

GestBiblio

Gestionnaire de Bibliothèque

Esseddik Ismaël | Rapport de projet Informatique

Master 1 Géomatique| JUIN 2017

Enseignant – C.Delgrange



Table des matières

[Avant-propos 1](#_Toc483842515)

[A/ Fonctionnement générale d’une bibliothèque universitaire 2](#_Toc483842516)

[1. documents et plan de classement 2](#_Toc483842517)

[2. Caractéristiques d’un lecteur, Emprunt et Retour 2](#_Toc483842518)

[3. Règlement de circulation et conditions de prêts 2](#_Toc483842519)

[4. Consultation du catalogue 3](#_Toc483842520)

[B/Développement de l’application 3](#_Toc483842521)

[1. Objectif de l’application 3](#_Toc483842522)

[2. Planification 3](#_Toc483842523)

[3. Hiérarchie des fonctionnalités de GestBiblio 3](#_Toc483842524)

[4. Problèmes rencontré et solution développée 4](#_Toc483842525)

[C/Perspective d’amelioration 5](#_Toc483842526)

[Annexe 6](#_Toc483842527)

[Diagrammes UML 6](#_Toc483842528)

[Outils à disposition 12](#_Toc483842529)

[Algorithmes et librairies python. 12](#_Toc483842530)

[***Isbnlib : Traitement des données sur les éditions d’ouvrages.*** 12](#_Toc483842531)

[***Tkinter : Traitement des interfaces Graphiques*** 12](#_Toc483842532)

[***re : Traitement des expression régulières*** 12](#_Toc483842533)

[***Datetime : Traitement des dates et intervalles de temps*** 12](#_Toc483842534)

[***Sqlite3 : Traitement de requêtes sur base de données*** 12](#_Toc483842535)

[*1)* *Requête sur Infos\_documents (Clé : ISBN)* 12](#_Toc483842536)

[*2)* *Requête sur Exemplaires (clé : Codebar)* 13](#_Toc483842537)

[*3)* *Requêtes sur Lecteurs (clé : Num\_etudiant)* 13](#_Toc483842538)

[*4)* *Requêtes sur Relations (clé : codebar)* 13](#_Toc483842539)

# 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]](#footnote-1) sous licence non libre intégralement développé en Web natif, qui m’a en grande partie inspiré 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 sigb open source dont quelque uns développé 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]](#footnote-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 SIGB

## documents et plan de classement

Toute édition de document est associée à un ISBN[[3]](#footnote-3), il fait référence au Titre, auteur, édition (année) et aux autres informations d’un ouvrage. C’est un numéro à 10 ou 13 chiffre réservé chaque Edition de livre. Etant unique, il permettra d’avoir la même information pour chaque exemplaire identique d’un même livre, on l’utiliser de sorte à éviter la duplication inutile de l’information dans la base de donnée. 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 : le codebar .

Les bibliothèques disposent de rayons organisés en diffèrent pôles avec chacun plusieurs champs de connaissances, il existe différente manière rangé par la cotation Dewey ainsi qu’un espace dédié aux archives.

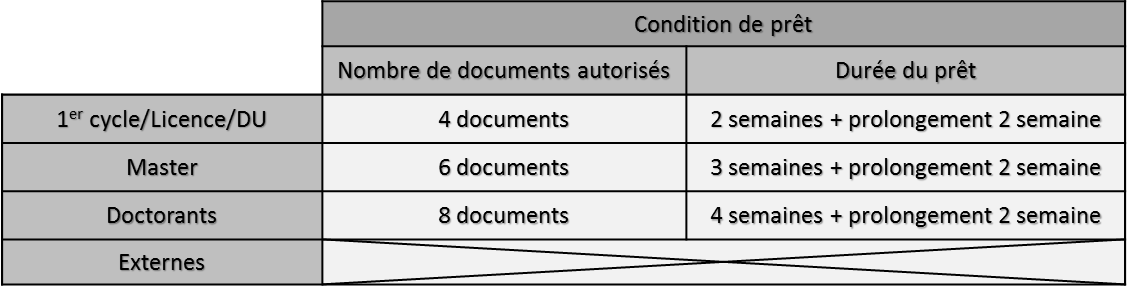
## Caractéristiques d’un lecteur

Un lecteur a un N° Etudiant qui sert d’identifiant unique, ainsi que diverse information relatives à son identité. Nom, Prénom, Date de naissance, Cycle, un Type de lecteur, N° Téléphone, un Etat (suspendu ou non), un Commentaire.

## Règlement de circulation et conditions de prêts

Voici une liste non exhaustive de règlements que l’on peut trouver dans certaine bibliothèque

* Tout livre a le 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 affichera 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.
* Les conditions de prêt varient selon les critères du lecteur :



## Consultation du catalogue

Il s’agit de pouvoir rechercher un document à partir de différents attributs et de consulter leurs informations

# B/Développement de l’application

## Objectif de l’application

Le but de mon projet est de programmer une partie du sigb. : Le gestionnaire administrateur. Ce dernier sera composé de :

Une **base de données** contenant :

* Les lecteurs inscrits ainsi que leurs statuts (lecteur interne/externe, coordonnées, information administratives etc.),
* 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** qui permettra toute les fonctionnalités : rechercher/modifier/intégrer et retirer des informations de la base de données mais aussi de faire des emprunt et des retours d’exemplaire

## Planification

## Hiérarchie des fonctionnalités de GestBiblio

en POO avec python pour la gestion de la base de données par l’administrateur qui permet :

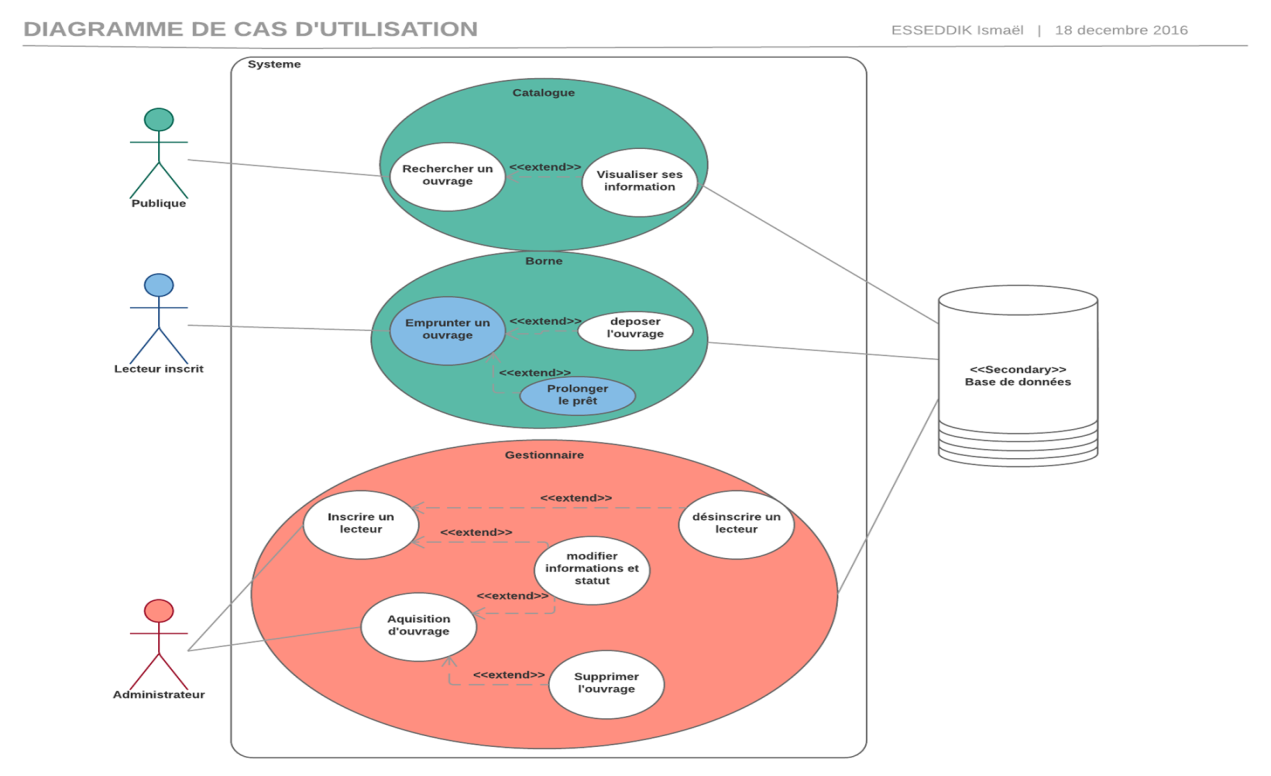
* Gestion des ouvrages (emprunt/retour/pénalité/avertissement).
* Gestion des lecteurs (inscrire/désinscrire un lecteur, modifier ses informations, modifier pénalité de retard).

## Problèmes rencontré et solution développée

# C/Perspective d’améliorations

# 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**.

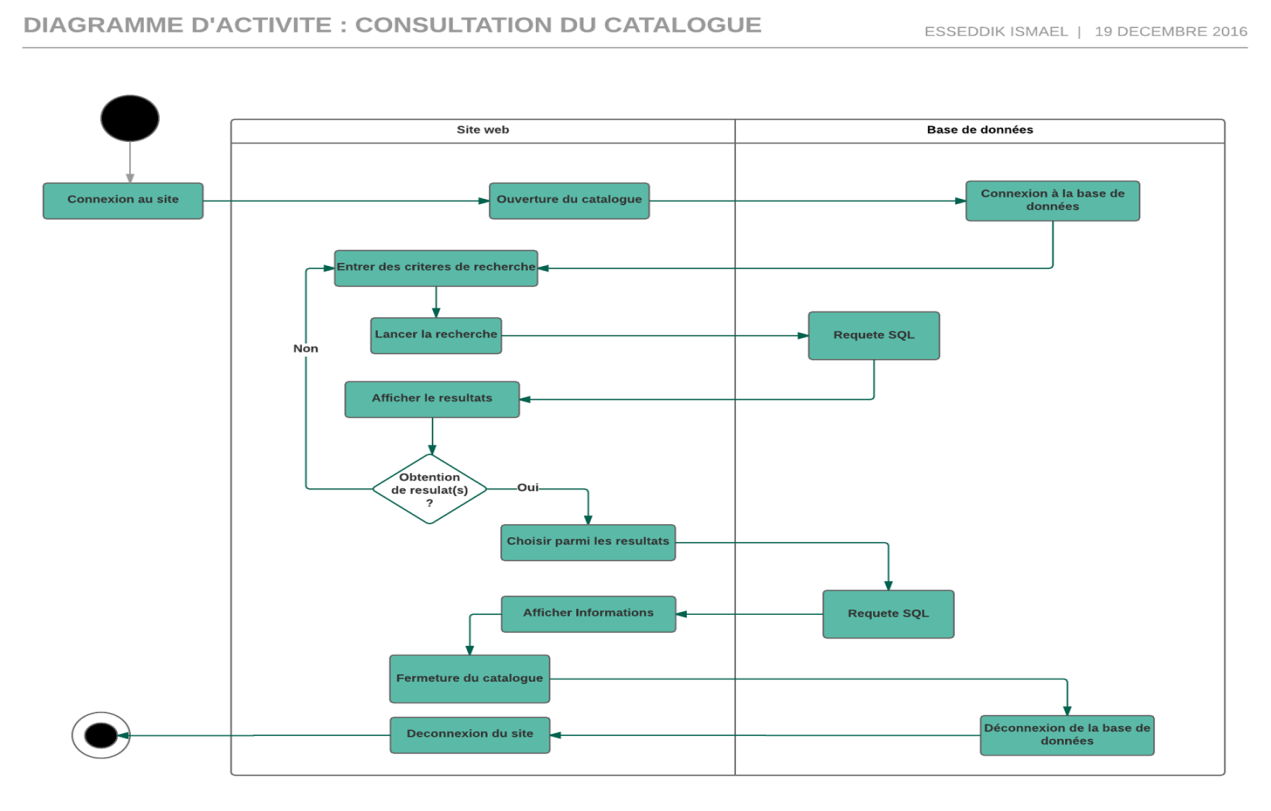


Figure 2 – Catalogue de consultation (public)

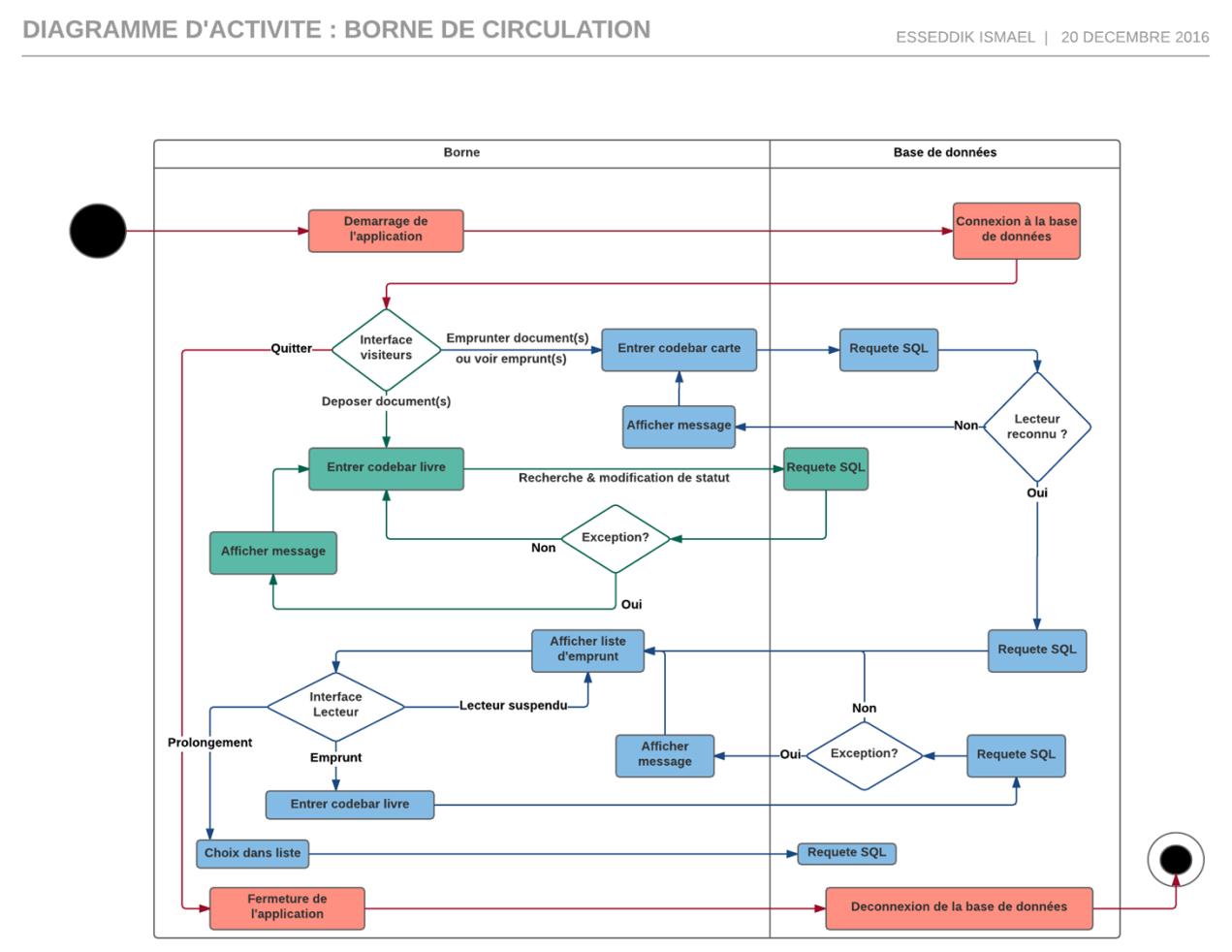


Figure 3 – c’est le rôle de l’administrateur de démarrer et d’arrêter la borne, n’importe qui peut venir déposer un livre mais pour emprunter il faudra s’identifier.

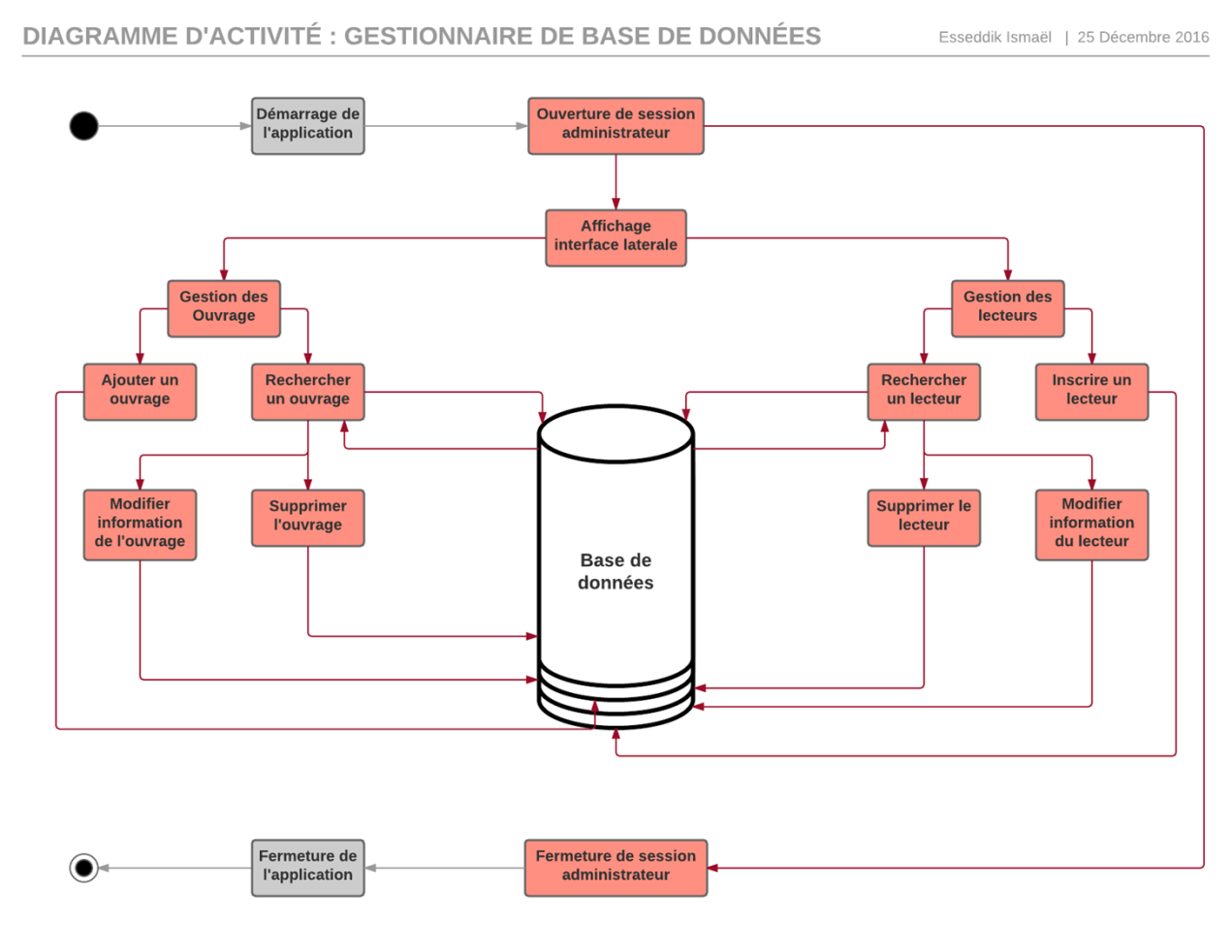


Figure 4 – les administrateurs sont en charge de gérer la base de donnée, par conséquent ils ont accès à toute les fonctionnalités.

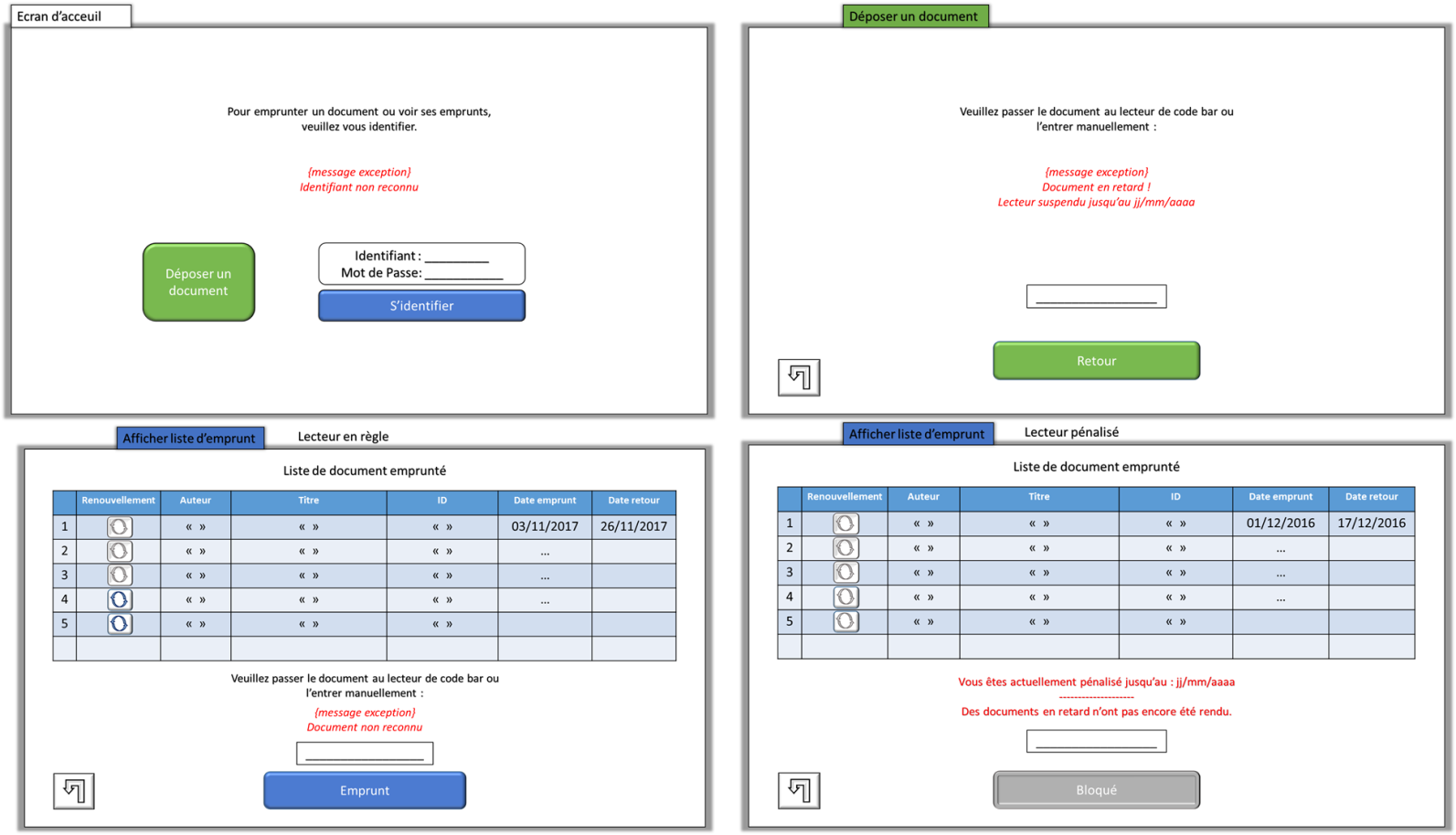


Figure 5 - Interface possible d’une borne d’emprunt

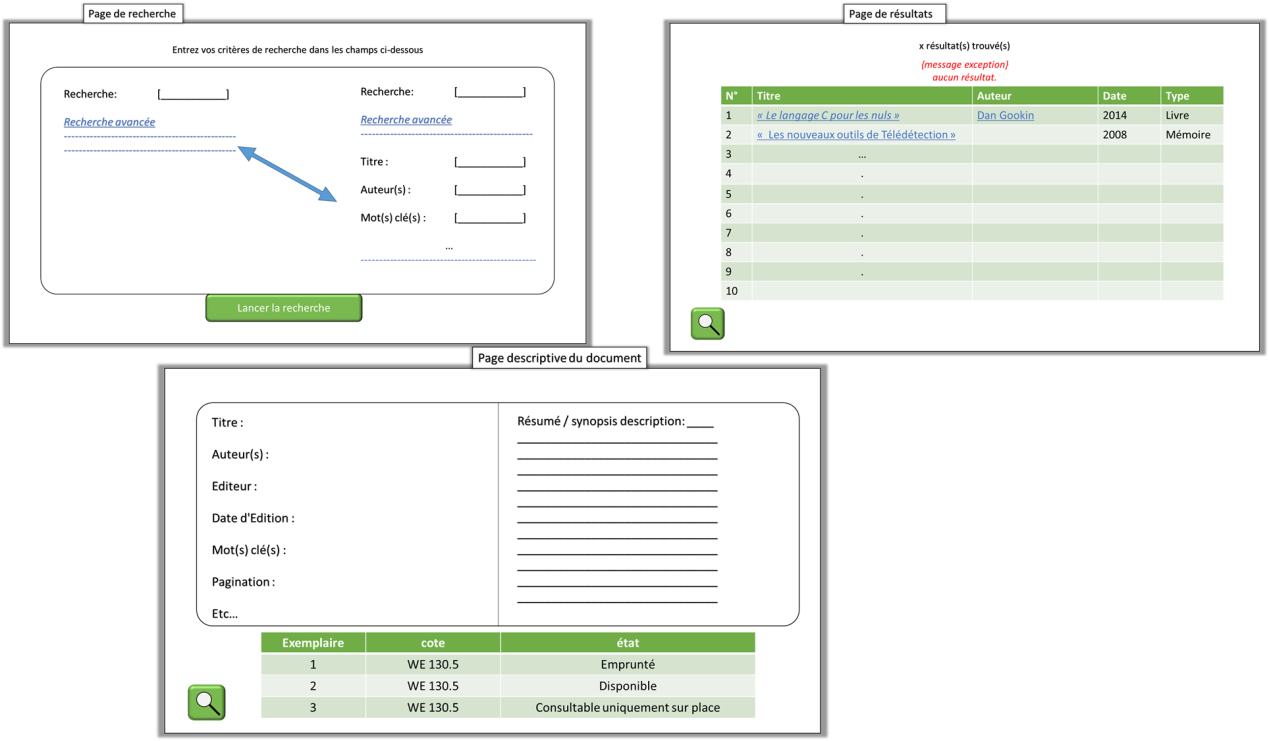


Figure 6 – Interface possible d’un catalogue en ligne

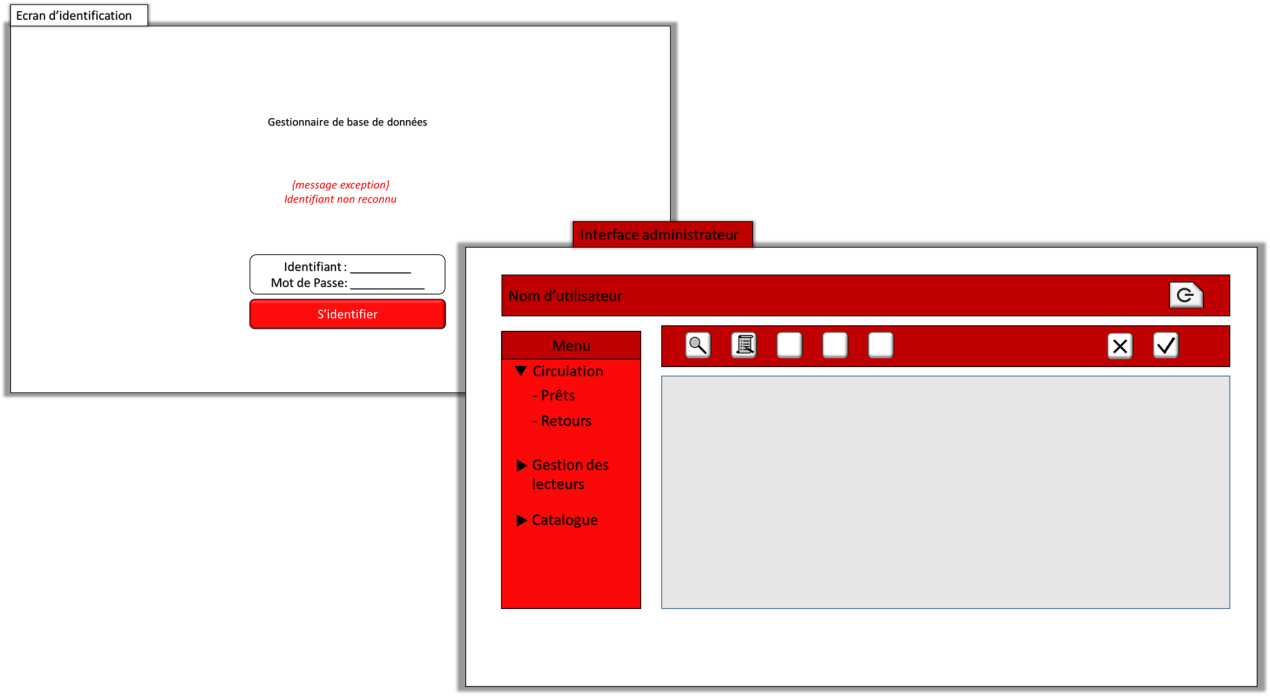


Figure 7 – Interface possible d’un gestionnaire de bibliothèque

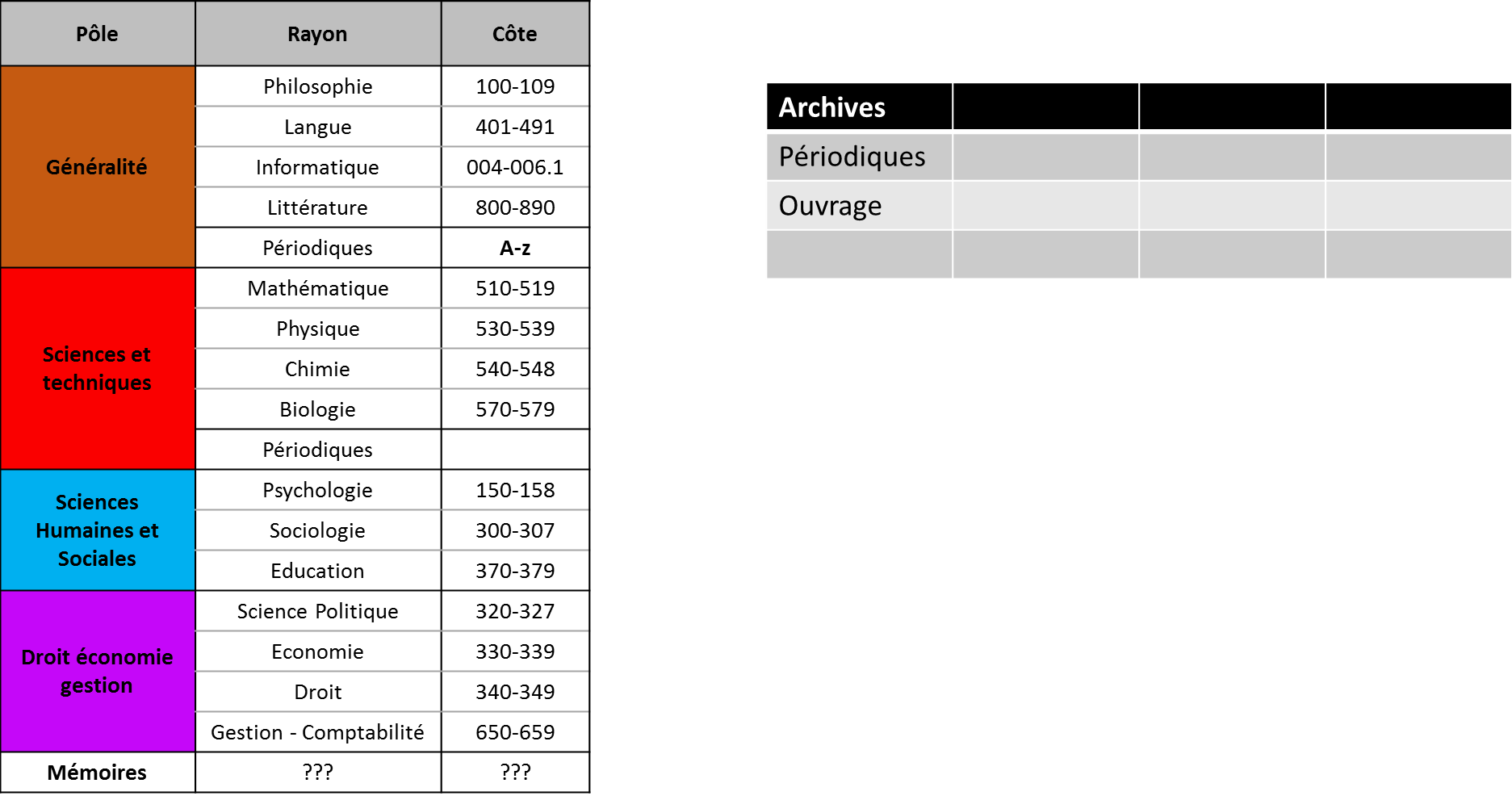


Figure 8 – Cotation Dewey

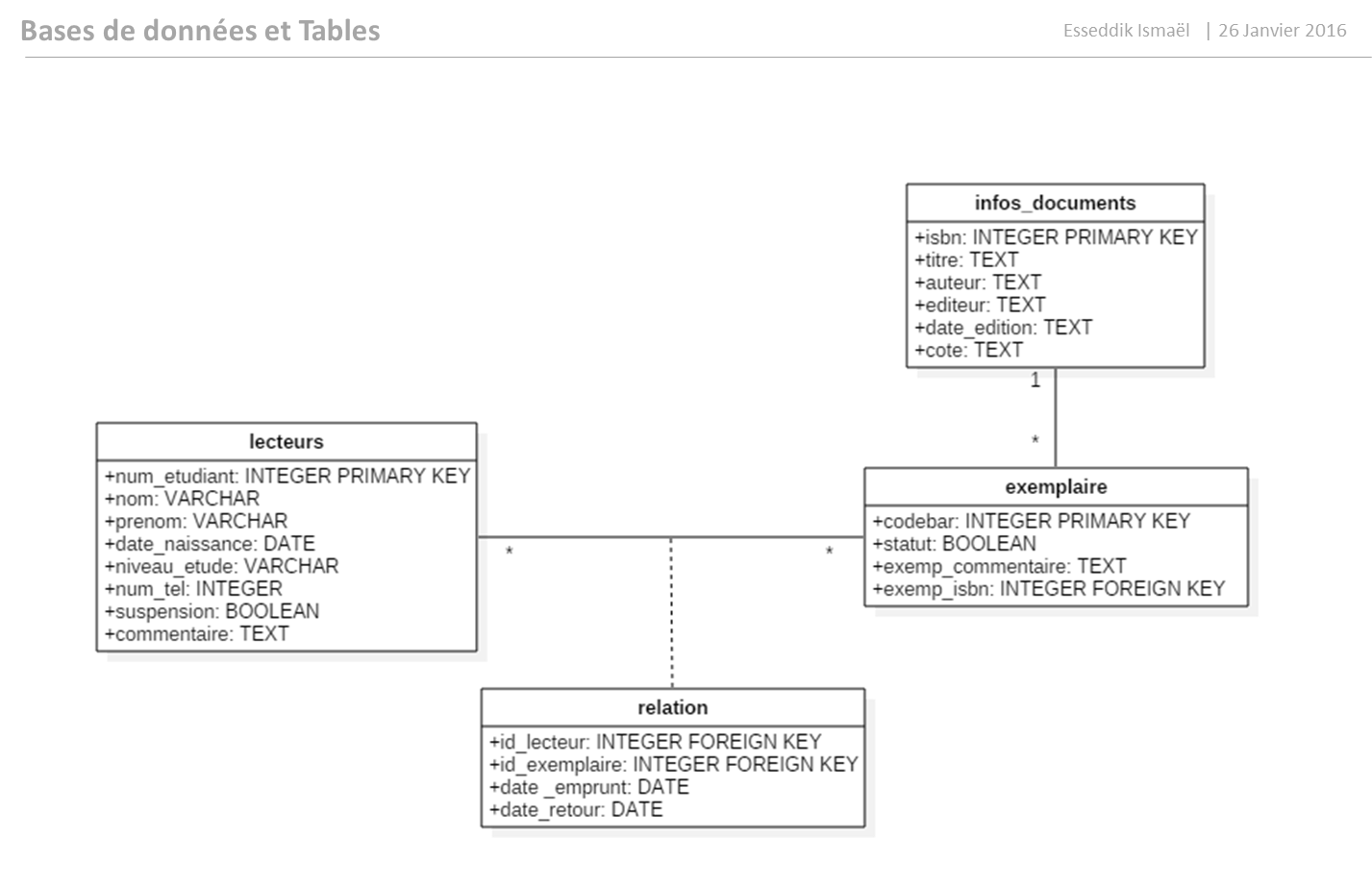


Figure 9 – Le contenu de la base de données est basée sur ce schéma.

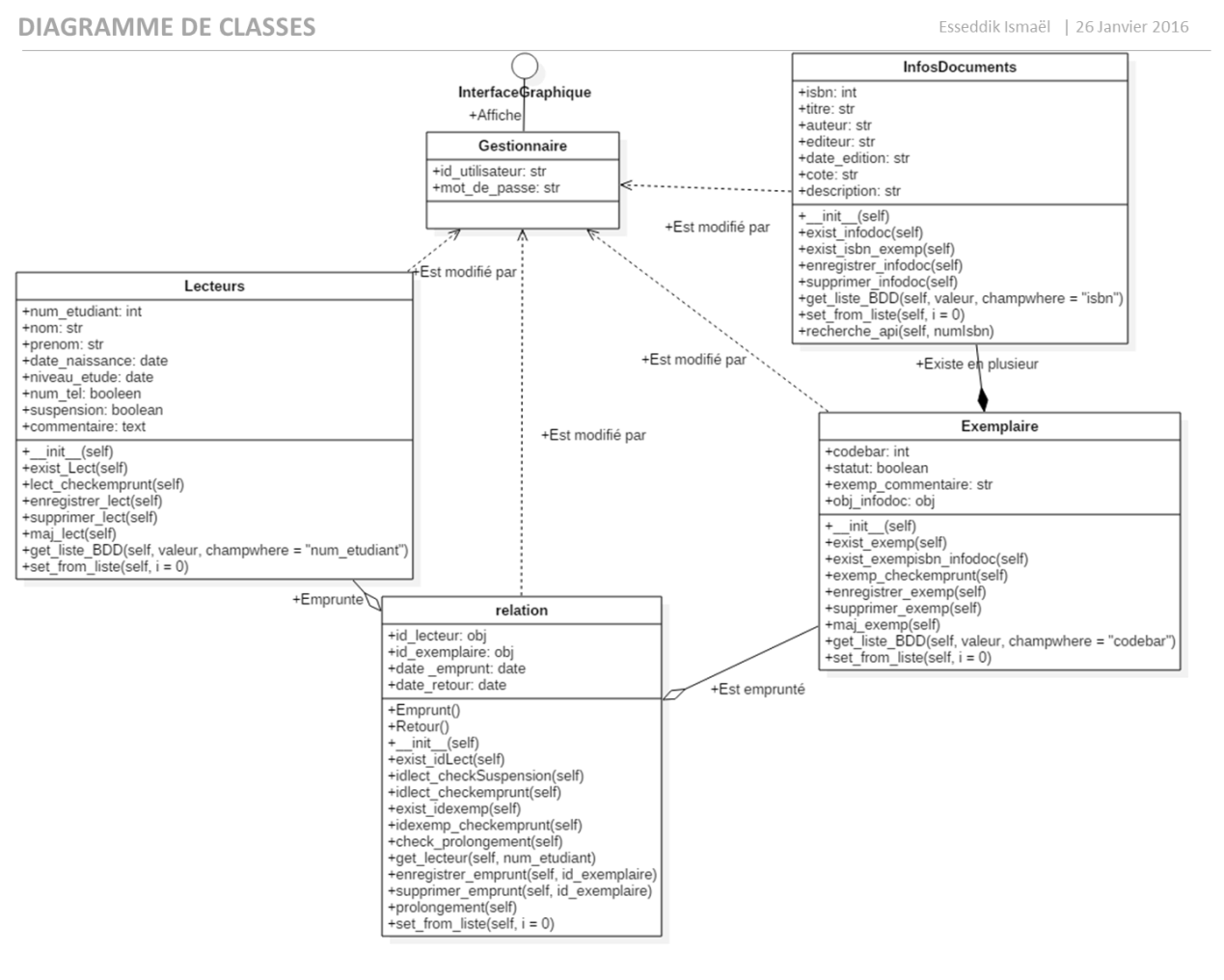


Figure 10 – Les fonctionnalités de classes résumées en uml

## Outils à 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** : Releve 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***

*Schéma de la hiérarchie des fenêtres*

## ***re : Traitement des expression régulières***

## ***Datetime : Traitement des dates et intervalles de temps***

## ***Sqlite3 : Traitement de requêtes sur base de données***

Parmis 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 donnée. Voici quelques exemples :

## *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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## *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

1. Si l’exemplaire existe déjà : afficher « exemplaire déjà existant »
2. 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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## *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 num\_etudiant** via une requête sql, pas de condition à vérifier, retourne un résultat ou  «none » si rien.
* Pour **enregistrer un num\_etudiant**, une seul condition

1. Si le num\_etudiant existe déjà : afficher « Lecteur déjà inscrit »

* Sinon : requête sql pour l’ajouter lui est ses données. (Nouvelle entrée)
* Pour **supprimer un num\_etudiant**, 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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## *Requêtes sur Relations (clé : codebar)*

* 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 emprunter 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.

1. Système de Gestion Intégré de Bibliothèques [↑](#footnote-ref-1)
2. Programmation orienté objet [↑](#footnote-ref-2)
3. International Standard Book Number [↑](#footnote-ref-3)