Institut National de Radioélectricité et de Cinématographie

Enseignement technique secondaire de qualification

Accès aux études supérieures

Avenue Jupiter, 188

1190 Forest

KACHING

AWA ESHGABADY

Pour l'obtention du certificat de qualification

Technicienne en informatique 6TQ.I

Tuteur : Abdelkader Ben Sallem

Année scolaire : 2025-2026

Remerciement :

Je tiens à remercier avant tout mes professeurs qui m’ont suivi pendant ces 3 ans au long de mon parcours. Merci pour tout ce que vous m'avez appris, j’ai vraiment aimé l'informatique et je ne regrette pas du tout d'avoir fait ce choix.

Ensuite, je veux dire un grand merci à mes amis qui comptent beaucoup pour moi. Merci d’avoir toujours été là pour m'aider, de m'avoir soutenu et de m’avoir redonné de la force quand j'en avais besoin.

Enfin, je remercie de tout mon cœur mes parents. Merci de n'avoir jamais douté de moi, de m'avoir encouragé tous les jours et d’avoir toujours cru en moi pour ce projet.

Table des matières

1. Introduction

Pour mon projet de fin d'année, j'ai choisi de réaliser un projet qui me tient vraiment à cœur. Je me suis directement inspiré de mon expérience en job étudiant pour effectuer une caisse enregistreuse que j'ai décidée de nommer **KACHING**.

Le but de ma caisse KACHING, c'est d'avoir un fonctionnement similaire à ce qu'on voit dans des magasins comme Carrefour, Zeeman ou Okay. J'ai voulu réaliser une application capable de gérer tout le processus : l'encaissement des produits avec l'ajout des quantités des produits et l'affichage jusqu'au paiement automatique et la sortie du ticket.

Mon application aura une interface principale « caissière », qui se connectera grâce à son propre ID détecté via Always Data. Elle aura donc la possibilité de gérer les produits/articles et d'avoir accès au paiement.

Et j'aurai une interface Administrateur qui récupérera les revenus à la fin de la journée, c'est-à-dire, chaque action faites pendant la journée. Dans ces revenus, s'afficheront tout le montant de fin de journée et les articles vendus par chaque employé.

Pour ce qui est des horaires, j'ai rajouté une barre de sécurité. L'application se fermera automatiquement à 18h et s'ouvrira à 6h. Seule l'interface Administrateur aura encore accès à ses fonctionnalités après la fermeture, tandis que les sessions des caissières seront bloquées jusqu'au matin.

2. Outils :

2.1 Les matériels utilisés

Matériel	Description	Quantité	Prix	Lien
Raspberry Pi 4	Modèle B , 4GB	1	117€	Raspberry Pi 4 model B 4GB LPDDR4 : Amazon.com.be: High-tech
Ordinateur portable	Hp Pavillon	1	A moi	///
Scanner code barre	Eyoyo Lecteur Codes Barres 1D 2D	1	22.99€	Lecteur Codes Barres Eyoyo 1D 2D USB avec Câble Inclus
Imprimante	HP Laser Jet 1020	1	Empreinte à l'école	///
Cable alimentation	Utilisée pour l'imrpimante	1	Empreinte à l'école	///
Total :	///	///	200€	///

2.2 Les logiciels utilisées

Logiciels	Description	Prix
Geany Programmer's	Utilisé pour coder sur Raspberry Pi	///
AlwaysData	Base de donnée en ligne	///
DB browser	Base de donnée en local	///
PythonAnywhere	Utilisé pour créer site de paiement en ligne	///
RealVNC	Affichage du Raspberry sur l'écran bureau	///
Gemini/Claude	Aide de la structure du code	///

2.3 Les langages utilisés

Langages	Description
CustomTkinter	Langage principale de mon code
Flask	Langage de mon site de payement

3. Explication des composants utilisés

Raspberry pi 4

Qu’est ce qu’un Raspberry Pi ?

Le Raspberry Pi est un micro-ordinateur polyvalent qui intègre tous ses composants essentiels sur une seul carte mère . Elle peut être servir d’ordinateur de bureau classique ou être utilisé pour des projets complexe tel que le mien, grâce à son accessibilité à distance.

Lancé en juin 2019, son objectif premier était d'offrir de nouvelles possibilités d'apprentissage dans le monde de l'informatique pour attirer les jeunes curieux. Aujourd'hui, sa performance et sa grande polyvalence en font une solution idéale, que ce soit pour découvrir la technologie ou pour construire des projets techniques personnalisés.

Le Raspberry contient un Processeur, Mémoire RAM, Connectivité , ports vidéo , USB, Stockage, Alimentation, GPIO.

Pourquoi l’utiliser dans mon projet ?

Le Raspberry Pi est le "cerveau" de mon projet. Il est indispensable pour faire fonctionner tout le service et la communication entre chaque composants.

Il lance l'application principale développée en CustomTkinter pour les employés. Il fait le pont entre la communication en permanence avec mon site Flask (PythonAnywhere), grâce au Raspberry Pi reçoit l’info dès qu’un client valide son paiement et met à jour l’interface caisse.

Il gère les reçus des données du scanner code barre et les infos du panier l’envoie direct à l’imprimante pour sortir le ticket de caisse automatiquement.

Ordinateur portable

Qu’est ce qu’un ordinateur portable ?

Un ordinateur est une machine électronique qui traite des informations de façon automatique. Il fonctionne en utilisant des programmes stockés dans sa mémoire qui lui donnent les instructions nécessaires pour accomplir des tâches.

- Il reçoit des informations grâce à des outils comme le clavier ou la souris.
- Il utilise un processeur, qui sert de cerveau, pour traiter toutes ces données.
- Il affiche ou donne le résultat final grâce à des outils de sortie comme l'écran.

Le système d'exploitation prend les commandes, puis les programmes lancés sont chargés dans la mémoire vive pour être utilisés par le processeur.

Pourquoi l’utiliser dans mon projet ?

Il est utilisé pour afficher mon application principale ce qui permet aux employés la faciliter d’utiliser leur espace de travail et le logiciel directement sur son écran intégré.

Il gère tout le fonctionnement du système, en assurant la communication avec mon site web pour le traitement des paiements.

Il permet d'afficher et de piloter les données provenant du Raspberry Pi, centralisant ainsi tout le contrôle du système dans une seule machine prête à l'emploi.

Scanner code barre Eyoyo

Qu’est ce qu’un Scanner ?

Le scanner de code barre Eyoyo est un appareil de lecture conçu pour capturer et décoder rapidement des informations. Il fonctionne en scannant les motifs visuels d'un QR code pour les transformer en données numériques que l'ordinateur peut comprendre et traiter.

Pourquoi l’utiliser dans mon projet ?

Il est utilisé pour reproduire le fonctionnement d'une vrai caisse enregistreuse. Au lieu de saisir manuellement les numéro d’article au clavier, le scanner lit directement les codes pour identifier les produits.

Il transmet les données au Raspberry, qui les enregistre immédiatement dans la caisse, rendant ainsi le processus beaucoup plus pratique.

Imprimante HP Laser Jet 1020

Qu’est ce qu’une Imprimante ?

L'imprimante HP LaserJet 1020 est une imprimante laser monochrome conçue pour imprimer rapidement des documents texte et des graphiques simples. Elle se connecte directement au Raspberry pi via un câble USB pour recevoir les infos à imprimer

Pourquoi l’utiliser dans mon projet ?

Une fois que le scanner a identifié le produit et que l'application a validé la transaction, l'imprimante génère physiquement le ticket de caisse. Elle permet ainsi de fournir une preuve d'achat immédiate et tangible aux clients, finalisant le processus de vente de manière professionnelle.

4. Explications des logiciels utilisés

Geany Programmer's

Geany est un logiciel d'édition de texte léger et rapide. Il permet d'écrire, de modifier et d'organiser le code source de nombreux langages de programmation dans une interface simple et claire.

Pourquoi l'utiliser dans mon projet ?

Dans ce projet, sert d'environnement de développement pour concevoir et modifier l'application. C'est l'outil dans lequel le code du programme est écrit, structuré et sauvegardé avant d'être lancé sur le Raspberry pi.

AlwaysData

Always data est un hébergeur web français reconnu pour sa fiabilité. Concrètement, il sert d'espace de stockage sécurisé sur Internet, ce qui permet d'accéder à des services et des données à distance, en dehors de la machine locale.

Pourquoi l'utiliser dans mon projet ?

Il est utilisé comme une base de données en ligne, sert à conserver et stocker toutes les infos importantes, comme l'identifiant dédié à chaque employé. Elle permet de sécuriser ces données en étant directement reliée à la page de connexion de l'application. AlwaysData vérifie l'ID saisi et autorise l'ouverture de la caisse uniquement lorsque l'employé est reconnu par cette base de données distante.

DB browser

DB Browser for SQLite est un logiciel local qui permet de créer et modifier des base de données SQLite. DB Browser est une interface graphique qui permet de voir et de changer facilement ce qu'il y a dedans.

Pourquoi l'utiliser dans mon projet ?

Contrairement à AlwaysData, elle me sert de base de données locale directement sur le Raspberry Pi. C'est dans cette base que sont stockés tous mes produits avec leur nom, leur prix et leur code-barres.

Quand l'employé scanne un article, le système retrouve instantanément le produit correspondant, l'affiche dans le tableau de la caisse et l'enregistre automatiquement dans les revenus. Sans elle, la caisse ne saurait ni ce qu'elle vend, ni à quel prix, et aucun suivi des ventes ne serait possible.

PythonAnywhere

PythonAnywhere est un service en ligne qui permet de faire tourner du code Python/Flask en permanence sur internet. Il héberge le code à distance et le maintient actif en continu, ce qui garantit qu'il reste accessible à tout moment.

Pourquoi l'utiliser dans mon projet ?

PythonAnywhere héberge le site de paiement développé avec Flask et le fait tourner en boucle derrière le projet. Cela assure que le site est toujours disponible au moment où le client scanne le QR code avec son téléphone pour payer.

RealVNC

RealVNC est un logiciel qui permet de prendre le contrôle d'un autre ordinateur à distance, depuis un écran différent.

Pourquoi l'utiliser dans mon projet ?

RealVNC me sert à afficher et contrôler mon application KACHING depuis mon ordinateur portable directement sur le Raspberry Pi.

Claude/ Gemini

Claude et Gemini sont des intelligences artificielles développées par Anthropic et Google. Elles sont capables d'analyser du code, de déte cter des erreurs et d'expliquer des concepts de programmation de manière simple et compréhensible par tous.

Pourquoi l'utiliser dans mon projet ?

Je les utilise comme un outil d'aide, elles m'ont aidé à mieux structurer et organiser mon projet, en me guidant sur la manière rendre propre et fonctionnels mes parties mes code de façon logique et cohérente.

5. Explication des langages utilisés

CustomTkinter

CustomTkinter est une bibliothèque Python qui permet de créer des interfaces graphiques modernes.

Elle est basée sur Tkinter, qui est l'outil de base de Python pour créer des fenêtres et des boutons, mais en apportant un design bien plus moderne et personnalisable.

Bibliothèque : pip install customtkinter

Pourquoi l'utiliser dans mon projet ?

J'ai choisi CustomTkinter plutôt que Tkinter car ce dernier propose des interfaces peu daté et peu professionnel pour une application caisse enregistreuse.

CustomTkinter permet d'obtenir un rendu plus propre. C'est donc lui qui est utilisé pour concevoir toute l'interface principale de mon application.

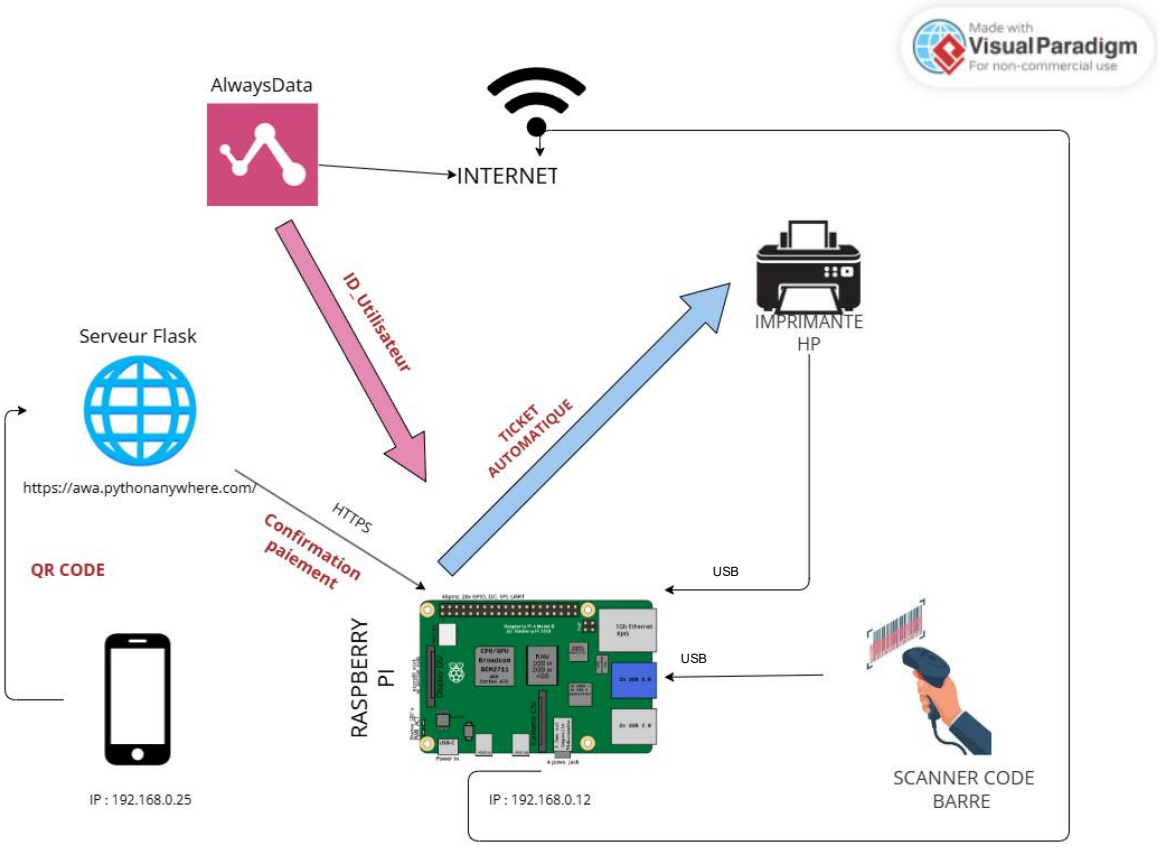
Flask

Flask est un framework Python qui permet de créer des sites web et des applications web. Il se concentre sur l'essentiel et permet de mettre en place un site web fonctionnel.

Pourquoi l'utiliser dans mon projet ?

Flask est utilisé pour créer et faire tourner la page de paiement vers laquelle le client est redirigé lorsqu'il scanne le QR code avec son téléphone. C'est lui qui gère toute la communication entre le téléphone du client et le Raspberry Pi. Lorsque le client valide son paiement sur la page Flask, l'information est directement transmise à la caisse.

6. Topologie Réseau



Explication :

Au centre, se trouve le Raspberry Pi 4 est le cerveau de mon projet, celui qui reçoit et envoie toutes les informations pour tous.

Le scanner est branché en USB sur le Pi 4 . Quand l'employé scanne un article, l'info est envoyée et va rechercher le produit dans sa base de données locale et l'afficher sur l'interface de la caisse.

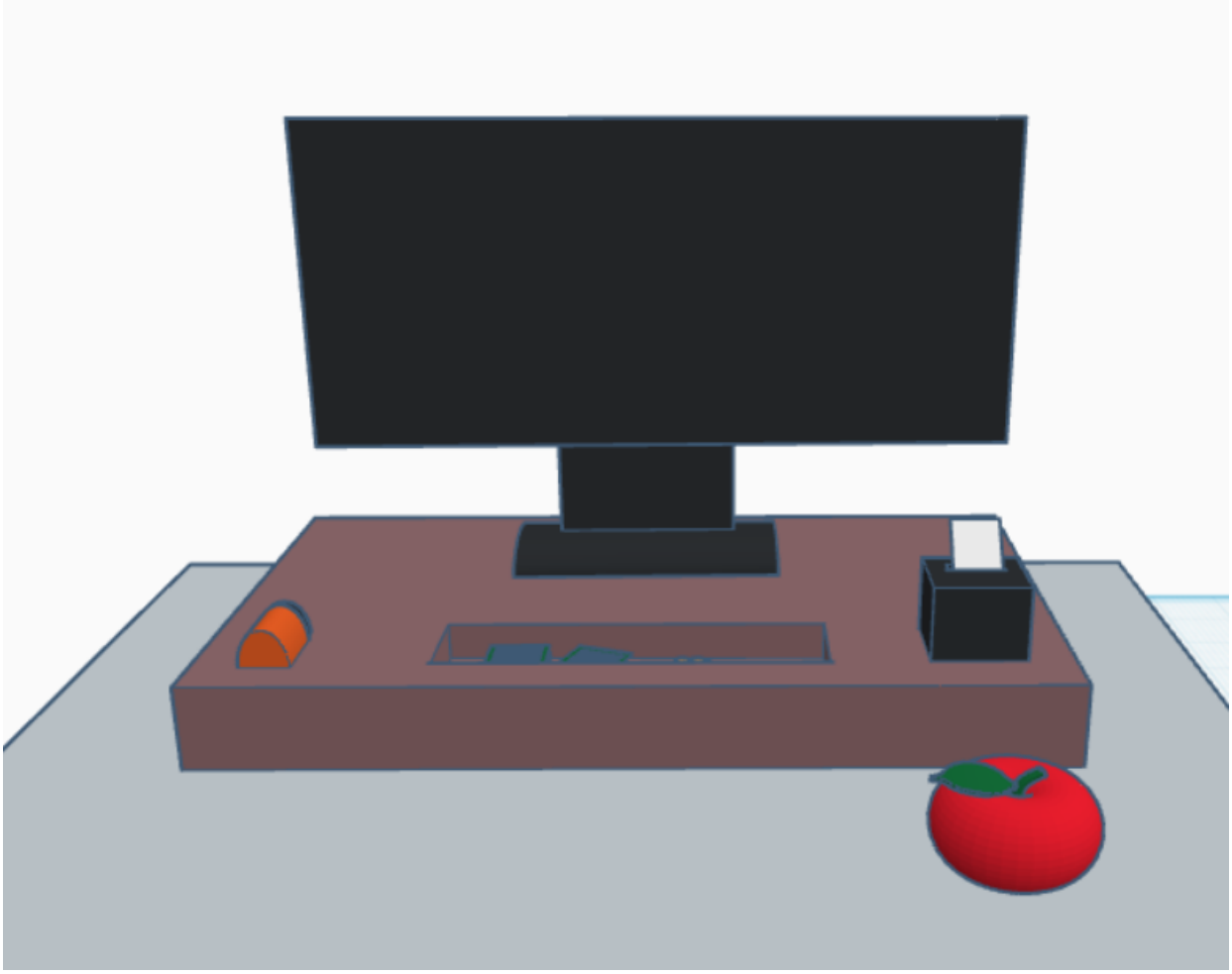
AlwaysData est connecté au Pi 4 via internet. C'est lui qui gère la connexion des employés. Quand un employé entre son ID sur l'écran de connexion, le Raspberry Pi vérifie auprès d'AlwaysData si cet ID est bien valide avant de lui ouvrir l'accès à la caisse.

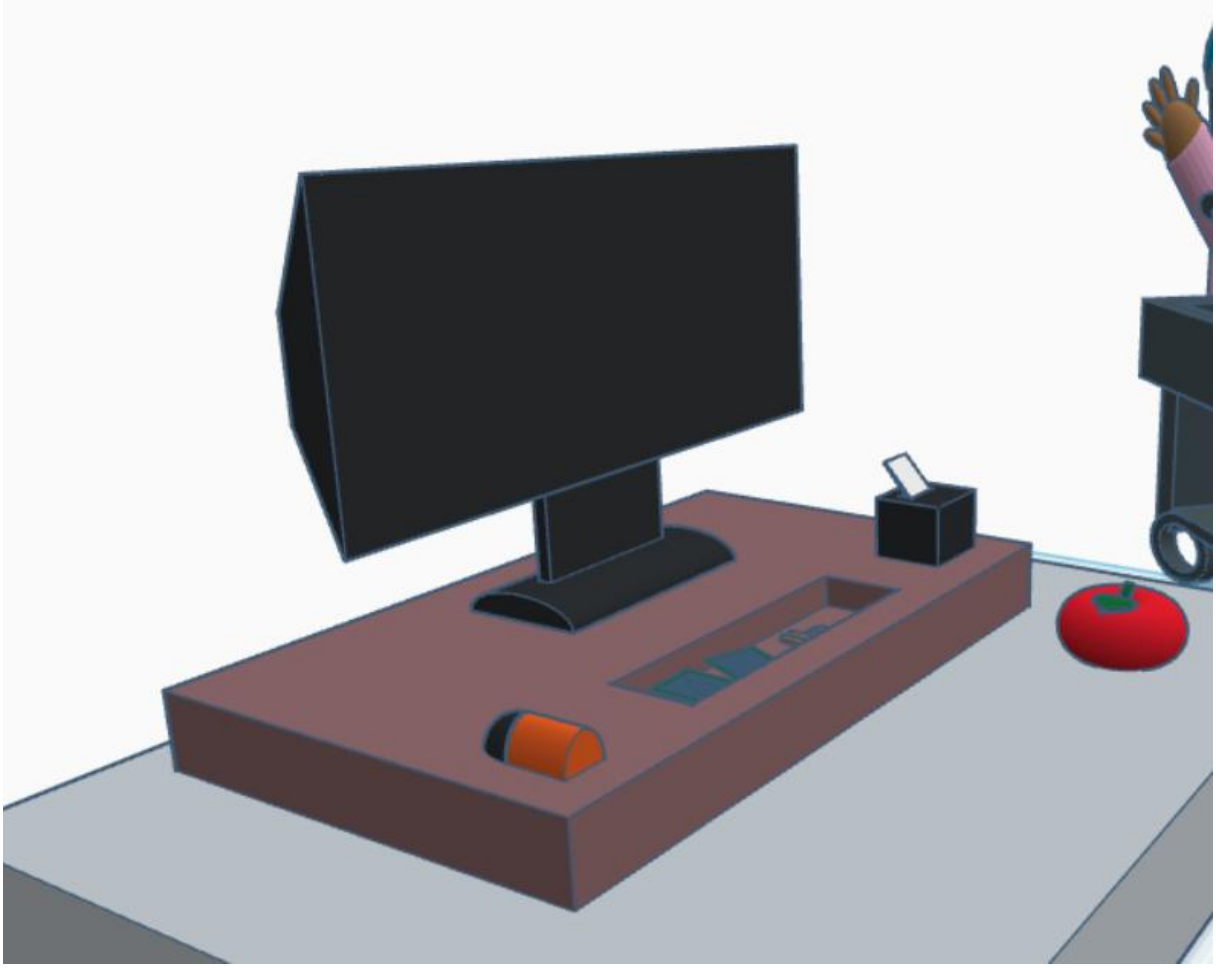
Le serveur Flask hébergé sur PythonAnywhere est lui aussi connecté au Pi 4 via internet, en HTTPS pour sécuriser la communication. Quand le client scanne le QR code avec son téléphone, il est redirigé vers ce serveur pour effectuer son paiement.

Une fois le paiement validé, le serveur Flask envoie automatiquement la confirmation au Raspberry Pi.

L'imprimante HP est également connectée au Pi 4. Dès que l'employé confirme le paiement, enverra les données à l’imprimante et un ticket de caisse sortira automatiquement.

7. Schéma 3D



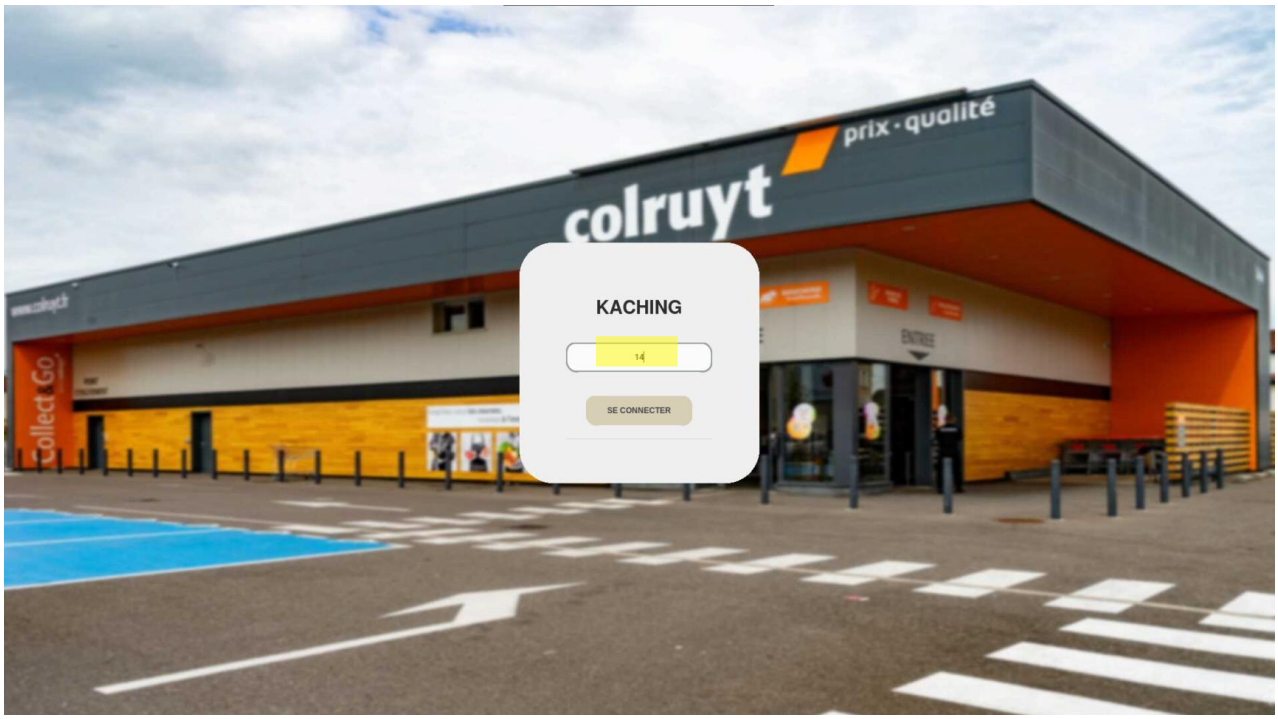


Explication :

Ce schéma représente une maquette 3D de la caisse enregistreuse pour illustrer à quoi ressemblerait en mieux mon projet. On peut y voir plusieurs éléments Au centre, un écran qui représente l'interface de la caisse sur laquelle l'employé travaille. Sur le plateau rouge représente le Raspberry pi qui est relié à tout. À gauche du plateau, se trouve le scanner code barre qui permet de scanner les articles, et à droite l'imprimante qui se charge d'imprimer automatiquement le ticket de caisse après chaque paiement validé.

8. Travail réaliser

8.1 Interface login



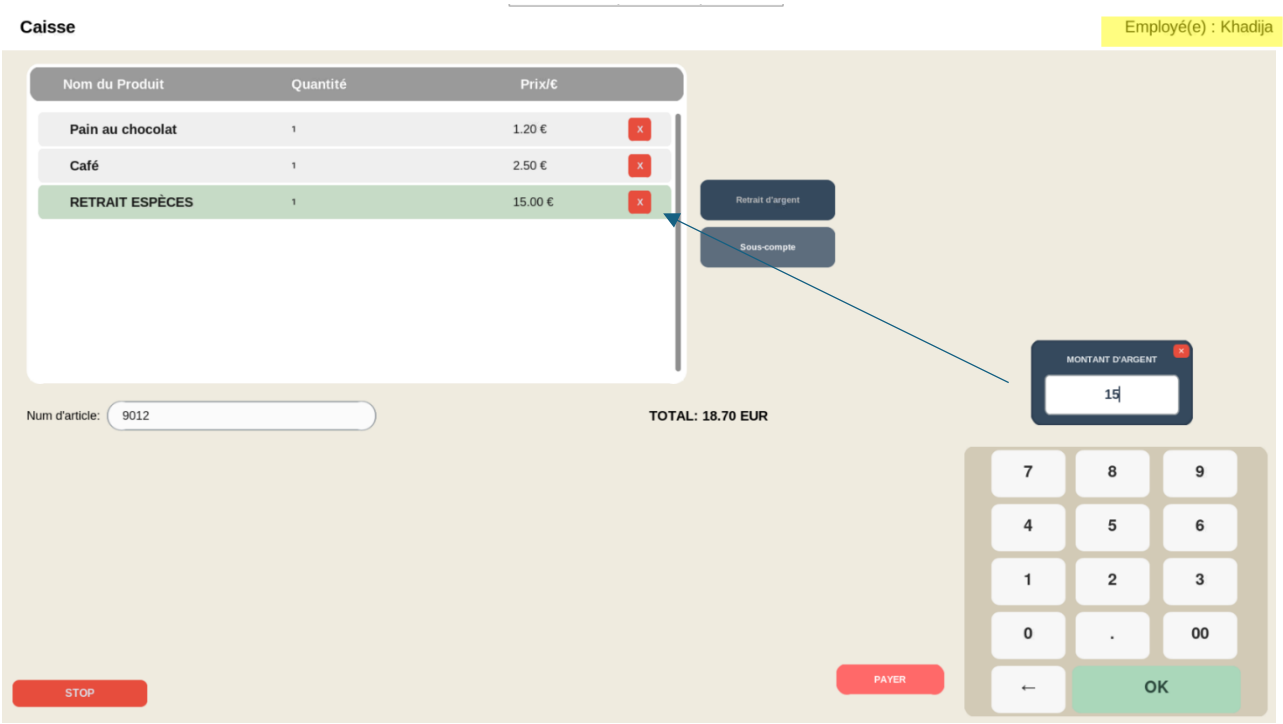
Explication :

Avant d'accéder à l'interface de la caisse, chaque employé doit s'identifier en entrant son numéro d'identifiant. Chaque employé possède son propre identifiant dans la base de données en ligne, par exemple l'ID 14 correspond bien à l'employée Khadija.

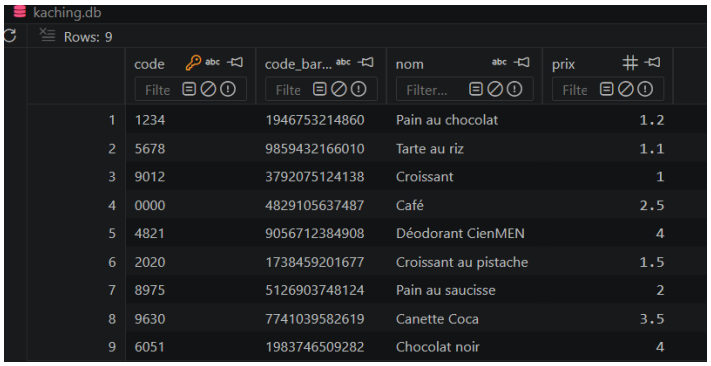
id_user	nom_user
1	ADMIN
2	AYMAN
10	CLOTHIDE
14	Khadija
27	AWA

Une fois l'identifiant entré et validé, le système va vérifier en temps réel auprès de la base de données en ligne (Always Data), si cet identifiant existe bien et correspond à un employé enregistré. Si oui, l'accès est accordé et l'employé est directement redirigé vers son interface de caisse. Si non, un message d'erreur affichera et bloquera son accès.

8.2 L’interface de Caisse



Explication :



Une fois connecté, l'employé est dirigé vers son interface de caisse. On peut remarquer en haut à droite qu’il affiche bien le nom de l'employé connecté « Khadija ».

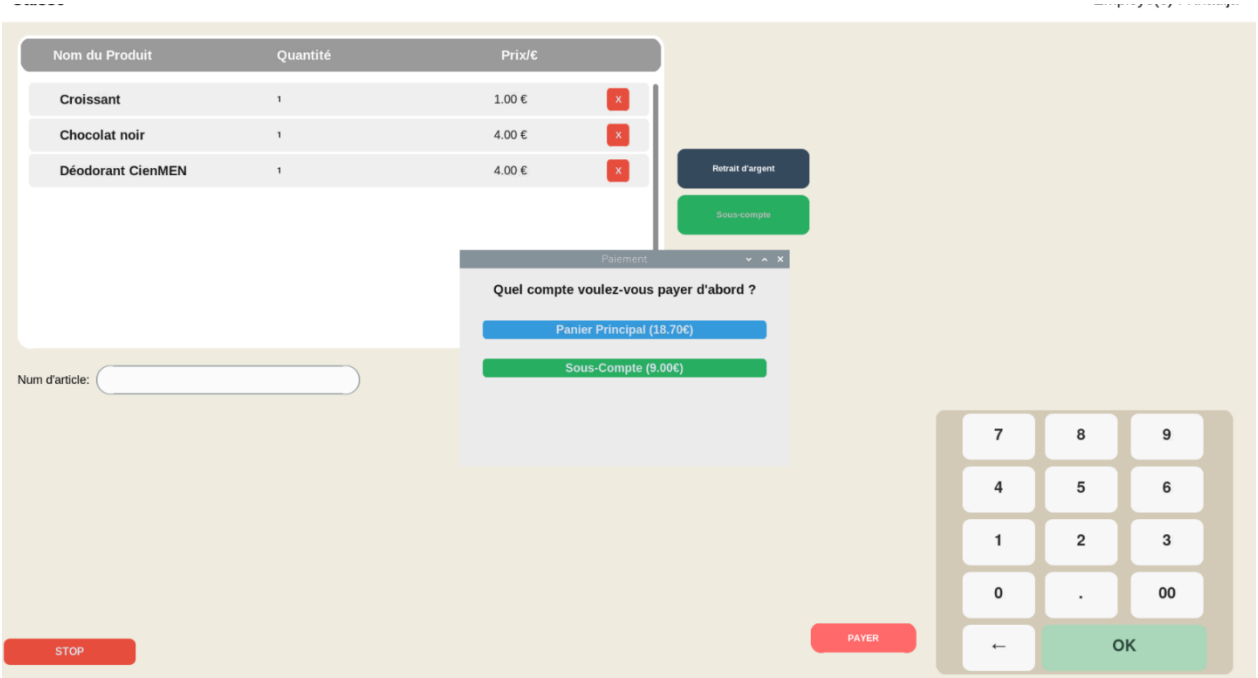
L'interface se compose d'un tableau des produits qui s’affiche automatiquement. Quand Khadija scanne un article ou entre manuellement son numéro dans le champ "Num d'article", le système va immédiatement rechercher dans la base de données SQLite le produit correspondant et l'affiche dans le tableau avec son nom, sa quantité et son prix. Le total se met à jour à chaque nouvel produit ajouté.

Pour rendre la caisse encore plus efficace, j'ai ajouté dans la base de données SQLite une colonne qui permet d'associer chaque code-barres à un produit précis. Lorsque le scanner lit le code-barres d'une cannette de Coca-Cola par exemple. Kaching.db (base de donnée) reconnaît le produit et l'affiche directement sous son nom dans le tableau. (Vidéo simule : https://youtu.be/H57CbeGkQPk)

L'interface de caisse propose deux fonctionnalités supplémentaires, un bouton "Retrait d'argent" pour les clients qui souhaitent retirer du liquide et un bouton "Sous-compte" pour ceux qui aimerait avoir d'un compte épargne pour payer séparément .

Enfin, j'ai ajouté un dispositive de clavier virtuel, ce qui permet à l'employé de tout contrôler depuis un écran tactile, sans avoir besoin d'un clavier ou d'une souris.

8.2.1 Sous compte



Explication :

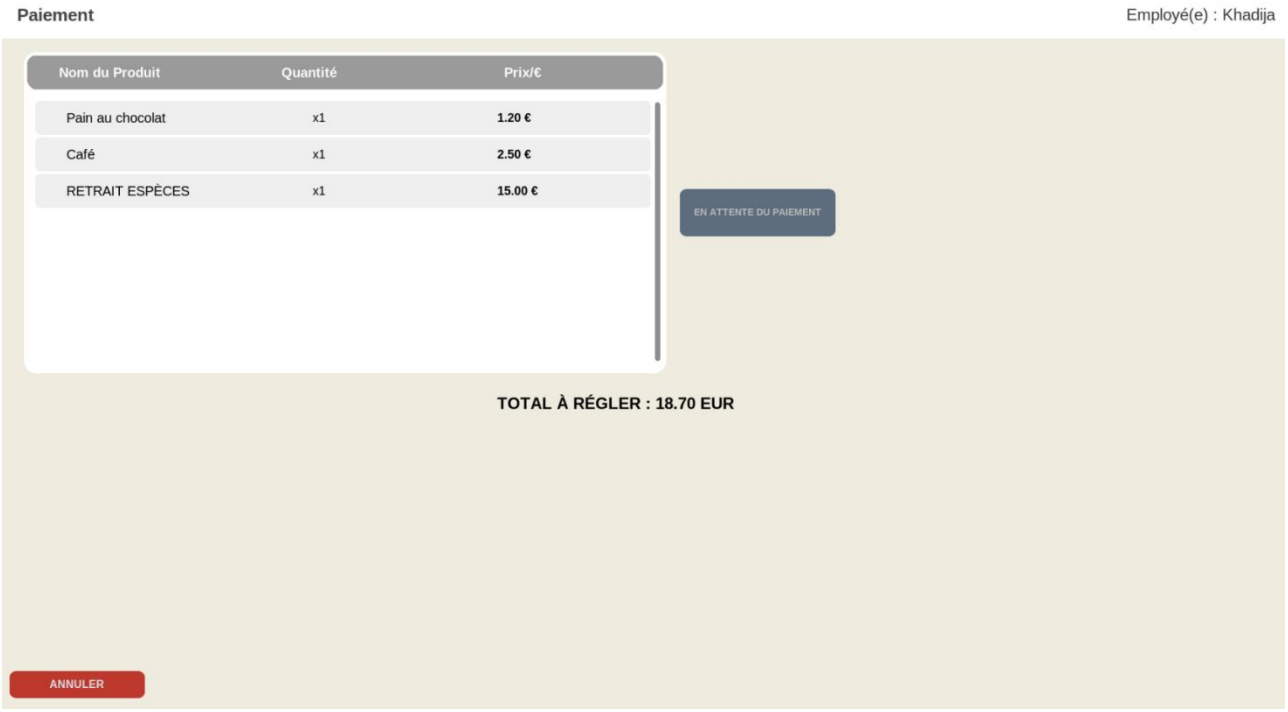
Lorsque le client dispose de deux comptes, une fenêtre s'ouvre automatiquement au moment du paiement pour demander à l'employé quel compte utiliser en premier.

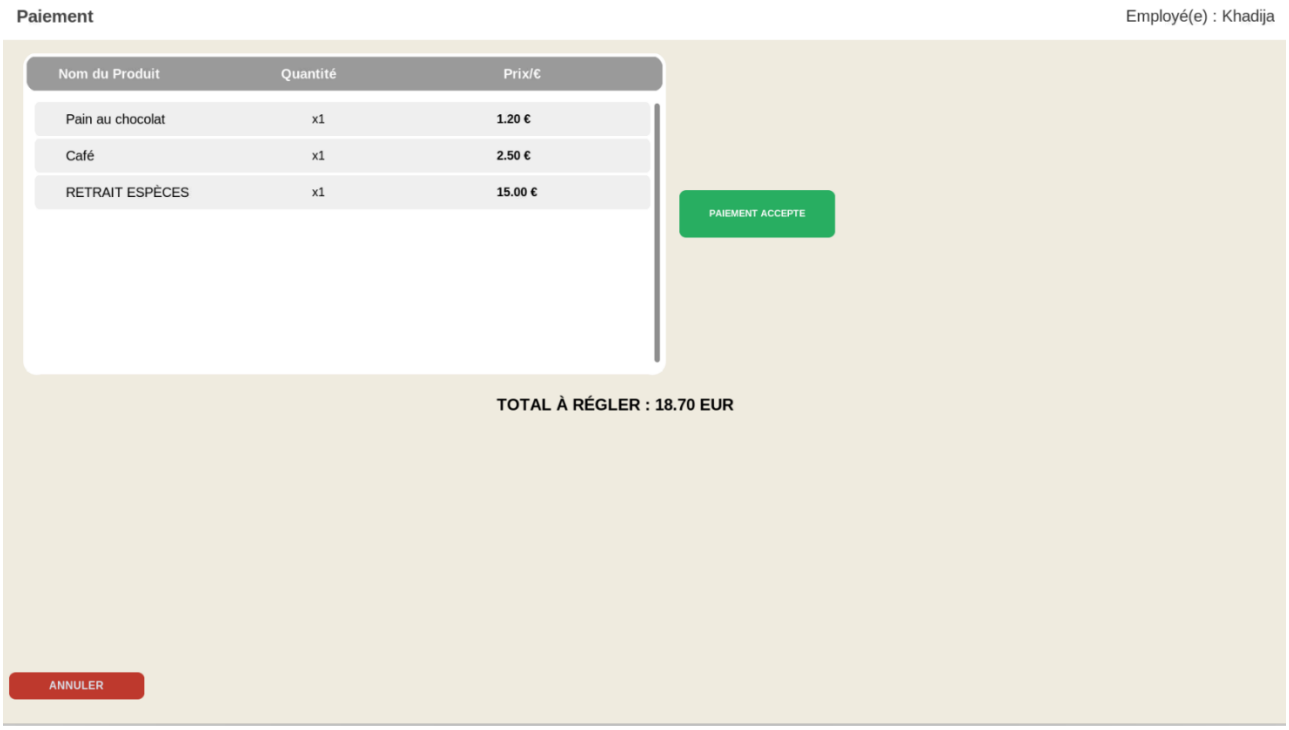
Comme on peut le voir sur l'image, l'employé a le choix entre le Panier Principal (bleu) et le Sous-Compte qui représente le compte épargne (vert).

Cela permet à l'employé de gérer facilement le paiement en fonction de ce que le client souhaite utiliser en priorité.

8.3 L’interface de paiement

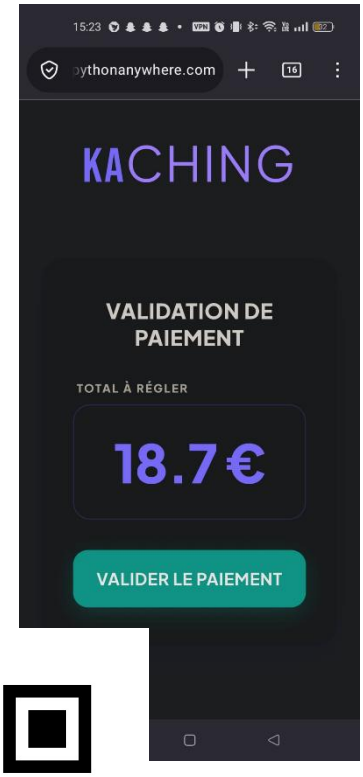
Panier Principale :



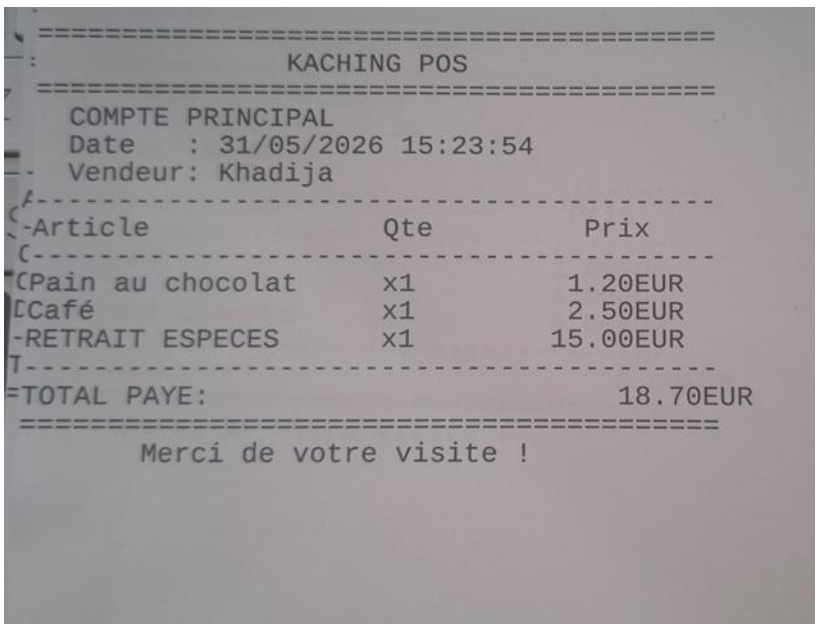


Explication :

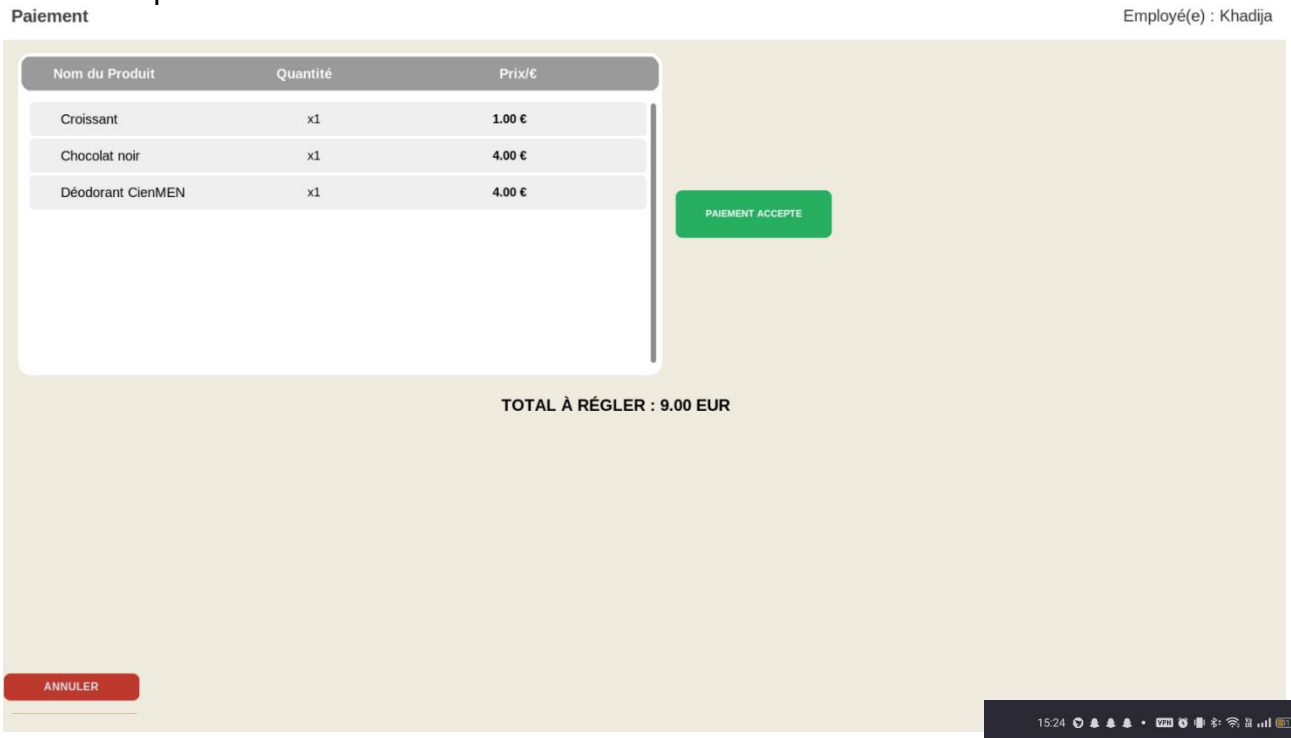
Une fois le panier sélectionné, vers l'écran de paiement. On peut y voir le panier des articles scannés ainsi que le montant total à régler, ici 18.70 EUR. Un bouton "En attente du paiement" s'affiche, indiquant que l'employé attente que le client paye son paiement.

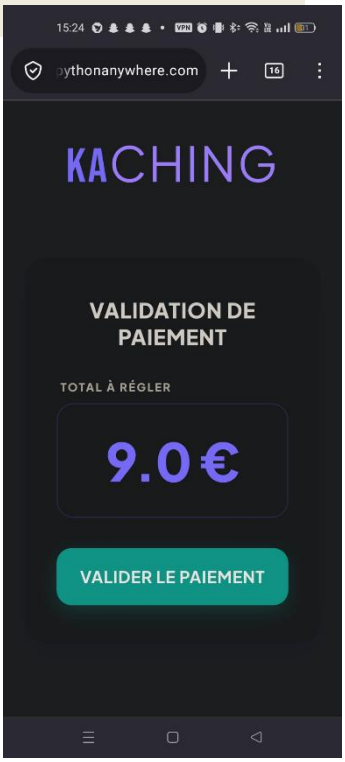


De son côté, le client n'a qu'à scanner le QR code avec son téléphone, , comme on peut le voir sur la deuxième image. Cette page affiche clairement le montant total à régler et un bouton "Valider le paiement". Dès que le client appuie sur ce bouton depuis son téléphone, la confirmation est instantanément transmise à la caisse et le bouton "En attente du paiement" se transforme automatiquement en "Paiement accepté". L'employé n'a plus qu'à appuyer dessus pour que le ticket de caisse s'imprime automatiquement.



Sous-Compte :

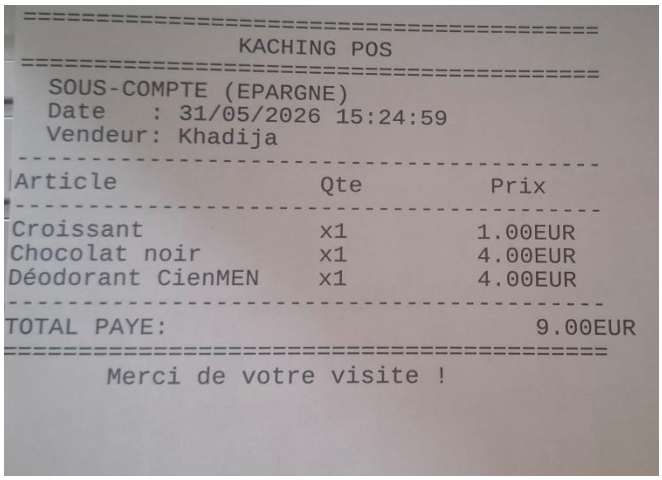




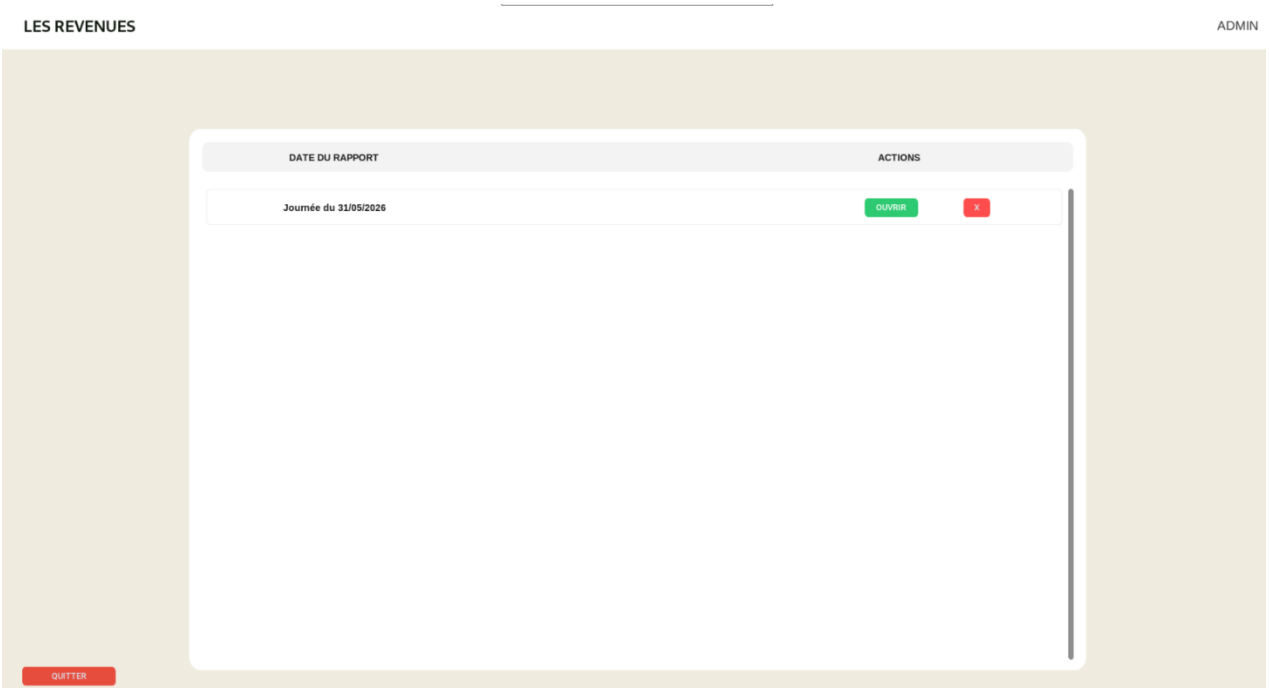
Explication :

Une fois le premier paiement validé, le deuxième compte s'affiche automatiquement à son tour sur l'interface de paiement, avec son propre montant à régler. Le client n'a qu'à scanner à nouveau le QR code et valider le paiement depuis son téléphone, exactement de la même façon que pour le premier.

Cela permet au client de payer ses deux paniers séparément et de recevoir en retour deux tickets de caisse différent, un pour chaque compte.



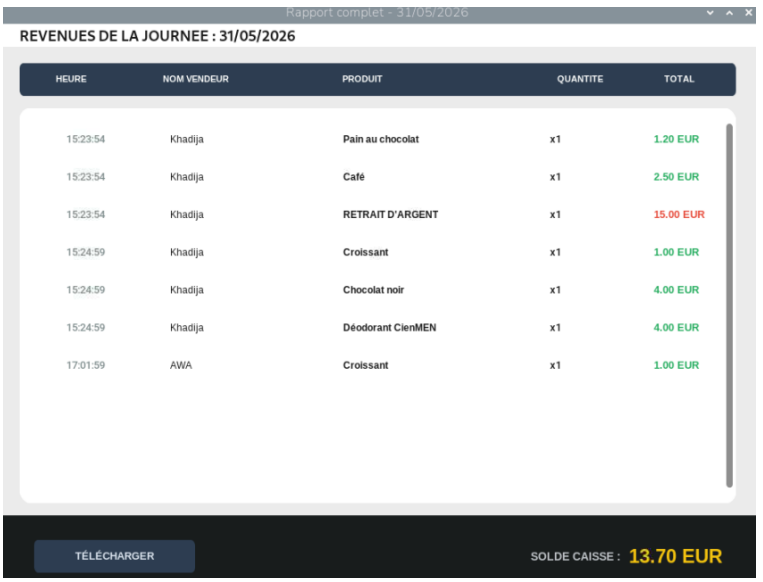
8.4 L’interface Administrateur



Explication :

J’ai aussi permis de créer une interface administrateur accessible que pour l’admin. Cela garantit que seul le responsable peut consulter les données de la journée.

Comme on peut le voir sur la deuxième image, chaque journée de travail génère un rapport , que l'admin peut ouvrir ou supprimer. En ouvrant le rapport, il peut consulter l'ensemble des actions effectuées durant la journée avec pour chaque opération l'heure exacte, le nom de l'employé concerné, le produit vendu, la quantité et le montant. On peut d'ailleurs remarquer que les retraits d'argent y sont également enregistrés et apparaissent en rouge pour les distinguer qu’il ne s’ajuste pas dans le total de revenue puisque. En bas du rapport, le solde total de la caisse pour la journée est affiché, et un bouton "Télécharger" permet d'exporter le rapport pour le conserver ou l'archiver.



9. Explication des codes importants

Fichier MAIN.PY

La synchronisation avec PythonAnywhere

```python
def demarrer_synchronisation_cloud(app_instance):
    """Récupère le statut depuis PythonAnywhere en boucle"""
    global session_client
    while True:
        try:
            response = requests.get("https://awa.pythonanywhere.com/api/statut_paiement", timeout=1)
            if response.status_code == 200:
                data = response.json()
                if data.get("paye_confirme"):
                    with session_lock:
                        # N'accepter la confirmation que si un paiement est réellement en cours
                        if session_client.get("statut") == "paiement":
                            session_client["paye_confirme"] = True
                            print("Synchro Cloud : Statut de paiement récupéré avec succès.")
        except Exception:
            pass  # Ignore les petites pertes de connexion pour éviter les crashs de l'interface
        time.sleep(0.5)
```

Cette fonction tourne en permanence en arrière-plan sans jamais s'arrêter, et interroge le serveur PythonAnywhere toutes les demi-secondes pour savoir si le client a bien appuyé sur le bouton "Valider le paiement" depuis son téléphone. Dès qu'elle détecte que le paiement a été confirmé, elle le signale immédiatement à l'application, ce qui déclenche automatiquement le basculement du bouton "En attente du paiement" en "Paiement accepté" sur l'écran de l'employé. Les erreurs de connexion sont ignorées silencieusement pour éviter tout crash de l'interface pendant cette vérification continue.

La navigation entre les pages :

```python
def show_frame(self, page_name):
    """AFFICHAGE NOM D'USER """
    frame = self._get_or_create_frame(page_name)

    if page_name in ["CaisseFrame", "AdminFrame", "PayeFrame"]:
        if hasattr(frame, "update_id_display"):
            frame.update_id_display(self.user_id, self.user_name)

    frame.tkraise()
```

Cette fonction gère tous les changements de page dans Kaching. À chaque fois qu'on bascule vers une nouvelle page, elle vérifie si cette page a besoin de connaître l'identité de l'employé connecté, et si c'est le cas, elle lui transmet automatiquement son identifiant et son nom avant de l'afficher.

Fichier LOGIN.PY

Horaire de la caisse

```
def attempt_login(self):
    user_id = self.id_entry.get().strip()
    if not user_id: return
    maintenant = datetime.now().hour
    # Si il est 18h ou plus, ou moins de 6h du matin
    if (maintenant >= 18 or maintenant < 6) and user_id != "1":
        messagebox.showwarning(
            "Caisse Fermée",
            f"L'accès à la caisse est verrouillé de 18h à 6h.\nIl est actuellement {datetime.now().strftime('%H:%M')}."
        )
        return

    # On cherche le nom dans Alwaysdata pour TOUS les utilisateurs
    nom_utilisateur = chercher_nom_utilisateur(user_id)

    if nom_utilisateur in ["Inconnu", "Erreur"]:
        self.id_entry.configure(border_width=2, border_color="#FF4B4B")
        return

    # Sauvegarde des infos dans le contrôleur
    self.controller.user_id = user_id
    self.controller.user_name = nom_utilisateur

    # Redirection selon l'ID
    # show_frame crée la frame si besoin (lazy loading) et appelle update_id_display
    if user_id == "1":
        self.controller.show_frame("AdminFrame")
        self.controller.frames["AdminFrame"].charger_sessions()
    else:
        self.controller.show_frame("CaisseFrame")
```

Cette partie du code ajoute une couche de sécurité à la caisse. Entre 18h et 6h du matin, aucun employé ne peut accéder à la caisse. Si quelqu'un tente de se connecter en dehors des heures d'ouverture, un message d'avertissement va s'afficher en lui indiquant l'heure actuelle et les heures d'accès autorisées. Seul l'admin (ID = 1) reste libre de se connecter à n'importe quelle heure pour consulter les rapports.

Fichier CAISSE.PY

L'ajout d'un article scanné

```
def ajouter_article(self, event):
        code = self.item_entry.get().strip()
        p = chercher_produit(code)
        if p:
            nom = p["nom"]
            prix = p["prix"]
            if nom in self.panier and self.panier[nom]["actif"]:
                if not nom.startswith("ret_"):
                    self.panier[nom]["quantite"] += 1
                    if "label_qte" in self.panier[nom]:
                        self.panier[nom]["label_qte"].configure(text=str(self.panier[nom]["quantite"]))
            else:
                is_retrait = nom.startswith("ret_")
                self.creer_ligne_tableau(nom, 1, prix, nom, is_retrait=is_retrait)
            self.recalculer_total()
        self.item_entry.delete(0, 'end')
```

Cette fonction est déclenchée à chaque fois que l'employé scanne un article ou entre son numéro manuellement. Elle récupère le code saisi, va chercher le produit dans la base de données SQLite, et vérifie ensuite si cet article est déjà présent dans le panier. Si c'est le cas, elle incrémente simplement la quantité de 1 et met à jour l'affichage dans le tableau. Si l'article n'est pas encore dans le panier, elle crée une nouvelle ligne dans le tableau avec son nom, sa quantité et son prix, puis recalcule automatiquement le total.

Sélection du panier

```
def demander_quel_compte(self):
    if self.compte_actif == 1:
        self.panier_principal, self.total_principal = self.panier, self.total_actuel
    else:
        self.panier_secondaire, self.total_secondaire = self.panier, self.total_actuel

    if self.total_secondaire <= 0:
        self.ouvrir_paiement_choisi(1)
    else:
        self.popup = ctk.CTkToplevel(self)
        self.popup.title("Paiement")
        self.popup.geometry("500x300")

        self.popup.withdraw()
        self.popup.update_idletasks()
        x = (self.popup.winfo_screenwidth() // 2) - 250
        y = (self.popup.winfo_screenheight() // 2) - 125
        self.popup.geometry(f"+{x}+{y}")
        self.popup.deiconify()
        self.popup.wait_visibility()
        self.popup.grab_set()
        ctk.CTkLabel(self.popup, text="Quel compte voulez-vous payer d'abord ?",font=("Arial", 20, "bold"), pady=20).pack()
        ctk.CTkButton(self.popup, text=f"Panier Principal ({self.total_principal:.2f}€)",
                      command=lambda: self.ouvrir_paiement_choisi(1), fg_color="#3498DB",font=("Arial", 18, "bold")).pack(pady=15, padx=35, fill="x")
        ctk.CTkButton(self.popup, text=f"Sous-Compte ({self.total_secondaire:.2f}€)",
                      command=lambda: self.ouvrir_paiement_choisi(2), fg_color="#27AE60",font=("Arial", 18, "bold")).pack(pady=15, padx=35, fill="x")
```

Quand l'employé appuie sur "Payer", cette fonction vérifie automatiquement si un sous-compte existe. Si oui, une fenêtre s'ouvre avec les deux paniers et leurs montants respectifs pour laisser le choix à l'employé.

Basculement du panier

```
def basculer_vers_compte(self, num):
    """Affiche le contenu du panier demandé (1 ou 2)"""
    if self.compte_actif == 1:
        self.panier_principal, self.total_principal = self.panier, self.total_actuel
    else:
        self.panier_secondaire, self.total_secondaire = self.panier, self.total_actuel

    if num == 1:
        self.compte_actif = 1
        self.panier = self.panier_principal
        self.sous_compte_btn.configure(state="normal", fg_color="#5D6D7E", text="Sous-compte")
    else:
        self.compte_actif = 2
        self.panier = self.panier_secondaire
        self.sous_compte_btn.configure(state="disabled", fg_color="#27AE60", text="Sous-compte")

    for child in self.scroll_table.winfo_children(): child.destroy()
    for k, v in self.panier.items(): self.creer_ligne_visuelle_refresh(k, v)
    self.recalculer_total()
```

Cette fonction gère le basculement entre le panier principal et le sous-compte. Quand l'employé appuie sur le bouton "Sous-compte", elle commence par sauvegarder l'état actuel du panier en cours puis charge le contenu de l'autre panier à la place. Le tableau est entièrement effacé et reconstruit avec les articles du nouveau panier sélectionné. Le bouton "Sous-compte" change de couleur pour indiquer visuellement sur quel compte l'employé est en train de travailler.

Fichier PAYE.PY

L'envoi du panier vers PythonAnywhere

```
def _envoyer_et_activer():
    try:
        requests.post(f"{CLOUD_URL}/api/maj_panier", json=donnees_cloud, timeout=5)
    except Exception:
        pass

    # Le cloud a maintenant paye_confirme=False → on peut activer l'attente
    from shared import session_lock, session_client
    with session_lock:
        session_client["statut"] = "paiement"
        session_client["panier"] = donnees_pour_le_site
        session_client["total"] = total_affiche
        session_client["paye_confirme"] = False
    self.after(0, lambda: setattr(self, "en_attente_paiement", True))

threading.Thread(target=_envoyer_et_activer, daemon=True).start()
```

Dès que l'interface de paiement s'affiche, le panier et le montant total sont envoyés automatiquement vers PythonAnywhere dans un thread séparé, ce qui évite de bloquer l'interface pendant l'envoi. Une fois l'envoi confirmé, Kaching active l'attente de confirmation du client.

```
import threading

session_lock = threading.Lock()
session_client = {
    "panier": [],
    "total": 0.0,
    "statut": "attente",
    "paye_confirme": False
}

#envoie au serveur flask
```

Géneration du ticket de caisse

Cette fonction génère et imprime le ticket de caisse dès que le paiement est validé. Elle récupère la date, l'heure et le nom de l'employé, puis récupère les articles du panier pour construire le ticket ligne par ligne avec le nom de chaque article, sa quantité et son prix. Une fois terminé, le ticket est envoyé directement à l'imprimante HP.

```
def imprimer_ticket(self):
    try:
        import subprocess
        maintenant = datetime.now().strftime("%d/%m/%Y %H:%M:%S")
        nom_employe = self.id_label.cget("text").replace("Employé(e) : ", "")
        label_compte = "COMPTE PRINCIPAL" if self.compte_en_cours_paiement == 1 else "SOUS-COMPTE (EPARGNE)"
        ticket = []
        ticket.append("========================================")
        ticket.append("              KACHING                   ")
        ticket.append("========================================")
        ticket.append(f"  {label_compte}")
        ticket.append(f"  Date   : {maintenant}")
        ticket.append(f"  Vendeur: {nom_employe}")
        ticket.append("----------------------------------------")
        ticket.append(f"{'Article':<20} {'Qte':<5} {'Prix':>10}")
        ticket.append("----------------------------------------")
        total = 0.0

        for nom, info in self.panier_actuel.items():
            if info["actif"]:
                if nom.startswith("ret_"):
                    nom_ticket = "RETRAIT ESPECES"
                    qte = 1
                    prix_total = info["prix_u"]
                else:
                    nom_ticket = nom[:20]
                    qte = info["quantite"]
                    prix_total = info["prix_u"] * qte
                total += prix_total
                ticket.append(f"{nom_ticket:<20} {('x'+str(qte)):<5} {prix_total:>9.2f}EUR")
        ticket.append("----------------------------------------")
        ticket.append(f"{'TOTAL PAYE:':<20} {total:>19.2f}EUR")
        ticket.append("========================================")
        ticket.append("       Merci de votre visite !          ")
        ticket.append("\n\n\n\n")
        texte_ticket = "\n".join(ticket)
```

Fichier ADMIN.PY

L'affichage détaillé d'une journée

```
def ouvrir_details_journalier(self, jour):
    win = ctk.CTkToplevel(self)
    win.title(f"Rapport complet - {jour}")
    win.geometry("950x700")
    win.after(100, win.focus_force)
    win.lift()
    win.focus_force()
    win.wait_visibility()

    top = ctk.CTkFrame(win, fg_color="white", height=70, corner_radius=0)
    top.pack(fill="x", pady=(0, 20))
    ctk.CTkLabel(top, text=f"REVENUES DE LA JOURNEE : {jour}", font=("Segoe UI", 18, "bold")).pack(side="left", padx=30)

    table_head = ctk.CTkFrame(win, fg_color="#2C3E50", height=40)
    table_head.pack(fill="x", padx=30)
    cols = [("HEURE", 0.05), ("NOM VENDEUR", 0.2), ("PRODUIT", 0.45), ("QUANTITE", 0.75), ("TOTAL", 0.9)]
    for t, p in cols:
        ctk.CTkLabel(table_head, text=t, text_color="white", font=("Arial", 11, "bold")).place(relx=p, rely=0.5, anchor="w")

    scroll = ctk.CTkScrollableFrame(win, fg_color="white", corner_radius=15)
    scroll.pack(fill="both", expand=True, padx=30, pady=15)

    total_jour = 0.0
    try:
        conn = sqlite3.connect("kaching.db")
        cursor = conn.cursor()
        cursor.execute("SELECT date_session, id_vendeur, produit, quantite, total_ligne FROM revenus WHERE date_session LIKE ? ORDER BY date_session ASC", (f"{jour}%",))
        rows = cursor.fetchall()
        conn.close()

        # --- LOGIQUE RETRAIT ---
        for r in rows:
            heure = r[0].split(" ")[1] if " " in r[0] else "--:--"
            nom_vendeur = chercher_nom_utilisateur(str(r[1]))
```

Cette fonction ouvre une nouvelle fenêtre affichant le rapport complet d'une journée. Elle se connecte à la base de données SQLite et récupère toutes les ventes enregistrées pour la date sélectionnée, triées par ordre chronologique.

Detaille des actions de la journée

```
def exporter_txt_journalier(self, jour, total_final):
    filename = f"Rapport_{jour.replace('/', '-')}.txt"
    path = filedialog.asksaveasfilename(defaultextension=".txt", initialfile=filename, title="Enregistrer le rapport")
    if not path: return

    try:
        now = datetime.datetime.now().strftime("%d/%m/%Y %H:%M")
        conn = sqlite3.connect("kaching.db")
        cursor = conn.cursor()
        cursor.execute("SELECT date_session, id_vendeur, produit, quantite, total_ligne FROM revenus WHERE date_session LIKE ? ORDER BY date_session ASC", (f"{jour}%",))
        ventes = cursor.fetchall()
        conn.close()

        with open(path, "w", encoding="utf-8") as f:
            f.write("================================================================================\n")
            f.write("                              RAPPORT DES REVENUES                              \n")
            f.write("                                   KACHING                                      \n")
            f.write("================================================================================\n\n")

            f.write(f" DATE DU RAPPORT : {jour}\n")
            f.write(f" GÉNÉRÉ LE       : {now}\n")
            f.write("\n" + "-" * 79 + "\n")

            header_line = f"{'HEURE':<8} | {'VENDEUR':<18} | {'PRODUIT':<25} | {'QUANTITÉ':<5} | {'TOTAL':<12}\n"
            f.write(header_line)
            f.write("-" * 79 + "\n")

            for v in ventes:
                heure = v[0].split(" ")[1] if " " in v[0] else "--:--"
                nom = chercher_nom_utilisateur(str(v[1]))
                nom_display = (nom[:15] + '..') if len(nom) > 17 else nom

                # Logique de nom pour le fichier texte aussi
                prod_brut = v[2]
                prod_clean = "RETRAIT D'ARGENT" if (prod_brut.startswith("ret_") or v[4] < 0) else prod_brut
                prod_display = (prod_clean[:22] + '..') if len(prod_clean) > 24 else prod_clean

                f.write(f"{heure:<8} | {nom_display:<18} | {prod_display:<25} | {v[3]:<5} | {v[4]:>8.2f} EUR\n")
```

Pour chaque ligne, elle récupère l'heure, va chercher le nom du vendeur auprès d'AlwaysData grâce à son identifiant, et affiche le produit avec son montant. Les

retraits d'argent sont détectés automatiquement et affichés en rouge, tandis que les ventes normales apparaissent en vert. Le total de la journée se calcule en additionnant uniquement les ventes réelles, sans compter les retraits d’argent.

Mes problèmes venues

1) **Transfert Windows vers Raspberry Pi**

Le projet a d'abord été développé sur Windows avant de tester sur le Raspberry Pi. Certaines bibliothèques ne fonctionnaient pas de la même façon sur les deux systèmes, ce qui m’a provoqué plusieurs modifications du code pour que tout fonctionne correctement sur le Raspberry Pi.

2) **Mettre le site accessible en local**

J'ai remarqué que le client ne pouvait payer que s'il était connecté au même réseau internet que le Raspberry pi, ce qui n'était pas professionnel. Donc j’ai décidé de le mettre sur PythonAnywhere pour le rendre accessible depuis n'importe quel réseau (4G/5G).

3) **Sous-compte**

Mon application ne distinguait pas correctement les deux comptes. Quand le premier paiement était validé, elle revenait directement vers la caisse sans prendre en compte le deuxième compte en attente.

4) **Imprimante thermique**

Au départ, j’avais choisi d’utiliser une imprimante thermique qui était plus pro mais j’ai remarqué qu’elle n’imprimait que du charabia et qu’elle n’était pas réellement compatible. Malgré même les nombreuse tentative(driver, demande au vendeur) donc j’ai décidé de passer vers une vrai imprimante.

5) **Crashs de l'application**

Lancer toutes les interfaces simultanément provoquait des crashs. La solution a été de créer toutes les pages une seule fois au démarrage et de n'en afficher qu'une à la fois, ce qui a permis de stabiliser l'application.

Conclusion

Ce projet m'a permis de créer une vraie caisse enregistreuse appelée Kaching, conçue pour tourner sur un Raspberry Pi 4. C'est un projet qui m'a demandé beaucoup de travail, de recherches et de patience, mais dont je suis vraiment fière du résultat.

J'ai appris à utiliser et combiner plusieurs technologies ensemble comme CustomTkinter pour l'interface, Flask et PythonAnywhere pour le paiement en ligne, AlwaysData pour la gestion des employés, SQLite pour stocker les articles et les revenus, et RealVNC pour contrôler le Raspberry Pi à distance. Le fait de faire communiquer tous ces outils entre eux était la partie la plus complexe du projet.

J'ai aussi rencontré pas mal de problèmes en cours de route, que ce soit le transfert de Windows vers le Raspberry Pi, les crashs de l'application, l'imprimante thermique incompatible ou encore les sous-compte. Chaque problème m'a obligé à chercher, tester et trouver des solutions par moi-même, ce qui m'a vraiment appris à développer mon autonomie et ma logique de programmation.

Si je devais améliorer le projet, j'ajouterais une vraie interface pour gérer les articles directement depuis l'application sans passer par la base de données et j'intégrerais un vrai système de paiement comme Stripe ou PayPal.

Pour finir, ce projet de TFE m'a vraiment permis de mettre en pratique tout ce que j'ai appris durant mes 3 années à l'Inraci. Il m'a confirmé que j'aime vraiment programmer et créer des applications. Je suis contente d'avoir choisi ce projet et d'avoir pu le mener à bien. Et j'espère en faire mon métier.

Bibliographie

1) Adafruit Industries. « *Mini thermal receipt printer*. » https://learn.adafruit.com/mini-thermal-receipt-printer/making-connections ,consulté le 10 avril 2026.

2) Circuit Digest. « *Interfacing USB barcode scanner with Raspberry Pi* » https://circuitdigest.com/microcontroller-projects/interfacing-usb-barcode-scanner-with-raspberry-pi-4, consulté le 15 février 2026.

3) Clubic. « *Avis Alwaysdata 2026 : que vaut cet hébergeur français et éco-responsable ?* ». https://www.clubic.com/avis-602821-avis-alwaysdata-2026-que-vaut-cet-hebergeur-francais-et-eco-responsable.html , consulté le 15 février 2026.

4) Hugatry's HackVlog. « *WiFi-Enabled Printer Using Raspberry Pi 3 and USB-printer* » https://www.youtube.com/watch?v=va6PWYZScZk , consulté le 21 mars 2026.

5) Le Pro du PC. « *Raspberry Pi 4 : guide complet pour exploiter ce mini-ordinateur polyvalent*. » https://www.leprodupc.fr/raspberry-pi-4-guide-complet-pour-exploiter-ce-mini-ordinateur-polyvalent/ , consulté le 5 mai 2026.

6) Rachid TANJAOUI. *« Raspberry pi et barcode scanner »* https://www.youtube.com/watch?v=S3xCwoWGQII , consulté le 15 février 2026.

7) Pier Aisa. *« 0300: Lettore Codici a Barre con Arduino o Raspberry ?? »* https://www.youtube.com/watch?v=2d1hc_2WVq0 , consulté le 16 février 2026.

8) PiddlerInTheRoot. *« Wireless Barcode Scanner (Raspberry Pi) »* https://www.youtube.com/watch?v=BEsUpozvJCU , consulté le 16 février 2026

Annexe


```python
import mysql.connector

def get_auth_connection():
    return mysql.connector.connect(
        host="mysql-lekskej.alwaysdata.net",
        user="lekskej_awa ",
        password="Iranienne2007",
        database="lekskej_caisse"
    )

def chercher_nom_utilisateur(user_id):
    try:
        conn = get_auth_connection()
        cursor = conn.cursor()
        # On récupère le nom_user correspondant à l'ID[cite: 14, 23]
        query = "SELECT nom_user FROM utilisateurs WHERE id_user = %s"
        cursor.execute(query, (user_id,))
        resultat = cursor.fetchone()
        conn.close()
        return resultat[0] if resultat else "Inconnu"
    except Exception as e:
        print(f"Erreur Alwaysdata : {e}")
        return "Erreur"
```

```python
def init_db():
    conn = sqlite3.connect("kaching.db")
    cursor = conn.cursor()

    # Table des produits
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS produits (
            code TEXT PRIMARY KEY,
            code_barre TEXT,
            nom TEXT NOT NULL,
            prix REAL NOT NULL
        )
    ''')

    # NOUVELLE TABLE : revenus pour l'admin
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS revenus (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            date_session TEXT,
            id_vendeur TEXT,
            produit TEXT,
            prix_u REAL,
            quantite INTEGER,
            total_ligne REAL
        )
    ''')
```