GestBiblio

Gestionnaire de Bibliothèque

Esseddik Ismaël | Rapport de projet Informatique
Master 1 Géomatique| JUIN 2017
Enseignant – C.Delgrange

ENSG Géomatique
ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES

UPEM
UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

Table des matières

Avant-propos

Ayant une connaissance du fonctionnement administratif d'une bibliothèque et travaillant en bibliothèque universitaire depuis deux ans, j'ai donc choisi de faire un gestionnaire destiné à un établissement d'enseignement dans le cadre du projet informatique de mon master 1. J'ai notamment fait usage d'un logiciel du nom de « Absysnet » : un S.I.G.B.[1] sous licence non libre intégralement développé en Web natif, qui m'a en grande partie inspirée pour l'élaboration de l'application. J'étais simplement curieux de savoir comment l'interaction se faisait entre une interface client et une base de données. Pour comprendre de quelle manière la circulation des documents se faisaient au sein de la base de données. Il existe déjà plusieurs s.i.g.b. open source dont quelques-uns développés en multiplateforme, mais aucun à ma connaissance n'utilise le langage python, mon objectif n'était pas de réinventer la roue, mais surtout d'expérimenter le langage python et le paradigme POO[2]. Sans rentrer dans les détails de la bibliothéconomie, ce rapport rappelle d'abord brièvement le fonctionnement général d'une bibliothèque, il décrit ensuite le déroulement de mes travaux sur l'application et se termine par des perspectives d'améliorations.

A/ Fonctionnement générale d'un S.I.G.B.

1. LE SYSTEME DE GESTION INTEGRE DE BIBLIOTHEQUE

Il est au moins constitué d'une base de données relationnelles, d'un logiciel d'interaction avec cette base de données et d'interfaces d'usages (publique, réservé ou purement administrative)[3].

2. DOCUMENTS ET PLAN DE CLASSEMENT

Toute édition de document est associée à un ISBN[4], il fait référence au titre, a au nom de l'auteur, a l'édition et aux autres informations d'un ouvrage. C'est un numéro à 10 ou 13 chiffres réservé chaque Edition de livre. Etant unique, il permettra d'avoir la même information pour chaque exemplaire d'un même livre, on l'utilisera de sorte à éviter la duplication inutile de l'information dans la base de données. Ainsi on peut avoir toutes les informations d'un document via la simple entrée du code ISBN.

Cependant, pour gérer les exemplaires on fera usage d'un autre identifiant pour discerner chaque exemplaire d'un même livre : un code bar quelconque.

Les bibliothèques disposent de rayons organisés en différent pôles avec chacun plusieurs champs de connaissances, il existe différente manières de coter les ouvrages, la cotation Dewey[5] est le plus souvent utilisé en occident.

[1] Système de Gestion Intégré de Bibliothèques.
[2] Programmation orienté objet.
[3] Voir Figure 1 à 4 & 6 à 8 en annexe.
[4] International Standard Book Number.
[5] Voir figure 8 en annexe

3. CARACTERISTIQUES D'UN LECTEUR

Un lecteur a un N° Etudiant qui sert d'identifiant unique, ainsi que diverses informations relatives à son identité. Nom, Prénom, Date de naissance, Cycle, un Type de lecteur, N° Téléphone, un état (suspendu ou non), un commentaire. On considèrera ici le lecteur comme individu inscrit et dont les informations sont contenues dans la base de données[6].

4. REGLEMENT DE CIRCULATION ET CONDITIONS DE PRETS

Voici une liste non exhaustive de règlements que l'on peut trouver dans certaines bibliothèques, ces dernières servent à faire en sorte que les documents de soit pas monopolisé par les mêmes emprunteurs et le gestionnaire est réglé pour les faire respecter au moment des saisies informatiques.

- Tout livre a droit à un seul et unique prolongement par emprunt.
- Tout livre déposé ne peut être réemprunté avant le lendemain.
- Emprunter un livre déjà emprunté par un autre lecteur est impossible.
- La remise d'un livre non emprunté affiche un message du type : « livre non emprunté ».
- Un lecteur ayant un retard sur ses remises sera suspendu pour une durée proportionnelle à son retard. *(Exemple : un livre rendu le 5 décembre qui aurait dû être rendu avant le 1er décembre suspend son lecteur du 1er au 09 décembre et pourra réemprunter le 10 décembre).*
- Les retards ne sont pas cumulables : chaque livre a sa période de suspension et c'est le plus grand retard qui est pris en compte pour définir la date de suspension.
- Les conditions de prêt varient selon les critères du lecteur :

	Condition de prêt	
	Nombre de documents autorisés	Durée du prêt
1er cycle/Licence/DU	4 documents	2 semaines + prolongement 2 semaine
Master	6 documents	3 semaines + prolongement 2 semaine
Doctorants	8 documents	4 semaines + prolongement 2 semaine
Externes		

B/Développement de l'application

1. OBJECTIF DE L'APPLICATION

Le but de mon projet était de programmer une partie du s.i.g.b.: Le gestionnaire administrateur. Ce dernier sera composé d'une base de données contenant :

- Les lecteurs inscrits ainsi que leurs statuts (lecteur interne/externe, coordonnées, information administratives etc.).

[6] Voir figure 10 en annexes

- Les documents et leur statuts (emprunté ou non, en rayon ou en magasin, quelle pôle, quelle côte, nombre de pages, type=livre/périodique/mémoire/cd ?

Ensuite le gestionnaire en interface graphique qui permettra d'exécuter toutes les fonctionnalités : rechercher, modifier, intégrer et retirer des informations de la base de données, mais aussi de faire des emprunts et des retours d'exemplaires dans le règlement générale énoncé dans la partie précédente.

2. PLANIFICATION DU PROJET

Le projet s'est globalement planifié dans la logique de la hiérarchie des objets : pas d'interface graphique sans fonctionnalités et pas de fonctionnalités sans base de données, j'ai donc d'abord fait l'intégralité de mes fonctionnalités avant de pouvoir les intégrer dans des objets en tant que méthodes puis en les reliant à d'autres objets de type interface graphique.

3. HIERARCHIE DES FONCTIONNALITES DE GESTBIBLIO

Au démarrage de l'application se fait une initialisation de la base de données, elle consiste d'une part en sa création si elle n'existe pas ainsi qu'une vérification des retards et d'une mise à jour des pénalités de suspension qui en découle. Ceci étant fait, l'interface graphique s'ouvre sur un écran d'accueil qui pourrait a posteriori servir d'identification de compte administrateur (mais pour le moment non actif).

On accède aux fonctionnalités via un menu qui représente le découpage du code en différentes classes. Chaque classe contient des méthodes et appartiennent à une table de la base de données, on aura alors par exemple un bouton gestionnaire d'exemplaire qui exécutera par ses méthodes (en majorité) des traitements sur la table exemplaire de la base de donnée. Autres exemples, les boutons emprunt et retour d'exemplaire concerne la relation entre lecteur et exemplaire, par conséquent ils font référence à la même table intitulée « relations ».

4. PROBLEMES RENCONTRE ET SOLUTION DEVELOPPEE

La grande difficulté pour moi a été de trouver une bonne organisation des objets en appliquant les notions d'héritages de classe et d'encapsulation. La configuration de départ qui voulait implémenter les objets les uns dans les autres a été un échec et m'a énormément retardé, j'ai aussi voulu distinguer les objets type interfaces graphiques de ceux qui interagissent avec la base de données, mais compte tenu du temps restant j'ai fini par fusionner ces deux tout en laissant une branche[7] associé au tests de fonctionnalités, en cas de « backup » nécessaire. Ainsi l'architecture a été revu de nombreuses fois, ce qui au final réduit le caractère orienté objet et se rapproche plus d'une programmation fonctionnelle et simplifiée.

Autre problème : En principe les acquisiteurs d'ouvrages disposent au préalable des informations concernant un ouvrage dans une base de donnée, il m'a fallu trouver une solution pour récupérer ces informations au moyen d'une API : celle de Googlebooks.

[7] Embranchement de GitHub, pour les travaux de groupes en parallèles.

C/Perspective d'améliorations

Pour conclure, les fonctionnalités sont présentes, mais pas toutes, en effet il faudrait pouvoir empêcher l'emprunt consécutif d'un même livre, implémenter un catalogue de recherche par champ… par ailleurs peut-être revoir le contenu ou la forme de la base de données relationnelle, certainement améliorer l'ergonomie de l'interface graphique, notamment via des écouteurs d'évènements complètement absent du script (en dehors de ceux intégrés aux boutons) ou encore permettre de prolonger. On pourrait aussi revoir la configuration orientée objet pour offrir de meilleure possibilités de reprendre les travaux. Finalement, cela dépendra des objectifs que l'on souhaite atteindre, il serait aussi intéressant de pouvoir intégrer à tout cela une composante spatiale qui, au moyen des nouvelles puces Rfid remplaçant les codes-barres dans certaines bibliothèques, permettrait un meilleur suivi de la circulation et de la gestion des inventaires.

Annexes

DIAGRAMMES UML

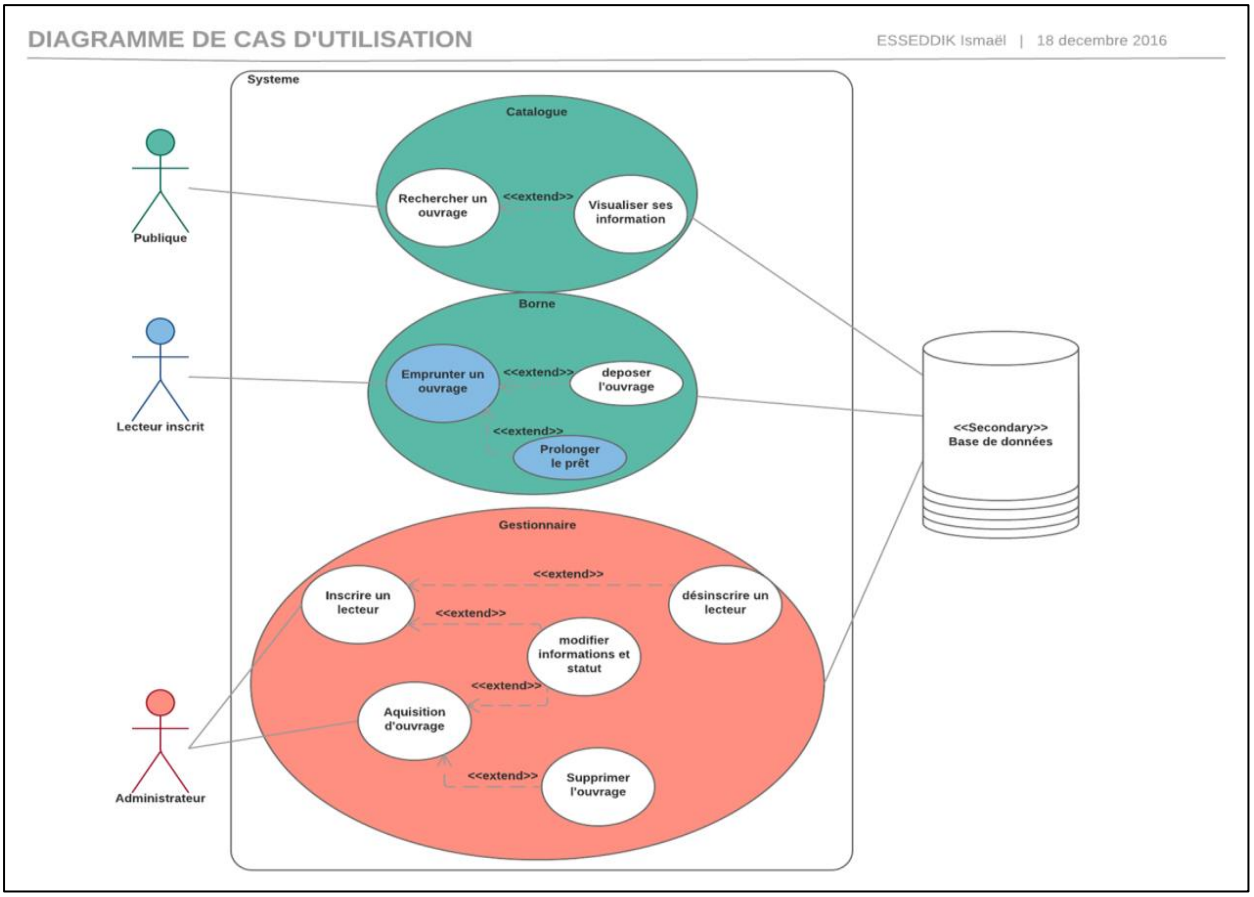

*Figure 1 – On distingue 3 type d'utilisateurs : le public, les lecteurs ayant droits inscrit dans la base de données et les administrateurs. Ainsi on représentera les droits tout publics en **vert**, les droits restreint aux lecteurs inscrit en **bleu** et les droits d'administrateurs encore plus restreints en **rouge**.*

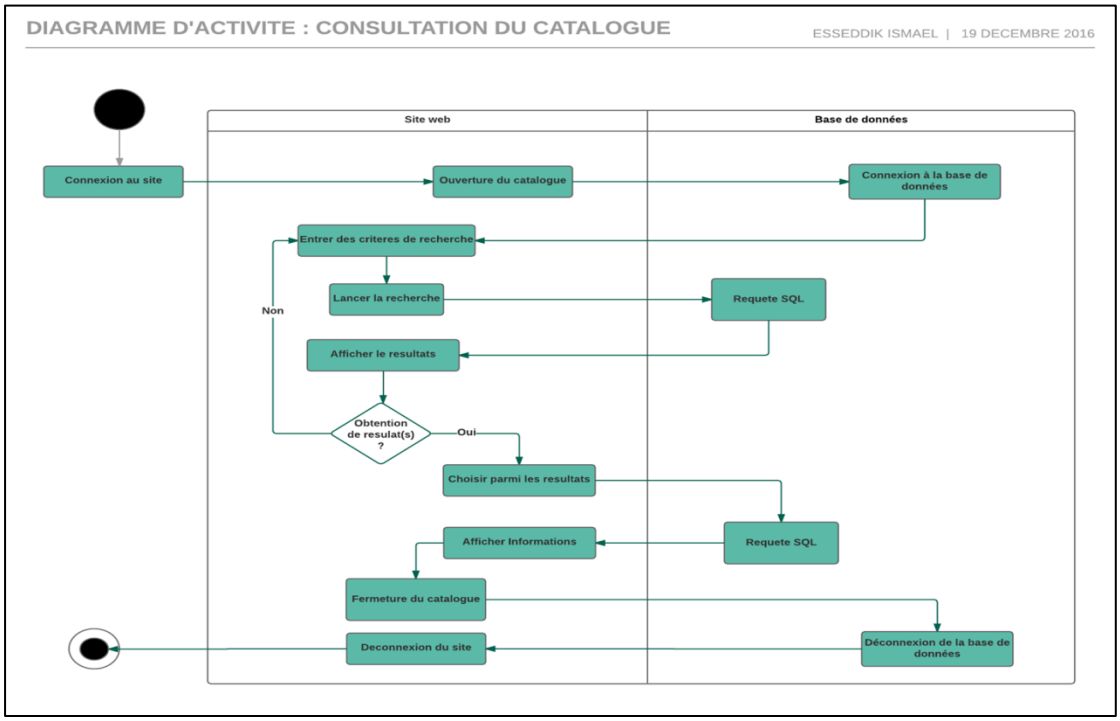

Figure 2 – Catalogue de consultation (public)

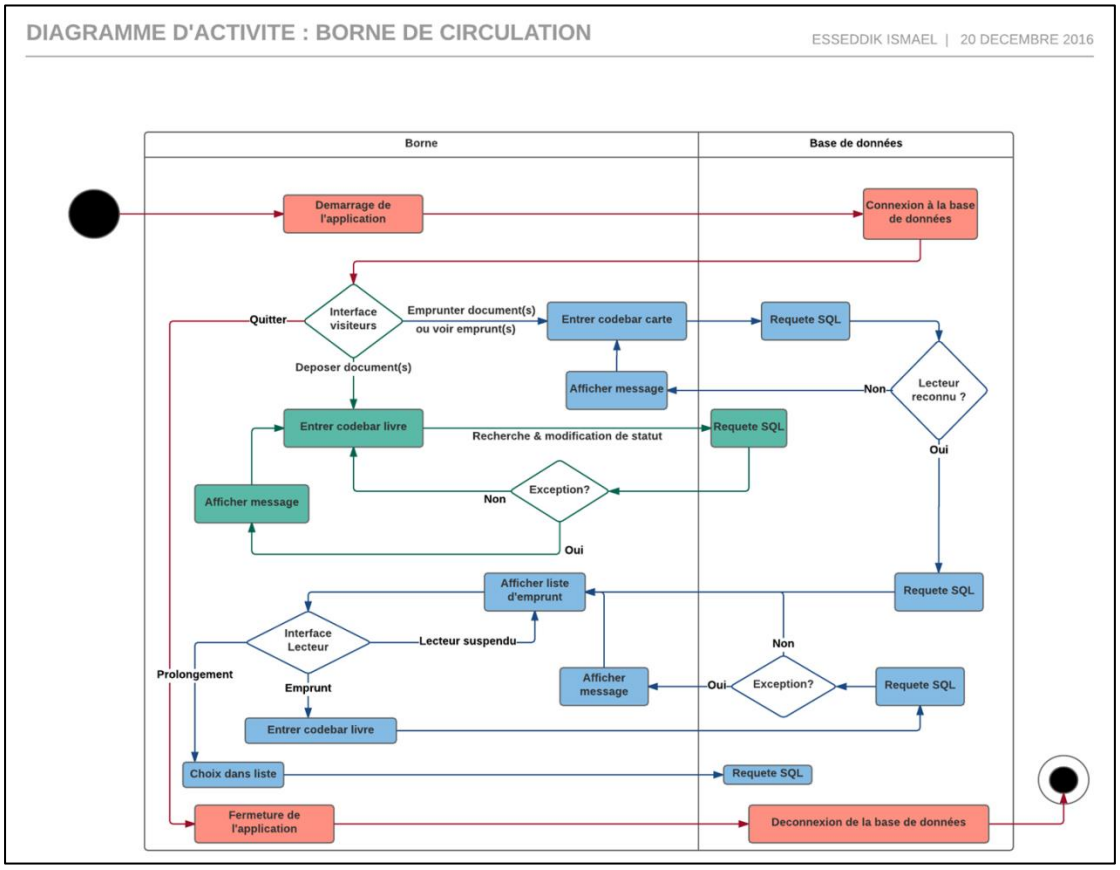

Figure 3 – C'est le rôle de l'administrateur de démarrer et d'arrêter la borne, n'importe qui peut venir déposer un livre mais pour emprunter il faudra s'identifier.

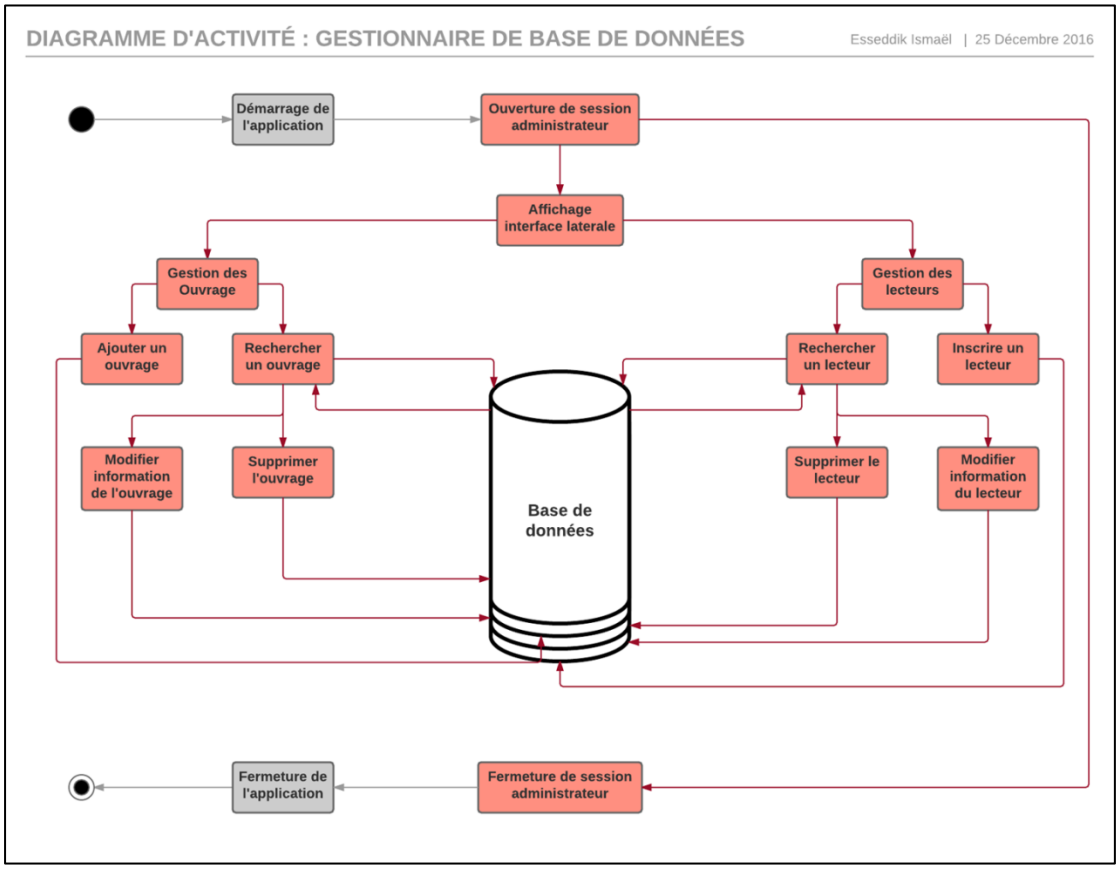

Figure 4 – Les administrateurs sont en charge de gérer la base de donnée, par conséquent ils ont accès à toute les fonctionnalités.

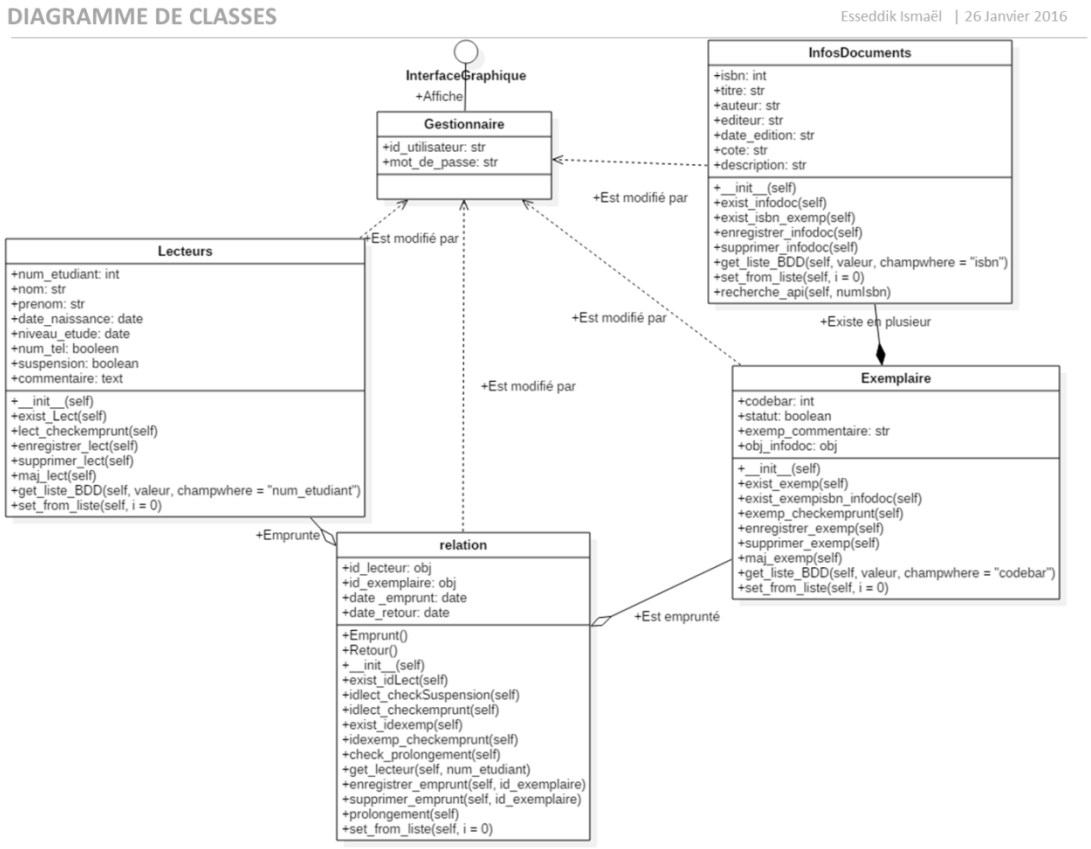

Figure 5 – Les fonctionnalités de classes résumées en uml

MAQUETTES D'INTERFACES GRAPHIQUES

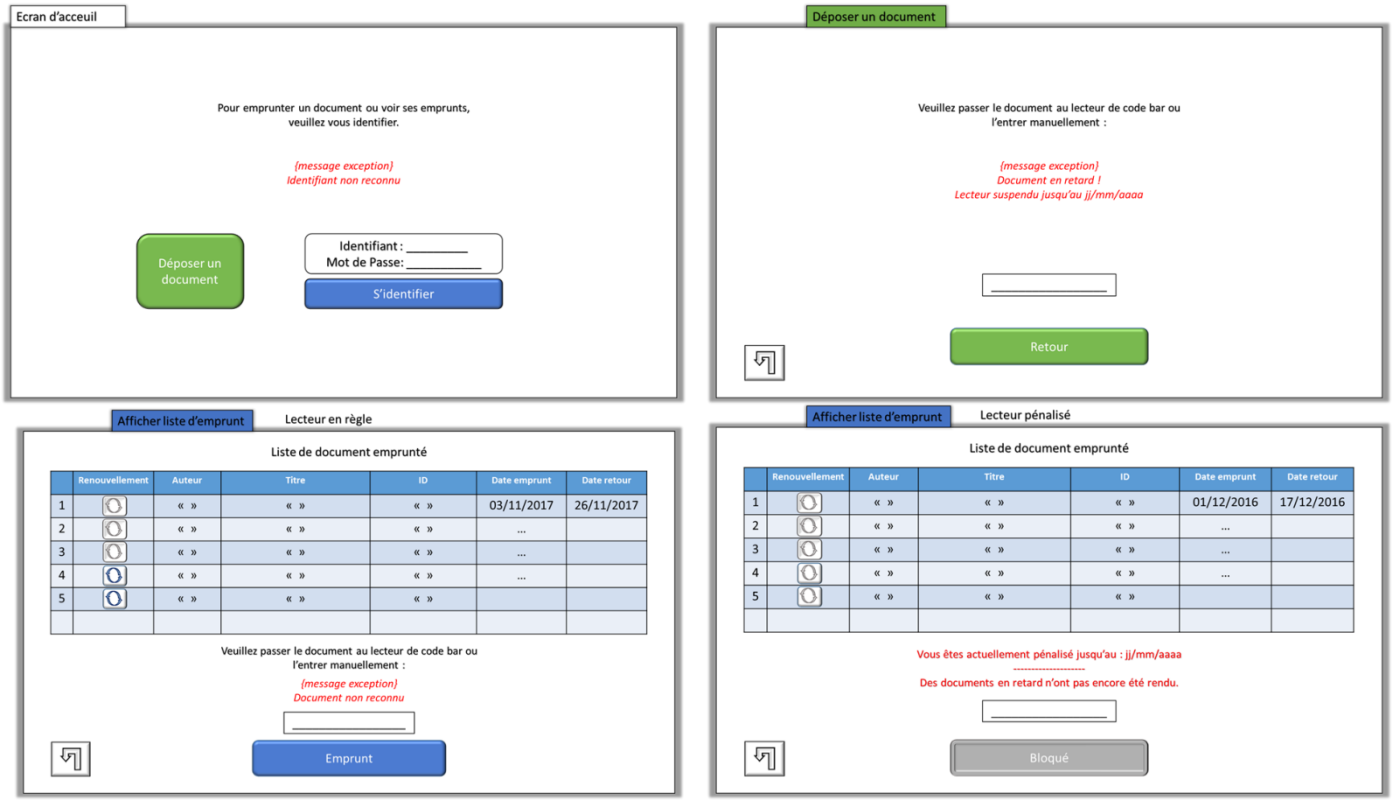

Figure 6 - Interface possible d'une borne d'emprunt.

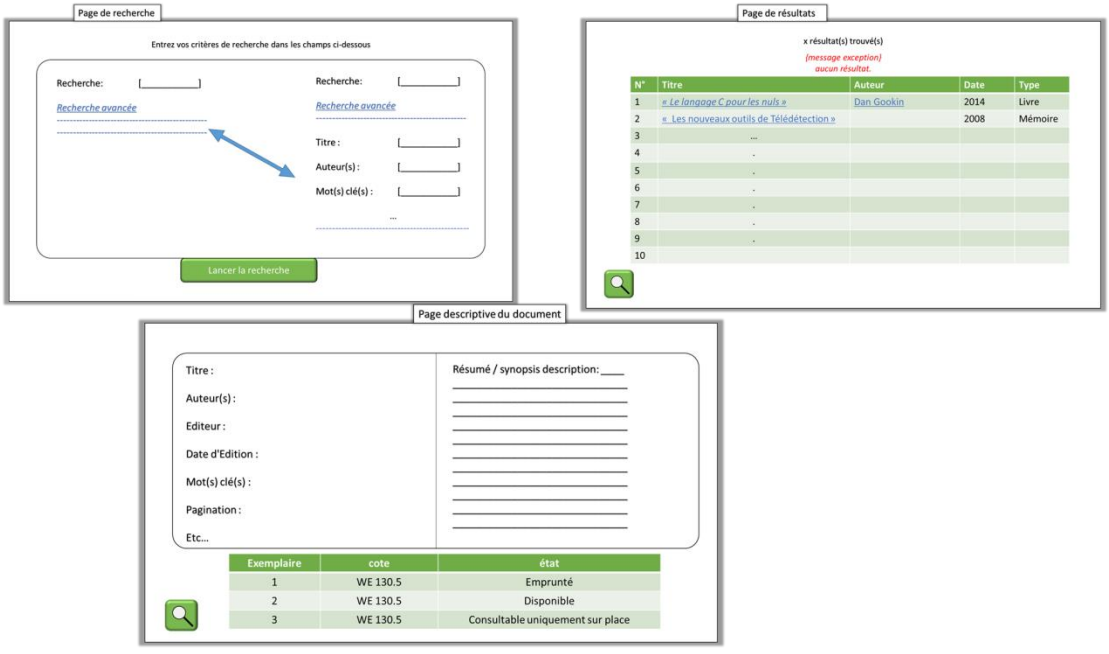

Figure 7 – Interface possible d'un catalogue en ligne.

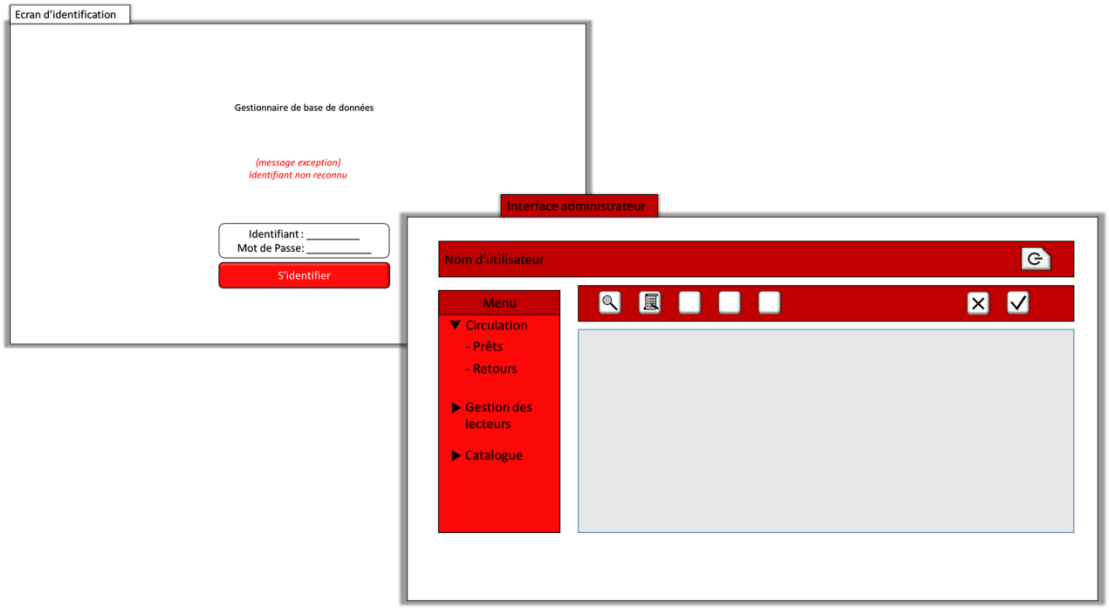

Figure 8 – Interface possible d'un gestionnaire de bibliothèque.

GESTION D'INVENTAIRE

Pôle	Rayon	Côte
Généralité	Philosophie	100-109
	Langue	401-491
	Informatique	004-006.1
	Littérature	800-890
	Périodiques	A-z
Sciences et techniques	Mathématique	510-519
	Physique	530-539
	Chimie	540-548
	Biologie	570-579
	Périodiques	
Sciences Humaines et Sociales	Psychologie	150-158
	Sociologie	300-307
	Education	370-379
Droit économie gestion	Science Politique	320-327
	Economie	330-339
	Droit	340-349
	Gestion - Comptabilité	650-659
Mémoires	???	???

Archives			
Périodiques			
Ouvrage			

Figure 9 – Cotation Dewey.

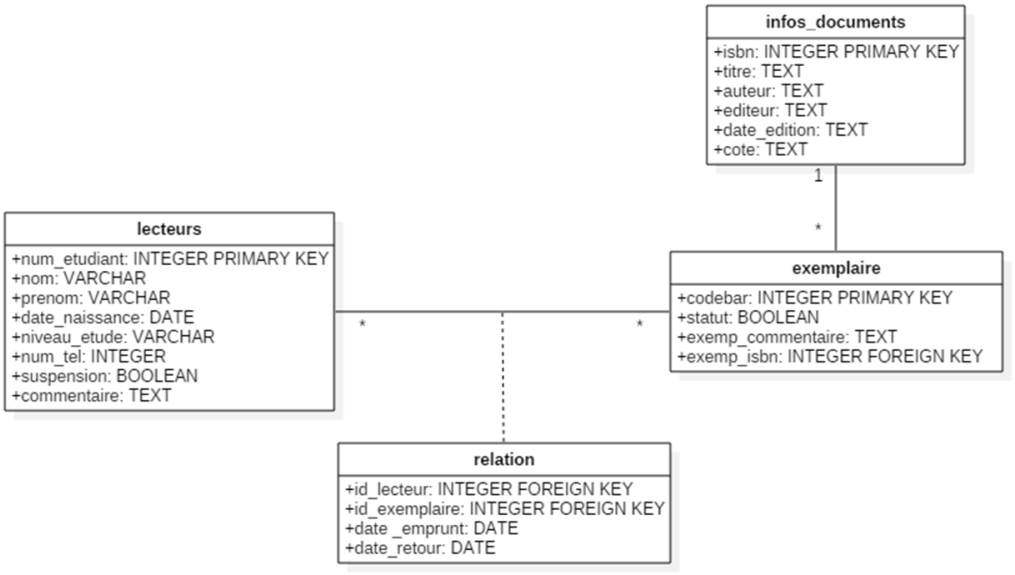

Figure 10 – Le contenu de la base de données est basée sur ce schéma.

OUTILS A DISPOSITION

- ASUS Transformer Book T100HAN (Windows 10).
- Microsoft Office pour la rédaction de documents.
- Lucid chart & StarUML pour l'U.M.L.
- IDLE Python Atom & Pycharm pour la programmation.
- GitHUB pour l'hébergement et la gestion suivie du projet.
- DB Browse for SQLite pour vérifier les opérations faites sur la base de données.

ALGORITHMES ET LIBRAIRIES PYTHON.

-Isbnlib : Traitement des données sur les éditions d'ouvrages.

Formatage isbn :

is_isbn13 / is_isbn10 : Relève la série de 10 ou 13 chiffres présente dans la chaine de caractère et vérifie sa validité.

to_isbn10 / to_isbn13 : Transforme la série de chiffre en variante 10 ou 13 caractères.

Réception de donnée API :

meta : renvoie les métadonnée du livre sous forme de tableau.

desc : renvoie une description (synopsis) du livre.

cover : renvoi dans un tableau une ou plusieurs vignette de la page de couverture (non utilisé).

-Tkinter : Traitement des interfaces Graphiques.
-re : Traitement des expressions régulières.
-Datetime : Traitement des dates et intervalles de temps.
-Sqlite3 : Traitement de requêtes sur base de données :

Parmi les fonctions en provenance de la vue « Interface graphique » certaines doivent faire appel à des contrôleurs qui vont par la vérification de conditions autoriser ou non l'exécution de requêtes sur la base de données. Voici quelques exemples :

1) Requête sur Infos_documents (Clé : ISBN) :

L'isbn est la base de toute l'information, pas de exemplaire sans isbn, pas d'emprunt ni de retour sans exemplaire.

- Pour **Rechercher un isbn** via une requête sql, pas de condition à vérifier, retourne un résultat ou «none » si inexistant

- Pour **enregistrer l'isbn**, 1 condition à vérifier :
    1. Si l'exemplaire existe déjà dans sa table ou est invalide : afficher « isbn déjà existant ou invalide»
    - Sinon : requête sql pour l'ajouter lui et ses données. (Nouvelle entrée)
- Pour **supprimer un isbn**, 2 conditions à vérifier :
    1. S'il n'existe pas : afficher «isbn inexistant »
    2. Si un exemplaire lui est associé : afficher « isbn associé à un ou plusieurs ouvrage»
    - Sinon : requête sql pour le supprimer lui et ses données

---

2) Requête sur Exemplaires (clé : Codebar) :

Il est impératif que l'isbn d'un exemplaire soit existant dans la base de données (dans la table info_documents) avant même que l'exemplaire associé n'y soit intégrer.

- Pour **Rechercher un codebar** via une requête sql pas de condition à vérifier, retourne un résultat ou «none » si rien.

- Pour **ajouter un exemplaire**, 2 conditions
    2. Si l'exemplaire existe déjà : afficher « exemplaire déjà existant »
    3. Si l'isbn n'existe pas dans la table info_doc : afficher « isbn non trouvé »
    - Sinon : requête sql pour l'ajouter lui et ses données. (Nouvelle entrée)
- Pour **supprimer un exemplaire**, 2 conditions à vérifier
    1. S'il n'existe pas : afficher « exemplaire inexistant »
    2. S'il est emprunté : afficher «l'exemplaire non rendu »
    - Sinon : requête sql pour le supprimer lui est ses données

---

3) Requêtes sur Lecteurs (clé : Num_etudiant) :

Il est impératif qu'un lecteur ait rendu tous ses livres avant de pouvoir être désinscrit de la base de données.

- Pour **Rechercher un numéro étudiant** via une requête sql, pas de condition à vérifier, retourne un résultat ou «none » si rien.
- Pour **enregistrer un numéro étudiant**, une seule condition
    1. Si le numéro étudiant existe déjà : afficher « Lecteur déjà inscrit »
    - Sinon : requête sql pour l'ajouter lui est ses données. (Nouvelle entrée)
- Pour **supprimer un numéro étudiant**, 2 conditions à vérifier
    1. S'il n'existe pas : afficher « Lecteur inexistant »
    2. S'il des exemplaires non rendu (présence dans la table relation): afficher «l'exemplaires non rendu »
    - Sinon : requête sql pour le supprimer lui est ses données

---

4) <u>Requêtes sur Relations (clé : codebar) :</u>

- Pour faire **emprunter un exemplaire**, 3 conditions à vérifier sur l'emprunteur :
    1. Si le lecteur n'existe pas : afficher « Lecteur inexistant »
    2. S'il est suspendu : afficher «Lecteur suspendu »
    3. S'il a atteint sa limite d'emprunt : afficher « limite d'emprunt atteinte »
    - Sinon : 2 sous condition à vérifier
        1. Si l'exemplaire a emprunté n'existe pas : afficher « document non identifié »
        2. S'il est déjà emprunté : afficher « exemplaire déjà emprunté
           Sinon : requête sql pour changer le statut de l'exemplaire & entrer une nouvelle entrée dans la table relation.
- Pour faire **retourner un exemplaire**, 2 conditions à vérifier :
    1. Si l'exemplaire n'existe pas : afficher « document non identifié »
    2. S'il n'est pas emprunté : afficher « exemplaire non emprunté »
    - Sinon : Changer le statut de l'exemplaire et retirer le champ associé de la table relation.

RAPPORT D'ACTIVITE DES SEANCES

Disponible à l'adresse suivante :
https://github.com/IsmaEsseddik/GestBiblio_M1_GEOM