 **REPUBLIQUE DU SENEGAL**

**UNIVERSITE IBA DER THIAM DE THIES**

**UFR Des Sciences Economiques et Sociales**

**&&**

**UFR Des Sciences et Technologies**

**Filière**

**Master en science de données et Applications**

**Un Peuple \_ Un But \_ Une Foi**

**Ministère de l’enseignement**

**Supérieur de la recherche et de l’innovation**

**PROJET :** **Application pour la Gestion des Produits Pharmaceutiques d’une Base de donnée Pharmacie**

**Membres**

* **Ndiaga Fall**
* **Yarame Ba**
* **Mouskeba BADJINKA**
* **Awa Sene**

**Professeur Mr Ly**

**Table des matières**

1. ***Introduction***
2. ***Description Fonctionnelle de l'Application***
3. ***Modélisation et Conception de la Base de Données***
4. ***Interface Utilisateur en Python***
5. ***Connexion entre Python et SQL***
6. ***Importation des bibliothèques***
7. ***Exemples de Cas d'Utilisation***
8. ***Conclusion et Perspectives***
9. *Introduction*

Cette application de gestion de pharmacie est conçue pour gérer efficacement le stock de produits pharmaceutiques en interfaçant Python avec MySQL. Elle permet d'ajouter, modifier, supprimer et rechercher des produits, ainsi que de visualiser l'inventaire et les dates de péremption des médicaments. L'interface graphique, développée avec customtkinter, offre une gestion fluide des opérations CRUD (Create, Read, Update, Delete) sur les produits.

1. **Fonctionnalités principales :**

* **Ajouter des produits** : Ajout de nouveaux articles avec nom, description, prix et quantité en stock.
* **Modifier des produits** : Mise à jour des informations de produit existantes.
* **Supprimer des produits** : Suppression de produits de la base de données.
* **Rechercher un produit**: Recherche des produits en stock ou rupture de stock.
* **Affichage de tous les produits** : Consultation de l'inventaire complet avec leurs détails.
* **Gestion du stock et des dates de péremption** : Suivi des quantités et expiration des produits.

1. **Objectif :**

Le but de ce projet est de créer une application complète qui permet de gérer efficacement les stocks d'une pharmacie. Elle simplifie la gestion des produits en pharmacie, améliore le suivi des stocks, et évite les ruptures ou les produits périmés.

1. **Contexte et justification :** Dans le secteur pharmaceutique, la gestion des stocks est un enjeu crucial pour assurer la disponibilité des médicaments et éviter les pertes liées aux produits périmés. Les pharmacies, souvent confrontées à une multitude de produits à gérer, doivent disposer d'un système efficace pour suivre les entrées, les sorties, ainsi que la date de péremption des médicaments. Une mauvaise gestion des stocks peut entraîner des ruptures ou des surplus, ce qui affecte la qualité du service offert aux patients. Ce projet vise à répondre à ces défis en proposant une application de gestion de stock adaptée aux besoins spécifiques des pharmacies, afin d'améliorer leur performance et leur organisation.
2. **Objectifs du projet :**

* **Automatiser la gestion des stocks** pour réduire les erreurs humaines et améliorer l'efficacité opérationnelle.
* **Faciliter le suivi des dates de péremption** pour éviter la vente de produits expirés.
* **Simplifier les opérations CRUD** (création, lecture, mise à jour, suppression) sur les produits afin de permettre une gestion fluide et intuitive des articles en stock.
* **Améliorer la visibilité des stocks en temps réel** afin d’éviter les ruptures de stock ou la surabondance de produits.
* **Créer une interface utilisateur intuitive** qui permet aux employés de la pharmacie d'accéder facilement aux informations sur les produits.

1. **Technologies utilisées :**

* **Python** : Langage principal pour le développement de l'application.
* **MySQL** : Base de données relationnelle pour stocker et gérer les informations des produits pharmaceutiques.
* **Django** : Framework web utilisé pour créer une interface administrateur permettant une gestion en ligne des stocks.
* **Customtkinter** : Bibliothèque Python pour concevoir l'interface graphique utilisateur.
* **Tkinter** : Utilisé pour la création d'interfaces graphiques simples et intuitives.

#### *Description Fonctionnelle de l'Application*

L’application de gestion de stock pour pharmacie propose une série de fonctionnalités essentielles pour une gestion fluide et précise des produits en stock. Voici un aperçu détaillé des principales fonctions :

#### *Ajouter un produit :*

Cette fonctionnalité permet à l'utilisateur d'insérer un nouveau produit dans la base de données. Les informations à renseigner incluent :

* **Nom du produit**
* **Description du produit**
* **Prix unitaire**
* **Quantité disponible** Une fois les données saisies et confirmées, le produit est ajouté au stock et devient accessible pour toutes les opérations futures.

#### *Modifier un produit :*

Cette fonctionnalité offre la possibilité d’éditer les informations d'un produit existant dans la base de données. L'utilisateur peut changer des éléments comme le nom, la description, le prix ou la quantité disponible. Cela permet d'ajuster les données en fonction des besoins ou des mises à jour.

#### *Supprimer un produit :*

L'utilisateur peut retirer un produit du stock via cette fonction. Si un produit n'est plus disponible ou ne doit plus être proposé, il peut être définitivement supprimé de la base de données, garantissant une vue claire et à jour des produits actifs.

#### *Rechercher un produit :*

Cette fonctionnalité permet de localiser rapidement un produit spécifique dans le stock en utilisant des critères de recherche tels que le nom ou la description. Cela aide l'utilisateur à accéder rapidement aux informations sur un produit particulier.

#### *Afficher tous les produits :*

L'utilisateur peut visualiser la liste complète des produits disponibles dans le stock. Cette vue globale présente tous les produits avec leurs informations principales (nom, description, prix, quantité), offrant ainsi un aperçu rapide de la situation du stock.

#### *Produits en rupture :*

Cette fonctionnalité génère une liste des produits dont la quantité en stock est nulle ou en dessous d'un seuil critique. Cela aide l'utilisateur à identifier les articles nécessitant un réapprovisionnement rapide, garantissant que la pharmacie ne manque jamais de produits essentiels.

Ces fonctionnalités assurent une gestion optimale des stocks, permettant à la pharmacie de maintenir un contrôle efficace sur ses produits tout en répondant aux besoins des clients.

La base de données de l'application de gestion de stock pour pharmacie est organisée de manière à assurer une gestion efficace des produits et des utilisateurs. Elle contient deux principales tables : gestion\_pharmacie\_produit et gestion\_pharmacie\_utilisateur, chacune jouant un rôle clé dans l'enregistrement des informations relatives aux produits et aux utilisateurs. Voici le schéma et la description des tables.

#### *Schéma de la base de données*

gestion\_pharmacie\_utilisateur gestion\_pharmacie\_produit

----------------------------- ---------------------------

| id\_utilisateur (PK) | | id\_produit (PK) |

| nom | | nom |

| email | | description |

| mot\_de\_passe | | prix |

| role | | quantite |

----------------------------- ---------------------------

#### *Modélisation et Conception de la Base de Données*

Relations

gestion\_pharmacie\_utilisateur --- gestion\_pharmacie\_produit

(relation potentielle selon l'usage)

#### *Description des tables*

1. **gestion\_pharmacie\_produit**
   * Cette table stocke les informations relatives aux produits disponibles dans la pharmacie.
   * **Champs :**
     + id\_produit (INT, PK) : Identifiant unique pour chaque produit.
     + nom (VARCHAR) : Nom du produit.
     + description (TEXT) : Détails ou description du produit.
     + prix (DECIMAL) : Prix unitaire du produit.
     + quantite (INT) : Quantité de produit disponible en stock.
   * **Clé primaire :** id\_produit est la clé primaire, assurant l’unicité de chaque produit dans la base de données.
2. **gestion\_pharmacie\_utilisateur**
   * Cette table gère les utilisateurs de l'application, qu'ils soient administrateurs ou autres rôles.
   * **Champs :**
     + id\_utilisateur (INT, PK) : Identifiant unique pour chaque utilisateur.
     + nom (VARCHAR) : Nom complet de l'utilisateur.
     + email (VARCHAR) : Adresse email de l'utilisateur, utilisée pour l'identification.
     + mot\_de\_passe (VARCHAR) : Mot de passe crypté pour la connexion sécurisée.
     + role (VARCHAR) : Le rôle de l'utilisateur (par exemple, administrateur ou utilisateur standard).
   * **Clé primaire :** id\_utilisateur est la clé primaire qui garantit que chaque utilisateur est unique dans la base.

#### *Clés primaires et étrangères*

* **Clés primaires :**
  + Dans la table gestion\_pharmacie\_produit, la clé primaire est id\_produit.
  + Dans la table gestion\_pharmacie\_utilisateur, la clé primaire est id\_utilisateur.
* **Clés étrangères :**
  + Si l'application nécessite une relation directe entre les utilisateurs et les produits (par exemple, un historique d'ajout ou de gestion des produits par des utilisateurs spécifiques), une clé étrangère id\_utilisateur peut être ajoutée à la table gestion\_pharmacie\_produit pour créer un lien entre les deux tables.
  + Dans ce cas, id\_utilisateur deviendrait une clé étrangère dans la table gestion\_pharmacie\_produit, pointant vers la clé primaire id\_utilisateur dans gestion\_pharmacie\_utilisateur.

Cela permettrait de suivre quel utilisateur a ajouté ou modifié un produit, renforçant la traçabilité et la gestion des produits dans la base de données.

#### *Interface Utilisateur en Python*

L’interface utilisateur (IU) de l’application de gestion de stock pour pharmacie a été conçue en utilisant CustomTkinter pour offrir une expérience fluide et intuitive à l'utilisateur. L’objectif principal est de permettre aux utilisateurs (administrateurs et gestionnaires de stock) d’interagir facilement avec la base de données pour ajouter, modifier, supprimer et consulter des produits.

#### *Présentation de l’interface graphique*

L'interface graphique est composée de plusieurs sections principales :

1. **Barre de navigation :**
   * Une barre latérale ou supérieure permet à l’utilisateur de naviguer facilement entre les différentes fonctionnalités telles que l'ajout de produits, la modification, la suppression, et la recherche. Elle offre un accès rapide à chaque fonctionnalité avec des boutons bien organisés et étiquetés.
2. **Tableau de bord des produits :**
   * Cette section affiche une liste de tous les produits stockés dans la base de données, avec des colonnes indiquant le nom du produit, sa description, son prix, et sa quantité disponible.
   * Les utilisateurs peuvent parcourir la liste des produits et filtrer ou rechercher des produits spécifiques à partir d'une barre de recherche.
3. **Formulaires de gestion des produits :**
   * **Ajout de produit :** Un formulaire dédié permet à l’utilisateur de saisir les informations essentielles d’un produit (nom, description, prix, quantité) avant de l’ajouter à la base de données.
   * **Modification de produit :** L’utilisateur peut sélectionner un produit existant dans la liste, et ses informations actuelles s'affichent dans le formulaire pour être modifiées.
   * **Suppression de produit :** Un bouton "Supprimer" permet de retirer un produit sélectionné de la base de données après confirmation.
4. **Notifications et messages :**
   * Des notifications apparaissent en haut ou en bas de l’interface pour informer l'utilisateur en cas de succès (ex. "Produit ajouté avec succès") ou d'erreur (ex. "Erreur : Produit introuvable").

#### *Explication des interactions*

Les interactions dans l'interface sont intuitives et bien définies :

1. **Ajouter un produit :**
   * L'utilisateur clique sur le bouton "Ajouter un produit", ce qui ouvre un formulaire.
   * L’utilisateur saisit le nom, la description, le prix et la quantité du produit.
   * Après validation (avec le bouton "Ajouter"), le produit est stocké dans la base de données, et la liste des produits est mise à jour automatiquement pour inclure le nouvel élément.
2. **Modifier un produit :**
   * L'utilisateur sélectionne un produit dans la liste affichée sur le tableau de bord.
   * Un clic sur le bouton "Modifier" permet de charger les informations actuelles du produit dans le formulaire de modification.
   * L’utilisateur modifie les champs souhaités (prix, quantité, etc.) et clique sur "Mettre à jour". Les changements sont reflétés dans la base de données et la liste des produits.
3. **Supprimer un produit :**
   * En sélectionnant un produit dans le tableau de bord, l’utilisateur peut cliquer sur "Supprimer".
   * Une fenêtre de confirmation s’ouvre pour éviter les suppressions accidentelles. Si l’utilisateur confirme, le produit est supprimé de la base de données, et la liste est actualisée.
4. **Rechercher un produit :**
   * Une barre de recherche est présente pour permettre à l'utilisateur de trouver un produit spécifique en tapant un mot-clé (nom ou description).
   * Les résultats s’affichent instantanément dans la liste des produits.
5. **Afficher les produits en rupture :**
   * Un bouton "Produits en rupture" permet d’afficher uniquement les produits dont la quantité est égale à zéro. Cela aide à identifier rapidement les articles nécessitant un réapprovisionnement.

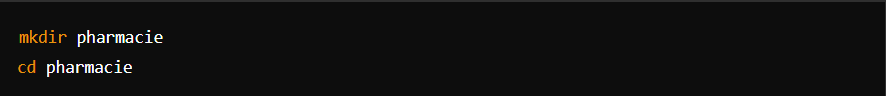
L'interface est donc centrée sur l'utilisateur avec des actions claires et des retours visuels (messages, notifications, mise à jour de la liste de produits) permettant une gestion efficace et en temps réel du stock de la pharmacie.

#### *Connexion entre Python et SQL*

Dans le cadre de ce projet, nous avons intégré Python et SQL en utilisant Django, un framework web puissant et flexible. L'objectif principal de cette intégration est de gérer efficacement les données relatives aux produits, commandes, fournisseurs, et utilisateurs au sein d'une application de gestion de pharmacie.

1. **Création projet django**

Commencez par créer un répertoire nommé pharmacie, puis accédez à ce répertoire dans votre terminal.



Installez Django via pip

Avec Django installé, vous pouvez créer le projet gestion\_pharmacie.



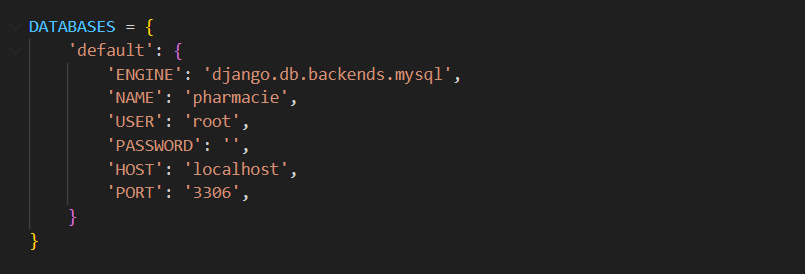
**Note**: Le point (.) à la fin de la commande place le projet directement dans le dossier courant (pharmacie), évitant la création d’un sous-répertoire supplémentaire.

Nous pouvons maintenant commencer à ajouter des applications pour la gestion des stocks, des produits, des utilisateurs, etc., en utilisant la commande suivante pour chaque nouvelle application :



### Configuration de la Base de Données

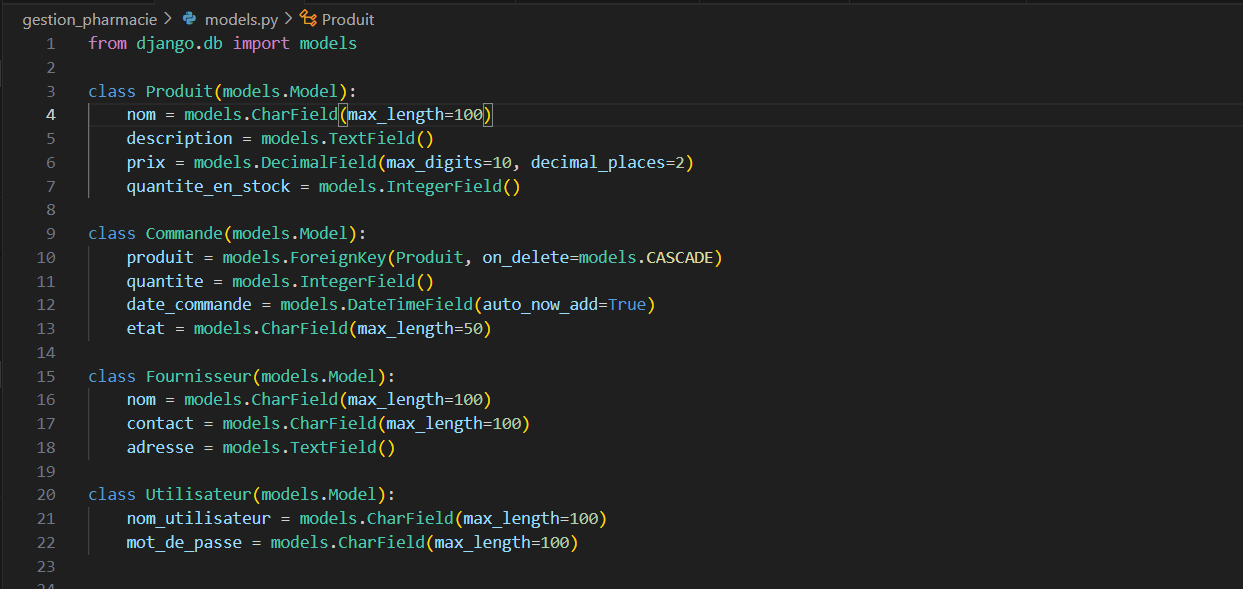
La configuration de la base de données est essentielle pour établir une connexion entre Django et MySQL. Cela se fait dans le fichier settings.py du projet. Voici les paramètres utilisés :



* **ENGINE** : Indique que nous utilisons MySQL comme moteur de base de données.
* **NAME** : Le nom de la base de données que nous avons créée pour stocker les informations de la pharmacie.
* **USER** : Le nom d'utilisateur pour accéder à la base de données.
* **PASSWORD** : Le mot de passe associé à l'utilisateur.
* **HOST** et **PORT** : Définissent l'adresse et le port de connexion à la base de données.

### Modèles de la Base de Données

Les modèles de données représentent la structure des tables dans la base de données. Dans notre application, nous avons défini plusieurs modèles correspondant aux entités essentielles de notre système :



* **Produit** : Représente les médicaments ou produits disponibles dans la pharmacie, incluant des attributs tels que le nom, la description, le prix et la quantité en stock.
* **Commande** : Modèle pour enregistrer les commandes passées, associant un produit à la quantité commandée et à la date de commande.
* **Fournisseur** : Stocke les informations relatives aux fournisseurs des produits.
* **Utilisateur** : Gère les informations des utilisateurs du système.

### Conclusion

L'intégration entre Python et SQL via Django permet de créer une application robuste et extensible pour la gestion des pharmacies. Grâce à la définition claire des modèles de données et à la configuration appropriée de la base de données, nous pouvons assurer une gestion efficace des informations, facilitant ainsi les opérations quotidiennes de la pharmacie.

### Importation des bibliothèques

from customtkinter import \*

import tkinter as tk

from tkinter import messagebox

import MySQLdb

from PIL import Image, ImageTk

import customtkinter as ctk

* **from customtkinter import \*** : Importe toutes les classes et fonctions de la bibliothèque customtkinter, qui permet de créer des interfaces graphiques esthétiques et personnalisables basées sur tkinter.
* **import tkinter as tk** : Importe la bibliothèque standard tkinter, qui est utilisée pour créer des interfaces graphiques en Python, et lui donne l'alias tk pour simplifier son utilisation.
* **from tkinter import messagebox** : Importe le module messagebox de tkinter, qui permet d'afficher des boîtes de dialogue pour informer ou avertir l'utilisateur.
* **import MySQLdb** : Importe la bibliothèque qui permet d'interagir avec une base de données MySQL.
* **from PIL import Image, ImageTk** : Importe les classes Image et ImageTk de la bibliothèque Pillow, qui est utilisée pour manipuler et afficher des images.
* **import customtkinter as ctk** : Importe customtkinter avec l'alias ctk, ce qui facilite l'utilisation de ses fonctionnalités.

### Fonctions pour la gestion des produits

#### *Fonction pour ajouter un produit*

def ajouter\_produit():

nom = champ\_nom.get()

description = champ\_description.get()

prix = champ\_prix.get()

quantite = champ\_quantite.get()

cursor.execute("INSERT INTO gestion\_pharmacie\_produit (nom, description, prix, quantite\_en\_stock) VALUES (%s, %s, %s, %s)",

(nom, description, prix, quantite))

db.commit()

messagebox.showinfo("Succès", "Produit ajouté avec succès")

champ\_nom.delete(0, tk.END)

champ\_description.delete(0, tk.END)

champ\_prix.delete(0, tk.END)

champ\_quantite.delete(0, tk.END)

* **Récupération des valeurs** : Récupère les valeurs saisies dans les champs d'entrée pour le nom, la description, le prix et la quantité du produit.
* **Insertion dans la base de données** : Utilise un curseur pour exécuter une requête SQL qui insère un nouveau produit dans la table gestion\_pharmacie\_produit.
* **Commit** : Sauvegarde les changements dans la base de données afin que l'insertion soit effective.
* **Message de succès** : Affiche une boîte de dialogue pour informer l'utilisateur que le produit a été ajouté avec succès.
* **Réinitialisation des champs** : Vide les champs d'entrée après l'ajout du produit pour permettre une nouvelle saisie.

#### *Fonction pour modifier un produit*

def modifier\_produit():

nom = champ\_nom1.get()

description = champ\_description1.get()

prix = champ\_prix1.get()

quantite = champ\_quantite1.get()

cursor.execute("UPDATE gestion\_pharmacie\_produit SET description=%s, prix=%s, quantite\_en\_stock=%s WHERE nom=%s",

(description, prix, quantite, nom))

db.commit()

messagebox.showinfo("Succès", "Produit modifié avec succès")

champ\_nom1.delete(0, tk.END)

champ\_description1.delete(0, tk.END)

champ\_prix1.delete(0, tk.END)

champ\_quantite1.delete(0, tk.END)

* **Modification des valeurs** : Récupère les nouvelles valeurs pour le nom, la description, le prix et la quantité du produit à partir des champs d'entrée.
* **Requête de mise à jour** : Exécute une requête SQL qui met à jour les informations d'un produit existant basé sur son nom.
* **Confirmation** : Affiche une boîte de dialogue pour confirmer que le produit a été modifié avec succès.

#### *Fonction pour supprimer un produit*

def supprimer\_produit():

nom = champ\_nom2.get()

cursor.execute("DELETE FROM gestion\_pharmacie\_produit WHERE nom=%s", (nom,))

db.commit()

messagebox.showinfo("Succès", "Produit supprimé avec succès")

champ\_nom2.delete(0, tk.END)

* **Suppression d'un produit** : Récupère le nom du produit à supprimer et exécute une requête SQL pour le retirer de la base de données.
* **Message de confirmation** : Affiche une boîte de dialogue pour informer l'utilisateur que le produit a été supprimé avec succès.

#### *Fonction pour rechercher un produit*

def rechercher\_produit():

nom = champ\_nom3.get()

cursor.execute("SELECT nom, description, prix, quantite\_en\_stock FROM gestion\_pharmacie\_produit WHERE nom=%s", (nom,))

resultat = cursor.fetchone()

if resultat:

nom, description, prix, quantite\_en\_stock = resultat

details = f"Nom: {nom}\nDescription: {description}\nPrix: {prix} FCFA\nQuantité en stock: {quantite\_en\_stock}"

messagebox.showinfo("Produit trouvé", details)

else:

messagebox.showwarning("Rupture de stock", "Le produit est en rupture de stock ou n'existe pas.")

champ\_nom3.delete(0, tk.END)

* **Recherche dans la base de données** : Exécute une requête SQL pour rechercher un produit par son nom.
* **Affichage des détails** : Si le produit est retrouvé, affiche ses détails dans une boîte de dialogue. Sinon, affiche un avertissement que le produit n'existe pas ou est en rupture de stock.
* **Réinitialisation des champs** : Vide le champ d'entrée après la recherche.

### Fonction pour charger une image avec opacité

def load\_image\_with\_opacity(path, opacity, new\_size):

image = Image.open(path).resize(new\_size, Image.Resampling.LANCZOS)

image = image.convert("RGBA")

alpha = image.split()[3]

alpha = alpha.point(lambda p: p \* opacity)

image.putalpha(alpha)

return ImageTk.PhotoImage(image)

* **Chargement et redimensionnement** : Charge une image à partir du chemin spécifié, la redimensionne pour s'adapter à la taille souhaitée.
* **Conversion de l'image** : Convertit l'image en format RGBA pour permettre la manipulation de la transparence.
* **Ajustement de l'opacité** : Modifie le canal alpha (transparence) de l'image en fonction du facteur d'opacité fourni.
* **Renvoi de l'image** : Retourne l'image prête à être affichée dans l'interface graphique.

### Fonction pour redimensionner l'image selon le canvas

def update\_image\_size(event):

canvas\_width = event.width

canvas\_height = event.height

image\_with\_opacity = load\_image\_with\_opacity(image\_path, 0.5, (canvas\_width, canvas\_height))

event.widget.image = image\_with\_opacity

event.widget.create\_image(0, 0, image=image\_with\_opacity, anchor="nw")

* **Événement de redimensionnement** : Cette fonction est appelée chaque fois que le canvas est redimensionné.
* **Mise à jour de l'image** : Redimensionne l'image selon les nouvelles dimensions du canvas et l'affiche.

### Connexion à la base de données

db = MySQLdb.connect(user='root', password='', host='localhost', database='pharmacie')

cursor = db.cursor()

* **Connexion** : Établit une connexion à la base de données MySQL avec les informations d'identification fournies.
* **Curseur** : Crée un curseur qui sera utilisé pour exécuter des requêtes SQL.

### Création de l'application

app = CTk()

app.geometry("1225x700")

app.title("Gestion de la Pharmacie")

* **Création de l'application** : Initialise la fenêtre principale de l'application avec une taille de 1225 pixels de large et 700 pixels de haut, et définit le titre de la fenêtre.

### Création de l'onglet de navigation

tabview = CTkTabview(master=app)

tabview.pack(padx=20, pady=20, fill='both', expand=True)

tabview.add("Ajouter des Produits")

tabview.add("Modifier des Produits")

tabview.add("Supprimer des Produits")

tabview.add("Rechercher")

tabview.add("Tous les produits")

tabview.add("Stock et date de péremption")

* **Création des onglets** : Utilise CTkTabview pour créer plusieurs onglets qui permettront à l'utilisateur d'accéder facilement aux différentes fonctionnalités de l'application.

### Création des canvases pour chaque onglet

canvas = tk.Canvas(tabview.tab("Ajouter des Produits"), width=1225, height=700)

canvas.pack(fill="both", expand=True)

# Répété pour d'autres onglets...

* **Canvases** : Crée un canvas pour chaque onglet, où des éléments graphiques peuvent être dessinés et organisés.

### Ajout d'une scrollbar pour l'onglet "Ajouter des Produits"

scrollbar = tk.Scrollbar(canvas)

scrollbar.pack(side="right", fill="y")

canvas.config(yscrollcommand=scrollbar.set)

scrollbar.config(command=canvas.yview)

* **Barre de défilement** : Crée une barre de défilement pour permettre à l'utilisateur de naviguer dans les produits ajoutés s'ils dépassent la taille de la fenêtre.

### Interface graphique pour ajouter un produit

champ\_nom = CTkEntry(tabview.tab("Ajouter des Produits"), placeholder\_text="Nom du produit")

champ\_nom.pack(pady=10)

champ\_description = CTkEntry(tabview.tab("Ajouter des Produits"), placeholder\_text="Description du produit")

champ\_description.pack(pady=10)

champ\_prix = CTkEntry(tabview.tab("Ajouter des Produits"), placeholder\_text="Prix du produit")

champ\_prix.pack(pady=10)

champ\_quantite = CTkEntry(tabview.tab("Ajouter des Produits"), placeholder\_text="Quantité")

champ\_quantite.pack(pady=10)

btn\_ajouter = CTkButton(tabview.tab("Ajouter des Produits"), text="Ajouter Produit", command=ajouter\_produit)

btn\_ajouter.pack(pady=10)

* **Champs d'entrée** : Crée des champs d'entrée pour permettre à l'utilisateur de saisir le nom, la description, le prix et la quantité d'un produit.
* **Bouton d'ajout** : Crée un bouton qui appelle la fonction ajouter\_produit lorsqu'il est cliqué.

### Autres interfaces pour les onglets

* **Répétez le processus ci-dessus pour les onglets de modification, suppression et recherche**, en adaptant les champs d'entrée et les boutons en conséquence.

### Exécution de l'application

app.mainloop()

* **Boucle principale** : Démarre la boucle principale de l'application, qui attend les interactions de l'utilisateur.

#### *Exemples de Cas d'Utilisation*

Dans le cadre de la gestion d'une pharmacie, il est crucial de disposer de fonctionnalités permettant de gérer efficacement les produits disponibles. Nous avons identifié trois cas d'utilisation principaux : l'ajout de nouveaux produits, la modification des informations sur les produits existants et l'affichage des produits disponibles. Ces cas d'utilisation sont essentiels pour assurer une gestion optimale du stock et une bonne expérience utilisateur.

### 1. Ajout de Produits

**Description :** Ce cas d'utilisation permet à un utilisateur, généralement un administrateur ou un gestionnaire de pharmacie, d'ajouter de nouveaux produits à la base de données.

**Acteurs :** Administrateur / Gestionnaire

**Préconditions :** L'utilisateur doit être authentifié et avoir les droits d'accès nécessaires pour ajouter des produits.

**Scénario principal :**

1. L'utilisateur se connecte à l'application.
2. L'utilisateur navigue vers la section « Ajouter un produit ».
3. L'utilisateur saisit les informations requises : nom, description, prix et quantité en stock.
4. L'utilisateur soumet le formulaire.
5. Le système enregistre les nouvelles informations dans la base de données.
6. Un message de confirmation s'affiche pour indiquer que le produit a été ajouté avec succès.

**Postconditions :** Le produit est ajouté à la base de données et est désormais visible dans la liste des produits.

### 2. Modification de Produits

**Description :** Ce cas d'utilisation permet à un utilisateur d'apporter des modifications aux informations d'un produit déjà existant dans la base de données.

**Acteurs :** Administrateur / Gestionnaire

**Préconditions :** L'utilisateur doit être authentifié et avoir les droits d'accès nécessaires pour modifier les produits.

**Scénario principal :**

1. L'utilisateur se connecte à l'application.
2. L'utilisateur navigue vers la section « Afficher les produits ».
3. L'utilisateur sélectionne le produit à modifier.
4. L'utilisateur accède au formulaire de modification.
5. L'utilisateur modifie les informations souhaitées (nom, description, prix ou quantité).
6. L'utilisateur soumet le formulaire.
7. Le système met à jour les informations du produit dans la base de données.
8. Un message de confirmation s'affiche pour indiquer que le produit a été modifié avec succès.

**Postconditions :** Les informations du produit sont mises à jour dans la base de données.

### 3. Affichage de Produits

**Description :** Ce cas d'utilisation permet à un utilisateur de visualiser la liste des produits disponibles dans la pharmacie.

**Acteurs :** Administrateur / Gestionnaire / Utilisateur

**Préconditions :** Aucun prérequis d'authentification n'est nécessaire pour afficher les produits.

**Scénario principal :**

1. L'utilisateur se connecte à l'application (facultatif).
2. L'utilisateur navigue vers la section « Afficher les produits ».
3. Le système récupère les données des produits depuis la base de données.
4. Le système affiche la liste des produits, incluant le nom, la description, le prix et la quantité en stock.
5. L'utilisateur peut filtrer ou rechercher des produits si cette fonctionnalité est disponible.

**Postconditions :** L'utilisateur visualise la liste des produits avec leurs détails.

### Conclusion

Les cas d'utilisation décrits illustrent les fonctionnalités fondamentales de gestion des produits dans l'application de pharmacie. Ils garantissent que les utilisateurs peuvent ajouter, modifier et consulter les informations relatives aux produits de manière intuitive et efficace, contribuant ainsi à l'optimisation des opérations de la pharmacie.

#### *Conclusion et Perspectives*

### Bilan

Le projet de développement d'une application de gestion de pharmacie a été mené à bien avec succès, atteignant les principaux objectifs fixés au début du processus. Grâce à une approche méthodique, nous avons réussi à concevoir et à implémenter une solution qui répond aux besoins spécifiques des utilisateurs tout en intégrant les meilleures pratiques en matière de développement de logiciels.

Les principales fonctionnalités, telles que l'ajout, la modification et l'affichage des produits, ont été mises en place avec une interface conviviale. Les tests approfondis ont confirmé la robustesse de l'application, garantissant son efficacité dans la gestion quotidienne des opérations d'une pharmacie.

### Améliorations Futures

Bien que l'application ait atteint un bon niveau de fonctionnalité, plusieurs pistes d'amélioration peuvent être envisagées pour enrichir l'expérience utilisateur et optimiser les performances de l'application :

1. **Intégration de Fonctionnalités Avancées :**
   * Ajout d'un système de gestion des stocks plus complexe, incluant des alertes automatiques pour les produits en rupture de stock.
   * Développement d'un module de rapport analytique permettant aux utilisateurs de visualiser les performances de vente et les tendances de consommation.
2. **Amélioration de l'Interface Utilisateur :**
   * Optimisation de l'interface pour une navigation encore plus intuitive, en tenant compte des retours des utilisateurs.
   * Intégration de fonctionnalités de personnalisation pour que les utilisateurs puissent adapter l'application à leurs besoins spécifiques.
3. **Sécurité Renforcée :**
   * Mise en place de protocoles de sécurité supplémentaires pour protéger les données sensibles des utilisateurs, notamment le chiffrement des informations personnelles et des transactions.
   * Intégration d'une authentification multi-facteurs pour renforcer la sécurité des comptes utilisateurs.
4. **Évolutivité et Maintenance :**
   * Préparation de l'application pour qu'elle soit facilement évolutive afin de s'adapter à la croissance de l'entreprise et à l'évolution des technologies.
   * Établissement d'un plan de maintenance régulière pour garantir la mise à jour continue des fonctionnalités et la correction des éventuelles failles de sécurité.
5. **Formation et Support Utilisateur :**
   * Mise en place de programmes de formation pour les utilisateurs afin de les familiariser avec l'application et ses nouvelles fonctionnalités.
   * Création d'un système de support technique efficace pour répondre rapidement aux problèmes rencontrés par les utilisateurs.

### Conclusion

En somme, ce projet a non seulement permis de développer une solution adaptée aux besoins d'une pharmacie, mais il a également mis en lumière l'importance de l'innovation continue et de l'adaptabilité dans le domaine de la technologie. Avec des améliorations futures envisagées, l'application de gestion de pharmacie a le potentiel de devenir un outil indispensable pour optimiser les opérations, améliorer la satisfaction client et favoriser la croissance de l'entreprise.