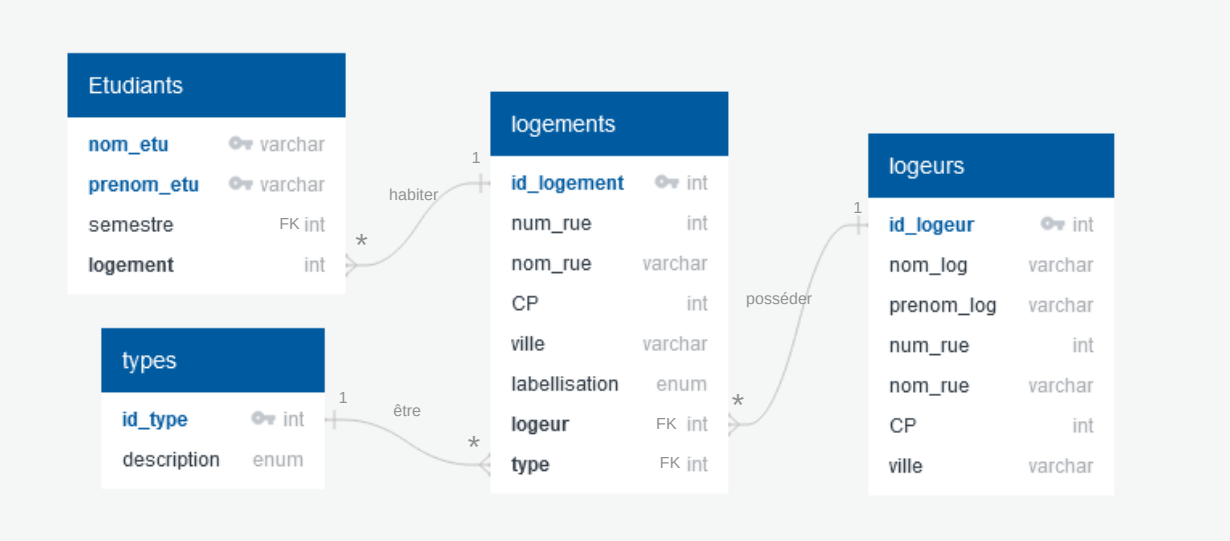
Bauvais Camille, Creuze Martin

TP6 : Base de données

Modèle relationnel de la base de données :



Code :

```
import sqlite3
from tkinter import *

class Fenetre(Tk):
    def __init__(self):
        Tk.__init__(self)
        self.frame = Frame(self)
        self.frame.grid(row=4, column=1, columnspan=4, sticky='w')
        self.nom_log = StringVar()
        self.prenom_log = StringVar()
        self.create()



    def create(self):
        Label(self, text="Nom du logeur : ").grid(row=1, column=1,
sticky='w')
        Entry(self, textvariable=self.nom_log).grid(row=1, column=2)
        Label(self, text="Prénom du logeur : ").grid(row=2, column=1,
sticky='w')
        Entry(self, textvariable=self.prenom_log).grid(row=2, column=2)
        Button(self, text="Valider", command=self.valider).grid(row=3,
column=1, padx=10, pady=10)
        Button(self, text="Réinitialiser", command=self.reinit).grid(row=3,
column=2, padx=10, pady=10)
        Button(self, text="Quitter", command=self.destroy).grid(row=3,
column=3, padx=10, pady=10)
```

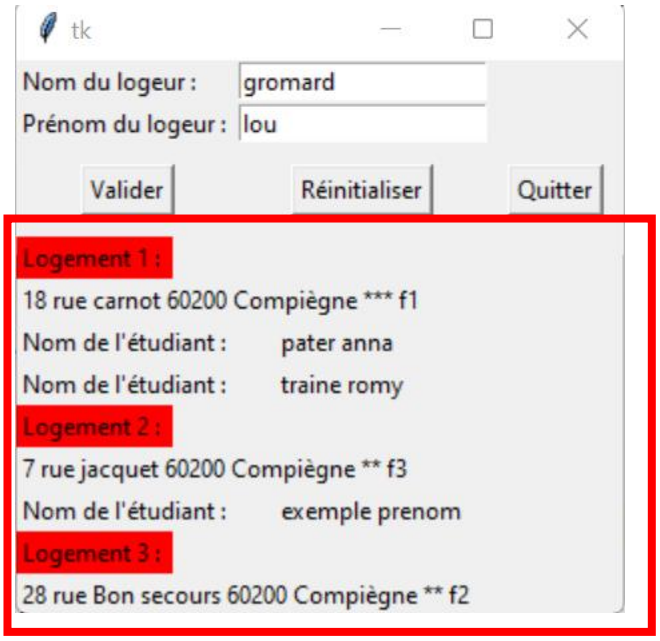
```
    def valider(self):
        self.frame.destroy()
        self.frame = Frame()
        self.frame.grid(row=4, column=1, columnspan=4, sticky='w')
        nom = self.nom_log.get()
        prenom = self.prenom_log.get()
        fichier_bdd = "TP6.sqlite"
        connexion = sqlite3.connect(fichier_bdd)
        curseur = connexion.cursor()
        curseur.execute(f"select logements.num_rue, logements.nom_rue,
logements.cp, logements.ville, logements.labellisation, types.description,
nom_etu, prenom_etu from logements inner join logeurs l on logements.logeur
= l.id_logeur inner join types on types.id_type = logements.type left join
etudiants on etudiants.logement = logements.id_logement where nom_log=? and
prenom_log=?",(nom, prenom))
        i = 1
        num_rue = 0
        nom_rue = ''
        cp = 0
        ville = ''
        ligne=1

        for elem in curseur.fetchall():
            if (num_rue != elem[0] or nom_rue != elem[1] or cp != elem[2]
or ville != elem[3]):
                num_rue = elem[0]
                nom_rue = elem[1]
                cp = elem[2]
                ville = elem[3]
                Label(self.frame, text=f"Logement {i} : ",
bg='red').grid(row=ligne, column=1, columnspan=3, sticky='w')
                ligne +=1
                etoile = ''
                for j in range(elem[4]):
                    etoile = etoile + '*'
                Label(self.frame, text=f"{num_rue} rue {nom_rue} {cp}
{ville} {etoile} {elem[5]}").grid(row=ligne, column=1, columnspan=3,
sticky='w')
                ligne +=1
                i += 1
            if (elem[6] and elem[7]):
                Label(self.frame, text=f"Nom de l'étudiant :
{elem[6]} {elem[7]}").grid(row=ligne, column=1, columnspan=3, sticky='w')
                ligne+=1
        connexion.close()

    def reinit(self):
        self.nom_log.set("")
        self.prenom_log.set("")
        self.frame.destroy()


if __name__ == '__main__':
    f = Fenetre()
    f.mainloop()
```

Pour la requête, nous réalisons un left join afin de récupérer le logement du logeur même s'il n'est habité par aucun étudiant.

Méthode utilisée pour l'affichage « résultat » et pour la réinitialisation :

Nous avons décidé de mettre en place un frame pour la partie contenant le résultat après avoir cliqué sur valider (voir affichage). Ainsi, lorsqu'on a besoin de réinitialiser la page, il suffit de détruire le frame, et de mettre à vide les champs pour le nom et le prénom. De plus, on détruit le frame à chaque fois que la fonction valider s'exécute pour que, si le nouveau propriétaire possède moins de logements que le premier, les dernières lignes de l'ancien propriétaire s'effacent.

Affichage :



Frame