Spécifications techniques

[**1) Introduction**](#_3ecbdg4tvhwb) **1**

[**2) Conception de l’architecture**](#_hd4hzv4wfhxo) **1**

[**3) Conception des composants logiciels**](#_r8ogj32rk4bx) **2**

[3.1) Application desktop](#_gsjrjyi4ctjl) 2

[3.2) Application mobile](#_2oqc7mey9bfx) 3

[3.3) Application gestion des stocks](#_88xhazty1j8i) 4

[**4) Conception de la structure des données**](#_z0olqr4j67ra) **4**

[**5) Détail de l’implémentation technique des fonctionnalités**](#_derafvlglj12) **6**

[**6) Choix technologiques : plateformes, frameworks, APIs, langages, etc.**](#_a3gs7aa9v841) **6**

# 

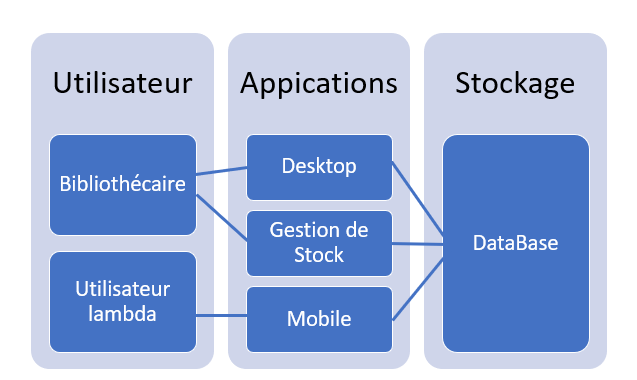
# 1) Introduction

Ce document définit les spécifications techniques des trois produits demandés par le client à savoir, une application desktop qui permet au bibliothécaire de rechercher des items et de gérer les emprunts.

Une application mobile qui permet aux emprunteurs de consulter ses prêts et d’être notifié lorsqu’il doit rendre l’item.

Enfin une application de gestion des stocks qui permet entre autres de consulter, ajouter, mettre à jour et supprimer un item, mais aussi de passer des commandes pour gérer son stock.

# 2) Conception de l’architecture



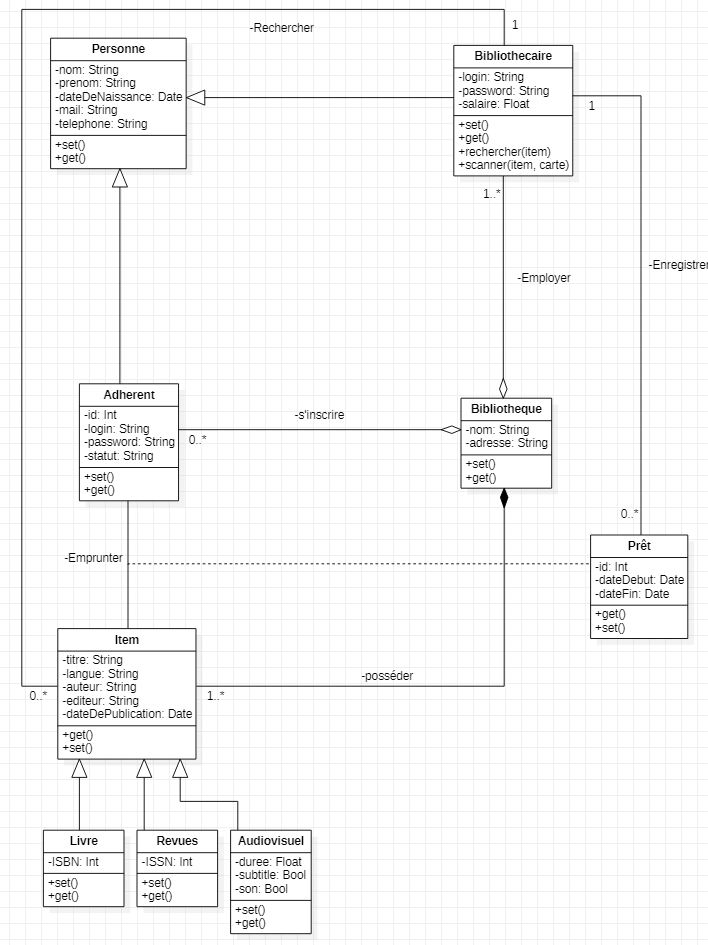
Comme le montre le schéma de l’architecture du projet (ci-dessus), celui-ci est composé de trois applications:

* L’application “Desktop” dans laquelle le bibliothécaire pourra gérer les emprunts
* L’application “Gestion de Stock" sera utilisée par la personne qui gère l’intendance.
* L’application “Mobile” dans laquelle les utilisateurs “lambda”, les “clients” pourront consulter leurs prêts et être notifiés lorsqu’ils doivent rendre un item.

Ces trois applications seront connectées à une même base de données (que nous nommerons le plus souvent DataBase pour des raisons pratiques). Cela permet aux trois applications d’être connectées entre elles; ainsi l’on ajoute un nouvel item dans l’application gestion des stocks, le bibliothécaire pourra immédiatement rechercher cet item dans l’application desktop.

# 3) Conception des composants logiciels

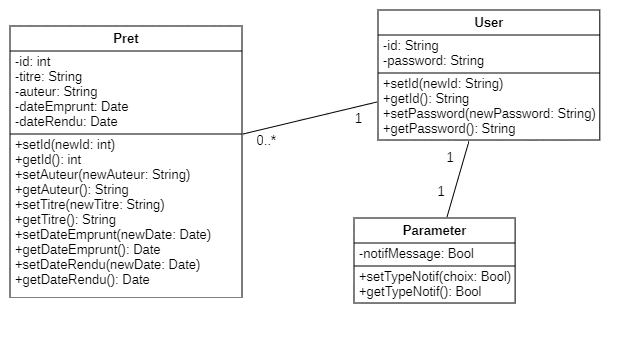
## 3.1) Application desktop



*Diagramme de classe de l’application desktop*

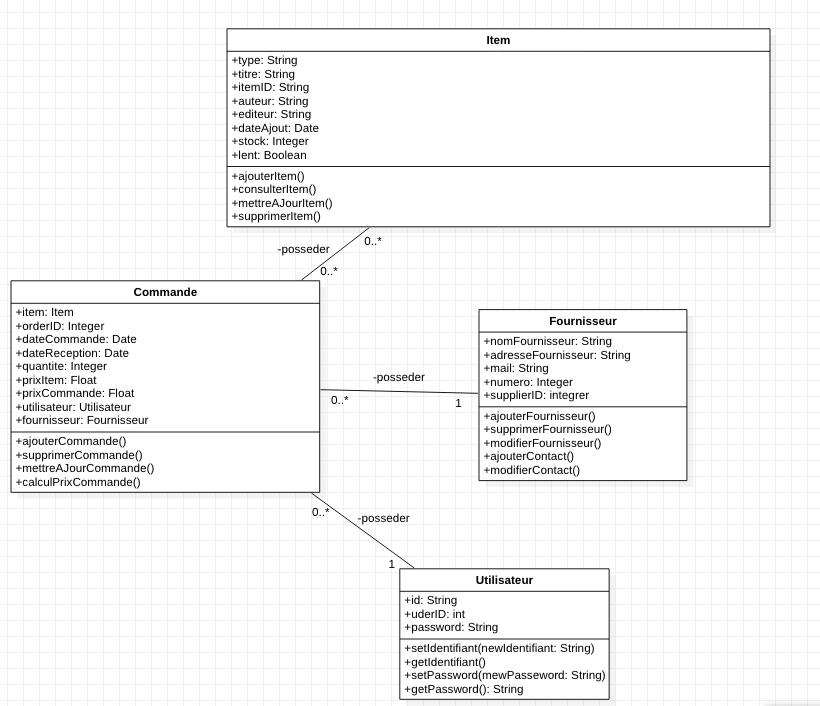
Afin d’améliorer sa lecture, toutes les classes n’ont pas été représentées (exemple: manuscrit). De même pour les get et les set car il sont peu pertinents et n'apportent rien. De plus, cela laisse une certaine liberté au programmeur qui peut donc les coder potentiellement plus efficacement.

## 3.2) Application mobile



*Diagramme de classe de l’application mobile*

## 3.3) Application gestion des stocks

*Diagramme de classe de l’application gestion des stocks*

# 4) Conception de la structure des données

Les trois applications du projet utilisent une base de données commune qui rassemble les informations des **items en stock**, des **prêts d’items**, les informations des **comptes des utilisateurs** et les informations des commandes.

La base de données se compose donc de **5** tables:

* Une table **Users** qui comporte toutes les informations liées aux utilisateurs inscrits à la bibliothèque:
* **userID** de type Integer: Numéro d’identification de l’utilisateur. *(clé primaire)*
* **id** de type String: Identifiant de l’utilisateur.
* **prenom** de type String: Prénom de l’utilisateur.
* **nom** de type String : Nom de l’utilisateur.
* **dateNaissance** de type Date: Date de naissance de l’utilisateur.
* **numTel** de type String: Numéro de téléphone de l’utilisateur.
* **password** de type String: Mot de passe de l’utilisateur.
* Une table **Items** qui comporte toutes les informations relatives sur les items en stock de la bibliothèque :
* **itemID** de type Integer: Le numéro d’identification de l’item. *(clé primaire)*
* **nom** de type String: Le nom/titre de l’item.
* **editeur** de type String: L’éditeur de l’item.
* **auteur** de type String: L’auteur de l’item.
* **dateAjout** de type Date: Date d’ajout de l’item.
* **type** de type String: Le type de l’item.
* **lent** de type Boolean: Indique si l’item est actuellement prêté ou non.
* **stock** de type Integer: Le nombre de cet item en stock.
* Une table **Loans** qui comporte toutes les informations liées aux prêts réalisés à la bibliothèque:
* **loanID** de type Integer: Numéro d’identification du prêt. *(clé primaire)*
* **itemID** de type Integer: Numéro d’identification de l’item. *(clé étrangère venant de la table Items)*
* **userID** de type Integer: Numéro d’identification de l’utilisateur qui a réalisé le prêt. *(clé étrangère venant de la table Users)*
* **dateDébut** de type Date: Date de début du prêt.
* **dateRendu** de type Date: Date de rendu du prêt.
* Une table **Orders** qui comporte toutes les informations relatives aux commandes:
* **orderID** de type Integer: Numéro d’identification de la commande.
* **itemID** de type Integer: Numéro d’identification de l’item. *(clé étrangère venant de la table Items)*
* **supplierID** de type Integer: Numéro d’identification de l’item. *(clé étrangère venant de la table Suppliers)*
* **orderDate** de type Date: Date de la réalisation de la commande.
* **receiptOrderDate** de type Date: Date de réception de la commande.
* **quantite** de type Integer: Quantité commandée.
* **itemPrice** de type Integer: Prix unitaire de l’item commandé.
* **totalPrice** *(calculé en auto)*: Prix total de la commande.
* Une table **Suppliers** qui comporte toutes les informations relatives aux fournisseurs:
* **supplierID** de type Integer: Numéro d’identification du fournisseur.
* **name** de type String: Nom du fournisseur.
* **address** de type String: Adresse du fournisseur.
* **mail** de type String: Mail du fournisseur.
* **phoneNumber** de type String: Numéro de téléphone du fournisseur.

# 5) Détail de l’implémentation technique des fonctionnalités

Les fonctionnalités des applications du projet sont disponibles dans le document “Diagramme des cas d’utilisation”. Nous vous invitons à vous y référer pour plus d'informations sur ce sujet.

Nos trois applications seront développées avec plusieurs outils différents et donc dans des langages différents; comme elles ne sont pas toutes sur les mêmes interfaces. Ainsi dans la partie suivante nous verrons les applications et langages que nous avons choisis pour réaliser notre projet de la meilleure des façons possible.

# 6) Choix technologiques : plateformes, frameworks, APIs, langages, etc.

Comme nous l’avons dit précédemment, nous avons trois applications qui ne seront pas sur les mêmes interfaces, l’une est utilisable sur mobile quand les deux autres sont utilisables sur l’ordinateur.

Ainsi, pour réaliser notre projet nous allons utiliser les applications suivantes :

* Visual Studio
* Microsoft SQL Server

Nous prévoyons de développer nos applications dans les langages suivants :

* Python
* KV
* SQL
* Html
* Css
* Php
* JavaScript

Les framework potentiellement utilisables pour la réalisation des différentes demandes du client sont les suivants :

* Kivy :

Ce framework utilise un langage appelé kv qui va servir à faire notre interface graphique.

Nous avons en partie choisi ce framework car il fonctionne avec le langage Python. Python étant un langage de haut niveau, il sera très rapide de développer une solution.

* Ionic **:**

Ionic est un framework permettant de réaliser des applications mobiles cross-plateform avec du code HTML, CSS, Javascript. Étant donné que nous avons une application iOS et Android à utiliser, Ionic est le meilleur choix.