|  |  |
| --- | --- |
|  | Institut de Formation Supérieure de la Ville de Wavre  **IFOSUP** |



Epreuve intégrée de la section

Technicien en programmation

2019-2020

**Application bureau : gestion d’hôtels**

Superviseur de projet :

L’Épreuve intégrée présenté par **Diaz Rincon Julieth** en vue de l'obtention du diplôme de technicien en programmation

**M. Vanconingsloo**

Table des matières

1. Introduction1
2. Description du projet1
3. Analyse1
4. DESIGN PRELIMINAIRE DE L’APPLICATION BUREAU 1
5. Page d’accueil 1
6. Page de gestion des hôtels2
7. Page gestion des villes2
8. *Page gestion des réservations2*
9. DIAGRAMME DE LA BASE DE DONNEE3
10. Structure des tables 2
11. RÉalisation4
12. **PAGE D’ACCUEIL 2**
13. **PAGE DE GESTION DES HOTELS2**
14. **PAGE GESTION DES VILLES2**
15. **PAGE GESTION DES RESERVATIONS2**
16. AmÉliorations Possibles7

Tapez le titre du chapitre (niveau 2)7

1. Conclusion7
2. Bibliographie7
3. Annexes7

# Introduction

J’ai décidé de réaliser mes études en technicien de programmation, car je voulais approfondir mes connaissances en Python. En effet, j’avais déjà eu l’occasion d’utiliser ce langage de programmation au cours du bachelier en informatique de gestion que je suis en parallèle, l’apprentissage de celui-ci n’était pas aussi approfondi que dans cette formation. Grâce à ce TFE je peux mettre en avant mes acquis dans ce langage.

Le sujet de mon TFE est la gestion d’une base de données d’un établissement commercial ici un Hôtel. Pour ce faire je réalise une base de données qui contient 3 tables grâce à l’outil « DB Browser for SQLite », un programme python avec une interface réalisée grâce à Tkinter qui permettra que l’utilisateur puisse facilement gérer sa clientèle.

L’objective que je veux atteindre grâce à ce travail est la réalisation d’une application de A à Z. La recherche d’informations, l’apprentissage d’utilisation de nouveaux outils et l’autonomie de travail. Ces objectives je ne les ai pas choisis au hasard, en effet ceux-ci sont souvent cités dans les différentes offres d’emplois du secteur.

# Description du projet

Cette application permet la gestion d’un groupe hôtelier composer de plusieurs enseignes d’hôtel.

Elle est composée d’une page d’accueil avec trois boutons permettant d’accéder à la gestion soit des :

* Hôtels
* Villes
* Réservations

La gestion de ses trois fonctionnalités, permettent : d’ajouter, modifier, supprimer ou annuler un champ à la table de la base de données correspondante grâce à des requêtes SQL.

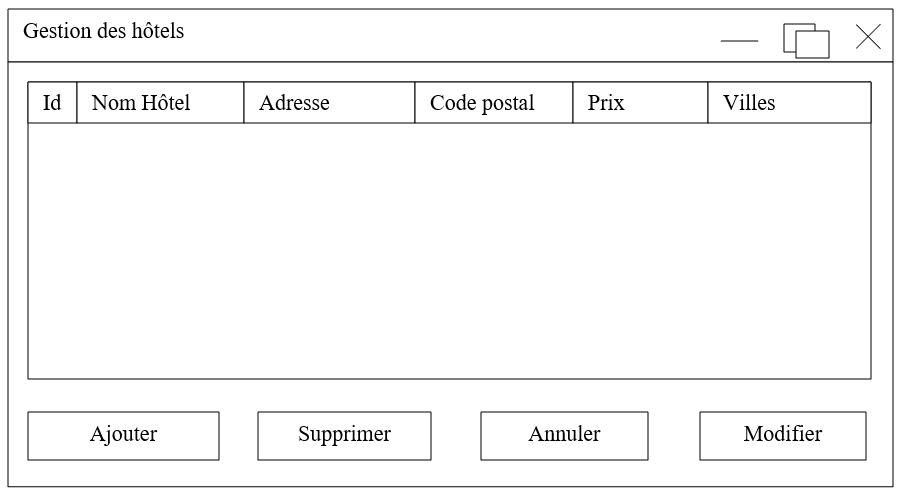
# Analyse

1. **Design préliminaire de l’application bureau**
2. Page d’accueil

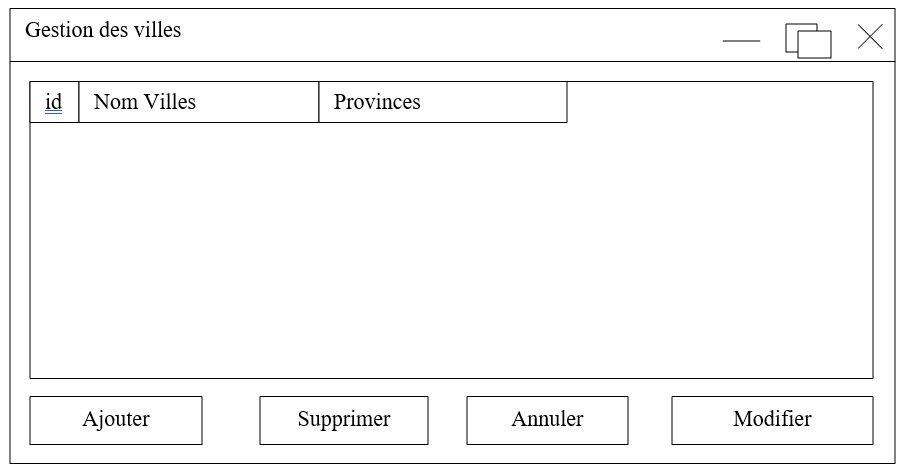
Une image contenant capture d’écran

Description générée automatiquement

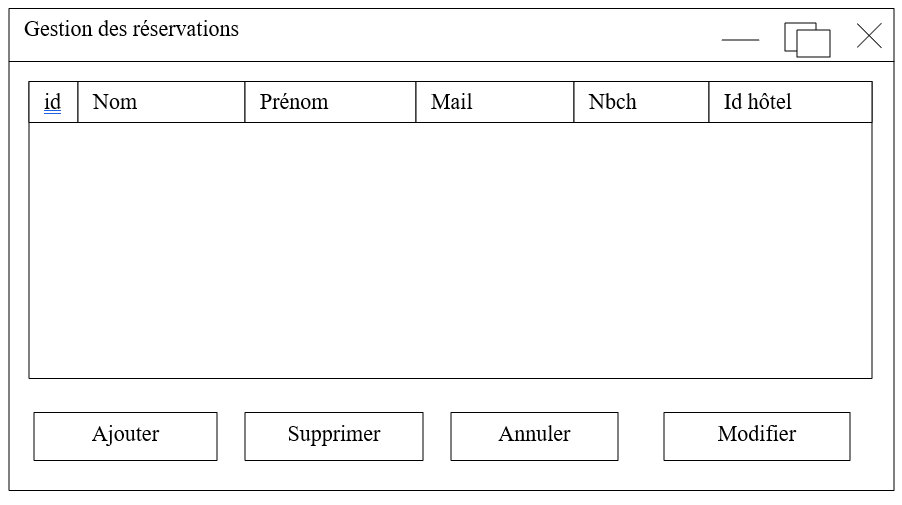
1. Page de gestion des Hôtels



1. Page de gestions des villes



1. Page de gestion des réservations



1. **Diagramme de la base de données**

Il s'agit d'une informatisation basique, de réservations de chambres d'hôtels. En résumé, pour les tables de référence (d'où partent les flèches symbolisant une dépendance référentielle)

* Hotel : chaque hôtel est identifié de manière unique (idhotel CP), possède un nom (nomhotel) et une adresse dans une ville (adresse, codepostal, idville CE) et propose des chambres à un prix unique (prixch)
* Reservation : chaque réservation est identifiée de manière unique (idreserv CP), et est effectuée par une personne (nom, prenom, mail) pour un certain nombre de chambres (nbch) dans un hôtel précis (idhotel CE).

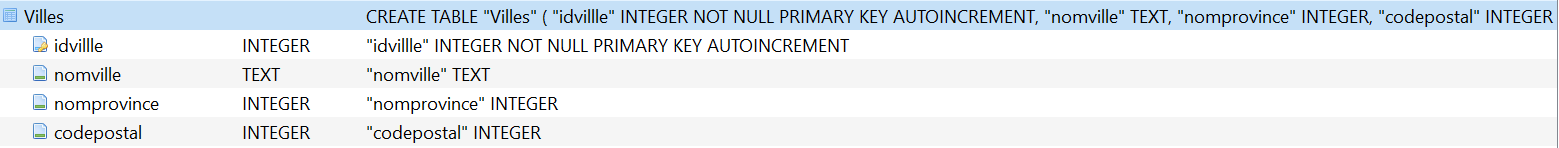
Une image contenant texte, moniteur, écran, assis

Description générée automatiquement

1. Structure des tables

Une image contenant capture d’écran

Description générée automatiquement



Une image contenant capture d’écran

Description générée automatiquement

# Réalisation

1. Page d’accueil

Une image contenant capture d’écran, moniteur

Description générée automatiquement

1. Page de gestion des hôtels

Une image contenant capture d’écran

Description générée automatiquement

1. Page de gestion des villes

Une image contenant capture d’écran

Description générée automatiquement

1. Page de gestion des réservations

Une image contenant capture d’écran

Description générée automatiquement

# Améliorations possibles

Une des améliorations qu’ont peux apporter est celle de l’extraction des différentes données et prix a payer par réservations. Pour cela j’ai fait une requête SQL :

SELECT idreserv ,nom ,prénom ,mail ,nbch (nombre de chambre réservé), idhotel, nomhotel,(Hotels.Prixch\*nbch) as payer FROM Info INNER JOIN Hotels ON Info.idhotel=Hotels.idhotel

Cela affiche :

Une image contenant moniteur, écran, assis, table

Description générée automatiquement

On voit grâce au résultat de cette requête qu’on pourrait crée un bouton permettant en python de crée un PDF afin de récupérer les différentes « Réservation ». Je n’ai pas ajouté cette fonctionnalité car malgré de nombreux essaie Tkinter n’a pas un package très ergonomique pour la liaison et l’extraction de données venant de plusieurs tables.

Une deuxième amélioration est la création d’une nouvelle table qu’ont pourrais compléter grâce à un formulaire. J’ai essayé d’implémenter celui-ci mais malheureusement les tables liées entre elles ne permette pas d’ajouter dans la table des données sans générer d’erreur d’encodage. Il faudrait pouvoir faire

# Conclusion

Ce projet est un pas de plus dans mon apprentissage. J’ai pu développer mes connaissances avec l’ajout progressif de chaque fonctionnalité, que j’ai apprise durant ma formation, de plus réaliser un projet python de A à Z est très satisfaisant et permet de mettre en forme l’ensemble des savoir appris.

Les difficultés rencontrées ont été : la fonctionnalité modification appliqué à la table Hotels. Le problème était la récupération de id à modifier qui était soi-disant non renseigné. J’ai à régler ce problème après plusieurs jours en me relisant et en retirant simplement deux parenthésés. Cela montre bien l’importances de la relecture de son code de plus à plusieurs reprises j’ai interchangé les def des hotels et celle des villes pour régler cela j’ai du donné des noms plus adapté et plus parlant à mes différentes def.

L’autre problème était de rendre l’application plus professionnelle j’ai dû faire beaucoup de recherche. Pour se faire je voulais avoir une image réaliste en fond d’écrans accueil, mais malheureusement à chaque fois que j’essayer de mettre une image plus réaliste celle-ci n’avait pas le bon format, elles étaient pour la plus par en jpg mais il n’y avait que les image png qui pouvais être utilisé. Finalement grâce à logiciel « paint » j’ai réussie à changer une image jpg en png.

Ce fut pour moi une grande réussite dans la mesure où j’ai su relever de nombreux défis, trouvé les bonnes informations aussi bien sur internet que dans mes cours, progresser assez vite et être plus minutieuse.

Cette épreuve intégrer fut donc une réussite et constitue pour moi le meilleur moyen de nous améliorer, de devenir autonome et de développer nos compétences aussi bien techniques que méthodologiques apprissent durant cette formation.

# Bibliographie

<https://www.sourcecodester.com/tutorials/python/11417/python-simple-inventory-system.html>

<https://python.doctor/page-tkinter-interface-graphique-python-tutoriel>

<https://docs.python.org/fr/3/library/tkinter.html>

<https://docs.python-guide.org/shipping/freezing/>

<https://helpdeskgeek.com/free-tools-review/4-tools-to-create-windows-installer-packages/>

<https://sql.sh/cours/insert-into>

# Annexes

1. from tkinter import \*  
   import tkinter.ttk as ttk  
   from tkinter.messagebox import \*  
   import sqlite3  
   from sqlite3 import Error  
     
     
   def connection(base):  
    global cur, db # attention global   
    try:  
    db = sqlite3.connect(base)  
    cur = db.cursor()  
    except Error as e:  
    print(e)  
    return None  
     
     
   def fenetre():  
    global fen  
    fen = Tk()  
    fen.title("Page d'Accueil")  
    fen.geometry("1000x600")  
    fen.configure(bg='#F0F0F0')  
     
    fen.image = PhotoImage(file='Image\hoteldelsol.png')  
     
    fen.canvas = Canvas(fen, width=1000, height=500, background='#F0F0F0', borderwidth=None)  
    fen.canvas.pack()  
    fen.canvas.create\_image(500, 200, image=fen.image)  
    fen.canvas.grid(row=0, columnspan=3)  
     
    Button(fen, text="Hotel", command=mainpagegestionhotel, font='Georgia', width=20, bg="lightgrey").grid(row=2, column=0)  
    Button(fen, text="Villes",command=mainpageville, font='Georgia', width=20, bg="lightgrey").grid(row=2, column=1)  
    Button(fen, text="Reservation",command=mainpagereserv, font='Georgia', width=20, bg="lightgrey").grid(row=2, column=2)  
     
   # --------------------------------------Hotel---------------------------------------------------------------------------  
     
     
   def mainpagegestionhotel():  
    global win, tables, cur  
    # création d'une fenêtre modal  
    win = Toplevel(fen)  
    win.title("Gestion des hôtels")  
     
     
    tables = ttk.Treeview(win)  
    tables.grid(row=0, columnspan=6, sticky=N + E + S + W, padx=10, pady=10) # remplit toute la zone de root  
     
    # Déclaration des colonnes  
    tables["columns"] = ("col1", "col2", "col3", "col4", "col5", "col6")  
     
    # taille des colonnes  
    tables.column("#1", width=30, minwidth=30)  
    tables.column("#2", width=200, minwidth=200)  
    tables.column("#3", width=200, minwidth=200)  
    tables.column("#4", width=300, minwidth=300)  
    tables.column("#5", width=300, minwidth=300)  
    tables.column("#5", width=300, minwidth=300)  
     
    # entête ( titre ) de la colonne  
    tables.heading("col1", text="id", anchor=W)  
    tables.heading("col2", text="Nom hotel", anchor=W)  
    tables.heading("col3", text="Adresse", anchor=W)  
    tables.heading("col4", text="Code postal", anchor=W)  
    tables.heading("col5", text="Prix", anchor=W)  
    tables.heading("col6", text="Villes", anchor=W)  
     
    # Paramètré la scrollbar  
    ascenseurY = ttk.Scrollbar(win, orient=VERTICAL, command=tables.yview)  
    # positionné la scrollbar  
    ascenseurY.grid(row=0, column=5, sticky='nse')  
     
    # configurer notre treeview  
    tables.config(show="headings", height=10, selectmode="browse",  
    yscrollcommand=ascenseurY.set) # height c'est le nombre de lignes a affiché  
    remplisagehotel()  
    # Boutons  
    Button(win, text="Ajouter", command=ajoutehotel).grid(row=6, column=1, padx=5, sticky=E + W)  
    Button(win, text="Modifier", command=modifh).grid(row=6, column=2, padx=5, sticky=E + W)  
    Button(win, text="Supprimer", command=supprimehotel).grid(row=6, column=3, padx=5, sticky=E + W)  
    Button(win, text="Annuler", command=win.destroy).grid(row=6, column=4, padx=5, sticky=E + W)  
     
     
   def ajoutehotel():  
    global win,tables, cur  
    # création d'une fenêtre modal  
    win = Toplevel(fen)  
    win.title("Gestion des hôtels")  
     
    # Déclaration des variables du formulaire  
    nomhotel = StringVar()  
    adresse = StringVar()  
    prixch = DoubleVar()  
    codepostal = IntVar()  
    idville = IntVar()  
     
    # etiquettes  
    Label(win, text="Nom Hotel").grid(row=1, column=0, pady=5, padx=5)  
    Label(win, text="Adresse").grid(row=2, column=0, pady=5, padx=5)  
    Label(win, text="Code posal").grid(row=3, column=0, pady=5, padx=5)  
    Label(win, text="Prix").grid(row=4, column=0, pady=5, padx=5)  
    Label(win, text="Villes").grid(row=5, column=0, pady=5, padx=5)  
     
    # Les entrys  
    Entry(win, textvariable=nomhotel, width=30).grid(row=1, column=1, columnspan=2, padx=5)  
    Entry(win, textvariable=adresse, width=30).grid(row=2, column=1, columnspan=2, padx=5)  
    Entry(win, textvariable=codepostal, width=30).grid(row=3, column=1, columnspan=2, padx=5)  
    Entry(win, textvariable=prixch, width=30).grid(row=4, column=1, columnspan=2, padx=5)  
    Entry(win, textvariable=idville, width=30).grid(row=5, column=1, columnspan=2, padx=5)  
     
    # Boutons  
    Button(win, text="Ajouter", command=lambda: ajoutdbhotel(nomhotel.get(), adresse.get(), codepostal.get(), prixch.get(),idville.get())).grid(row=6, column=1, padx=5, sticky=E + W)  
    Button(win, text="Annuler", command=win.destroy).grid(row=6, column=2, padx=5, pady=10, sticky=E + W)  
     
     
   def ajoutdbhotel(nomhotel, adresse, codepostal, prixch, idville): # sql autre fichier c'est mieux  
    global win, tables  
    win.destroy()  
    if nomhotel != "" and adresse != "":  
    query = '''  
    INSERT INTO Hotels (nomhotel, adresse, codepostal,prixch)  
    VALUES (?, ?, ?, ?)'''  
     
    cur.execute(query, (nomhotel, adresse, codepostal, prixch)) # re crée un cur avec la db  
    db.commit()  
    remplisagehotel()  
     
     
   def modifiehotel():  
    global win,tables, cur,winmh  
    # création d'une fenêtre modal  
    winmh = Toplevel(fen)  
    winmh.title("Gestion des hôtels")  
     
    # Déclaration des variables du formulaire  
    nomhotel=StringVar()  
    adresse = StringVar()  
    prixch = DoubleVar()  
    codepostal = IntVar()  
    idville = IntVar()  
     
    # etiquettes  
    Label(winmh, text="Nom hôtel").grid(row=1, column=0, pady=5, padx=5)  
    Label(winmh, text="Adresse").grid(row=2, column=0, pady=5, padx=5)  
    Label(winmh, text="Code posal").grid(row=3, column=0, pady=5, padx=5)  
    Label(winmh, text="Prix").grid(row=4, column=0, pady=5, padx=5)  
    Label(winmh, text="Villes").grid(row=5, column=0, pady=5, padx=5)  
     
    # Les entrys  
    Entry(winmh, textvariable=nomhotel, width=30).grid(row=1, column=1, columnspan=2, padx=5)  
    Entry(winmh, textvariable=adresse, width=30).grid(row=2, column=1, columnspan=2, padx=5)  
    Entry(winmh, textvariable=codepostal, width=30).grid(row=3, column=1, columnspan=2, padx=5)  
    Entry(winmh, textvariable=prixch, width=30).grid(row=4, column=1, columnspan=2, padx=5)  
    Entry(winmh, textvariable=idville, width=30).grid(row=5, column=1, columnspan=2, padx=5)  
     
    # Boutons  
    Button(winmh, text="Modifier", command=lambda: modifdbhotel(nomhotel.get(), adresse.get(), codepostal.get(), prixch.get())).grid(row=6, column=1, padx=5, sticky=E + W)  
    Button(winmh, text="Annuler", command=winmh.destroy).grid(row=6, column=2, padx=5, pady=10, sticky=E + W)  
     
     
   def modifh():  
    global winmh,idmodifierh,tables  
    if tables.focus():  
    nomhotel = tables.item(tables.focus())["values"][1]  
    idmodifierh = tables.item(tables.focus())["values"][0]  
    confirm = askokcancel("Ok ou annuler", "Modifier" + " " + nomhotel + " ?")  
    if confirm:  
    modifiehotel()  
    db.commit()  
    remplisagehotel()  
    else :  
    showinfo("Attention", "Veuillez sélectionner \nune ligne avant de \nmodifier")  
     
     
   def modifdbhotel(nomhotel, adresse, codepostal, prixch):  
    query='''  
    UPDATE Hotels SET nomhotel=?, adresse=?, codepostal=?, prixch=? WHERE idhotel=? '''  
    cur.execute(query, (nomhotel, adresse, codepostal, prixch, idmodifierh))  
    db.commit()  
    winmh.destroy()  
    remplisagehotel()  
     
     
   def supprimehotel():  
    global tables  
    if tables.focus():  
    nomhotel = tables.item(tables.focus())["values"][1]  
    adresse = tables.item(tables.focus())["values"][2]  
    idAsupprimer = tables.item(tables.focus())["values"][0]  
    confirm = askokcancel("Ok ou annuler", "Suppression de " + nomhotel + " " + adresse + " ?")  
    if confirm:  
    # supprimer le record  
    cur.execute("DELETE FROM Hotels WHERE idhotel = ?", (idAsupprimer,))  
    db.commit()  
    remplisagehotel()  
    else:  
    showinfo("Attention", "Veuillez sélectionner \nune ligne avant de \nsupprimer")  
     
     
   def remplisagehotel():  
    global tables, cur  
     
    # vider la liste  
    tables.delete(\*tables.get\_children())  
     
    # récupérer la liste de la db  
    cur.execute("SELECT \* FROM Hotels")  
    Hotels= cur.fetchall()  
     
    # Ajouter les etudiants dans le treeview  
    for Hotel in Hotels:  
    tables.insert('', 'end', values=(Hotel[0], Hotel[1], Hotel[2], Hotel[3], Hotel[4], Hotel[5]))  
     
   # --------------------------------------Villes--------------------------------------------------------------------------  
     
     
   def mainpageville():  
    global win, tables, cur  
     
    # création d'une fenêtre modal  
    winV = Toplevel(fen)  
    winV.title("Gestion des Villes")  
     
    tables = ttk.Treeview(winV)  
    tables.grid(row=0, columnspan=4, sticky=N + E + S + W, padx=10, pady=10) # remplit toute la zone de root  
     
    # Déclaration des colonnes  
    tables["columns"] = ("col1", "col2", "col3")  
     
    # taille des colonnes  
    tables.column("#1", width=30, minwidth=30)  
    tables.column("#2", width=200, minwidth=200)  
    tables.column("#3", width=200, minwidth=200)  
     
    # entête ( titre ) de la colonne  
    tables.heading("col1", text="idVille", anchor=W)  
    tables.heading("col2", text="Nom Ville", anchor=W)  
    tables.heading("col3", text="Nom Province", anchor=W)  
     
    # Paramètré la scrollbar  
    ascenseurY = ttk.Scrollbar(winV, orient=VERTICAL, command=tables.yview)  
    # positionné la scrollbar  
    ascenseurY.grid(row=0, column=4, sticky='nse')  
     
    # configurer notre treeview  
    tables.config(show="headings", height=10, selectmode="browse", yscrollcommand=ascenseurY.set) # height c'est le nombre de lignes a affiché  
    remplisagevilles()  
    # Boutons  
    Button(winV, text="Ajouter", command=ajouteville, width=3).grid(row=6, column=1, padx=2, sticky=E + W)  
    Button(winV, text="Modifier", command=modifv, width=3).grid(row=6, column=2, padx=2, sticky=E + W)  
    Button(winV, text="Supprimer", command=supprimeville, width=3).grid(row=6, column=3, padx=2, sticky=E + W)  
    Button(winV, text="Annuler", command=winV.destroy, width=15).grid(row=6, column=4, padx=2, sticky=E + W)  
     
     
   def ajouteville():  
    global winA,tables, cur  
    # création d'une fenêtre modal  
    winA = Toplevel(fen)  
    winA.title("Gestion des hôtels")  
     
    # Déclaration des variables du formulaire  
    idville = IntVar()  
    nomville = StringVar()  
    nomprovince = StringVar()  
     
    # etiquettes  
    Label(winA, text="Id Ville").grid(row=1, column=0, pady=5, padx=5)  
    Label(winA, text="Ville").grid(row=2, column=0, pady=5, padx=5)  
    Label(winA, text="Province").grid(row=3, column=0, pady=5, padx=5)  
     
    # Les entrys  
    Entry(winA, textvariable=idville, width=30).grid(row=1, column=1, columnspan=2, padx=5)  
    Entry(winA, textvariable=nomville, width=30).grid(row=2, column=1, columnspan=2, padx=5)  
    Entry(winA, textvariable=nomprovince, width=30).grid(row=3, column=1, columnspan=2, padx=5)  
     
    # Boutons  
    Button(winA, text="Ajouter", command=lambda: ajoutdbville(nomville.get(), nomprovince.get())).grid(row=4, column=1, padx=5, sticky=E + W)  
    Button(winA, text="Annuler", command=winA.destroy).grid(row=4, column=2, padx=5, pady=10, sticky=E + W)  
     
     
   def ajoutdbville(nomville, nomprovince): # sql autre fichier c'est mieux  
    global winA, tables  
    winA.destroy()  
    if nomville != "" and nomprovince != "":  
    query = '''  
    INSERT INTO Villes (nomville, nomprovince)  
    VALUES (?, ?)'''  
     
    cur.execute(query, (nomville, nomprovince)) # re crée un cur avec la db  
    db.commit()  
    remplisagevilles()  
     
     
   def modifieville():  
    global winM,tables, cur  
    # création d'une fenêtre modal  
    winM = Toplevel(fen)  
    winM.title("Gestion des hôtels")  
     
    # Déclaration des variables du formulaire  
    idville = IntVar()  
    nomville = StringVar()  
    nomprovince = StringVar()  
     
    # etiquettes  
    Label(winM, text="Id Ville").grid(row=1, column=0, pady=5, padx=5)  
    Label(winM, text="Ville").grid(row=2, column=0, pady=5, padx=5)  
    Label(winM, text="Province").grid(row=3, column=0, pady=5, padx=5)  
     
    # Les entrys  
    Entry(winM, textvariable=idville, width=30).grid(row=1, column=1, columnspan=2, padx=5)  
    Entry(winM, textvariable=nomville, width=30).grid(row=2, column=1, columnspan=2, padx=5)  
    Entry(winM, textvariable=nomprovince, width=30).grid(row=3, column=1, columnspan=2, padx=5)  
     
    # Boutons  
    Button(winM, text="Modifier", command=lambda: modifdbville(nomville.get(), nomprovince.get())).grid(row=4, column=1, padx=5, sticky=E + W)  
    Button(winM, text="Annuler", command=winM.destroy).grid(row=4, column=2, padx=5, pady=10, sticky=E + W)  
     
     
   def modifv():  
    global tables, fen, idAmodifierv  
    if (tables.focus()) :  
    nomville = tables.item(tables.focus())["values"][1]  
    idAmodifierv = tables.item(tables.focus())["values"][0]  
    confirm = askokcancel("Ok ou annuler", "Modifier" + " " + nomville + " ?")  
    if confirm:  
    modifieville()  
    db.commit()  
    remplisagevilles()  
    else :  
    showinfo("Attention", "Veuillez sélectionner \nune ligne avant de \nmodifier")  
     
     
   def modifdbville(nomville, nomprovince):  
    global winM  
    winM.destroy()  
    if nomville != "" or nomprovince !="":  
    query='''  
    UPDATE Villes SET nomville=?, nomprovince=? WHERE idvillle=? '''  
    cur.execute(query, (nomville, nomprovince, idAmodifierv))  
    db.commit()  
    remplisagevilles()  
     
     
   def supprimeville():  
    global tables  
    if tables.focus():  
    nomville = tables.item(tables.focus())["values"][1]  
    nomprovince = tables.item(tables.focus())["values"][2]  
    idAsupprimer = tables.item(tables.focus())["values"][0]  
    confirm = askokcancel("Ok ou annuler", "Suppression de " + nomville + " " + nomprovince + " ?")  
    if confirm:  
    # supprimer le record  
    cur.execute("DELETE FROM Villes WHERE idvillle = ?", (idAsupprimer,))  
    db.commit()  
    remplisagevilles()  
    else:  
    showinfo("Attention", "Veuillez sélectionner \nune ligne avant de \nsupprimer")  
     
     
   def remplisagevilles():  
    global tables, cur  
     
    # vider la liste  
    tables.delete(\*tables.get\_children())  
     
    # récupérer la liste de la db  
    cur.execute("SELECT \* FROM Villes")  
    Villes= cur.fetchall()  
     
    # Ajouter les etudiants dans le treeview  
    for Ville in Villes:  
    tables.insert('', 'end', values=(Ville[0], Ville[1], Ville[2]))  
     
     
   # --------------------------------------Reservation--------------------------------------------------------------------------  
   def mainpagereserv():  
    global winr, tables, cur  
     
    # création d'une fenêtre modal  
    winr = Toplevel(fen)  
    winr.title("Gestion des reservations")  
     
    tables = ttk.Treeview(winr)  
    tables.grid(row=0, columnspan=6, sticky=N + E + S + W, padx=10, pady=10) # remplit toute la zone de root  
     
    # Déclaration des colonnes  
    tables["columns"] = ("col1", "col2", "col3","col4", "col5", "col6")  
     
    # taille des colonnes  
    tables.column("#1", width=30, minwidth=30)  
    tables.column("#2", width=100, minwidth=100)  
    tables.column("#3", width=100, minwidth=100)  
    tables.column("#4", width=150, minwidth=150)  
    tables.column("#5", width=80, minwidth=80)  
    tables.column("#6", width=80, minwidth=80)  
     
    # entête ( titre ) de la colonne  
    tables.heading("col1", text="Idreserv", anchor=W)  
    tables.heading("col2", text="Nom ", anchor=W)  
    tables.heading("col3", text="Prénom", anchor=W)  
    tables.heading("col4", text="Mail", anchor=W)  
    tables.heading("col5", text="Nombre de chambre ", anchor=W)  
    tables.heading("col6", text="Id Hotel", anchor=W)  
     
    # Paramètré la scrollbar  
    ascenseurY = ttk.Scrollbar(winr, orient=VERTICAL, command=tables.yview)  
    # positionné la scrollbar  
    ascenseurY.grid(row=0, column=6, sticky='nse')  
     
    # configurer notre treeview  
    tables.config(show="headings", height=10, selectmode="browse", yscrollcommand=ascenseurY.set) # height c'est le nombre de lignes a affiché  
    remplisagereserv()  
    # Boutons  
    Button(winr, text="Ajouter",command=ajoutereserv, width=3).grid(row=6, column=1, padx=2, sticky=E + W)  
    Button(winr, text="Modifier",command=modifr, width=3).grid(row=6, column=2, padx=2, sticky=E + W)  
    Button(winr, text="Supprimer",command=supprimereserv, width=3).grid(row=6, column=3, padx=2, sticky=E + W)  
    Button(winr, text="Annuler", command=winr.destroy, width=3).grid(row=6, column=4, padx=2, sticky=E + W)  
     
     
   def ajoutereserv():  
    global winAr,tables, cur  
    # création d'une fenêtre modal  
    winAr = Toplevel(fen)  
    winAr.title("Gestion des hôtels")  
     
    # Déclaration des variables du formulaire  
    idreserv = IntVar()  
    nom = StringVar()  
    prenom = StringVar()  
    mail = StringVar()  
    nbch = IntVar()  
    idhotel = IntVar()  
     
    # etiquettes  
    Label(winAr, text="Id Reservation").grid(row=1, column=0, pady=5, padx=5)  
    Label(winAr, text="Nom").grid(row=2, column=0, pady=5, padx=5)  
    Label(winAr, text="Prénom").grid(row=3, column=0, pady=5, padx=5)  
    Label(winAr, text="Mail").grid(row=4, column=0, pady=5, padx=5)  
    Label(winAr, text="Nombre de chambre").grid(row=5, column=0, pady=5, padx=5)  
    Label(winAr, text="Id hotel").grid(row=6, column=0, pady=5, padx=5)  
     
    # Les entrys  
    Entry(winAr, textvariable=idreserv, width=30).grid(row=1, column=1, columnspan=2, padx=5)  
    Entry(winAr, textvariable=nom, width=30).grid(row=2, column=1, columnspan=2, padx=5)  
    Entry(winAr, textvariable=prenom, width=30).grid(row=3, column=1, columnspan=2, padx=5)  
    Entry(winAr, textvariable=mail, width=30).grid(row=4, column=1, columnspan=2, padx=5)  
    Entry(winAr, textvariable=nbch, width=30).grid(row=5, column=1, columnspan=2, padx=5)  
    Entry(winAr, textvariable=idhotel, width=30).grid(row=6, column=1, columnspan=2, padx=5)  
     
    # Boutons  
    Button(winAr, text="Ajouter", command=lambda: ajoutdbreserv(idreserv.get(),nom.get(), prenom.get(),mail.get(),nbch.get(),idhotel.get())).grid(row=7, column=1, padx=5, sticky=E + W)  
    Button(winAr, text="Annuler", command=winAr.destroy).grid(row=7, column=2, padx=5, pady=10, sticky=E + W)  
     
     
   def ajoutdbreserv(idreserv, nom,prenom,mail,nbch,idhotel): # sql autre fichier c'est mieux  
    global winAr, tables  
    winAr.destroy()  
    if nom != "" and prenom != "":  
    query = '''  
    INSERT INTO Reservation (idreserv, nom,prenom,mail,nbch,idhotel)  
    VALUES (?, ?,?,?,?,?)'''  
     
    cur.execute(query, (idreserv,nom, prenom,mail,nbch,idhotel)) # re crée un cur avec la db  
    db.commit()  
    remplisagereserv()  
     
     
   def modifreserv():  
    global winMr,tables, cur  
    # création d'une fenêtre modal  
    winMr = Toplevel(fen)  
    winMr.title("Gestion des hôtels")  
     
    # Déclaration des variables du formulaire  
    idreserv = IntVar()  
    nom = StringVar()  
    prenom = StringVar()  
    mail = StringVar()  
    nbch = IntVar()  
    idhotel = IntVar()  
     
    # etiquettes  
    Label(winMr, text="Id Reservation").grid(row=1, column=0, pady=5, padx=5)  
    Label(winMr, text="Nom").grid(row=2, column=0, pady=5, padx=5)  
    Label(winMr, text="Prénom").grid(row=3, column=0, pady=5, padx=5)  
    Label(winMr, text="Mail").grid(row=4, column=0, pady=5, padx=5)  
    Label(winMr, text="Nombre de chambre").grid(row=5, column=0, pady=5, padx=5)  
    Label(winMr, text="Id hotel").grid(row=6, column=0, pady=5, padx=5)  
     
    # Les entrys  
    Entry(winMr, textvariable=idreserv, width=30).grid(row=1, column=1, columnspan=2, padx=5)  
    Entry(winMr, textvariable=nom, width=30).grid(row=2, column=1, columnspan=2, padx=5)  
    Entry(winMr, textvariable=prenom, width=30).grid(row=3, column=1, columnspan=2, padx=5)  
    Entry(winMr, textvariable=mail, width=30).grid(row=4, column=1, columnspan=2, padx=5)  
    Entry(winMr, textvariable=nbch, width=30).grid(row=5, column=1, columnspan=2, padx=5)  
    Entry(winMr, textvariable=idhotel, width=30).grid(row=6, column=1, columnspan=2, padx=5)  
     
    # Boutons  
    Button(winMr, text="Modifier", command=lambda: modifdbreserv(idreserv.get(),nom.get(), prenom.get(),mail.get(),nbch.get(),idhotel.get())).grid(row=7, column=1, padx=5, sticky=E + W)  
    Button(winMr, text="Annuler", command=winMr.destroy).grid(row=7, column=2, padx=5, pady=10, sticky=E + W)  
     
     
   def modifr():  
    global tables, fen, idAmodifierr  
    if (tables.focus()) :  
    nom = tables.item(tables.focus())["values"][1]  
    idAmodifierr = tables.item(tables.focus())["values"][0]  
    confirm = askokcancel("Ok ou annuler", "Modifier" + " " + nom + " ?")  
    if confirm:  
    modifreserv()  
    db.commit()  
    remplisagereserv()  
    else :  
    showinfo("Attention", "Veuillez sélectionner \nune ligne avant de \nmodifier")  
     
     
   def modifdbreserv(idreserv,nom, prenom,mail,nbch,idhotel):  
    global winMr  
    winMr.destroy()  
    if nom != "" or prenom !="":  
    query='''  
    UPDATE Reservation SET idreserv=?, nom=?, prenom=?,mail=?,nbch=?,idhotel=? WHERE idreserv=? '''  
    cur.execute(query, (idreserv, nom,prenom,mail,nbch,idhotel,idAmodifierr))  
    db.commit()  
    remplisagevilles()  
     
     
   def supprimereserv():  
    global tables  
    if tables.focus():  
    nom = tables.item(tables.focus())["values"][1]  
    prenom = tables.item(tables.focus())["values"][2]  
    idAsupprimersr = tables.item(tables.focus())["values"][0]  
    confirm = askokcancel("Ok ou annuler", "Suppression de " + nom + " " + prenom + " ?")  
    if confirm:  
    # supprimer le record  
    cur.execute("DELETE FROM Reservation WHERE idreserv = ?", (idAsupprimersr,))  
    db.commit()  
    remplisagereserv()  
    else:  
    showinfo("Attention", "Veuillez sélectionner \nune ligne avant de \nsupprimer")  
     
     
     
   def remplisagereserv():  
    global tables, cur  
     
    # vider la liste  
    tables.delete(\*tables.get\_children())  
     
    # récupérer la liste de la db  
    cur.execute("SELECT \* FROM Reservation")  
    Reservations= cur.fetchall()  
     
    # Ajouter les etudiants dans le treeview  
    for reservation in Reservations:  
    tables.insert('', 'end', values=(reservation[0], reservation[1], reservation[2],reservation[3], reservation[4], reservation[5]))  
     
     
   if \_\_name\_\_ == '\_\_main\_\_':  
    connection("database\Hotel.db")  
    fenetre()  
    mainloop()