les conflits via Github. Pendant les vacances, nous avons partagé notre travail via github et effectué des réunions sur Zoom.

Répartition des tâches		
Tâche	Cloé	Louis
Diagramme des classes & modèle relationnel	x	x
Maquettage de l'interface IHM	x	
Préparation et passation des tests utilisateurs	x	x
Création du projet App/Dal/Domain	x	x
Création des fichiers MySQL	x	
Codage de la classe Administrateur, Albums et Utilisateur	x	x
Création et code des userControls, des form d'accueil, de connexion et de création de compte	x	x
Création et code du form administrateur et d'ajout d'un album	x	x
Création du modèle MVP		x
Procédure des tests		x
Rédaction du rapport	x	x

Figure n°11: Tableau de répartition

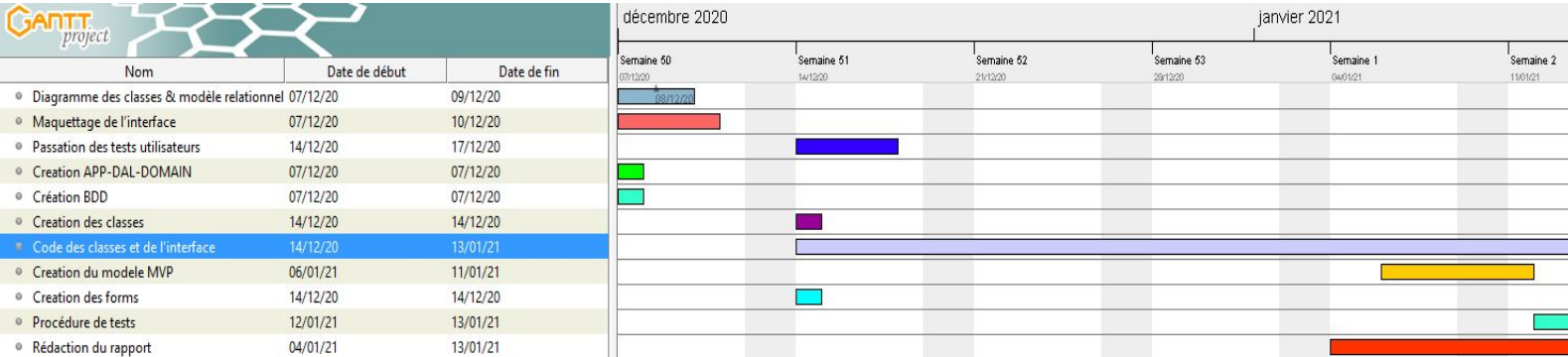


Figure n°12: Diagramme de gantt

IX- Bilan et perspectives du projet

Pour conclure, ce projet nous a permis de monter en compétences sur le langage C# et plus particulièrement en programmation événementielle. Des notions que nous avons vu en cours magistraux et appliquées en travaux dirigés ont pu prendre du sens dans ce projet, plus proche de ce que nous pourrions être amené à développer plus tard. En particulier, voulant proposer un affichage des albums autrement qu'avec une `dataGridView`, nous avons appris à générer des controls en fonction du contenu de la base de données et à s'éloigner de l'utilisation de l'interface graphique proposée par Visual Studio. Ayant pour ambition de découvrir une autre architecture que celle vue en cours, nous avons aussi découvert l'approche MVP des applications winforms, nous apprenant à mieux séparer les responsabilités parmi les classes de la solution et découvert une utilité concrète des interfaces. Enfin, voulant proposer une interface UX, nous avons pu concrètement mettre en application nos notions de Conception Centrée Utilisateur en partant des maquettes et en arrivant au produit final. Cette volonté nous a aussi permis de nous familiariser avec des pattern de design graphique pour proposer une application la plus esthétique possible.

Pour les améliorations du projet, nous aurions envisagé de proposer d'accéder à une fiche technique plus détaillée des albums (en cliquant dessus) comprenant par exemple leur synopsis et l'histoire/anecdotes autour de leur création et de leur vie (qui pourraient être ajoutés par les utilisateurs). Aussi, nous aurions voulu créer une classe *Auteur*, et ajouté un onglet pour parcourir l'ensemble des auteurs de l'application et leurs albums. Enfin, nous aurions aimé hébergé la base de données en ligne de l'application pour permettre aux potentiels utilisateurs de s'auto-enrichir leur application.