# RAPPORT D'EXAMEN PRATIQUE

**Développement d'une application console de gestion de la bibliothèque du Dakar Institute of Technology**

**Formation : Master 1 Intelligence Artificielle (M1 IA)**
**Module : Programmation Orientée Objet en Python**

Présenté par :

- OROU BOUYAGUI Bio Mourou
- LY Abdoulaye
- TRAORE Djibrael
- SAWADOGO Moumini

Période : 19 Décembre 2025 – 09 Janvier 2026

# Table des matières

## 1. Introduction générale

La gestion efficace des ressources documentaires est essentielle pour soutenir les activités académiques et pédagogiques. Les bibliothèques jouent un rôle central en offrant aux étudiants, enseignants et personnels un accès rapide et fiable aux ouvrages et documents dont ils ont besoin. Dans ce projet, nous nous intéressons à la digitalisation de la bibliothèque du Dakar Institute of Technology (DIT).

### 1.1 Contexte du projet

Actuellement, la bibliothèque du DIT fonctionne de manière entièrement manuelle : les livres sont enregistrés sur des cahiers, les emprunts sont suivis approximativement, et il est difficile de produire des statistiques fiables. Cette situation entraîne des risques d'erreurs, une absence de traçabilité et limite la qualité du service offert aux usagers.
Pour remédier à ces limites, il est nécessaire de mettre en place une application informatique capable d'automatiser l'enregistrement des livres, la gestion des emprunts et des réservations, et de générer des rapports précis et exploitables.

### 1.2 Objectifs du projet

L'objectif principal de ce projet est de développer une solution digitale permettant d'améliorer et d'automatiser de la gestion de la bibliothèque du Dakar Institute of Technology (DIT).

## 2. Analyse du problème et modélisation

### 2.1 Analyse fonctionnelle

L'analyse fonctionnelle permet d'identifier les principaux besoins métiers que l'application doit satisfaire afin d'assurer une gestion efficace et fiable de la bibliothèque.

- **Gestion des livres et des exemplaires**
  Le système doit permettre l'enregistrement, la modification et la suppression des livres, tout en prenant en compte la gestion des exemplaires multiples d'un même ouvrage. Chaque livre doit être identifié de manière unique (notamment par son ISBN) et associé à un statut indiquant sa disponibilité (disponible, emprunté, réservé, perdu ou endommagé). Le système doit également assurer le suivi du nombre d'emprunts par livre.

- **Gestion des utilisateurs**

  L'application doit permettre la création et la gestion des différents types d'utilisateurs de la bibliothèque, à savoir les étudiants, les enseignants et le personnel administratif. Chaque utilisateur doit disposer d'un identifiant unique et d'un historique détaillé de ses emprunts. Le système doit également appliquer des règles spécifiques, telles que la limitation du nombre d'emprunts en fonction du type d'utilisateur.


- **Gestion des emprunts et des retours**

  Le système doit gérer les opérations d'emprunt et de retour des livres. Avant chaque emprunt, la disponibilité de l'ouvrage doit être vérifiée automatiquement. Le système doit enregistrer les dates d'emprunt et de retour prévues, détecter les retards éventuels et appliquer les règles de pénalités définies. La fonctionnalité de renouvellement des emprunts doit également être prise en charge.


- **Système de réservation**

  L'application doit permettre aux utilisateurs de réserver un livre. Lorsqu'un livre réservé devient disponible, le système doit notifier automatiquement l'utilisateur concerné à travers un fichier texte.


- **Recherche avancée**

  Cette fonctionnalité est implémentée partout dans les autres fonctionnalités Le système doit offrir des fonctionnalités de recherche avancée permettant de retrouver rapidement un ouvrage. Les recherches doivent pouvoir être effectuées selon

plusieurs critères, tels que le titre, l'auteur, la catégorie, l'année de publication, l'ISBN, la disponibilité ou encore des mots-clés.

- **Rapports et statistiques**
  L'application doit être capable de générer des rapports et des statistiques afin de fournir une vision globale de l'activité de la bibliothèque.

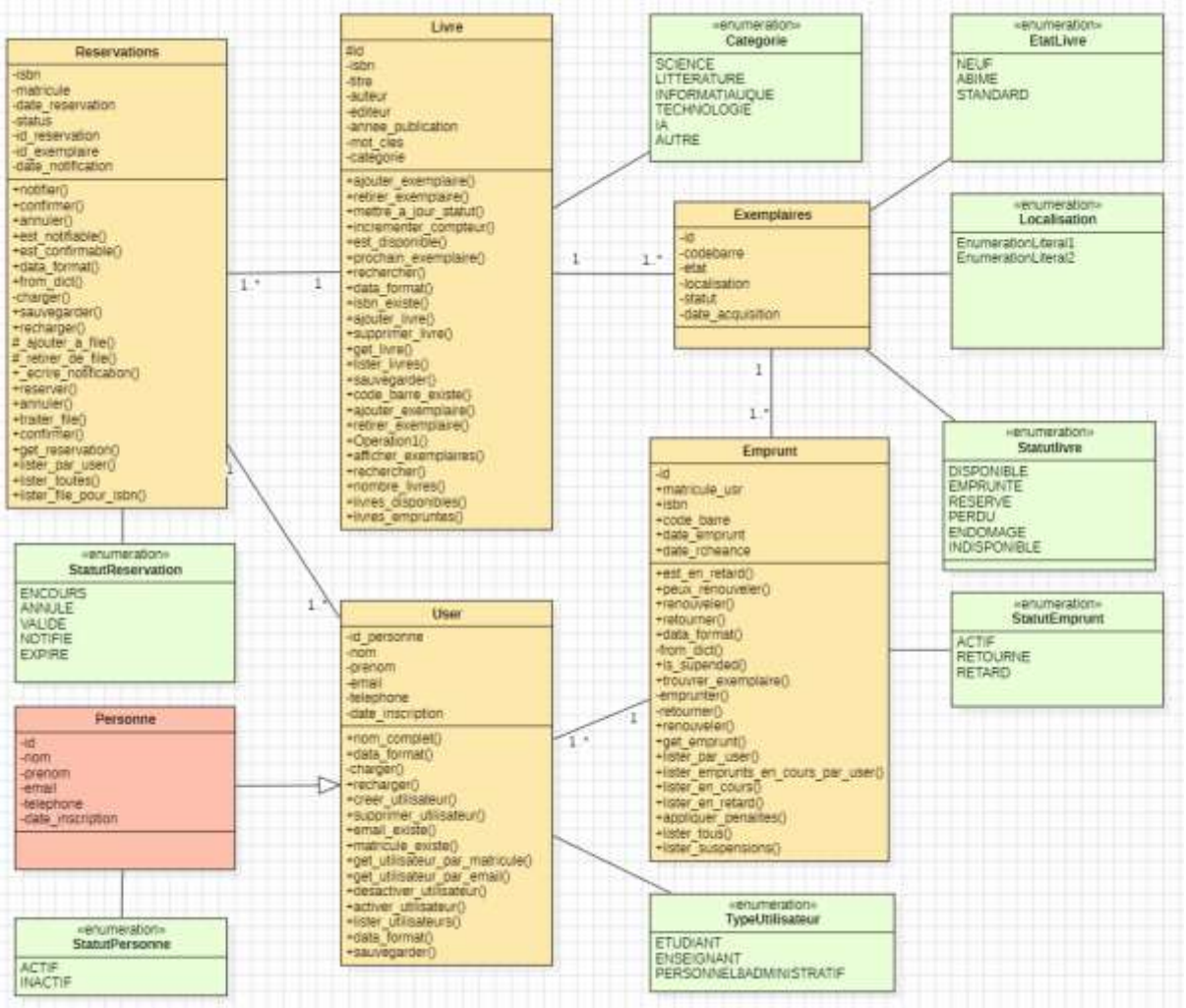
```
=== Statistiques ===
1) Afficher le tableau de bord complet
2) Imprimer le rapport dans un fichier texte
0) Retour
Choix:
Choix invalide
```

- **Journalisation et sauvegarde des données**
  Les autres fonctionnalités font ponctuellement appel à cette dernière. Le système doit assurer la sauvegarde automatique des données après chaque opération importante. Les informations relatives aux livres, aux utilisateurs, aux emprunts et aux réservations doivent être persistées dans des fichiers. Par ailleurs, un fichier de journalisation doit enregistrer l'ensemble des actions effectuées afin de garantir la traçabilité, la sécurité et la fiabilité du système.

```
1  [2026-01-08 21:54:57] INFO | SYSTEM | NOTIFICATION_RESERVATION | 9782212679571 | Notification envoyée
2  [2026-01-09 06:36:50] INFO | Admin | AJOUT_LIVRE | 9782412070048 | Ajout du livre 'Apprendre Python'
3  [2026-01-09 06:44:41] INFO | SYSTEM | NOTIFICATION_RESERVATION | 9782412070048 | Notification envoyée
4  [2026-01-09 06:44:41] INFO | Admin | AJOUT_EXEMPLAIRE | 9782412070048 | Ajout de l'exemplaire 12895 a
5  [2026-01-09 06:45:47] INFO | SYSTEM | NOTIFICATION_RESERVATION | 9782412070048 | Notification envoyée
6  [2026-01-09 06:45:47] INFO | Admin | AJOUT_EXEMPLAIRE | 9782412070048 | Ajout de l'exemplaire 01222 a
7  [2026-01-09 07:04:15] ERROR | Admin | LISTE_EMPRUNTS | N/A | Erreur lors de la liste des emprunts en
8  [2026-01-09 07:16:04] INFO | Admin | LISTE_EMPRUNTS | N/A | 4 emprunt(s) en cours affiché(s)
9  [2026-01-09 07:23:52] INFO | Admin | LISTE_EMPRUNTS | N/A | 4 emprunt(s) en cours affiché(s)
10 [2026-01-09 07:24:04] INFO | Admin | LISTE_TOUS_EMPRUNTS | N/A | 7 emprunt(s) historisés
11 [2026-01-09 07:27:23] INFO | Admin | LISTE_EMPRUNTS_RETARD | N/A | 0 en retard
12 [2026-01-09 07:40:05] WARNING | Admin | AFFICHER_RESERVATION | 0 | Non trouvée
13 [2026-01-09 07:40:13] INFO | Admin | LISTE_RESERVATIONS | N/A | 9 réservation(s) affichée(s)
14 [2026-01-09 07:40:28] INFO | Admin | AFFICHER_RESERVATION | RES-25YKHH0A | Détail affiché
15 [2026-01-09 07:42:46] INFO | Admin | AFFICHER_RESERVATION | RES-25YKHH0A | Détail affiché
16 [2026-01-09 07:43:08] INFO | Admin | AFFICHER_RESERVATION | RES-25YKHH0A | Détail affiché
17 [2026-01-09 07:44:16] INFO | Admin | AFFICHER_RESERVATION | RES-25YKHH0A | Détail affiché
18 [2026-01-09 08:15:17] WARNING | Admin | CREER_RESERVATION | 0 | Erreur de validation lors de la créat
19 [2026-01-09 08:15:23] INFO | Admin | LISTE_RESERVATIONS | N/A | 9 réservation(s) affichée(s)
20
```

### 2.2 Analyse orientée objet

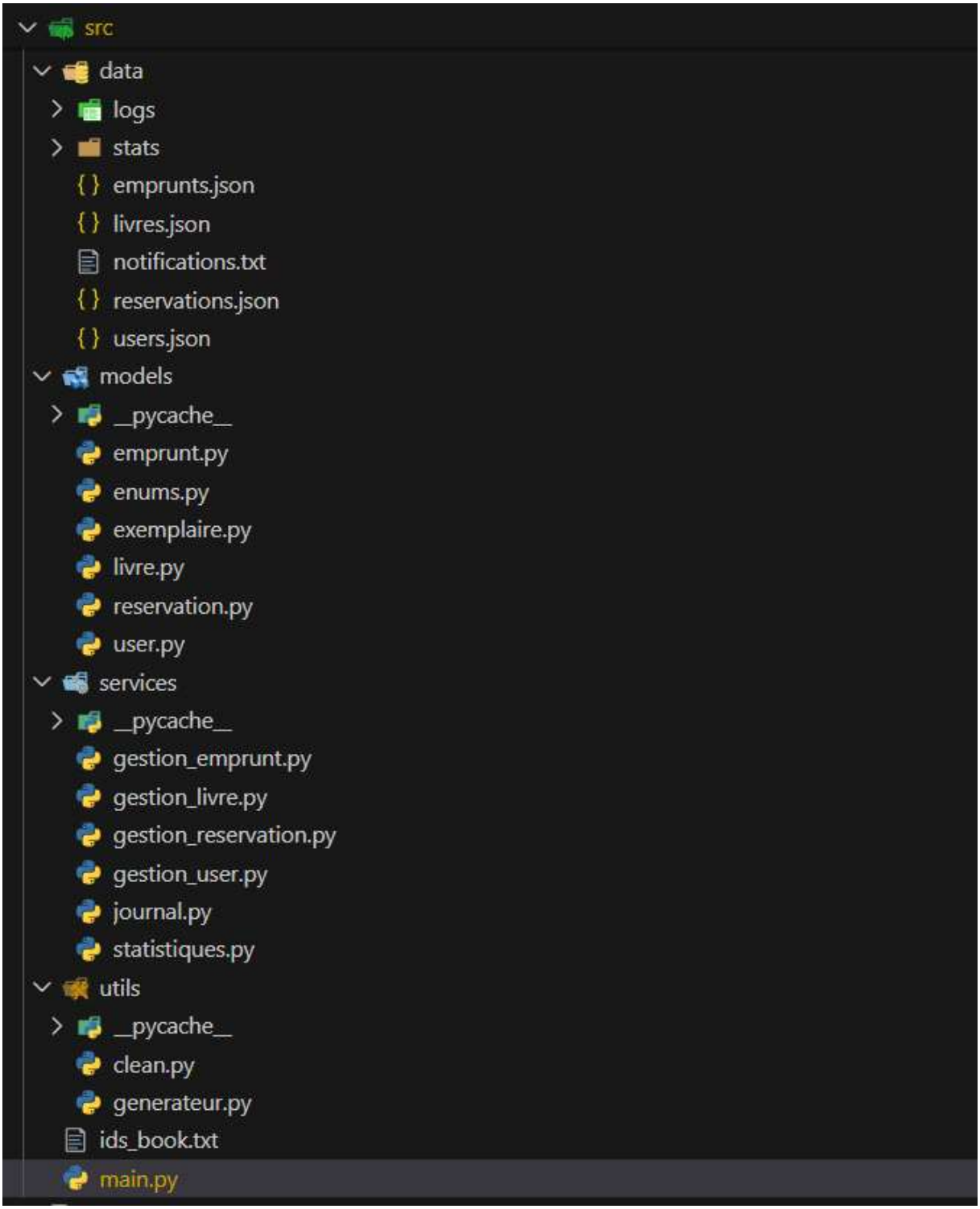
Le système a été modélisé selon une approche orientée objet afin de représenter fidèlement les entités réelles d'une bibliothèque et leurs interactions. Le diagramme de classes UML met en évidence les principales classes du système, notamment Livre, Exemplaire, Emprunt, Réservation, User et Personne, ainsi que leurs relations.
Cette modélisation permet de structurer clairement les responsabilités de chaque classe, de favoriser la réutilisation du code et d'assurer une meilleure maintenabilité de l'application.


Reservations
-isbn
-matricule
-date_reservation
-status
-id_reservation
-id_exemplaire
-date_notification
+notifier()
+confirmer()
+annuler()
+est_notifiable()
+est_confirmable()
+data_format()
+from_dict()
-charger()
+sauvegarder()
+recharger()
#_ajouter_a_file()
#_retirer_de_file()
+_ecrire_notification()
+reserver()
+annuler()
+traiter_file()
+confirmer()
+get_reservation()
+lister_par_user()
+lister_toutes()
+lister_file_pour_isbn()
«enumeration»
StatutReservation
ENCOURS
ANNULE
VALIDE
NOTIFIE
EXPIRE
Personne
-id
-nom
-prenom
-email
-telephone
-date_inscription
«enumeration»
StatutPersonne
ACTIF
INACTIF
Livre
#id
-isbn
-titre
-auteur
-editeur
-annee_publication
-mot_cles
-categorie
+ajouter_exemplaire()
+retirer_exemplaire()
+mettre_a_jour_statut()
+incrementer_compteur()
+est_disponible()
+prochain_exemplaire()
+rechercher()
+data_format()
+isbn_existe()
+ajouter_livre()
+supprimer_livre()
+get_livre()
+lister_livres()
+sauvegarder()
+code_barre_existe()
+ajouter_exemplaire()
+retirer_exemplaire()
+Operation1()
+afficher_exemplaires()
+rechercher()
+nombre_livres()
+livres_disponibles()
+livres_empruntes()
User
-id_personne
-nom
-prenom
-email
-telephone
-date_inscription
+nom_complet()
+data_format()
-charger()
+recharger()
+creer_utilisateur()
+supprimer_utilisateur()
+email_existe()
+matricule_existe()
+get_utilisateur_par_matricule()
+get_utilisateur_par_email()
+desactiver_utilisateur()
+activer_utilisateur()
+lister_utilisateurs()
+data_format()
+sauvegarder()
«enumeration»
Categorie
SCIENCE
LITTERATURE
INFORMATIAUQUE
TECHNOLOGIE
IA
AUTRE
Exemplaires
-id
-codebarre
-etat
-localisation
-statut
-date_acquisition
Emprunt
-id
+matricule_usr
+isbn
+code_barre
+date_emprunt
+date_echeance
+est_en_retard()
+peux_renouveler()
+renouveler()
+retourner()
+data_format()
-from_dict()
+is_supended()
+trouver_exemplaire()
-emprunter()
-retourner()
+renouveler()
+get_emprunt()
+lister_par_user()
+lister_emprunts_en_cours_par_user()
+lister_en_cours()
+lister_en_retard()
+appliquer_penalites()
+lister_tous()
+lister_suspensions()
«enumeration»
TypeUtilisateur
ETUDIANT
ENSEIGNANT
PERSONNELADMINISTRATIF
«enumeration»
EtatLivre
NEUF
ABIME
STANDARD
«enumeration»
Localisation
EnumerationLiteral1
EnumerationLiteral2
«enumeration»
StatutIvre
DISPONIBLE
EMPRUNTE
RESERVE
PERDU
ENDOMAGE
INDISPONIBLE
«enumeration»
StatutEmprunt
ACTIF
RETOURNE
RETARD
1
1..*

### 3. Architecture logicielle et organisation du projet

#### 3.1 Vision globale de l'architecture

L'application de gestion de bibliothèque a été conçue selon une **architecture modulaire orientée objet**, visant à séparer clairement les responsabilités entre les différentes composantes du système.



### 3.2 Organisation de l'arborescence du projet

L'application est structurée de manière modulaire afin de garantir une bonne lisibilité du code, une maintenance facilitée et une séparation claire des responsabilités.

- **Dossier data/ – Persistance des données**
  **Le dossier data** est dédié au stockage des données persistantes de l'application. Il contient principalement des fichiers JSON (livres.json, users.json, emprunts.json, reservations.json) qui assurent la sauvegarde des informations. Il inclut également des sous-dossiers pour la journalisation (logs/), les statistiques (stats/) ainsi qu'un fichier de notifications (notifications.txt). Cette organisation permet de centraliser toutes les données manipulées par le système.

- **Dossier src/models/ – Modélisation métier**
  **Le dossier models** regroupe les classes représentant les entités principales du système telles que *Livre*, *Exemplaire*, *User*, *Emprunt* et *Reservation*. Ces classes définissent les attributs et les comportements fondamentaux de chaque entité, en cohérence avec le diagramme de classes UML, et constituent le cœur de la modélisation orientée objet de l'application.

- **Dossier src/services/ – Logique applicative**
  Le dossier services contient les modules chargés de la gestion des règles métier. Il regroupe notamment les services de gestion des livres, des utilisateurs, des emprunts et des réservations, ainsi que les modules de journalisation et de statistiques. Cette séparation permet d'isoler la logique de traitement, de réduire le couplage et de rendre le code plus évolutif.

- **Dossier src/utils/ – Fonctions utilitaires**
  Le dossier utils rassemble des fonctions utilitaires utilisées à différents niveaux de l'application, telles que le nettoyage de la console, la génération d'identifiants ou la gestion de fichiers auxiliaires. Ces outils transversaux évitent les duplications de code et améliorent la cohérence globale du projet.

- **Fichier main.py – Point d'entrée de l'application**
  Le fichier main.py constitue le point de démarrage de l'application. Il orchestre l'exécution globale, initialise les différents services et assure l'interaction avec l'utilisateur via l'interface en ligne de commande.

### 3.3 Choix de la modularisation

La modularisation repose sur le principe de **séparation des responsabilités**.
Chaque module est indépendant et spécialisé, ce qui permet :

- une maintenance facilitée,
- une meilleure organisation du code,
- une extensibilité future (interface graphique, API, base de données).

## 4. Description des classes

Cette section présente les différentes classes du système de gestion de bibliothèque, leurs responsabilités respectives ainsi que leurs principales caractéristiques.

### 4.1 Classe Livre

**Rôle**

La classe **Livre** représente une œuvre bibliographique enregistrée dans la bibliothèque. Elle centralise les informations descriptives d'un ouvrage et assure la gestion de ses exemplaires.
Elle permet notamment de vérifier la disponibilité d'un livre, de gérer les exemplaires associés, d'effectuer des recherches et de suivre le nombre d'emprunts réalisés.

**Attributs**

- id : identifiant interne du livre
- isbn : numéro ISBN unique
- titre : titre du livre
- auteur : auteur du livre
- annee_publication : année de publication
- editeur : maison d'édition
- categorie : catégorie du livre (énumération Categorie)
- mot_cles : mots-clés associés
- nombreEmprunt : compteur du nombre d'emprunts
- nombreExemplaireDisponible : nombre d'exemplaires disponibles
- dateAjout : date d'ajout du livre dans le système

**Méthodes principales**

- incrementer_compteur() : incrémente le nombre d'emprunts
- est_disponible() : vérifie la disponibilité du livre
- chercher_exemplaire() : recherche un exemplaire disponible
- ajouter_livre() / supprimer_livre() : gestion du catalogue
- code_barre_existe() : vérifie l'existence d'un code-barres
- ajouter_exemplaire() / retirer_exemplaire() : gestion des exemplaires
- afficher_exemplaires() : affiche les exemplaires associés

- rechercher() : recherche de livres

### 4.2 Classe Exemplaires

**Rôle**

La classe **Exemplaire** représente une copie physique d'un livre. Chaque exemplaire est identifié par un code-barres unique et possède un état et un statut propres.
Cette classe permet de suivre précisément l'emplacement, l'état et la disponibilité de chaque exemplaire indépendamment du livre auquel il appartient.

**Attributs**

- id : identifiant de l'exemplaire
- codebarre : code-barres unique
- etat : état physique (énumération EtatLivre)
- statut : statut d'emprunt (énumération StatutLivre)
- localisation : emplacement physique
- date_acquisition : date d'acquisition

**Méthodes principales**

Les principales méthodes de cette classe se retrouve dans la classe Livre

- changer_etat() : met à jour l'état physique
- changer_statut() : met à jour le statut d'emprunt

### 4.3 Classe Reservation

**Rôle**

La classe **Reservation** gère le processus de réservation des livres indisponibles. Elle permet d'enregistrer les demandes des utilisateurs, de suivre leur statut et de notifier automatiquement l'utilisateur lorsqu'un exemplaire devient disponible.
Le traitement des réservations suit une logique FIFO (premier arrivé, premier servi), garantissant une gestion équitable des demandes.

**Attributs**

- isbn : ISBN du livre réservé
- matricule : identifiant de l'utilisateur
- date_reservation : date de réservation
- statut : statut de la réservation (StatutReservation)
- id_reservation : identifiant unique
- id_exemplaire : exemplaire concerné
- date_notification : date de notification

**Méthodes principales**

- charger() Charge les réservations depuis le fichier de persistance et initialise la liste des réservations.
- sauvegarder() Enregistre l'ensemble des réservations dans le fichier de stockage.
- lister() Retourne la liste complète des réservations enregistrées.
- _reservations_actives(isbn)
  Récupère les réservations actives d'un livre donné, triées par date de réservation.
- prochaine_reservation(isbn) Identifie la prochaine réservation à traiter pour un livre donné.
- reservation_existe(isbn, matricule) Vérifie l'existence d'une réservation active pour un utilisateur et un livre donnés.
- reserver_livre(livre, matricule)
  Crée une nouvelle réservation pour un livre indisponible.
- consommer_reservation(isbn, matricule) Supprime la réservation lorsque l'utilisateur notifié procède à l'emprunt du livre.
- annuler_reservation(isbn, matricule, livre=None) Annule une réservation active et libère l'exemplaire associé si nécessaire.
- notifier_si_disponible(livre) Notifie l'utilisateur lorsqu'un exemplaire devient disponible et réserve cet exemplaire.
- ecrire_notification(reservation, titre_livre) Génère un message de notification indiquant la disponibilité du livre réservé.

### 4.4 Classe Emprunt

**Rôle**

La classe **Emprunt** gère le cycle de vie d'un emprunt, depuis l'enregistrement jusqu'au retour du livre. Elle permet de calculer automatiquement les retards, d'appliquer les pénalités et de mettre à jour le statut des emprunts.
Elle joue un rôle central dans le respect des règles de prêt définies par la bibliothèque.

**Attributs**

- id : identifiant de l'emprunt
- isbn : ISBN du livre
- matricule : identifiant de l'utilisateur
- date_emprunt : date d'emprunt
- date_retour_prevue : date prévue de retour
- date_retour : date réelle de retour
- statut : statut de l'emprunt (StatutEmprunt)
- penalite_jour : pénalité par jour de retard
- nb_renouvellement : nombre de renouvellements

**Méthodes principales**

- calculer_retard() : calcule le nombre de jours de retard
- calculer_penalite_jours() : calcule les pénalités
- est_en_retard() : vérifie si l'emprunt est en retard
- renouveler() : renouvelle un emprunt
- retourner() : clôture l'emprunt
- trouver_exemplaire() : identifie l'exemplaire associé
- mettre_a_jour_retards() : met à jour les statuts

### 4.5 Classe User

**Rôle**

La classe **User** représente un utilisateur de la bibliothèque. Elle gère les informations d'identification, les règles d'emprunt et l'historique des opérations effectuées par chaque utilisateur.

Elle permet également d'activer ou de désactiver un utilisateur et de vérifier les contraintes liées aux emprunts.

**Attributs**

- numInscription : identifiant utilisateur
- nombreEmpruntMax : limite d'emprunts
- penalite : pénalité cumulée

**Méthodes principales**

- enregistrer_emprunt() : enregistre un emprunt
- enregistrer_retour() : enregistre un retour
- email_existe() / matricule_existe() : vérifications
- get_utilisateur_par_matricule()
- get_utilisateur_par_email()
- activer_utilisateur() / desactiver_utilisateur()
- lister_utilisateurs()
- sauvegarder()

## 5.Explication détaillée des étapes de développement

L'implémentation des fonctionnalités a été réalisée de manière progressive, en respectant la logique définie lors de la modélisation. Chaque fonctionnalité a été développée, testée puis améliorée avant de passer à la suivante.
Cette approche a permis de limiter les erreurs, de mieux comprendre les interactions entre les classes et d'assurer la stabilité globale de l'application.

### 5.1. Étape 1 : Analyse des besoins fonctionnels

La première étape a consisté à analyser les besoins du système à partir du cahier des charges. Cette analyse a permis d'identifier les fonctionnalités essentielles, notamment

- la gestion des livres et de leurs exemplaires,
- la gestion des utilisateurs,
- le processus d'emprunt et de retour,
- le système de réservation,
- la recherche avancée,
- la production de statistiques.

Cette phase a servi de base à la définition des entités métier et des interactions entre elles. Nous avons voulu aussi mettre en pratiques le cours de gestion de projet informatique. L'équipe a été divisée en : Scrum Master, Product owner, Développer ; tout en progressant par des Sprint de 3 jours.

### 5.2. Étape 2 : Modélisation orientée objet

À partir de l'analyse fonctionnelle, une modélisation orientée objet a été réalisée. Les principales entités du système ont été représentées sous forme de classes, avec :

- leurs attributs,
- leurs méthodes,
- et leurs relations (associations, héritage).

Cette modélisation a été formalisée à travers un diagramme de classes UML, garantissant une vision claire et structurée du système avant le codage.

### 5.3. Étape 3 : Mise en place de l'architecture du projet

Une architecture modulaire a ensuite été définie afin d'organiser le projet de manière cohérente.
Le code a été structuré en plusieurs modules distincts :

- **models** pour les classes métier,
- **services** pour la logique applicative,
- **utils** pour les fonctions transversales,
- **data** pour la persistance des données.

Cette organisation permet une séparation claire des responsabilités et facilite la maintenance et l'évolution de l'application.

### 5.4. Étape 4 : Implémentation des classes métier

Les classes métier ont été implémentées en priorité, conformément au diagramme UML. Chaque classe encapsule ses attributs et méthodes principales, garantissant :

- l'intégrité des données,
- le respect des règles métier,
- et la cohérence des interactions entre objets.

Cette étape a permis de disposer d'une base solide avant l'ajout des fonctionnalités complexes.

### 5.5. Étape 5 : Développement de la logique applicative

Une fois les classes métier en place, les fonctionnalités ont été développées au niveau des services.

Cette phase inclut :

- la gestion des emprunts et des retours,
- le contrôle des disponibilités,
- la gestion des réservations,
- le calcul automatique des pénalités,
- la mise à jour des statuts.

Les services jouent un rôle central dans la coordination des opérations entre les différentes entités.

### 5.6. Étape 6 : Mise en œuvre de la persistance des données

La persistance des données a été assurée à l'aide de fichiers JSON. Chaque opération critique (ajout, modification, suppression) déclenche une sauvegarde automatique, garantissant la cohérence et la durabilité des données.

### 5.7. Étape 7 : Gestion des erreurs et validation des données

Des mécanismes de validation et de gestion des erreurs ont été intégrés afin de :

- prévenir les incohérences,
- bloquer les actions invalides,
- informer clairement l'utilisateur via la console.

Cette étape améliore la robustesse et la fiabilité globale du système.

### 5.8. Étape 8 : Tests fonctionnels et amélioration continue

Enfin, des tests fonctionnels ont été réalisés à chaque étape du développement afin de :

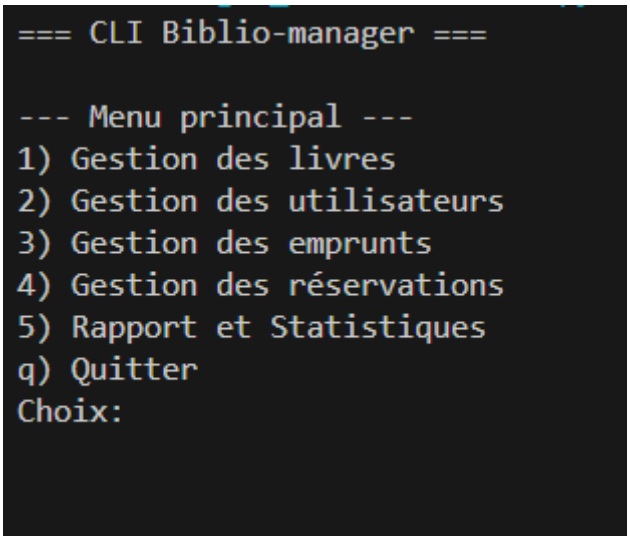
- vérifier le bon fonctionnement des fonctionnalités,
- corriger les anomalies détectées,
- améliorer progressivement l'ergonomie et la stabilité de l'application.

Cette approche itérative a permis d'aboutir à une application cohérente, fiable et conforme aux objectifs initiaux.

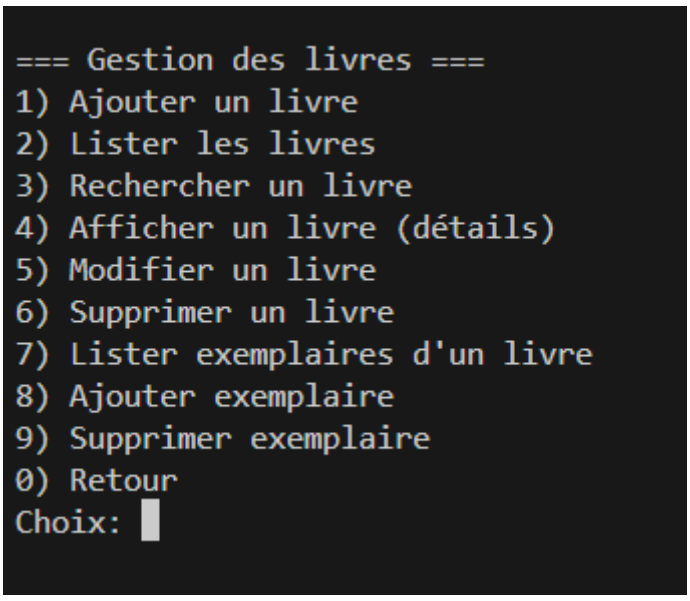
## 6. Illustration des principales fonctionnalités de l'application

### 6.1. Menu principal

C'est la principale porte d'entrée de l'application, il donne accès aux principales fonctionnalités de l'application




### 6.2. Gestion des livres et exemplaires (sous menu)



2. liste des livres

```
Choix: 2
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| ISBN          | Titre           | Auteur          | Éditeur   | Année       | Catégorie   | Statut      | Mots-clés                         |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| 9782409038426 | Raspberry Pi    | Kevin SARTOR    | eni       | 2023        | Autre       | Disponible  | electronique,python,domotique,esp8266 |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| 9782212679571 | Maths avec Python | Peter Farrell | EYROLLES  | 2020        | Autre       | Disponible  | python                            |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| 9782100813728 | IA triomphes    | Melanie Mitchell | Dunod    | 2021        | Autre       | Disponible  | ia,elect                          |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| 9782100847693 | DL avec Keras   | Aurélien Géron  | Dunod     | 2024        | Autre       | Disponible  | ia,ml,dl                          |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| 9782416016417 | IA, python      | EYROLLES        | EYROLLES  | 2024        | Autre       | Disponible  | python                            |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| 9782412098370 | IA en low-code  | Abel Michael    | F1RST     | 2024        | Autre       | Disponible  | ia,low,code                       |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| 9782710811817 | Data Science    | Filipe AFONSO   | Technip   | 2018        | Autre       | Disponible  | stats,data,python,math            |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
| 9782412070048 | Apprendre Python | Eurode         | Dunod     | 2024        | Informatique | Disponible | programme,python,scripts          |
+---------------+-----------------+-----------------+-----------+-------------+-------------+-----------------------------------+
```

3. Afficher un livre

```
Choix: 4
ISBN: 9782412098370
Titre: IA en low-code
Auteur: Abel Michael
Editeur: F1RST
Année: 2024
Nombre exemplaires: 1
Disponibles: 1
Statut: Disponible
Exemplaires:
 - ID: 00145 | statut: disponible | etat: bon
```

7. lister les exemplaires d'un livre

```
Choix: 7
ISBN du livre: 9782412098370
+-------------+-------------+
| Code barre | Statut      |
+-------------+-------------+
| 00145      | disponible |
+-------------+-------------+
```

### 6.3. Gestion des utilisateurs

1. liste des utilisateurs

```
Choix: 1
Matricule    | Nom Complet          | Email                    | Statut     | Type
-------------------------------------------------------------------------------------
U-5DIZ5WRI   | Runner Auto          | auto.runner@example.com  | inactif    | Etudiant
U-3HSBZLV9   | Yo Bio Mourou        | mourou@gmail.com         | actif      | Enseignant
U-Y569PEJE   | Carole HONHE         | bignon.caro@gmail.com    | actif      | Etudiant
U-P2Z63IIM   | Abel DAKOTO          | abel.da@gmail.co         | actif      | Enseignant
U-X15N63YN   | Honorine GOUNOU      | honor@gmail.com          | actif      | Etudiant
```

2.Créer un utilisateur

```
Choix: 2
Nom: SAWADOGO
Prénom: Moumini
Email: moumiabf@outlook.fr
Téléphone: 75653338

Types d'utilisateurs disponibles :
1) Étudiant
2) Enseignant
3) Personnel administratif
Choisissez un type (1-3) : 1
Utilisateur créé. Matricule: U-Q6YOE0NZ | Type: Etudiant
```

3. Modifier un utilisateur

```
Choix: 3
Matricule: U-Q6YOE0NZ

Modification de : Moumini SAWADOGO (actuel : Etudiant, statut: actif)
Laisser vide pour ne pas modifier.
Nouveau nom: OUEDRAOGO
Nouveau prénom: Moumini
Nouvel email:
Nouveau téléphone:

Types disponibles :
1) Étudiant
2) Professeur
3) Personnel administratif
Actuel : Etudiant
Nouveau type (1-3, ou Entrée pour conserver) : 2
Type mis à jour : Enseignant
Utilisateur modifié et sauvegardé.
```

4. Supprimer un utilisateur

```
Choix: 4
Matricule: U-Q6YOE0NZ
Utilisateur supprimé.
```

5.Rechercher par matricule

```
Choix: 5
Matricule: U-5DIZ5WRI
Matricule: U-5DIZ5WRI | Nom: Runner Auto | Email: auto.runner@example.com | Statut: inactif
```

6.Rechercher par mail

```
Choix: 6
Email: honor@gmail.com
Matricule: U-X15N63YN | Nom: Honorine GOUNOU | Email: honor@gmail.com | Statut: actif
```

7.Activer un utilisateur

```
Matricule    | Nom Complet          | Email                    | Statut     | Type
-------------------------------------------------------------------------------------
U-5DIZ5WRI   | Runner Auto          | auto.runner@example.com  | inactif    | Etudiant
U-3HSBZLV9   | Yo Bio Mourou        | mourou@gmail.com         | actif      | Enseignant
U-Y569PEJE   | Carole HONHE         | bignon.caro@gmail.com    | actif      | Etudiant
U-P2Z63IIM   | Abel DAKOTO          | abel.da@gmail.co         | actif      | Enseignant
U-X15N63YN   | Honorine GOUNOU      | honor@gmail.com          | actif      | Etudiant
```

```
Choix: 7
Matricule: U-5DIZ5WRI
Utilisateur activé.
```

```
Choix: 1
Matricule    | Nom Complet          | Email                    | Statut     | Type
-------------------------------------------------------------------------------------
U-5DIZ5WRI   | Runner Auto          | auto.runner@example.com  | actif      | Etudiant
U-3HSBZLV9   | Yo Bio Mourou        | mourou@gmail.com         | actif      | Enseignant
U-Y569PEJE   | Carole HONHE         | bignon.caro@gmail.com    | actif      | Etudiant
U-P2Z63IIM   | Abel DAKOTO          | abel.da@gmail.co         | actif      | Enseignant
U-X15N63YN   | Honorine GOUNOU      | honor@gmail.com          | actif      | Etudiant
```

8.Désactiver un utilisateur

| Matricule | Nom Complet | Email | Statut | Type |
|---|---|---|---|---|
| U-5DIZ5WRI | Runner Auto | auto.runner@example.com | actif | Etudiant |
| U-3HSBZLV9 | Yo Bio Mourou | mourou@gmail.com | actif | Enseignant |
| U-Y569PEJE | Carole HONHE | bignon.caro@gmail.com | actif | Etudiant |
| U-P2Z63IIM | Abel DAKOTO | abel.da@gmail.co | actif | Enseignant |
| U-X15N63YN | Honorine GOUNOU | honor@gmail.com | actif | Etudiant |

```
Choix: 8
Matricule: U-X15N63YN
Utilisateur désactivé.
```

```
Choix: 1
Matricule    | Nom Complet          | Email                    | Statut     | Type
-------------------------------------------------------------------------------
U-5DIZ5WRI   | Runner Auto          | auto.runner@example.com  | actif      | Etudiant
U-3HSBZLV9   | Yo Bio Mourou        | mourou@gmail.com         | actif      | Enseignant
U-Y569PEJE   | Carole HONHE         | bignon.caro@gmail.com    | actif      | Etudiant
U-P2Z63IIM   | Abel DAKOTO          | abel.da@gmail.co         | actif      | Enseignant
U-X15N63YN   | Honorine GOUNOU      | honor@gmail.com          | inactif    | Etudiant
```

### 6.4. Gestion des emprunts

1) Lister emprunts en cours

```
Choix: 1
+-------------+-----------+---------------+--------------+-----------------+-----------+--------------+---------------+
| ID          | Matricule | Nom et Prénom | ISBN         | Titre           | Exemplaire | Date emprunt | Date échéance |
+-------------+-----------+---------------+--------------+-----------------+-----------+--------------+---------------+
| EMP-S2NNZSHK | U-5DIZ5WRI | Auto Runner   | 9782416016417 | IA, python      | 00003     | 2026-01-07   | 2026-01-21    |
+-------------+-----------+---------------+--------------+-----------------+-----------+--------------+---------------+
| EMP-IHUBQPET | U-P2Z63IIM | DAKOTO Abel   | 9782212679571 | Maths avec Python | 00147   | 2026-01-08   | 2026-01-22    |
+-------------+-----------+---------------+--------------+-----------------+-----------+--------------+---------------+
| EMP-8PMZ4RHY | U-P2Z63IIM | DAKOTO Abel   | 9782710811817 | Data Science    | 12421     | 2026-01-08   | 2026-01-22    |
+-------------+-----------+---------------+--------------+-----------------+-----------+--------------+---------------+
| EMP-Q1ZF6X0M | U-3HSBZLV9 | Bio Mourou Yo | 9782416016417 | IA, python      | 00002     | 2026-01-08   | 2026-01-22    |
+-------------+-----------+---------------+--------------+-----------------+-----------+--------------+---------------+
```

2) Lister emprunts d'un utilisateur

```
Choix: 2
Matricule utilisateur: U-3HSBZLV9
+--------------+---------------+------------------+------------+--------------+---------------+
| ID           | ISBN          | Titre            | Exemplaire | Date emprunt | Date échéance |
+--------------+---------------+------------------+------------+--------------+---------------+
| EMP-XO3R3FG1 | 9782412070048 | Apprendre Python | 00001      | 2026-01-04   | 2026-01-18    |
+--------------+---------------+------------------+------------+--------------+---------------+
| EMP-Q1ZF6X0M | 9782416016417 | IA, python       | 00002      | 2026-01-08   | 2026-01-22    |
+--------------+---------------+------------------+------------+--------------+---------------+
```

3) Lister tous les emprunts

```
Choix: 3
+-----------------+-----------+-----------------+---------------+-----------------+-----------+-----------------+-----------------+-----------+
| ID              | Matricule | Nom et Prénom   | ISBN          | Titre           | Exemplaire | Date emprunt | Date échéance | Date retour | Statut    |
+-----------------+-----------+-----------------+---------------+-----------------+-----------+-----------------+-----------------+-----------+
| EMP-QRVMBLEB    | U-5DIZ5WRI | Auto Runner    | 9782412070048 | Apprendre Python | 00001     | 2026-01-04      | 2026-01-18      | 2026-01-04 | retourne  |
| EMP-X03R3FG1    | U-3HSBZLV9 | Bio Mourou Yo  | 9782412070048 | Apprendre Python | 00001     | 2026-01-04      | 2026-01-18      | 2026-01-04 | retourne  |
| EMP-52NNZSHE    | U-5DIZ5WRI | Auto Runner    | 9782416016417 | IA, python      | 00003     | 2026-01-07      | 2026-01-21      |           | emprunte  |
| EMP-MJ0XFGTF    | U-5DIZ5WRI | Auto Runner    | 9782416016417 | IA, python      | 00002     | 2026-01-07      | 2026-01-21      | 2026-01-08 | retourne  |
| EMP-IHUBQPET    | U-P2Z63IIM | DAKOTO Abel    | 9782212679571 | Maths avec Python | 00147     | 2026-01-08      | 2026-01-22      |           | emprunte  |
| EMP-8PMZ4R0Y    | U-P2Z63IIM | DAKOTO Abel    | 9782710011017 | Data Science    | 12421     | 2026-01-08      | 2026-01-22      |           | emprunte  |
| EMP-Q1ZFGX0M    | U-3HSBZLV9 | Bio Mourou Yo  | 9782416016417 | IA, python      | 00002     | 2026-01-08      | 2026-01-22      |           | emprunte  |
+-----------------+-----------+-----------------+---------------+-----------------+-----------+-----------------+-----------------+-----------+
```

4) Lister emprunts en retard

5) Afficher emprunt par ID

```
Choix: 5
ID emprunt: EMP-QRVMBLEB
ID emprunt      : EMP-QRVMBLEB
Utilisateur     : Auto Runner (U-5DIZ5WRI)
Livre           : Apprendre Python (9782412070048)
Exemplaire      : 00001
Date emprunt    : 2026-01-04 10:21
Date échéance   : 2026-01-18 10:21
Date retour     : 2026-01-04 10:21
Statut          : retourne
```

6) Lister suspensions

```
Choix: 6
Aucune suspension
```

7) Emprunter

```
ID User: U-3HSBZLV9
ISBN du livre: 9782212679571
Code barre exemplaire : 00147
Emprunt enregistré.
```

8) Retourner

```
ID emprunt: EMP-Q1ZF6X0M
Retour enregistré.
```

9) Renouveler emprunt

```
Choix: 9
ID emprunt: EMP-Q1ZF6X0M
Jours supplémentaires (par défaut 7):
Renouvellement impossible (déjà retourné, en retard, ou limite de renouvellements atteinte).
```

### 6.5. Gestion des réservations

1) Créer une réservation

```
Choix: 1
Matricule utilisateur: U-3HSBZLV9
ISBN: 9782212679571
Réservation créée : RES-B7G71168 (statut=en_attente)
```

2) Lister toutes les réservations

```
| RES-DEF4PLPW | U3        | Utilisateur supprimé | 1111111111    | Livre supprimé    | 2026-01-04 10:13 | en_attente |
| RES-FFC5Y0LY | U-5DIZ5WRI | Auto Runner         | 9782412070048 | Apprendre Python  | 2026-01-04 10:17 | notifie    |
| RES-PLG7BQOX | U-5DIZ5WRI | Auto Runner         | 9782412070048 | Apprendre Python  | 2026-01-04 10:21 | notifie    |
| RES-4KK9DK9W | 1         | Utilisateur supprimé | 01            | Livre supprimé    | 2026-01-04 11:56 | annule     |
| RES-7RB21AHK | U-P2Z63IIM | DAKOTO Abel         | 9782412098370 | IA en low-code    | 2026-01-08 14:21 | en_attente |
| RES-25YKHH0A | U-P2Z63IIM | DAKOTO Abel         | 9782212679571 | Maths avec Python | 2026-01-08 15:34 | confirme   |
| RES-G1WPHZU3 | H54419    | Utilisateur supprimé | 9782212679571 | Maths avec Python | 2026-01-08 15:36 | notifie    |
| RES-B7G71168 | U-3HSBZLV9 | Bio Mourou Yo       | 9782212679571 | Maths avec Python | 2026-01-09 22:43 | en_attente |
```

3) Lister réservations par utilisateur

```
Choix: 3
Matricule utilisateur: U-3HSBZLV9
+--------------+---------------+-------------------+------------------+------------+
| ID           | ISBN          | Titre             | Date             | Statut     |
+--------------+---------------+-------------------+------------------+------------+
| RES-B7G71168 | 9782212679571 | Maths avec Python | 2026-01-09 22:43 | en_attente |
+--------------+---------------+-------------------+------------------+------------+
```

4) Afficher réservation par ID

```
Choix: 4
ID réservation: RES-7RB21AHK
-----------------------------------
Détails de la réservation :
-----------------------------------
ID réservation : RES-7RB21AHK
Utilisateur    : DAKOTO Abel (U-P2Z63IIM)
Livre          : IA en low-code (9782412098370)
Date           : 2026-01-08 14:21
Statut         : en_attente
```

5) Annuler une réservation

```
Choix: 2
```

| ID | Matricule | Nom et Prénom | ISBN | Titre | Date | Statut |
|---|---|---|---|---|---|---|
| RES-8S8JAC95 | U-TEST | Utilisateur supprimé | 9782412070048 | Apprendre Python | 2026-01-04 09:55 | annule |
| RES-SZ53WUNI | U-CLI | Utilisateur supprimé | 9782412070048 | Apprendre Python | 2026-01-04 10:11 | notifie |
| RES-DEF4PLPW | U3 | Utilisateur supprimé | 1111111111 | Livre supprimé | 2026-01-04 10:13 | en_attente |
| RES-FFC5Y0LY | U-5DIZ5WRI | Auto Runner | 9782412070048 | Apprendre Python | 2026-01-04 10:17 | notifie |
| RES-PLG7BQOX | U-5DIZ5WRI | Auto Runner | 9782412070048 | Apprendre Python | 2026-01-04 10:21 | notifie |
| RES-4KK9DK9W | 1 | Utilisateur supprimé | 01 | Livre supprimé | 2026-01-04 11:56 | annule |
| RES-7RB21AHK | U-P2Z63IIM | DAKOTO Abel | 9782412098370 | IA en low-code | 2026-01-08 14:21 | en_attente |
| RES-25YKHH0A | U-P2Z63IIM | DAKOTO Abel | 9782212679571 | Maths avec Python | 2026-01-08 15:34 | confirme |

```
Choix: 5
ID réservation: RES-7RB21AHK
Réservation annulée.
```

| RES-8S8JAC95 | U-TEST | Utilisateur supprimé | 9782412070048 | Apprendre Python | 2026-01-04 09:55 | annule |
|---|---|---|---|---|---|---|
| RES-SZ53WUNI | U-CLI | Utilisateur supprimé | 9782412070048 | Apprendre Python | 2026-01-04 10:11 | notifie |
| RES-DEF4PLPW | U3 | Utilisateur supprimé | 1111111111 | Livre supprimé | 2026-01-04 10:13 | en_attente |
| RES-FFC5Y0LY | U-5DIZ5WRI | Auto Runner | 9782412070048 | Apprendre Python | 2026-01-04 10:17 | notifie |
| RES-PLG7BQOX | U-5DIZ5WRI | Auto Runner | 9782412070048 | Apprendre Python | 2026-01-04 10:21 | notifie |
| RES-4KK9DK9W | 1 | Utilisateur supprimé | 01 | Livre supprimé | 2026-01-04 11:56 | annule |
| RES-7RB21AHK | U-P2Z63IIM | DAKOTO Abel | 9782412098370 | IA en low-code | 2026-01-08 14:21 | annule |
| RES-25YKHH0A | U-P2Z63IIM | DAKOTO Abel | 9782212679571 | Maths avec Python | 2026-01-08 15:34 | confirme |
| RES-G1WPHZU3 | H54419 | Utilisateur supprimé | 9782212679571 | Maths avec Python | 2026-01-08 15:36 | notifie |
| RES-B7G71168 | U-3HSBZLV9 | Bio Mourou Yo | 9782212679571 | Maths avec Python | 2026-01-09 22:43 | en_attente |

6) Traiter la file d'un ISBN (notifier tête)

```
Choix: 6
ISBN: 9782212679571
Aucune notification envoyée livre est indisponible ou la file est vide.
```

7) Confirmer une réservation

```
Choix: 7
ID réservation à confirmer: RES-PLG7BQOX
Réservation RES-PLG7BQOX confirmée et emprunt créé (ID: EMP-5DW00XVV)
Réservation confirmée.
```

### 6.6. Rapport et statistiques

1) Afficher le tableau de bord complet

```
================================================================
                TOP 5 DES LIVRES LES PLUS EMPRUNTÉS
================================================================
Rang   Titre                          Emprunts
----------------------------------------------------------
1      Apprendre Python               3
2      IA, python                     3
3      Maths avec Python              2
4      Data Science                   1

================================================================
              TOP 5 DES UTILISATEURS LES PLUS ACTIFS
================================================================
Rang   Nom                            Emprunts
----------------------------------------------------------
1      Runner Auto                    4
2      Yo Bio Mourou                  3
3      Abel DAKOTO                    2

================================================================


=== Statistiques ===
1) Afficher le tableau de bord complet
2) Imprimer le rapport dans un fichier texte
0) Retour
Choix:
```

2) Imprimer le rapport dans un fichier texte

## 6. Difficultés rencontrées et solutions apportées

### 6.1. Difficultés rencontrées

- Le projet a mis en évidence des différences de niveau en programmation Python entre les membres du groupe, en particulier sur les notions de la maitrise de la programmation orientée objet et d'organisation du code.

- Le travail à distance a également limité la fluidité des échanges, rendant parfois la coordination et la résolution des problèmes plus lentes.

- Par ailleurs, la compréhension des exigences fonctionnelles et techniques du projet a nécessité un temps d'analyse important avant de pouvoir les traduire correctement en fonctionnalités opérationnelles.

- Enfin, certaines parties du système ont demandé des explications techniques approfondies, notamment sur l'architecture globale, le rôle des classes et leurs interactions, ce qui a allongé le temps de travail mais renforcé la compréhension collective du projet.

### 6.2. Solutions apportées

- Afin d'harmoniser le niveau du groupe, une approche pédagogique progressive de gestion de projet informatique dans le contexte **scrum** a été adoptée, avec un développement réalisé sprint par sprint, appuyé par des explications simples et des exemples concrets.

- La communication a été renforcée grâce à des échanges réguliers et à l'utilisation d'un espace de discussion (Zoom et google Drive), permettant de lever rapidement les blocages et d'assurer une cohérence technique malgré le travail à distance.

- Une analyse approfondie de l'énoncé a également été menée, incluant la reformulation des exigences et une phase de modélisation préalable, afin d'aligner le développement sur les objectifs du projet.

- Enfin, le projet a été conduit de manière itérative, avec des tests fréquents et une documentation claire du code, facilitant la compréhension, la correction et l'évolution de l'application.

**7. Limites et perspectives**

- L'application fonctionne uniquement en mode console, ce qui est adapté au cadre pédagogique mais limite l'ergonomie et l'accessibilité pour les utilisateurs non techniques.

- Le stockage des données via des fichiers JSON simplifie la mise en œuvre, mais présente des limites en termes de performances, de fiabilité et de gestion simultanée des données

- L'ajout d'une interface graphique permettrait de rendre l'application plus intuitive et d'améliorer la visualisation des informations.

- La migration vers une base de données offrirait une meilleure gestion des données, des performances accrues et une solution plus adaptée à une utilisation à grande échelle.

**Conclusion**

Ce projet a permis de concevoir une application de gestion de bibliothèque basée sur les principes fondamentaux de la programmation orientée objet. Les fonctionnalités principales, telles que la gestion des livres, des utilisateurs, des emprunts et des réservations, ont été mises en œuvre de manière cohérente et fonctionnelle.

Au-delà de l'aspect technique, ce travail a contribué à renforcer la compréhension de la conception logicielle, de l'organisation d'un projet modulaire et de la mise en pratique des règles métier dans une application réelle. Il constitue une base solide pour des évolutions futures, tant dans un cadre académique que professionnel.

Ce travail représente une base évolutive pouvant être réutilisée ou étendue dans des contextes académiques ou professionnels futurs.