Institut National de Radioélectricité et de Cinématographie

Enseignement technique secondaire de qualification

Accès aux études supérieures

Avenue Jupiter, 188

1190 Forest

Gestion d’emprunt de livres (Bibliothèques)

SmartLib

Projet personnel de Bako Abdelwadoud

Pour l’obtention du certificat de qualification

Technicien(ne) en informatique

Année scolaire : 2025-2026

Remerciement :

Je tenais d'abord avant de commencer mon rapport, à remercier certaines personnes en commençant par Monsieur Janah pour m'avoir aidé et proposé plusieurs idées à rajouter dans mon TFE ainsi que des documentations à lire pour pouvoir utiliser si possible dans mon projet.

Je voulais aussi remercier Monsieur Mdiker pour le cours qui nous a transmis. Il m'a été nécessaire car j'utilise beaucoup de protocoles qu'on a vu au cours.

Sans oublier mes autres professeurs d'informatique qui m'ont donné beaucoup de connaissance durant ces 3 années.

Il y'a aussi mon père qui n'a pas arrêté de me soutenir malgré les moments où j'avais une baisse de motivation, il m'a toujours accompagné et a pu tester mon projet quand celui-ci était fini.

Table des matières

1. Introduction

1.1 Mise en contexte

La gestion des bibliothèques dans les établissements scolaires repose encore souvent sur des méthodes manuelles, même si elles marchent, elles présentent des limites en termes d'efficacité, de traçabilité et d'accessibilité. L'enregistrement des prêts, le suivi des retours et la consultation des disponibilités nécessitent une intervention humaine régulière, source de perte de temps et d'erreurs potentielles. Les solutions logicielles professionnels ont un coût budgétaire élevée pour des petites structures. Mon projet SmartLib propose une version plus économique et autonome basé sur des logiciels gratuit et libres d'accès.

1.2 Problématique

La question de ce travail consiste à savoir comment faire un système de gestion de livres complet qui va être fiable et économique capable de remplacer les anciennes méthodes sans contraintes techniques ou des dépenses financière. Il s'agit de développer une solution qui va intégrer l'identification des usagers, la gestion des ouvrages, la génération de document PDF et la sauvegarde des données, le tout en respectant un budget matériel et en utilisant des logicielles gratuites.

1.3 Objectif du projet

L'objectif principal est de réaliser un système fonctionnel de gestion de livre automatisée, exploitables sans intervention d'une personne. Cette gestion inclut l'authentification des utilisateurs via un badge, la gestion des références d'ouvrages, l'enregistrement automatique des emprunts et retours, la génération de reçus au format numérique, l'envoie de notification par courriel et un historique des activités.

1.4 Méthodologie

La réalisation du projet a été progressive, commençant par ce dont j'ai besoin et les technologies nécessaires, suivi d'une phase prototype fais en virtualisation sur VMware puis des tests du matériel tels que le badge. Le développement avec les logiciels a été fais petit à petit en testant chaque logiciel avant de les intégrés dans le projet.

2. Matériels

2.1 Matériels utilisé

2.1.1 Raspberry Pi 4

C’est quoi un Raspberry Pi 4 ?

Le Raspberry Pi 4 est un micro-ordinateur auxquelles tout est mis sur une seule carte mère et qui utilise comme système d’exploitation Linux. Il est pratique pour sa taille et offre assez de puissance pour ce type de projet.

Le Raspberry Pi 4 contient un port Ethernet, 4 ports USB, un port jack, une sortie micro SD, un port USB-C pour l’alimentation et des pin GPIO et GND.

Le Raspberry Pi 4 a été officiellement commercialisé le 24 juin 2019 avec plusieurs options différentes de mémoires vives 2GB, 4GB ou 8GB de RAM.

Ajout dans mon projet

Mon Raspberry Pi 4 permet d’héberger mon application de gestion de livres. Elle se connecte au réseau local, permet l’exécution de l’application Python/Tkinter en mode kiosque, la communication avec le module RFID via l’interface SPI, le stockage de la base de données SQLite, la génération des PDF via ReportLab, le partage de fichier avec Samba et une planification de tâches avec Cron

2.1.2 Capteur RFID-RC522

C’est quoi un capteur RFID-RC522 ?

Le module RFID-RC522 est un lecteur/encodeur RFID fonctionnant à 13,56 Mhz. Le RFID (Radio Frequency identification) est une technologie d’identification automatique sans contact qui utilises des ondes radio pour lire des informations stockes dans un tag (badges, cartes, étiquettes, etc …)

Le processus de lecture du badge se déroule par l’alimentation du lecteur qui va émettre un champ électromagnétique à 13,56 MHz.

Le badge capté l’énergie du champ via son antenne et s’alimente sans pile nécessaire et la communication se fait par la transmission de l’identifiant du badge avec son UID unique au lecteur. Le Raspberry Pi 4 va récupére l’UID via SPI et le comparé avec la base de données.

Ajout dans mon projet

J’ai choisi ce capteur pour son cout qui très faibles, en plus de cela il est très facile à intégrer dans ce projet car chaque élève aura un badge avec son propre UID et il est compatible avec l’alimentation du Raspberry Pi 4.

2.1.3 Câbles GPIO (Dupont)

C’est quoi un câble GPIO ?

Un câble GPIO aussi dit câbles Dupont, est un fil électrique qui peut avoir 2 embouts différents mâles-femelles ou 2 embouts les mêmes mâles-mâles ou femelles-femelles. Il permet de relier des composants électroniques.

Ajout dans mon projet

Dans mon projet j’utilises les câbles GPIO pour alimenter mon capteur RFID-RC522 (7 utilisées) et de transmettre les données du capteur à mon Raspberry Pi 4.

2.1.4 Badges/Cartes MIFARE Classic 1k

C’est quoi un Badges/Cartes MIFARE Classic 1k ?

Les Badges MIFARE Classic 1k sont les supports d’identification utilisés par les utilisateurs du système. Chaque Badges contient une Puce RFID et une antenne, permettant une communication sans contact avec le lecteur RC522.

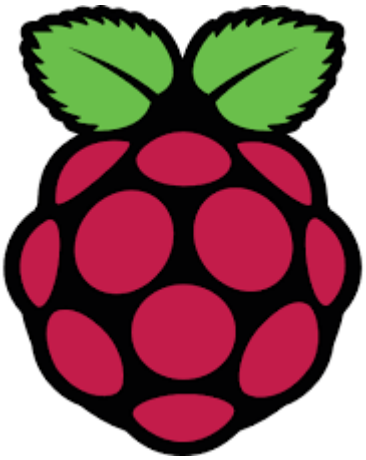
Ajout dans mon projet

Chaque élève va se voir attribuer un badge personnel. Le système enregistre dans la base de données L’UID du badge, le nom complet de l’élève. Lors de l’emprunt l’élève approche son badge du lecteur le système va lire l’UID, il cherche l’UID dans la table user de ma base de données, s’il le trouve l’UID l’emprunt est autorisé dans le cas contraire l’emprunt est refusé.

2.2 Logiciels utilisés

2.2.1 Raspberry Pi OS

Qu'est-ce que Raspberry Pi OS ?



Le système d'exploitation utilisé est Raspberry Pi OS, une distribution Linux dérivée de Debian optimisé pour du micro-ordinateur. Cette plateforme nous donne un environnement stable et documenté, incluant un gestionnaire de paquet qui permet l'installation rapide des dépendances logicielles. Le langage de programmation principale est python dans sa version trois.

Ajout dans mon projet

Raspberry Pi OS est nécessaire car elle est le cœur de mon projet est permet les liaisons avec d'autres protocoles, base de données, capteurs etc.

2.2.2 Python3

Qu'est-ce que Python3 ?

Pyhton3 est la 3ème version de Python (langage de programmation) elle est riche pour sa bibliothèque native et sa communauté qui partage de nouvelles informations. Python permet de structurer le code et de gérer les entrées-sorties, de manipuler les bases de données et interagir avec les interfaces du système.

Ajout dans mon projet

Python est aussi nécessaire pour la réalisation du projet car c'est grâce à ce langage programmation que je peux coder les modules et intégrer de nouveaux protocoles.

2.2.3 Tkinter/Thonny

Qu'est-ce que Tkinter ?

Thonny est un environnement de développement open source. Tkinter est une bibliothèque graphique native de Python qui permet de créer des fenêtres, des tableaux, des boutons, des champs de saisie et de gérer les interactions avec l'utilisateur. Elle permet une exécution immédiate avec l'environnement Debian

Ajout dans mon projet

Tkinter dans mon projet va permettre de créer une application avec toute les références de livres qu'on peut emprunter ainsi que la quantité des livres il y'a aussi une partie administrateur avec l'historique de tous les livres, l'ajout et suppression de badge, l'ajout de livres et des backups.

2.3 Base de données

2.3.1 SQLite

Qu'est-ce que SQLite ?

SQLite est un système de gestion de base de données léger. Il n'a pas besoin d'un serveur pour fonctionner il va enregistrer toutes les données dans un fichier avec l'extension « .db » qui va être stocké localement sur le Raspberry Pi.

Ajout dans mon projet

Dans mon projet, j'utilise SQLite pour stocker des informations. Mon fichier se nomme smartlib.db et posséde 4 tables différentes loans (emprunts), user (utilisateurs/RFID), books (livres) et sqlite_sequence (système)

2.3.2 DB Browser for SQLite

Qu'est-ce que DB Browser for SQLite ?

DB Browser for SQLite est un logiciel gratuit et libre de droit qui permet de visualiser, crée où modifier les bases de données SQLite.

Ajout dans mon projet

DB Browser for SQLite m'a permis de visualiser ma base de données lors de plusieurs tests d'emprunt pour voir que tout fonctionnait correctement.

2.3.3 AlwaysData

Qu'est-ce que AlwaysData ?

AlwaysData est un hébergeur web gratuit qui fournit un espace de stockage distant accessible via les protocoles SSH et SFTP. Cette plateforme permet de stocker et de gérer des fichiers en ligne de manière sécurisé.

Ajout dans mon projet

Dans le cadre de mon projet, AlwaysData va jouer un rôle important car elle va héberger des copies de la base de données SQLite. Cette solution permet une sécurité si mon il y'a un problème avec le Raspberry Pi ou corruption du système il y'aura une copie de la base de données sur AlwaysData. L'intégration s'éffectue via la bibliothèque python Paramiko qui établit une connexion SFTP entre le Raspberry Pi et l'hébergeur permettant le transfert des fichiers de sauvegarde. La sauvegarde se fait manuellement depuis l'interface administrateur.

2.4 Protocoles et Services (Modules)

2.4.1 mDNS (Avahi)

Qu'est-ce que mDNS

mDNS (multicast DNS) est un protocole de résolutions de noms qui permet aux appareils d'un réseau local de se découvrir et de communiquer en utilisant des noms simples plutôt que des adresses IP numériques. Cette technologie élimine le besoin de configurer manuellement les adresses IP.

Ajout dans le projet

Dans mon projet SmartLib, mDNS est activé pour attribuer au Raspberry Pi un nom de domaine local tel que smartlib.local. Les administrateurs peuvent ainsi accéder au partage de fichier en utilisant ce nom au lieu de l'adresse IP.

2.4.2 Samba

Qu'est-ce que Samba ?

Samba est un logicielle open source qui implémente les protocoles de partage de fichiers et d'impression utilisés nativement par les systèmes Windows. Elle permet à des machines Linux de partager des dossiers accessibles depuis n'importe quel poste sur le réseau local.

Ajout dans le projet

Samba, dans SmartLib est configuré pour partager le dossier contenant les reçus PDF générés. Les administrateurs peuvent ainsi se connecter depuis n'importe quel ordinateur du réseau, les consulter et les imprimer.

2.4.3 SMTPLib

Qu’est-ce que SMTP ?

SMTPLib est un module de Python qui implémente le protocole SMTP (Simple Mail Transfer Protocol) utilisé pour l’envoie de courriels. Il permet d’établir une connexion avec un serveur de messagerie. Ce module est intégré à Python ce qui évite une installation.

Ajout dans le projet

Dans SmartLib, SMTPLib est utilisé pour envoyer automatiquement un courriel à chaque emprunt de livre. Le message contient les détails de l’emprunt et inclut en pièce jointe le reçu PDF. L’envoie est effectué de manière asynchrone pour ne pas bloquer l’interface utilisateur pendant la transmission.

2.4.4 Logging/SysLogHandler

Qu’est-ce que Logging et SysLogHandler ?

Le module Logging de Python fournit un système complet de journalisation des évènements d’une application permettant de tracer les opérations, les erreurs et les informations de débogage. SysLogHandler est un composant de ce module qui permet d’envoyer les journaux vers le service de jouranlisation du système d’exploitation.

Ajout dans le projet

Ce module est configuré pour enregistrer chaque événement comme la détection du badge, l’emprunt d’un livre, le retour d’un livre, connexion à l’interface d’administration ou erreur de lecture.

2.4.5 ReportLab

Qu’est-ce que ReportLab ?

ReportLab est une bibliothèque Python spécialisée dans la génération de documents au format PDF. Elle permet de créer des fichiers structurés avec une mise en page précise, d’y insérer du texte formaté, des tableaux et des graphiques.

Ajout dans le projet

ReportLab est employée pour générer automatiquement un reçu d’emprunt à chaque fois qu’un livre est prêté. Le document PDF crée contient les informations essentielles tels que le nom de l’emprunteur, le titre du livre avec son auteur, la date d’emprunt et la date de retour prévue. Ce reçu est envoyé par courriel.

2.4.6 Threading

Qu’est-ce que Threading ?

Le module Threading de Python permet d’exécuter plusieurs tâches simultanément au sein d’un même programme, grâce à la création de threads d’exécution parallèles. Cette fonctionnalité est bien pour maintenir la réactivité d’une application.

Ajout dans le projet

Dans SmartLib, Threading est utilisé pour exécuter l'envoi des courriels en arrière-plan. Lorsqu'un emprunt est validé, le reçu PDF est généré puis l'envoi du message est transmis à un thread séparé. Cela permet à l'interface de rester immédiatement disponible pour l'opération suivante, sans attendre la fin de la transmission du courriel, ce qui améliore significativement le confort d'utilisation.

2.4.7 Cron

Qu’est-ce que Cron ?

Cron est un service intégré aux systèmes de type Unix qui permet de planifier l’exécution automatique de commande ou de scripts à des intervalles qu’on va définir. C’est utile pour des tâches de maintenance, les sauvegardes régulières et les nettoyages automatisé.

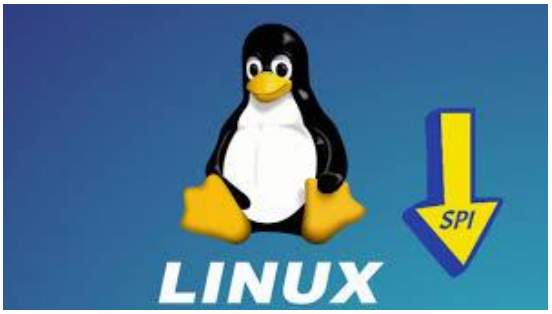
Ajout dans le projet

Dans SmartLib, Cron est configuré pour exécuter quotidiennement une sauvegarde de la base de données à une heure de faible activité. Il planifie également la suppression automatique des fichiers de sauvegarde trop anciens et le nettoyage des reçus PDF dépassant une certaine durée de conservation. C’est utile car il n’y pas besoin d’intervention humaine.

2.4.8 SPI (Linux)

Qu’est-ce que SPI ?

SPI (Serial Peripheral Interface) est un protocole de communication série synchrone qui est très utilisé pour relier un microcontrôleur ou microprocesseur à des périphériques externes tels que des capteurs, écrans ou des modules de communication. Il permet un transfert rapide sur de courtes distances en utilisant des câbles GPIO.



Ajout dans le projet

Le protocole SPI est utilisé pour la communication entre le Raspberry Pi et le module RFID-RC522. Il transmet les commandes de lecture du lecteur et reçoit les identifiant des badges détectés.

2.5. Autres Applications

2.5.1 RealVNC

Qu’est-ce que RealVNC ?

RealVNC est une application de bureau à distance qui permet de visualiser et de contrôle un ordinateur depuis un autre ordinateur via internet ou un réseau local.

Ajout dans mon projet

Elle me permet de me connecter à distance sans devoir tout le temps être obligatoirement sur mon Raspberry Pi.

2.5.2 VMware Workstation

Qu’est-ce que VMware Workstation ?

VMware Workstation est un logiciel de virtualisation qui permet de créer et d’exécuter plusieurs systèmes d’exploitation (machines virtuelles sur un seul ordinateur physique.

Ajout dans mon projet

Grâce à VMware Workstation j’ai pu d’abord faire des tests pour la création de mon application et le code Python avant de passer en physique sur mon Raspberry Pi

3. Prix

3.1 Prix du projet

3.1.1 Prix du matériel

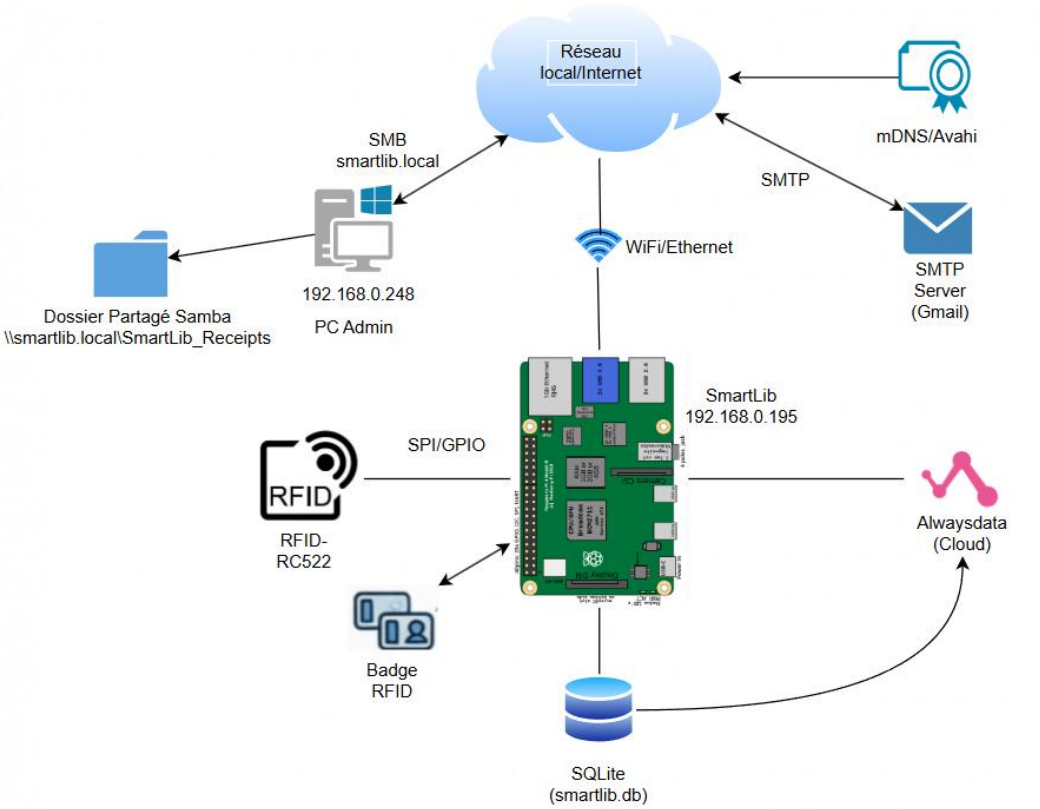
COMPOSANT	PRIX UNITAIRE	QUANTITE	TOTAL
RASPBERRY PI 4 MODELE B, 8GB	55€	1	55€
CÂBLES FEMELLES-FEMELLES	0,06€	7	0,42€
RFID-RC522	0,68€	1	0,68€
BADGE RFID MIFARE	0,99€	4	3,96€
TOTAL MATÉRIEL			60€

3.1.2 Prix des logiciels

LOGICIEL	PRIX UNITAIRE	QUANTITE	TOTAL
THONNY	0€	1	0 €
ALWAYSDATA	0€	1	0€
REALVNC	0€	1	0€
VMWARE WORKSTATION	0€	1	0€
SQLITE	0€	1	0€
DB BROWSER FOR SQLITE	0€	1	0€
TOTAL LOGICIEL			0€

4. Schéma

4.1 Topologie réseaux



Explication de la topologie :

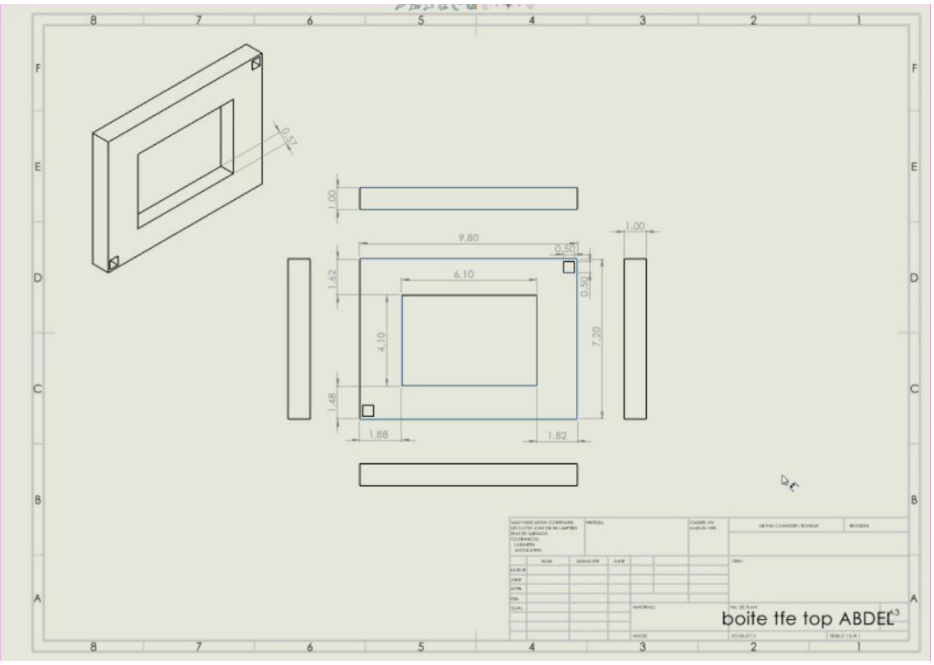
Comme vous pouvez le voir au centre le Raspberry Pi qui va exécuter l'application Python/Tkinter et va stocker la base de données SQLite localement et va transmettre aussi la base de données vers AlwaysData. L'administrateur peut accéder à tout les reçus via Samba depuis un autre ordinateur en local. Le serveur SMTP qui va faire l'envoie de notifications par courriels.

4.2 Schéma mécanique

4.2.1 Schéma mécanique (plan)

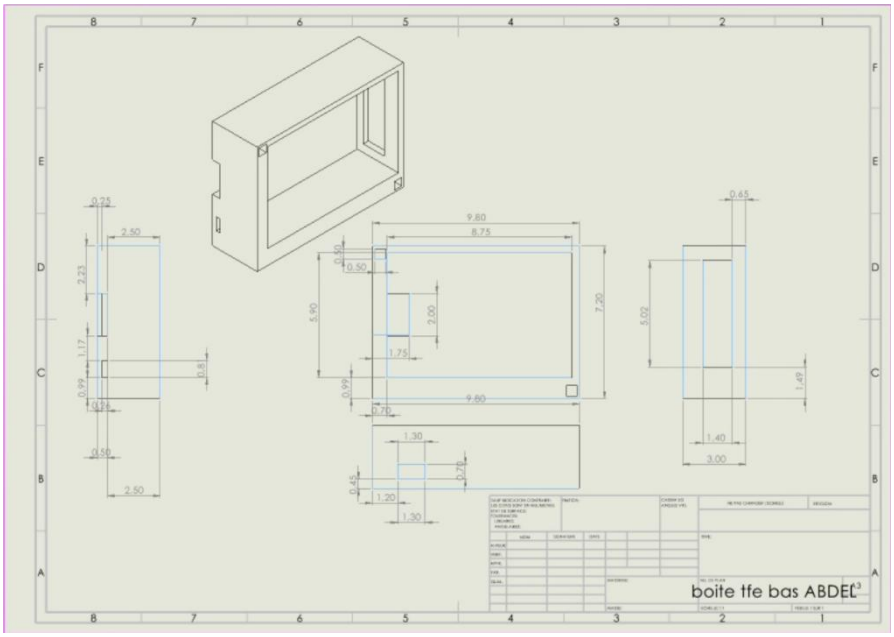
<u>Vue du dessus</u>

Sur ce schéma on peut voir un trou de 6 cm sur 4 cm qui va accueillir le lecteur RFID-RC522 pour réaliser tous les emprunts



<u>Vue du bas</u>

Sur ce schéma on peut voir la boite dans lequel va être le Raspberry Pi ainsi que les trous pour faire passer la clé USB avec mon OS ainsi que le câble Ethernet.

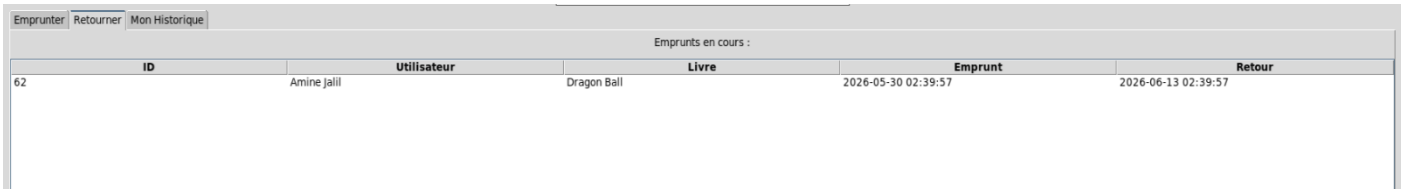


5. Le travail réalisé

5.1 Explication des interfaces

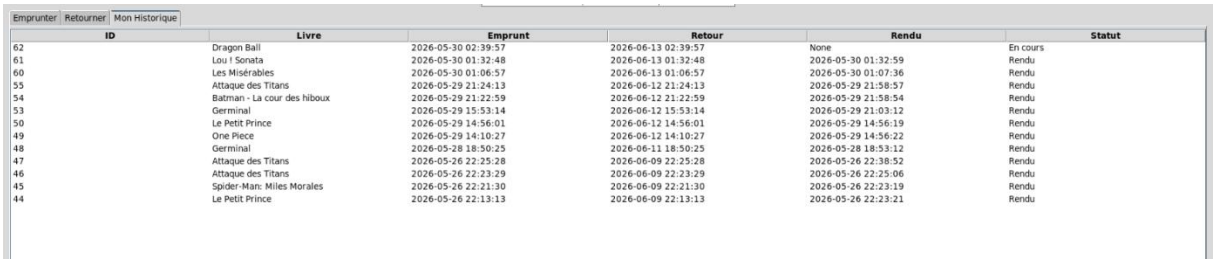
5.1.1 Interface principale (client)

L'écran principale contient tous les livres qu'il y a ainsi que le nombre de disponibilités par livres, on peut aussi effectuer une recherche et filtrer par genre. Il y'a aussi 2 autres onglets « Retourner » et « Mon historique ». Il y'a aussi 2 boutons « Emprunter » et « Admin ».



1) Dans l'onglet « Retourner » on peut apercevoir les emprunts en cours ainsi quand ils doivent remettre le livre emprunter.

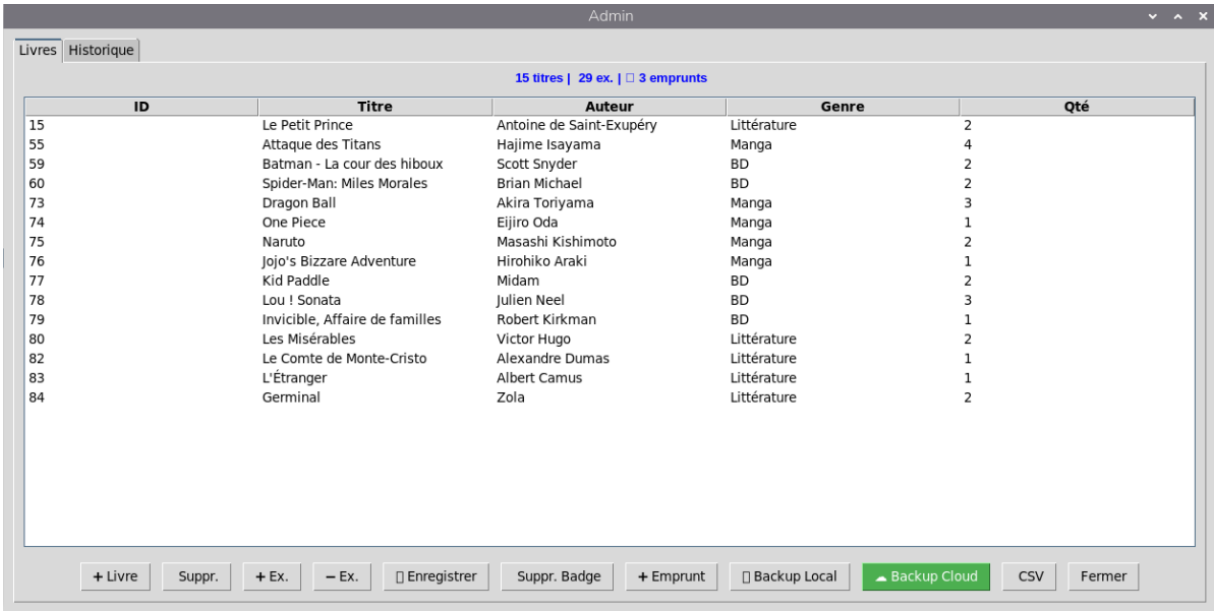
2)



ID	Livre	Emprunt	Retour	Rendu	Statut
62	Dragon Ball	2026-05-30 02:39:57	2026-06-13 02:39:57	None	En cours
61	Lou ! Sonata	2026-05-30 01:32:48	2026-06-13 01:32:48	2026-05-30 01:32:59	Rendu
60	Les Misérables	2026-05-30 01:06:57	2026-06-13 01:06:57	2026-05-30 01:07:36	Rendu
55	Attaque des Titans	2026-05-29 21:24:13	2026-06-12 21:24:13	2026-05-29 21:58:57	Rendu
54	Batman - La cour des hiboux	2026-05-29 21:22:59	2026-06-12 21:22:59	2026-05-29 21:58:54	Rendu
53	Germinal	2026-05-29 15:53:14	2026-06-12 15:53:14	2026-05-29 21:03:12	Rendu
50	Le Petit Prince	2026-05-29 14:56:01	2026-06-12 14:56:01	2026-05-29 14:56:19	Rendu
49	One Piece	2026-05-29 14:10:27	2026-06-12 14:10:27	2026-05-29 14:56:22	Rendu
48	Germinal	2026-05-28 18:50:25	2026-06-11 18:50:25	2026-05-28 18:53:12	Rendu
47	Attaque des Titans	2026-05-26 22:25:28	2026-06-09 22:25:28	2026-05-26 22:38:52	Rendu
46	Attaque des Titans	2026-05-26 22:23:29	2026-06-09 22:23:29	2026-05-26 22:25:06	Rendu
45	Spider-Man: Miles Morales	2026-05-26 22:21:30	2026-06-09 22:21:30	2026-05-26 22:23:19	Rendu
44	Le Petit Prince	2026-05-26 22:13:13	2026-06-09 22:13:13	2026-05-26 22:23:21	Rendu

Dans l'onglet « Mon historique » on peut voir l'historique de tous les emprunts que le profil a fait et voir le statut de ces emprunts.

5.1.2 Interface administrateur



Dans l'interface administrateur il y'a 2 onglet « Livres » et « Historique », dans l'onglet « Livre » on peut y voir tous les livres qui sont disponibles on peut aussi en rajouter grâce au bouton « +Livre » et supprimer un livre avec le bouton « Suppr. »

Lorsqu'on appuie sur ce bouton une fenêtre s'ouvre des champs de saisie qu'on doit remplir pour introduire le nouveau livre.

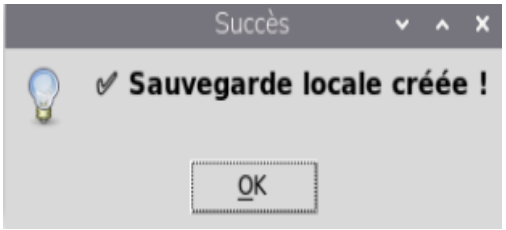
Ensuite, les 2 boutons suivant « +Ex » et « -Ex » sont des boutons pour soit ajouter des exemplaires d'un livre ou pour retirer un exemplaire.

Les 2 prochains boutons à suivre, sont les boutons « Enregistrer » et « Suppr. Badge », ces boutons sert soit à enregistrer un badge RFID alors il faut introduire le nom de l'élève.

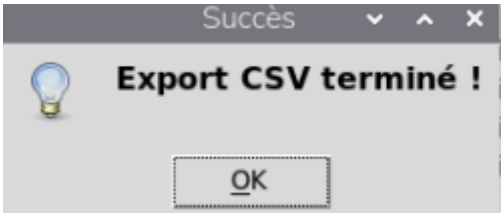
Ou bien il faut supprimer un badge à l'aide du bouton « Suppr. Badge ».

Sur cette page si plusieurs badges son enregistrer il y'a une liste déroulante pour voir tous les badges.

Il reste encore les boutons « backup local » et « backup cloud », ces 2 boutons servent tous les 2 faire une sauvegarde de la base de données mais une va être copier en local et l'autre va être copier vers l'hébergeur AlwaysData.



Il reste plus que le bouton CSV qui permet de faire un exports CSV.



5.2 Explication du fonctionnement de la gestion de livres (code)

5.2.1 Implémentation de ma base de données

```
USER_HOME = "/home/abdel"
DB_PATH = "/home/abdel/tfe/smartlib.db"
```

Sur cette image DB_PATH montre le chemin de ma base de données SQLite.

```
#Base de donnée
def init_db():
    conn = sqlite3.connect(DB_PATH)
    c = conn.cursor()
```

Sur cette image on voit clairement que la fonction « conn = sqlite3.connect(DB_PATH) va initialiser la base de données avec le chemin montré plus tôt. Le sqlité3.connect établit la connexion avec la base de données.

```
def get_available_books(genre_filter=None, search_query=None):
    conn = sqlite3.connect(DB_PATH); c = conn.cursor()
```

Par exemple, cette ligne de code va voir dans la base de données SQLite les livres qui sont disponibles ou non.

5.2.2 Implémentation de l'envoie de courriel

```
SMTP_EMAIL = "abdelwadoud.bako@gmail.com"
SMTP_PASSWORD = "uhcamcmeuuonrlvt"
DESTINATAIRE = "abdelwadoud.bako@inraci.be"
CURRENT_USER = "Invité"
```

```python
#EMAIL
def send_email_with_pdf(pdf_path, book_title):
    if not SMTP_EMAIL or not SMTP_PASSWORD: return

    wait = 0
    while not os.path.exists(pdf_path) and wait < 5:
        time.sleep(0.5); wait += 0.5

    if not os.path.exists(pdf_path):
        logger.error(f"❌ PDF introuvable pour email: {pdf_path}")
        return

    try:
        msg = MIMEMultipart()
        msg['From'], msg['To'], msg['Subject'] = SMTP_EMAIL, DESTINATAIRE, f"📚 SmartLib : {book_title}"
        msg.attach(MIMEText(f"Livre emprunté : {book_title}\nDate : {datetime.now().strftime('%d/%m/%Y %H:%M')}", 'plain'))

        with open(pdf_path, "rb") as f:
            part = MIMEBase('application', 'octet-stream')
            part.set_payload(f.read())
        encoders.encode_base64(part)
        part.add_header('Content-Disposition', f'attachment; filename="{os.path.basename(pdf_path)}"')
        part.add_header('Content-Type', 'application/pdf')
        msg.attach(part)

        s = smtplib.SMTP('smtp.gmail.com', 587)
        s.starttls()
        s.login(SMTP_EMAIL, SMTP_PASSWORD)
        s.sendmail(SMTP_EMAIL, DESTINATAIRE, msg.as_string())
        s.quit()
        logger.info(f"📧 Email envoyé avec PDF: {os.path.basename(pdf_path)}")
    except Exception as e:
        logger.error(f"❌ Erreur Email: {e}")
```

Pour envoyer des mails à l'admin pour voir tous les emprunts j'ai dû installer la bibliothèque SMTPLib la fonction pour l'envoie se fait en plusieurs étapes. Tout d'abord la fonction vérifie les informations Gmail soit bien configuré puis patiente la création du PDF. Lorsque ceci est fait la fonction crée un courriel avec le corps du message dans lequel il va attachée le PDF avec ce message. Quand tout ceci est fait c'est le moment de l'envoie du mail sur le port 587 avec smtplib.SMTP.

5.2.3 Implémentation du RFID-RC522

```python
#RFID
def read_rfid():
    try:
        from mfrc522 import SimpleMFRC522
        r = SimpleMFRC522()
        uid, _ = r.read()
        logger.info(f"✅ RFID détecté - UID: {uid}")
        return str(uid)
    except Exception as e:
        logger.error(f"❌ Erreur lecture RFID: {e}")
        return None
    finally: GPIO.cleanup()

def register_student_rfid():
    win = tk.Toplevel(root); win.title("📝 Enregistrer Badge"); win.geometry("350x200")
    tk.Label(win, text="Nom élève :").pack(pady=5)
    name_e = tk.Entry(win, width=35); name_e.pack(pady=5)
    def save():
        name = name_e.get().strip()
        if not name: return messagebox.showwarning("Erreur", "Nom requis")
        uid = read_rfid()
        if not uid: return messagebox.showerror("Erreur", "Lecture échouée")
        conn = sqlite3.connect(DB_PATH); c = conn.cursor()
        try:
            c.execute("INSERT INTO users VALUES (?,?,?)", (uid, name, ""))
            conn.commit()
            logger.info(f"✅ Nouveau badge enregistré: {name} (UID: {uid})")
            messagebox.showinfo("Succès", f"✅ {name} enregistré\nUID: {uid}"); win.destroy()
        except sqlite3.IntegrityError: messagebox.showinfo("Info", "Badge déjà enregistré")
        finally: conn.close()
    tk.Button(win, text="📡 Lire & Enregistrer", command=save, bg="green", fg="white").pack(pady=10)
```

Dans la fonction « read_rfid » on va d'abord importer le module SimpleMFRC522 qui permet de communiquer avec le lecteur RFID-RC522 puis va initialiser le lecteur lorsqu'on passe le badge sur le lecteur le système va journaliser le tout dans les logs. Puis la fonction « register_student_rfid » permet d'enregister un nouvel élève en associant son nom à son badge RFID dans la base de données.

5.2.4 Génération des PDF (ReportLab)

```python
#PDF GÉNÉRATION
def generate_receipt(book_id, borrow_date, due_date, username):
    conn = sqlite3.connect(DB_PATH); c = conn.cursor()
    c.execute("SELECT title, author, genre FROM books WHERE id=?", (book_id,))
    b = c.fetchone(); conn.close()
    if not b: return None
    title, author, genre = b
    fname = f"reçu_emprunt_{book_id}_{datetime.now().strftime('%Y%m%d_%H%M')}.pdf"
    local = os.path.join(USER_HOME, fname)
    samba = os.path.join(SAMBAPATH, fname)
    os.makedirs(SAMBAPATH, exist_ok=True)

    pdf = canvas.Canvas(local, pagesize=A4)
    w, h = A4
    pdf.setFont("Helvetica-Bold", 16); pdf.drawString(50, h-50, "□ SmartLib - Reçu")
    pdf.setFont("Helvetica", 12)
    pdf.drawString(50, h-70, f"Date : {datetime.now().strftime('%d/%m/%Y %H:%M')}")
    pdf.drawString(50, h-90, f"Utilisateur : {username}")
    pdf.setFont("Helvetica-Bold", 12); pdf.drawString(50, h-120, "Livre :")
    pdf.setFont("Helvetica", 12)
    pdf.drawString(70, h-140, f"Titre : {title}")
    pdf.drawString(70, h-160, f"Auteur : {author}")
    pdf.drawString(70, h-180, f"Genre : {genre}")
    pdf.drawString(70, h-200, f"Emprunté le : {borrow_date}")
    pdf.drawString(70, h-220, f"À rendre : {due_date}")
    pdf.save()

    try: shutil.copy2(local, samba)
    except Exception as e: print(f"Erreur Samba: {e}")
    return local, title
```

Cette fonction génère automatiquement un reçu PDF à chaque emprunt de livre. Elle récupère les informations du livre, crée un document structuré avec ReportLab, le sauvegarde localement, le copie vers le dossier partagé Samba, puis retourne les données nécessaires à l'envoi par courriel.

Dans cette fonction, il y'a une connexion avec la base de données pour récupérer les informations de celle-ci. Lorsque ceci est fait elle vérifie que le livre existe bien dans la base de données. Quand le livres est bien là il commence la création du PDF avec cette ligne de code « pdf = canvas.Canvas(local, pagesize=A4), puis met les informations du livres avec le titre, l'auteur, le genre, emprunté à quelle date et à remettre à quelle date le tout avec le nom de l'utilisateur.

La fonction va aussi créer une copie des PDF en local et sur le dossier partagé SAMBA.

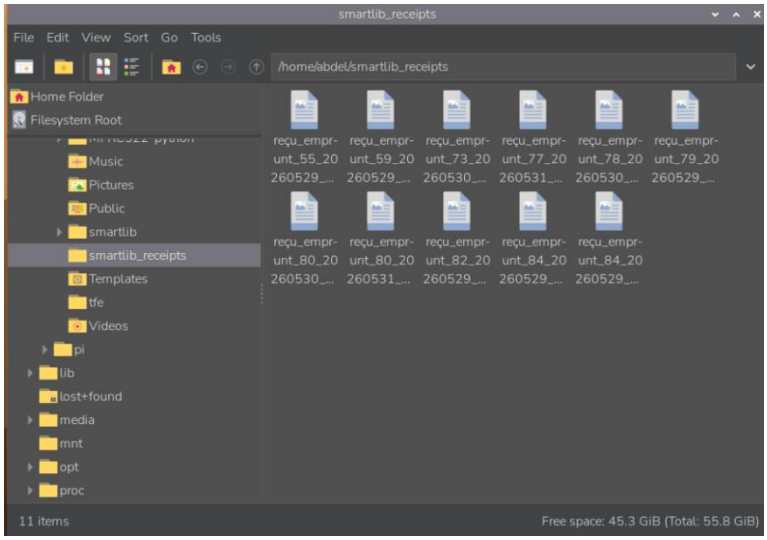
5.3 Fonctionnalités

5.3.1 Utilisation de Samba et mDNS

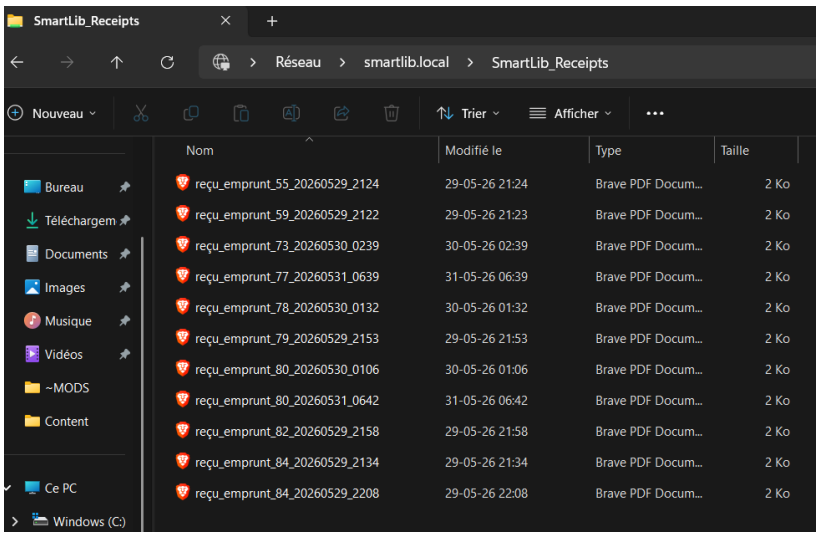
Tout d'abord, dans mon application j'utilise Samba pour le partage de dossier ainsi je peux me connecter localement à ce fichier depuis un autre ordinateur. Le partage que j'ai fait me permet de me connecter au dossier avec tout les reçus créer. Samba peut permettre à l'administrateur de se connecter à distance sans devoir aller directement sur le Raspberry Pi.

Grâce à mDNS, l'utilisation de Samba est plus simple car avec mDNS je n'ai plus besoin d'une IP pour me connecter à distance mais juste d'un nom de domaine comme « \\smartlib.local\SmartLib_Receipts » qui va se connecter à mon partage de dossier

1) Ici on peut apercevoir des emprunts qui on été créer dans le dossier « smartlib_receipts »



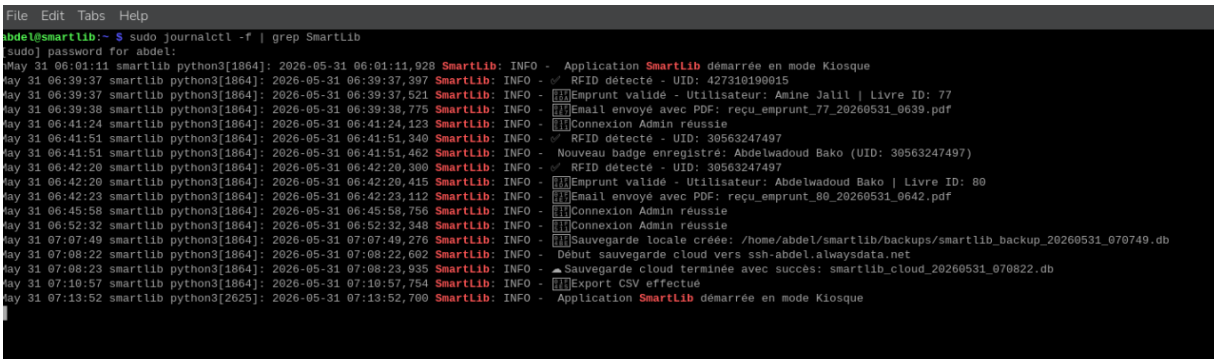
2) Dans la 2 image on peut voir le partage de dossier depuis une machine Windows. On peut aussi voir le chemin sans l'adresse IP mais «. local »



5.3.2 Utilisation de Logging SysLogHandler

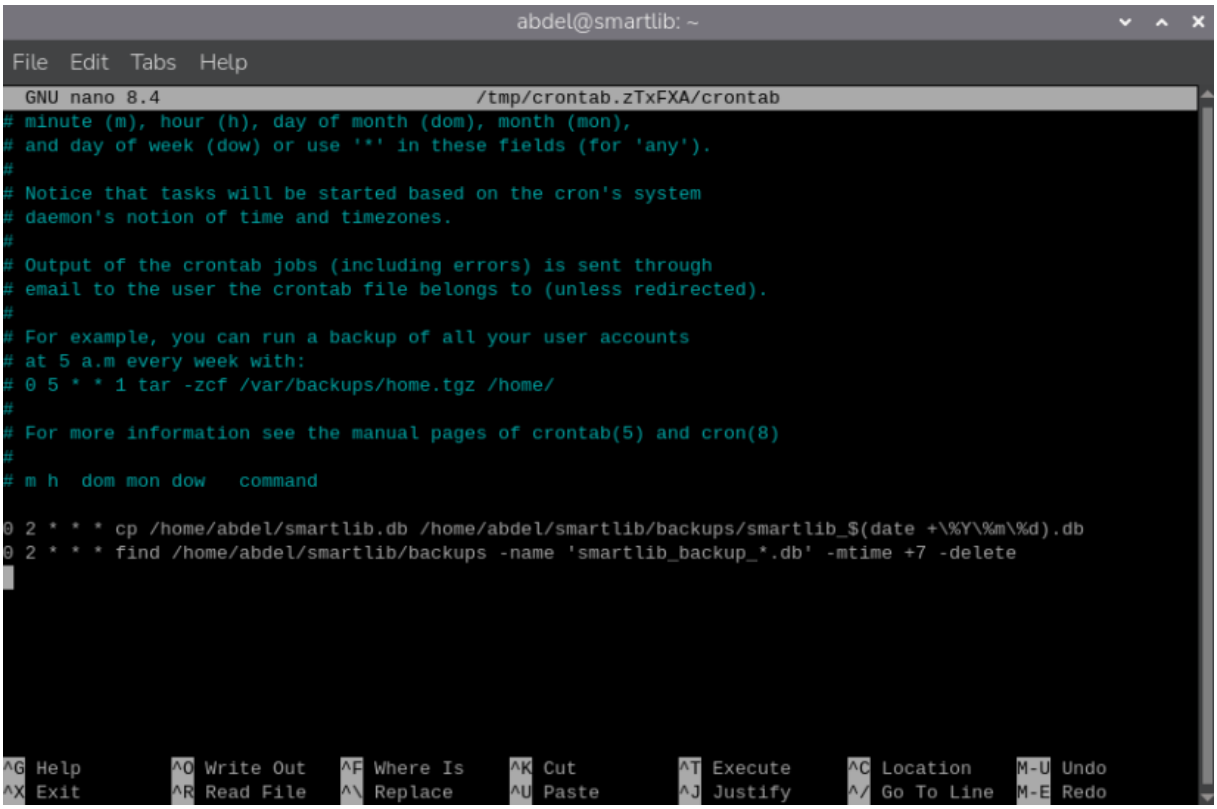
```python
#Configuration Syslog
def setup_logging():
    logger = logging.getLogger("SmartLib")
    logger.setLevel(logging.INFO)
    try:
        handler = logging.handlers.SysLogHandler(address='/dev/log')
        formatter = logging.Formatter('%(asctime)s SmartLib: %(levelname)s - %(message)s')
        handler.setFormatter(formatter)
        logger.addHandler(handler)
    except Exception as e:
        print(f"Syslog non dispo, bascule sur console: {e}")
        console = logging.StreamHandler()
        logger.addHandler(console)
    return logger

logger = setup_logging()
```

Le rôle de cette fonction configure le système de journalisation (logging) de SmartLib. Elle permet d'enregistrer tous les événements importants tels que les emprunts, les erreurs, les connexion).



Ici on peut apercevoir tous les logs qui s'affichent avec tout les RFID qui sont scannés, les emprunts réaliser, l'ajout d'un badge, les courriels envoyés, les export CSV, la sauvegarde dans le cloud de AlwaysData et le démarrage de l'application en mode Kiosque. Le logger va être mis dans le code dans plusieurs fonctions pour récupérer chaque élément du code.

5.3.3 Cron et Sauvegarde

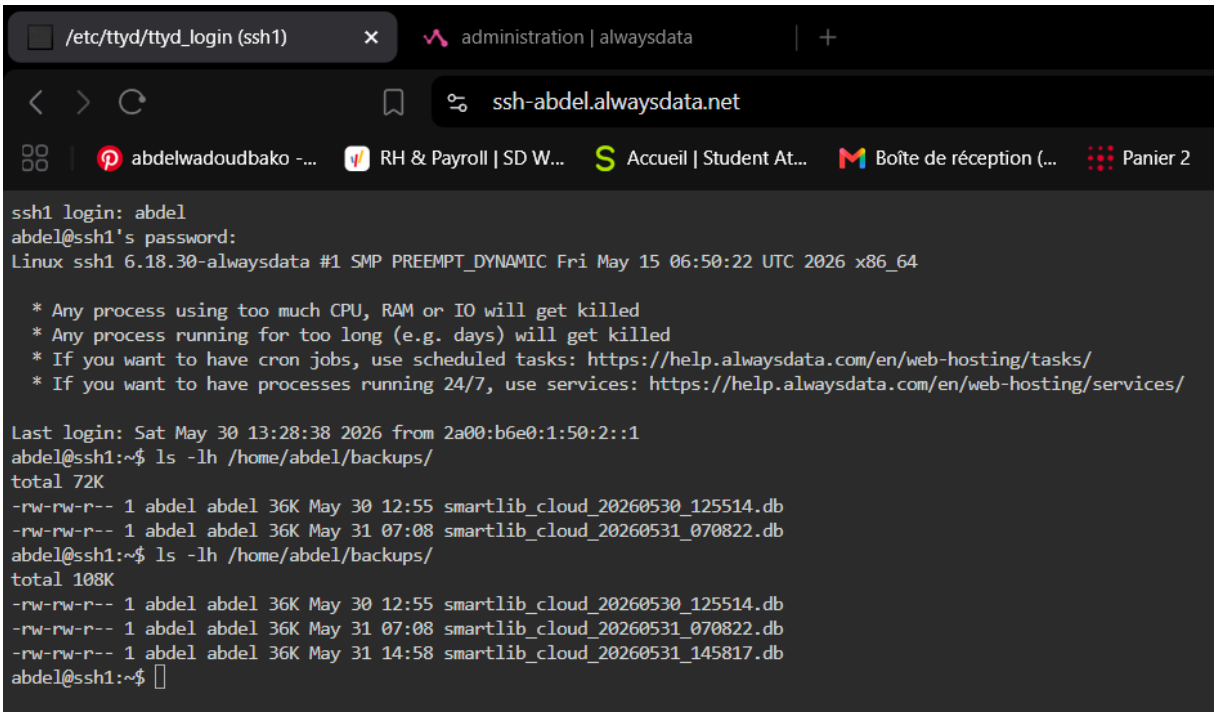


Grâce à la planification de tâche (Cron, je fais une suppression automatique des vieux backups de la base de données pour éviter une surcharge du stockage du Raspberry Pi.

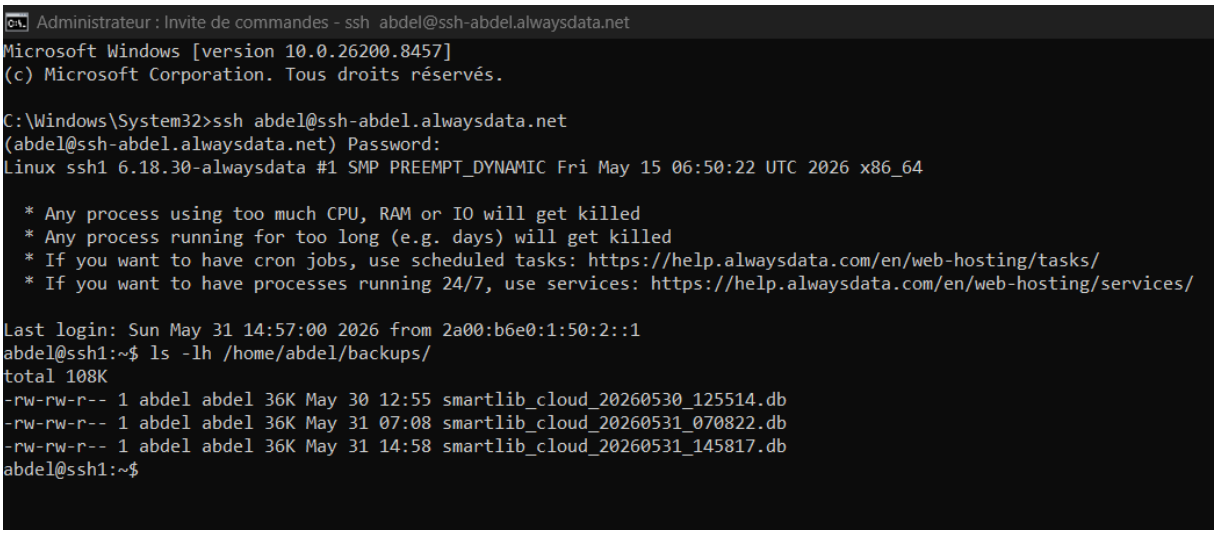
```python
#Nettoyage des PDF expirés
def cleanup_old_pdfs(days=30):
    now = datetime.now()
    for path in [USER_HOME, SAMBAPATH]:
        if not os.path.exists(path): continue
        for filename in os.listdir(path):
            if filename.startswith("reçu_emprunt_") and filename.endswith(".pdf"):
                try:
                    parts = filename.split("_")
                    date_str = parts[-2] + "_" + parts[-1].replace(".pdf", "")
                    file_date = datetime.strptime(date_str, "%Y%m%d_%H%M")
                    if (now - file_date).days > days:
                        os.remove(os.path.join(path, filename))
                        logger.info(f"  PDF supprimé (nettoyage auto): {filename}")
                except Exception as e:
                    logger.error(f"Erreur suppression PDF: {e}")
```

Dans cette fonction on reprend la date de maintenant, puis on va suivre le chemin des reçus des emprunts ici c'est « SAMBAPATH » car c'est une variable qui contient le chemin du dossier des reçus puis vérifie les PDF vieux de plus de 30 jours

5.3.4 Utilisation de AlwaysData



L'utilisation de AlwaysData, ici elle me permet de stocker et de voir les backups de ma base de données. Elle me sert de sauvegarde externe qui est accessible à distance via les protocoles SSH et SFTP. Sur cette image je me connecte directement depuis mon moteur de recherche Brave et je suis connecté en SSH.



Voici la connexion AlwaysData depuis l'invite de commande Windows. Je me connecte toujours en SSH mais cette fois ci précise le nom de l'utilisateur.

6. Conclusion

6.1 Résumé du travail

Le projet SmartLib à permis de développer un système complet de gestion de livres automatisé qui fonctionne sur Raspberry Pi 4. L'application intègre un module RFID pour l'authentification des utilisateurs, une interface graphique développé par python Tkinter, une base de données SQLite, une génération automatique des reçus PDF envoyés par courriel et partagé sur le réseau via Samba. Un système de journalisation (log) avec SysLog et une synchronisation cloud vers AlwaysData.

6.2 Objectifs

Les objectifs ne sont pas totalement atteints malgré une grosse partie réalisée avec tout ce que j'ai déjà cité mais je voulais intégrer le protocole LDAP (Lightweight Directory Acces Protocol). Je voulais que le Raspberry Pi dépende d'un serveur central Active Directory pour authentifier les élèves. Malheureusement la version de mon Raspberry Pi est une version ancienne donc ne veut pas prendre en compte la dépendance LDAP malgré une installation forcée du module celui-ci ne voulais pas fonctionner malgré les tests sur une machine virtuelle et sur une machine physique celui-ci ne voulait pas s'effectuer.

6.3 Améliorations possibles

Plusieurs améliorations sont possibles comme une interface web qui permettrait une consultation à distance des livres disponibles, l'intégration du LDAP qui améliorerait la gestion des utilisateurs. Une application mobile pour les administrateurs pour pouvoir avoir l'historique et les emprunts réaliser.

7. Sitographie

7.1 Liens / Documentations

Administration | alwaysdata. (s. d.). https://admin.alwaysdata.com/login/?next=/ssh/ , consulté le 14 mai 2026

Miguelbalboa. (s. d.). *GitHub - miguelbalboa/rfid : Arduino RFID Library for MFRC522*. GitHub. https://github.com/miguelbalboa/rfid, consulté le 10 janvier 2026

Avahi (Français) - ArchWiki. (s. d.). https://wiki.archlinux.org/title/Avahi_(Fran%C3%A7ais), consulté le 15 avril 2026

Google LLC. (2024). Gmail SMTP configuration. Google Workspace Administrator Help. https://support.google.com/a/answer/176600, consulté le 17 avril 2026

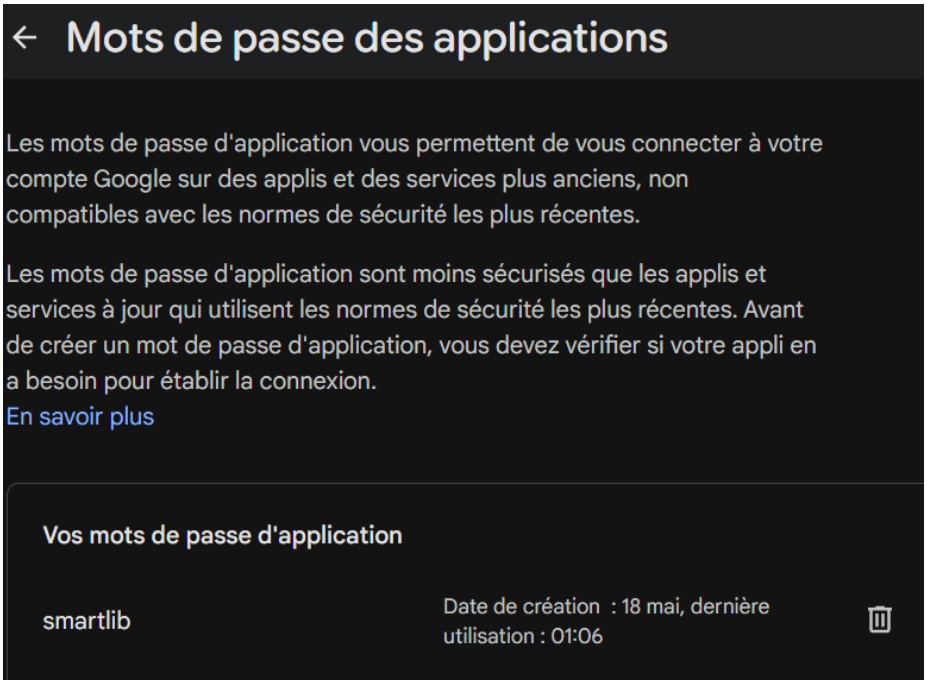
Wikitheo. (2026, 23 mars). *Comment configurer un serveur SMTP pour Gmail (Guide complet)* [Vidéo]. YouTube. https://www.youtube.com/watch?v=L5sZef-Bg7U, consulté le consulté le 17 avril 2026

Welcome to Paramiko ! — Paramiko documentation. (s. d.). https://www.paramiko.org/, consulté le 15 avril 2026

Python 3.14 documentation. (s. d.). Python Documentation. https://docs.python.org/3/, consulté 20 décembre

8. Annexes

Création de mot de passe d'application

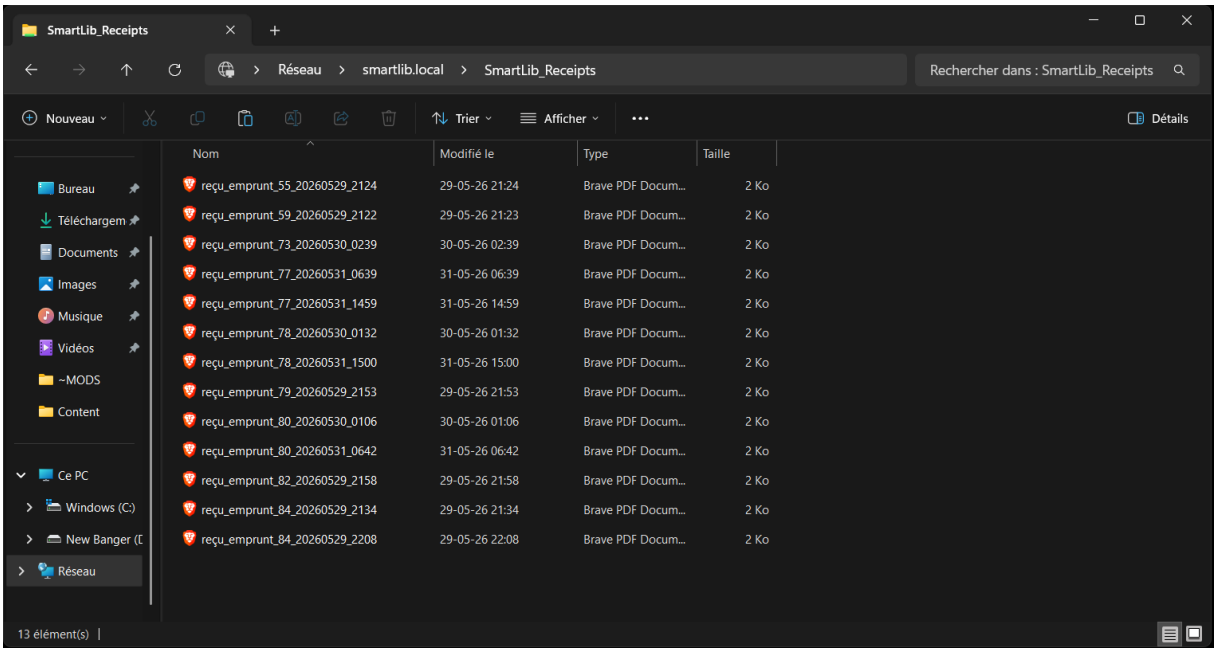


Configuration Samba

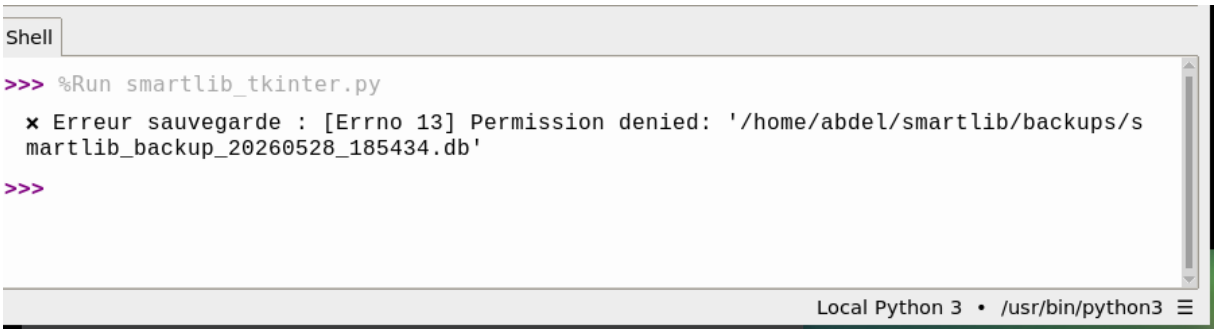
```
[SmartLib_Receipts]
   path = /home/abdel/smartlib_receipts
   browseable = yes
   writable = yes
   guest ok = yes
   read only = no
   valid users = abdel
```

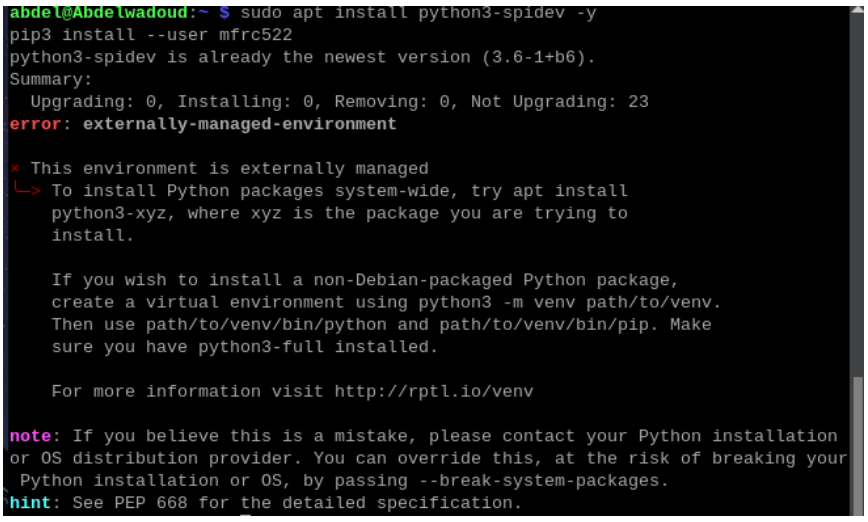
```
# === Configuration ===
USER_HOME = "/home/abdel"
DB_PATH = "/home/abdel/tfe/smartlib.db"
SAMBAPATH = os.path.join(USER_HOME, "smartlib_receipts")
```

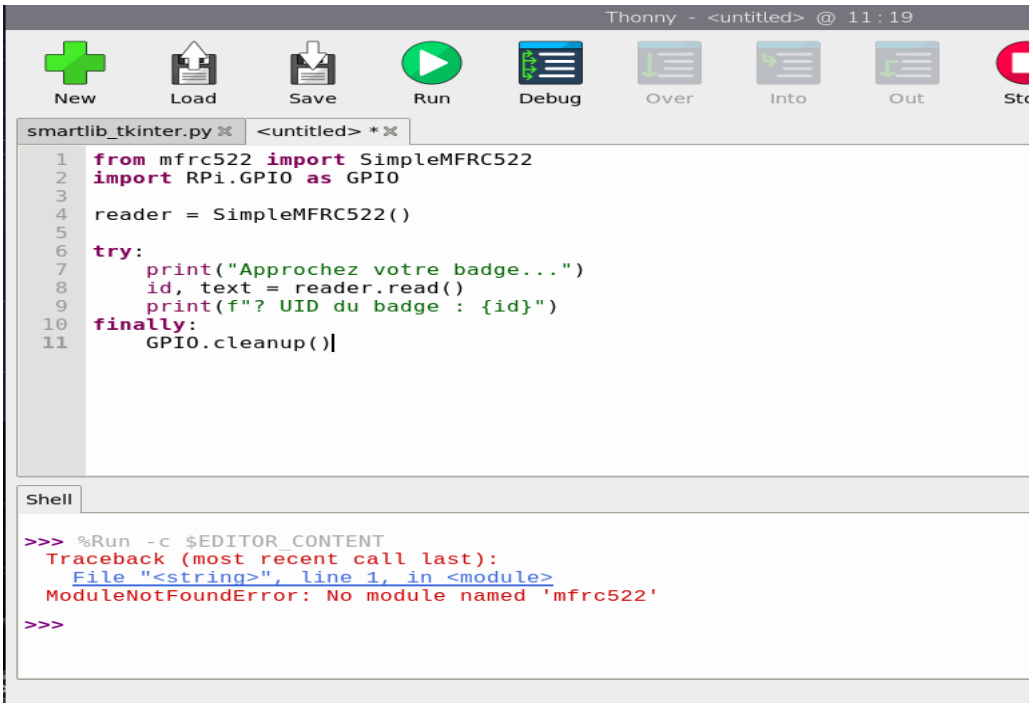
```
#======================= Global Settings =======================

[global]
netbios name = smartlib
```



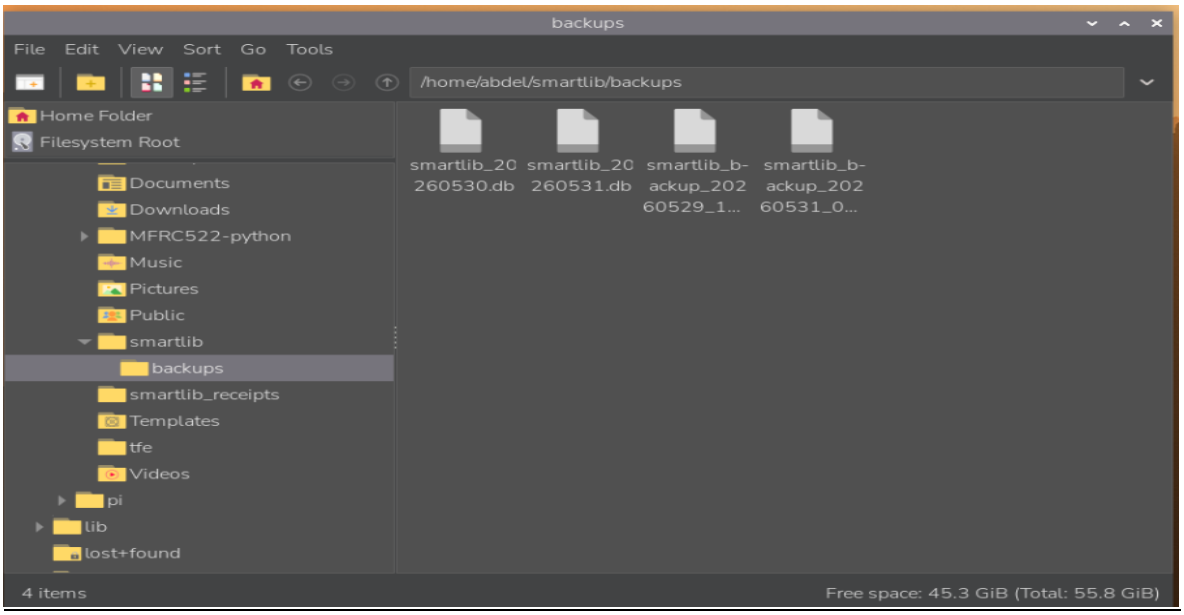
Les erreurs pendant la réalisation

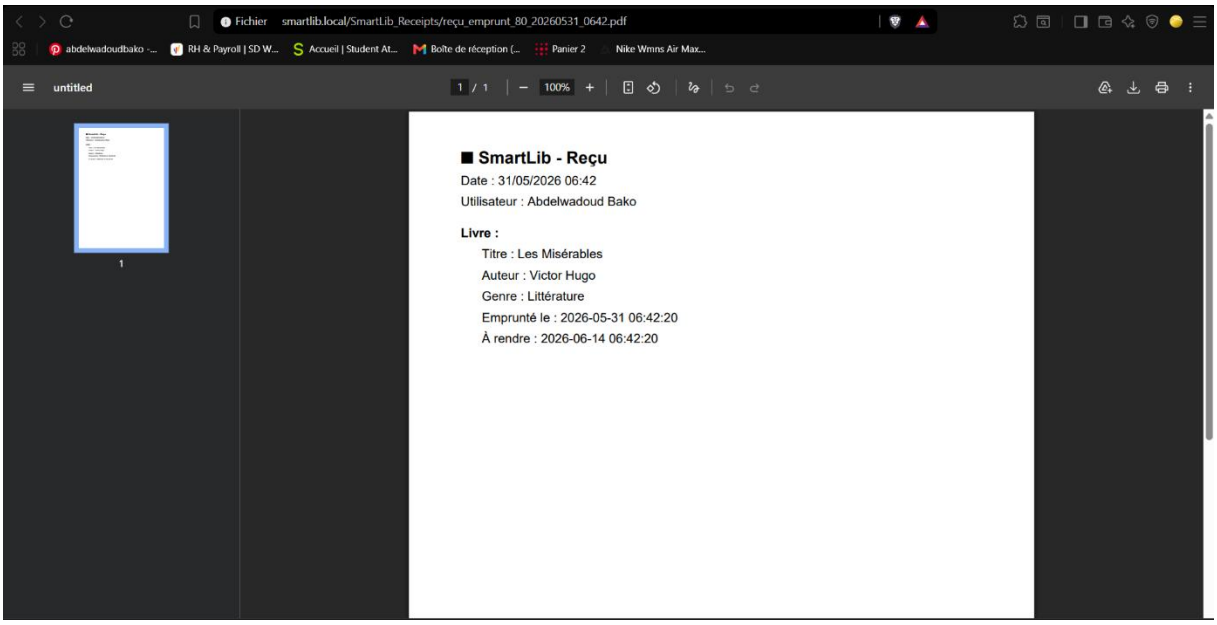


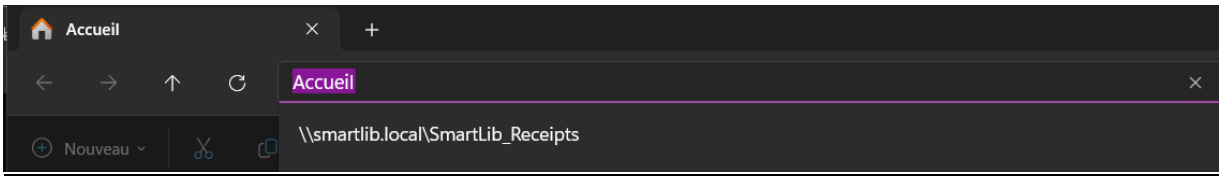



Thonny - <untitled> @ 11 : 19
New
Load
Save
Run
Debug
Over
Into
Out
smartlib_tkinter.py
<untitled> *
1 from mfrc522 import SimpleMFRC522
2 import RPi.GPIO as GPIO
3
4 reader = SimpleMFRC522()
5
6 try:
7 print("Approchez votre badge...")
8 id, text = reader.read()
9 print(f"? UID du badge : {id}")
10 finally:
11 GPIO.cleanup()
Shell
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'mfrc522'
>>>

Autres


backups
File Edit View Sort Go Tools
/home/abdel/smartlib/backups
Home Folder
Filesystem Root
Documents
Downloads
MFRC522-python
Music
Pictures
Public
smartlib
backups
smartlib_receipts
Templates
tfe
Videos
pi
lib
lost+found
smartlib_20260530.db
smartlib_20260531.db
smartlib_backup_20260529_1...
smartlib_backup_20260531_0...
4 items
Free space: 45.3 GiB (Total: 55.8 GiB)






```python
# smartlib_tkinter.py
import tkinter as tk
from tkinter import messagebox, ttk, simpledialog
import sqlite3
from datetime import datetime, timedelta
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4
from reportlab.lib import colors
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
import os
import csv
import threading
import shutil
import time
import RPi.GPIO as GPIO
import logging
import logging.handlers
import paramiko
```

```
#(ALWAYSDATA)
REMOTE_HOST = "ssh-abdel.alwaysdata.net"
REMOTE_PORT = 22
REMOTE_USER = "abdel"
REMOTE_PASSWORD = "inraci2026"
REMOTE_PATH = "/home/abdel/backups/"
```

```
#Configuration
USER_HOME = "/home/abdel"
DB_PATH = "/home/abdel/tfe/smartlib.db"
SAMBAPATH = os.path.join(USER_HOME, "smartlib_receipts")

SMTP_EMAIL = "abdelwadoud.bako@gmail.com"
SMTP_PASSWORD = "uhcamcmeuuonrlvt"
DESTINATAIRE = "abdelwadoud.bako@inraci.be"
CURRENT_USER = "Invité"
```

```
#Interface Principale
root = tk.Tk()
root.title("📚 SmartLib v5.1 - Cloud")
root.geometry("900x650")
root.attributes('-fullscreen', True)
root.bind('<Control-q>', lambda e: root.destroy())

logger.info(" Application SmartLib démarrée en mode Kiosque")
cleanup_old_pdfs(30)

nb = ttk.Notebook(root); nb.pack(fill="both", expand=True, padx=10, pady=10)

fb = ttk.Frame(nb); nb.add(fb, text="Emprunter")
sf = ttk.Frame(fb); sf.pack(pady=5)
tk.Label(sf, text="Recherche :").pack(side="left")
sv = tk.StringVar(); tk.Entry(sf, textvariable=sv, width=30).pack(side="left")
sv.trace("w", lambda *a: refresh_all_views(sv.get()))
gf = tk.Frame(fb); gf.pack(pady=5)
tk.Label(gf, text="Genre :").pack(side="left", padx=5)
genre_combo = ttk.Combobox(gf, state="readonly", width=20); genre_combo.pack(side="left")
genre_combo.bind("<<ComboboxSelected>>", lambda e: refresh_all_views(sv.get()))
tree_borrow = ttk.Treeview(fb, columns=("ID","Titre","Auteur","Genre","Dispo"), show="headings", height=10)
for c in ("ID","Titre","Auteur","Genre","Dispo"): tree_borrow.heading(c, text=c); tree_borrow.column(c, width=140)
tree_borrow.pack(pady=5, fill="both", expand=True)
bf = tk.Frame(fb); bf.pack(pady=5)
tk.Button(bf, text="Emprunter", command=lambda: borrow_book(tree_borrow.item(tree_borrow.selection())['values'][0] if tree_borrow.selection() else None)).pack(side="left", padx=5)
tk.Button(bf, text="🔒 Admin", command=open_admin_panel).pack(side="left", padx=5)

fr = ttk.Frame(nb); nb.add(fr, text="Retourner")
tk.Label(fr, text="Emprunts en cours :").pack(pady=5)
tree_return = ttk.Treeview(fr, columns=("ID","Utilisateur","Livre","Emprunt","Retour"), show="headings", height=8)
for c in ("ID","Utilisateur","Livre","Emprunt","Retour"): tree_return.heading(c, text=c)
tree_return.pack(pady=5, fill="both", expand=True)
tk.Button(fr, text="Retourner", command=lambda: return_book(tree_return.item(tree_return.selection())['values'][0] if tree_return.selection() else None)).pack(pady=5)

fh = ttk.Frame(nb); nb.add(fh, text="Mon Historique")
tree_my_history = ttk.Treeview(fh, columns=("ID","Livre","Emprunt","Retour","Rendu","Statut"), show="headings", height=10)
for c in ("ID","Livre","Emprunt","Retour","Rendu","Statut"): tree_my_history.heading(c, text=c); tree_my_history.column(c, width=140)
tree_my_history.pack(pady=5, fill="both", expand=True)
```