**ПРИЛОЖЕНИЕ Г. ТЕКСТ ПРОГРАММЫ**

host = "127.0.0.1"  
user = "postgres"  
password = "Sasaha228"  
db\_name = "kurs"  
  
list\_typegsm = ['Код ГСМ', 'Название ГСМ', 'Единица измерения','Марка ГСМ']  
list\_vendorgsm=['Код поставщика', 'Название производителя', 'Адрес производителя', 'Код ГСМ']  
list\_companydrivers=['Табельный номер\n водителя', 'ФИО водителя', 'Государственный но\nмер прикрепленного авто', 'Дата приема\n на работу', '\tДата выдачи\n водительсокого удостоверения', '\tДата действия\n водительсокого удостоверения', 'Номер водительского\n удостоверения', 'Категория водительского\n удостоверения']  
list\_comptechnmeans=['НомГосРегистр', 'Марка авто', 'Номер кузова', 'ЕдИмз', 'Грузоподъёмность', 'Год выпусмка', 'Первичная стоимость', 'Код %', 'Остаточная стоимость']  
list\_deliverycontract=['Номер договора','Дата заключения\n договора','Код ГСМ','Код поставщика','Единица\n измерения','Цена','Количество','Стоимость','Ставка НДС','Сумма НДС','Сумма с НДС']  
list\_naryad=['Номер наряда','Дата составления','Табельный номер\nводителя','ФИО водителя','Государственный номер\nавто','Номер водительского\n удостоверения','Производимая работа']  
list\_pl=['Номер путевого листа','Дата составления ПЛ','Код ГСМ','Табельный номер водителя','Номер водительского\nудостоверения','Государственный номер авто','Пробег, км','Единица измерения','Остаток ГСМ','Объем полученных ГСМ','Объем потраченных ГСМ']  
list\_prihod=['Номер документа','Дата составленеия','Номер ТТН','Дата ТТН','Код ГСМ','Код поставщика','Единица измерения','Количество']  
list\_rashod=['Номер документа','Дата составления','Номер ПЛ','Дата ПЛ','Код ГСМ','Единица измерения','Количество']  
list\_ksu=['Номер документа','Дата составления\nКСУ','Номер склада','Номер цистерны','Код ГСМ','Код поставщика','Единица измерения','Стоимость единицы','Остаток на начало\nпериода','Количество прихода','Количество расхода','Остаток на конец\nпериода']  
list\_ttn=['Номер документа','Дата ТТН','Код ГСМ','Код поставщика','Единица измерения','Количество']  
  
  
slist\_typegsm = ['КодГСМ','НазвГСМ','ЕдИзм','МаркГСМ']  
slist\_vendorgsm = ['КодПост','НазвПроизв','АдрПроизв','КодГСМ']  
slist\_companydrivers = ['ТабНомВод','ФИОвод','ГосНомПрикрАвто','ДатПриемНаРаб','ДатВыдВодУдост','ДатДействВодУдост','НомВодУдост','КатВодУдост']  
slist\_comptechnmeans = ['НомГосРег', 'МаркАвт', 'НомКуз', 'ЕдИзм', 'Грузоп', 'ГодВып', 'ПервСтоим', 'Код %', 'ОстаСтоим']  
slist\_deliverycontract = ['НомДог','ДатЗаклДог','КодГСМ','КодПост','ЕдИзм','Цена','Кол','Стоим','СтавкНДС','СуммНДС','Сумм c НДС']  
slist\_naryad = ['НомНар','ДатСост','ТабНомВод','ФИОвод','ГосНомАвто','НомВодУдост','ПроизвРаб']  
slist\_pl = ['НомПутЛист','ДатаСостПЛ','КодГСМ','ТабНомВод','НомВодУдост','ГосНомАвто','Пробег, км','ЕдИзм','ОстГСМ','ОбПолГСМ','ОбПотрГСМ']  
slist\_prihod = ['НомДок','ДатСост','НомТТН','ДатТТН','КодГСМ','КодПост','ЕдИзм','Кол']  
slist\_rashod = ['НомДок','ДатCост','НомПЛ','ДатПЛ','КодГСМ','ЕдИзм','Кол']  
slist\_ksu = ['НомДок','ДатСостКСУ','НомСкл','НомЦист','КодГСМ','КодПост','ЕдИзм','CтоимЕд','ОстНаНачПер','КолПрих','КолРасх','ОстаНаКонПер']  
slist\_ttn = ['НомДок','ДатТТН','КодГСМ','КодПост','ЕдИзм','Кол']  
  
searhListINT = ["Цена", "Кол", "Стоим",'ОбПолГСМ',  
 "СтавкНДС", "Сумма НДС", "Сумма с НДС",'СуммНДС',  
 "CтоимЕд", "ОстНаНачПер", "КолПрих", "КолРасх",  
 "ОстаНаКонПер", "Грузоп", "ГодВып", "ПервСтоим", "Код %", "ОстаСтоим","Пробег, км","ОстГСМ","ОбПотрГСМ"]  
searhSQLListINT = {  
 ('ttn','Кол'):'amount',  
 ('rashfile','Кол'):'amount',  
 ('prihfile', 'Кол'):'amount',  
 ('ksu','СтоимЕд'):'ed\_price',  
 ('ksu','ОстНаНачПер'):'start\_bal',  
 ('ksu','КолПрих'):'amount\_prih\_gsm',  
 ('ksu','КолРасх'):'amount\_rash\_gsm',  
 ('ksu','ОстаНаКонПер'):'end\_bal',  
 ('deliverycontract','Цена'):'price',  
 ('deliverycontract','Кол'):'amount',  
 ('deliverycontract','Стоим'):'stoim',  
 ('deliverycontract','СтавкНДС'):'rate\_nds',  
 ('deliverycontract','СуммНДС'):'price\_nds',  
 ('deliverycontract','Сумм с НДС'):'price\_of\_nds',  
 ('comptechnmeans','Грузоп'):'load\_capacity',  
 ('comptechnmeans','ГодВып'):'year\_of\_product',  
 ('comptechnmeans','ПервСтоим'):'first\_cost',  
 ('comptechnmeans','Код %'):'kod\_porc',  
 ('comptechnmeans','ОстаСтоим'):'last\_cost',  
 ('pl','ОбПолГСМ'):'amount',  
 ('pl','Пробег, км'):'probeg',  
 ('pl','ОстГСМ'):'ostatok',  
 ('pl','ОбПотрГСМ'):'potr'  
}  
  
searhList = ['КодГСМ','НазвГСМ','ЕдИзм','МаркГСМ',  
 'КодПост','НазвПроизв','АдрПроизв',  
 'ТабНомВод','ФИОвод','ГосНомПрикрАвто',  
 'ДатПриемНаРаб','ДатВыдВодУдост','ДатДействВодУдост',  
 'НомВодУдост','КатВодУдост','НомГосРег','МаркАвт',  
 'НомКуз','НомДог','ДатЗаклДог','НомНар','НомПутЛист',  
 'ДатаСостПЛ','НомДок','ДатСост','НомТТН','ДатТТН','НомПЛ',  
 'ДатПЛ','НомСкл','НомЦист','ДатСостКСУ','ПроизвРаб']  
searhSQLList = {  
 ('vendorgsm','КодПост'):'code\_post',  
 ('vendorgsm','НазвПроизв'):'name\_proizv',  
 ('vendorgsm','АдрПроизв'):'addres\_proizv',  
 ('vendorgsm','КодГСМ'):'code\_gsm',  
 ('typegsm','КодГСМ'):'code\_gsm',  
 ('typegsm','НазвГСМ'):'name\_gsm',  
 ('typegsm','ЕдИзм'):'unit',  
 ('typegsm','МаркГСМ'):'mark\_gsm',  
 ('companydrivers','ТабНомВод'):'tab\_number',  
 ('companydrivers','ФИОвод'):'drivers\_name',  
 ('companydrivers','ГосНомПрикрАвто'):'national\_avto\_num',  
 ('companydrivers','ДатПриемНаРаб'):'date\_of\_hire',  
 ('companydrivers','ДатВыдВодУдост'):'date\_driverlicens',  
 ('companydrivers','ДатДействВодУдост'):'validity\_day\_drlic',  
 ('companydrivers','НомВодУдост'):'num\_drivlicens',  
 ('companydrivers','КатВодУдост'):'category\_drivlicens',  
 ('comptechnmeans','НомГосРег'):'national\_avto\_num',  
 ('comptechnmeans','МаркАвт'):'auto\_mark',  
 ('comptechnmeans','НомКуз'):'body\_number',  
 ('comptechnmeans','ЕдИзм'):'untill',  
 ('deliverycontract','НомДог'):'contract\_number',  
 ('deliverycontract','ДатЗаклДог'):'date\_contract',  
 ('deliverycontract','КодГСМ'):'code\_gsm',  
 ('deliverycontract','КодПост'):'code\_post',  
 ('naryad','НомНар'):'nar\_number',  
 ('naryad','ТабНомВод'):'tab\_number',  
 ('naryad','ФИОвод'):'drivers\_name',  
 ('naryad','ГосНомАвто'):'national\_avto\_num',  
 ('naryad','НомВодУдост'):'num\_drivlicens',  
 ('naryad','ПроизвРаб'):'rabot',  
 ('naryad','ДатСост'):'date\_nar',  
 ('pl','НомПутЛист'):'pl\_number',  
 ('pl','ДатаСостПЛ'):'date\_pl',  
 ('pl','ТабНомВод'):'tab\_number',  
 ('pl','НомВодУдост'):'num\_drivlicens',  
 ('pl','ГосНомАвто'):'national\_avto\_num',  
 ('pl','ЕдИзм'):'unit',  
 ('pl','КодГСМ'):'сode\_gsm',  
 ('prihfile','НомДок'):'doc\_number\_prih',  
 ('prihfile','ДатСост'):'date\_sost',  
 ('prihfile','НомТТН'):'ttn\_number',  
 ('prihfile','ДатТТН'):'date\_zakl\_ttn',  
 ('prihfile','КодГСМ'):'code\_gsm',  
 ('prihfile','КодПост'):'code\_post',  
 ('prihfile','ЕдИзм'):'untill',  
 ('rashfile','НомДок'):'prih\_number',  
 ('rashfile','НомПЛ'):'pl\_number',  
 ('rashfile','ДатПЛ'):'date\_pl',  
 ('rashfile','ДатСост'):'date\_sost',  
 ('rashfile','КодГСМ'):'code\_gsm',  
 ('rashfile','ЕдИзм'):'unit',  
 ('ksu','НомДок'):'ksu\_doc',  
 ('ksu','КодГСМ'):'code\_gsm',  
 ('ksu','КодПост'):'code\_post',  
 ('ksu','НомСкл'):'sklad\_number',  
 ('ksu','НомЦист'):'tanker\_number',  
 ('ksu','ДатСостКСУ'):'date\_ksu',  
 ('ksu','ЕдИзм'):'unit',  
 ('ttn','НомДок'):'ttn\_number',  
 ('ttn','ДатТТН'):'date\_zakl\_ttn',  
 ('ttn','КодГСМ'):'code\_gsm',  
 ('ttn','КодПост'):'code\_post',  
 ('ttn','ЕдИзм'):'untill'  
}  
  
searhComboboxList = {  
 'typegsm':slist\_typegsm,  
 'vendorgsm':slist\_vendorgsm,  
 'companydrivers':slist\_companydrivers,  
 'deliverycontract':slist\_deliverycontract,  
 'naryad':slist\_naryad,  
 'pl':slist\_pl,  
 'prihfile':slist\_prihod,  
 'rashfile':slist\_rashod,  
 'ksu':slist\_ksu,  
 'ttn':slist\_ttn  
}  
  
searhComboboxList1 = {  
 'typegsm':list\_typegsm,  
 'vendorgsm':list\_vendorgsm,  
 'companydrivers':list\_companydrivers,  
 'deliverycontract':list\_deliverycontract,  
 'naryad':list\_naryad,  
 'pl':list\_pl,  
 'prihfile':list\_prihod,  
 'rashfile':list\_rashod,  
 'ksu':list\_ksu,  
 'ttn':list\_ttn  
}  
  
list\_otch1 = [[],[]]

import tkinter as tk  
from tkinter import \*  
from tkinter import ttk  
from PIL import Image, ImageTk  
import psycopg2  
from config import \*  
from functools import partial  
import time  
import subprocess  
import os  
  
  
conn = psycopg2.connect(  
 host = host,  
 user = user,  
 password = password,  
 database = db\_name,  
 port = "5432"  
 )  
conn.autocommit = True  
  
class progrload(tk.Frame):  
 def \_\_init\_\_(self, win):  
 super().\_\_init\_\_(win)  
 self.progbar()  
  
 def progbar(self):  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 200  
 h = (h // 2) - 200  
  
 win.title('АРМ')  
 win.geometry('500x400+{}+{}'.format(w, h))  
 win.resizable(False, False)  
 self.frame = tk.Frame(win, bg= "#4d4f4c")  
 self.frame.place(relwidth=1, relheight=1)  
  
 value\_var = IntVar()  
 value = 10  
  
 img = Image.open("logo2.png")  
 self.tkimage = ImageTk.PhotoImage(img)  
 self.l3 = tk.Label(self.frame, image=self.tkimage, bg="#4d4f4c")  
 self.l3.pack(expand=1)  
  
 self.progressbar = ttk.Progressbar(orient="horizontal", variable=value\_var, maximum=100)  
 self.progressbar.pack(side = tk.BOTTOM, fill = tk.X)  
  
 self.label = ttk.Label(self.frame, textvariable=value\_var)  
 self.progressbar.start()  
  
 while True:  
 self.frame.update()  
 if value\_var.get() == 80:  
 self.progressbar.stop()  
 loginSystem(win)  
 break  
  
class loginSystem(tk.Frame):  
  
 def \_\_init\_\_(self,logWin):  
 super().\_\_init\_\_(logWin)  
 self.loginSystem()  
  
 def show\_password(self):  
 self.inputPassword.config(show="")  
  
 def hide\_password(self):  
 self.inputPassword.config(show="\*")  
  
 def loginSystem(self):  
  
 #методы возвращают размеры экрана, на котором запущено окно  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w=(w//2)-200  
 h=(h//2)-200  
  
 win.title('Авторизация')  
 win.geometry('400x150+{}+{}'.format(w, h))  
 win.resizable(False,False)  
  
 self.frame = tk.Frame(win)  
 self.frame.place(relwidth=1, relheight=1)  
  
 self.lab\_Login = tk.Label(self.frame, text = "Логин", font = 10)  
 self.lab\_Login.place(x=40,y=15)  
  
 self.lab\_Password = tk.Label(self.frame,text="Пароль",font = 15)  
 self.lab\_Password .place(x=40,y=45)  
  
 self.inputLogin = ttk.Entry(self.frame, width=15)  
 self.inputLogin.place(x=130,y=15)  
  
 self.inputPassword = ttk.Entry(self.frame, width=15, show = "\*")  
 self.inputPassword.place(x=130,y=45)  
  
 self.show\_button = tk.Button(self.frame, text="Показать пароль",command=self.show\_password)  
 self.show\_button.place(x=40,y=78)  
  
 self.hide\_button = tk.Button(self.frame, text="Скрыть пароль", fg="black",width=14, font=('',12), command=self.hide\_password)  
 self.hide\_button.place(x=220,y=78)  
  
 self.connButton = tk.Button(self.frame, text="Войти",fg="black",width=10, font=('',12),command=self.checkLogin)  
 self.connButton.pack(side=tk.BOTTOM, pady = 10)  
  
 def checkLogin(self):  
 global Polzovatel  
 Polzovatel=self.inputLogin.get()  
 if Polzovatel == "admin" and self.inputPassword.get() == "admin":  
 self.destroy()  
 self.frame.destroy()  
 mainProgramm(win)  
 else:  
  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 200  
 h = (h // 2) - 200  
 errorWindow = tk.Toplevel(self)  
 errorWindow.title("Ошибка входа")  
 errorWindow.geometry('300x150+{}+{}'.format(w, h))  
 errorWindow.resizable(False, False)  
  
 self.errorWindowFrame= tk.Frame(errorWindow)  
 self.errorWindowFrame.place(relwidth=1,relheight=1)  
  
 self.errorLabel = tk.Label(self.errorWindowFrame, text="Неверный логин или пароль!\nПовторите попытку снова",font=('',14))  
 self.errorLabel.pack(expand=1,pady=35)  
  
 self.repeatButton=tk.Button(self.errorWindowFrame, text="Повторить",width=20,font=('',12),command=errorWindow.destroy)  
 self.repeatButton.pack(side=tk.BOTTOM,pady=5)  
  
class mainProgramm(tk.Frame):  
  
 def \_\_init\_\_(self,win):  
 super().\_\_init\_\_(win)  
 self.startMain()  
  
 def startMain(self):  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2)-400  
 h = (h // 2)-400  
  
 win.title('АРМ заведующий')  
 win.geometry('800x600+{}+{}'.format(w,h))  
 win.resizable(False, False)  
  
 self.frameMain = tk.Frame(win)  
 self.frameMain.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Frame(self.frameMain,bg="#107eaf",width=300,height=600)  
 self.l1.pack(side=tk.RIGHT, fill=tk.Y)  
 self.l2 = tk.Label(self.frameMain,bg="#4d4f4c",width=500,height=600)  
 self.l2.pack(side=tk.LEFT, fill=tk.Y)  
  
 self.spiskiButton = tk.Button(self.frameMain,text = "Справочные документы",fg="black",width=18, font=('', 15),command=self.spiskiApp)  
 self.spiskiButton.place(x=555, y = 100)  
  
 self.docButton = tk.Button(self.frameMain,text="Оперативные документы", fg="black",width=18,font=('',15), command= self.docApp)  
 self.docButton.place(x=555,y=200)  
  
 self.othetButton = tk.Button(self.frameMain,text="Отчётные документы",fg="black",width=18,font=('',15),command = self.othWindowSp)  
 self.othetButton.place(x=555,y=300)  
  
 self.arhbd = tk.Button(self.frameMain,text="Восстановление БД", fg="black", width=18, font=('',15), command = self.arhbutton)  
 self.arhbd.place(x=555, y =400)  
  
 self.ifno = tk.Button(self.frameMain, text="Инфо", fg="black", width=18, font=('', 15),  
 command=self.infowind)  
 self.ifno.place(x=555, y=435)  
  
 self.closeApp = tk.Button(self.frameMain,text="Выход",fg="black",width=18,font=('',15),command=self.closeApp)  
 self.closeApp.place(x=555,y=475)  
  
 img = Image.open("logo2.png")  
 self.tkimage = ImageTk.PhotoImage(img)  
 self.l3=tk.Label(self.frameMain,image=self.tkimage,bg="#4d4f4c")  
 self.l3.place(x=50,y=75)  
  
 self.infoUser = tk.Label(self.frameMain,text=f"Пользователь:\t{Polzovatel}",font=('',16), bg="#4d4f4d")  
 self.infoUser.place(x=50,y=400)  
  
 self.post = tk.Label(self.frameMain,text="Должность:\tзаведущий заправкой ГСМ",font=('',16), bg="#4d4f4d")  
 self.post.place(x=50,y=425)  
  
 named\_tuple = time.localtime()  
 time\_string = time.strftime("%m/%d/%Y", named\_tuple)  
 self.LogTime=tk.Label(self.frameMain,text=f"Дата входа:\t{time\_string}",font=('',16), bg="#4d4f4d")  
 self.LogTime.place(x=50,y=450)  
  
 def spiskiApp(self):  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 400  
 h = (h // 2) - 400  
  
 spiskiAppWindow = tk.Toplevel(self)  
 spiskiAppWindow.title('Справочные документы')  
 spiskiAppWindow.geometry('800x600+{}+{}'.format(w, h))  
 spiskiAppWindow.resizable(False, False)  
  
 self.spiskiFrame = tk.Frame(spiskiAppWindow)  
 self.spiskiFrame.place(relwidth=1, relheight=1)  
  
 self.topLine = tk.Label(self.spiskiFrame, bg="#107eaf", height=5)  
 self.topLine.pack(side=tk.TOP, fill=tk.X)  
  
 self.topText = tk.Label(self.spiskiFrame, text="Справочные документы", bg="#107eaf", font=('', 18))  
 self.topText.place(x=305,y=30)  
  
 self.spiskButton1 = tk.Button(self.spiskiFrame, text ="Виды ГСМ", bd=0, justify=CENTER, height=3, font=('',18), command=partial(self.viewDB, list\_typegsm, "typegsm", "Справочник вида ГСМ"))  
 self.spiskButton1.pack(side = tk.TOP, fill = tk.X)  
  
 self.spiskButton2 = tk.Button(self.spiskiFrame, text = "Поставщики ГСМ", bd=0, justify=CENTER, height=3, font=('',18), command=partial(self.viewDB, list\_vendorgsm, "vendorgsm", "Справочник поставщиков ГСМ"))  
 self.spiskButton2.pack(side=tk.TOP, fill = tk.X)  
  
 self.spiskButton3 = tk.Button(self.spiskiFrame, text= "Водители предприятия", bd=0, justify=CENTER, height=3, font=('',18), command=partial(self.viewDB, list\_companydrivers, "companydrivers", "Справочник водителей предприятия"))  
 self.spiskButton3.pack(side=tk.TOP, fill = tk.X)  
  
 self.spiskButton4 = tk.Button(self.spiskiFrame, text = "Технические средства предприятия", bd=0, justify=CENTER, height=3, font=('',18), command=partial(self.viewDB, list\_comptechnmeans, "comptechnmeans", "Справочник технических средств предприятия"))  
 self.spiskButton4.pack(side=tk.TOP, fill= tk.X)  
  
 self.botLine = tk.Label(self.spiskiFrame, bg="#107eaf", height=5)  
 self.botLine.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.closeB = tk.Button(self.spiskiFrame, text='Закрыть', width=5, font=('', 18), command=spiskiAppWindow.destroy)  
 self.closeB.place(x=360, y=535)  
  
 def docApp(self):  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 400  
 h = (h // 2) - 400  
  
 docAppWindow = tk.Toplevel(self)  
 docAppWindow.title('Оперативные документы')  
 docAppWindow.geometry('800x700+{}+{}'.format(w, h))  
 docAppWindow.resizable(False, False)  
  
 self.docFrame = tk.Frame(docAppWindow)  
 self.docFrame.place(relwidth=1, relheight=1)  
  
 self.topLine = tk.Label(self.docFrame,bg="#107eaf",height=5)  
 self.topLine.pack(side=tk.TOP, fill = tk.X)  
  
 self.topLine = tk.Label(self.docFrame,text="Оперативные документы", bg="#107eaf", font = ('',18))  
 self.topLine.place(x=300,y=30)  
  
 self.docButton1 = tk.Button(self.docFrame, text = 'Договор на поставку', bd = 0, justify=CENTER, height=3, font=('',18), command=partial(self.viewDB, list\_deliverycontract, "deliverycontract", "Договор на поставку"))  
 self.docButton1.pack(side = tk.TOP, fill = X)  
  
 self.docButton2 = tk.Button(self.docFrame, text = 'Товарно-транспортная накладная', bd = 0, justify=CENTER, height=3, font=('',18), command=partial(self.viewDB, list\_ttn, "ttn", "ТТН"))  
 self.docButton2.pack(side = tk.TOP, fill = X)  
  
 self.docButton3 = tk.Button(self.docFrame, text = 'Карточка складского учета', bd = 0, justify=CENTER, height=3, font=('',18), command=partial(self.viewDB, list\_ksu, "ksu", "Карточная складского учета"))  
 self.docButton3.pack(side=tk.TOP, fill = X)  
  
 self.docButton4 = tk.Button(self.docFrame, text = 'Путевой лист', bd = 0, justify=CENTER, height=3, font=('',18), command=partial(self.viewDB, list\_pl, "pl", "ПЛ"))  
 self.docButton4.pack(side=tk.TOP, fill = X)  
  
 self.docButton5 = tk.Button(self.docFrame, text='Наряд', bd=0, justify=CENTER, height=3, font=('', 18), command=partial(self.viewDB, list\_naryad, "naryad", "Наряд"))  
 self.docButton5.pack(side=tk.TOP, fill=X)  
  
 self.docButton6 = tk.Button(self.docFrame, text='Файл прихода', bd=0, justify=CENTER, height=3, font=('', 18),  
 command=partial(self.viewDB, list\_prihod, "prihfile", "Файл прихода"))  
 self.docButton6.pack(side=tk.TOP, fill=X)  
  
 self.docButton7 = tk.Button(self.docFrame, text='Файл расхода', bd=0, justify=CENTER, height=3, font=('', 18),  
 command=partial(self.viewDB, list\_rashod, "rashfile", "Файл расхода"))  
 self.docButton7.pack(side=tk.TOP, fill=X)  
  
  
  
 self.botLine = tk.Label(self.docFrame, bg="#107eaf", height=5)  
 self.botLine.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.closeB = tk.Button(self.docFrame, text='Закрыть', width=5, font=('', 18), command=docAppWindow.destroy)  
 self.closeB.place(x=360, y=635)  
  
 def othWindowSp(self):  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 400  
 h = (h // 2) - 400  
  
 othWind = tk.Toplevel(self)  
 othWind.title("Отчётные документы")  
 othWind.geometry("800x600+{}+{}".format(w,h))  
 othWind.resizable(False,False)  
  
 self.othWindow = tk.Frame(othWind)  
 self.othWindow.place(relheight=1,relwidth=1)  
  
 self.topLine = tk.Label(self.othWindow, bg="#107eaf", height=5)  
 self.topLine.pack(side=tk.TOP, fill=tk.X)  
  
 self.topText = tk.Label(self.othWindow, text="Отчётные документы", bg="#107eaf", font=('', 18))  
 self.topText.place(x=315,y=30)  
  
 self.otButton1 = tk.Button(self.othWindow,text = 'Отчёт по заключенным договорам на поставку ГСМ',bd=0,justify=CENTER,height=3,font=('',18), command = self.otch1)  
 self.otButton1.pack(side=tk.TOP,fill=X)  
  
 self.otButton2 = tk.Button(self.othWindow,text='Отчёт о движении ГСМ на складе',bd=0,justify=CENTER, height=3, font=('', 18), command = self.otch2)  
 self.otButton2.pack(side=tk.TOP,fill=X)  
  
 self.otButton3 = tk.Button(self.othWindow, text='Отчёт по водителям',bd=0, justify=CENTER, height=3,font=('',18), command = self.otch3SQL)  
 self.otButton3.pack(side=tk.TOP, fill=X)  
  
 self.otButton4 = tk.Button(self.othWindow, text='Отчёт по путевым листам',bd=0, justify=CENTER, height=3,font=('', 18), command = self.otch4)  
 self.otButton4.pack(side=tk.TOP, fill=X)  
  
 self.botLine = tk.Label(self.othWindow, bg="#107eaf", height=5)  
 self.botLine.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.closeB = tk.Button(self.othWindow, text='Закрыть', width=5, font=('', 18), command=othWind.destroy)  
 self.closeB.place(x=360,y=535)  
  
 def closeApp(self):  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 200  
 h = (h // 2) - 200  
  
 clWin = tk.Toplevel(self)  
 clWin.title("Выход из АРМ")  
 clWin.geometry('300x150+{}+{}'.format(w, h))  
 clWin.resizable(False, False)  
  
 self.closeWindow = tk.Frame(clWin)  
 self.closeWindow.place(relwidth=1, relheight=1)  
  
 self.textCloseWindow = tk.Label(self.closeWindow, text="Вы дейтвительно хотите выйти из АРМ?",font=('',14))  
 self.textCloseWindow.place(x=15,y=40)  
  
 self.yesButton = tk.Button(self.closeWindow, text="Да",width=12 ,font = ('',12),command=self.rebot)  
 self.yesButton.place(x=15,y=100)  
  
 self.noButton = tk.Button(self.closeWindow, text="Нет",fg='red', width=12, font=('', 12),command=clWin.destroy)  
 self.noButton.place(x=150,y=100)  
  
 def viewDB(self, column\_names, tablename, tablenamerus):  
 self.viewTableDataBases = tk.Toplevel(self)  
 self.viewTableDataBases.title(f"{tablenamerus}")  
 screen\_width = self.viewTableDataBases.winfo\_screenwidth()  
 self.viewTableDataBases.geometry(f'{screen\_width}x800')  
 self.viewTableDataBases.rowconfigure(index=0, weight=1)  
 self.viewTableDataBases.columnconfigure(index=0, weight=1)  
 self.viewTableDataBases.resizable(False, False)  
  
 self.viewDB\_frame = tk.Frame(self.viewTableDataBases)  
 self.viewDB\_frame.place(relwidth=1, relheight=1)  
  
 data = []  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""SELECT \* FROM "{tablename}" """)  
 data = [row for row in cursor.fetchall()]  
 except Exception as \_ex:  
 print("ТАБЛИЦА НЕ ПОДТЯНУЛАСЬ")  
  
 self.tree = ttk.Treeview(self.viewDB\_frame, height=37, columns=column\_names, show="headings")  
 self.tree.pack(fill=X)  
  
 total\_width = 0  
 for i in column\_names:  
 self.tree.heading(f"{i}", text=f"{i}")  
 if i == '№':  
 self.tree.column(f"{i}", stretch=False)  
 self.tree.column(f"{i}", width=50)  
 total\_width += 50;  
 else:  
 column\_width = screen\_width // len(column\_names)  
 self.tree.column(f"{i}", width=column\_width, stretch=True)  
 total\_width += column\_width  
  
 for row in data:  
 self.tree.insert('', tk.END, values=tuple(row))  
 for i, value in enumerate(row):  
 max\_width = max([len(str(val)) for j, val in enumerate(row)] + [len(column\_names[i])])  
 column\_width = screen\_width // len(column\_names)  
 self.tree.column(column\_names[i], width=max\_width + 20, anchor=CENTER)  
  
  
 if tablename == "prihfile" or tablename == "rashfile":  
 self.blueLab = tk.Label(self.viewDB\_frame, bg="#107eaf", height=35)  
 self.blueLab.pack(side=tk.BOTTOM, fill=tk.X)  
 self.searchButton = tk.Button(self.viewDB\_frame, text="Поиск", bd=0, justify=CENTER, width=12, font=('', 18), command =partial(self.serCH, tablename))  
 self.searchButton.place(x=100, y=720)  
  
 self.closeButton = tk.Button(self.viewDB\_frame, text="Закрыть", bd=0, justify=CENTER, width=12, font=('', 18),  
 command=self.reboot)  
 self.closeButton.place(x=300, y=720)  
 elif tablename == "ksu":  
 self.blueLab = tk.Label(self.viewDB\_frame, bg="#107eaf", height=35)  
 self.blueLab.pack(side=tk.BOTTOM, fill=tk.X)  
 self.inputButton = tk.Button(self.viewDB\_frame, text="Добавить", bd=0, justify=CENTER, width=12,  
 font=('', 18),  
 command=partial(self.inputTableWindows, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=100, y=720)  
  
 self.searchButton = tk.Button(self.viewDB\_frame, text="Поиск", bd=0, justify=CENTER, width=12, font=('', 18), command =partial(self.serCH, tablename))  
 self.searchButton.place(x=300, y=720)  
  
 self.closeButton = tk.Button(self.viewDB\_frame, text="Закрыть", bd=0, justify=CENTER, width=12, font=('', 18),  
 command=self.reboot)  
 self.closeButton.place(x=500, y=720)  
 else:  
  
 self.blueLab = tk.Label(self.viewDB\_frame, bg="#107eaf", height=35)  
 self.blueLab.pack(side=tk.BOTTOM, fill = tk.X)  
  
 self.inputButton = tk.Button(self.viewDB\_frame, text="Добавить", bd=0, justify=CENTER, width=12, font=('', 18),  
 command=partial(self.inputTableWindows, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=100, y=720)  
  
 self.changeButton = tk.Button(self.viewDB\_frame, text="Изменить", bd=0, justify=CENTER, width=12, font=('', 18), command = partial(self.upDATE, tablename,column\_names, tablenamerus))  
 self.changeButton.place(x=300, y=720)  
  
 self.deleteButton = tk.Button(self.viewDB\_frame, text="Удаление", bd=0, justify=CENTER, width=12, font=('', 18),command = partial(self.DELButton,column\_names, tablename, tablenamerus))  
 self.deleteButton.place(x=500, y=720)  
  
 self.searchButton = tk.Button(self.viewDB\_frame, text="Поиск", bd=0, justify=CENTER, width=12, font=('', 18), command =partial(self.serCH, tablename))  
 self.searchButton.place(x=700, y=720)  
  
 self.closeButton = tk.Button(self.viewDB\_frame, text="Закрыть", bd=0, justify=CENTER, width=12, font=('', 18),  
 command=self.reboot)  
 self.closeButton.place(x=900, y=720)  
  
 def infowind(self):  
 self.otch1SQL = tk.Toplevel(self)  
 self.otch1SQL.title(f"Инфо")  
 screen\_width = self.otch1SQL.winfo\_screenwidth()  
 self.otch1SQL.geometry(f'400x300')  
 self.otch1SQL.rowconfigure(index=0, weight=1)  
 self.otch1SQL.columnconfigure(index=0, weight=1)  
 self.otch1SQL.resizable(False, False)  
  
 self.viewDB\_otch1SQL = tk.Frame(self.otch1SQL)  
 self.viewDB\_otch1SQL.place(relwidth=1, relheight=1)  
  
 self.txet = tk.Text(self.viewDB\_otch1SQL, width=20, wrap=WORD)  
 self.txet.insert(1.0, f"\tНаименование АРМ:\tАРМ заведующего ГСМ \n \tВерсия программы\t1.0.0\n \tИнформация о разработчике:\t\t\tСтудент 3 курса\n\t\t\t\t Группы АС-59\n\t\t\t\t Сахацкий А.С.")  
  
 self.txet.tag\_config('title', justify=CENTER,  
 font=("", 18, ''))  
 self.txet.pack(side=tk.TOP, fill=tk.X)  
  
 def rebot(a, \_event=None):  
 a.destroy()  
 os.environ['PGPASSWORD'] = f'{password}'  
 cmd = f'pg\_dump -h {host} -p 5432 -U {user} -Fc {db\_name} > {db\_name}.dump'  
 subprocess.call(cmd, shell=True)  
 del os.environ['PGPASSWORD']  
 loginSystem(win)  
 def reboot(a, \_event=None):  
 a.destroy()  
 mainProgramm(win)  
  
 def inputTableWindows(self, column\_names, tablename, tablenamerus):  
 if tablename == "typegsm":  
 buflist = list\_typegsm  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление в справочник ГСМ")  
 inputTableWin.geometry('400x500')  
 inputTableWin.resizable(False, False)  
  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l1=tk.Label(self.inTable, text= f"{buflist[0]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable,width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text= f"{buflist[1]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e=ttk.Entry(self.inTable,width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=3, font=('',18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e=ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.inTable, text="Добавить",fg="black",width=18,font=('',15), command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=90,y=400)  
  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black",width=18,font=('',15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=90, y=450)  
  
 if tablename == "vendorgsm":  
 buflist = list\_vendorgsm  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление в справочник поставщиков")  
 inputTableWin.geometry('400x500')  
 inputTableWin.resizable(False, False)  
  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.inTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable, width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.inTable, width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.inTable, text="Добавить", fg="black", width=18, font=('', 15), command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=90, y=400)  
  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=90, y=450)  
  
 if tablename == "companydrivers":  
 buflist = list\_companydrivers  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление в справочник водителей предприятия")  
 inputTableWin.geometry('410x800')  
 inputTableWin.resizable(False, False)  
  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.inTable, text=f"Табельный номер\n водителя", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable, width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text=f"ФИО водителя", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.inTable, width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"Государственный номер прикрепленного авто", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"Дата приема на работу", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.inTable, text=f"Дата выдачи водительсокого удостоверения", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.inTable, width=15)  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.inTable, text=f"Дата действия водительсокого удостоверения", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.inTable, width=15)  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.inTable, text=f"Номер водительского удостоверения", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.inTable, width=15)  
 self.l7e.pack(fill=tk.X)  
  
 self.l8 = tk.Label(self.inTable, text=f"Категория водительского удостоверения", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l8.pack(side=tk.TOP, fill=tk.X)  
 self.l8e = ttk.Entry(self.inTable, width=15)  
 self.l8e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.inTable, text="Добавить", fg="black", width=15, font=('', 15), command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=10, y=750)  
  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", width=15, font=('', 15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=200, y=750)  
  
 if tablename == "comptechnmeans":  
 buflist = list\_comptechnmeans  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление в справочник технических средств")  
 inputTableWin.geometry('410x800')  
 inputTableWin.resizable(False, False)  
  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.inTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2,font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable, width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.inTable, width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.inTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.inTable, width=15)  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.inTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.inTable, width=15)  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.inTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.inTable, width=15)  
 self.l7e.pack(fill=tk.X)  
  
 self.l9 = tk.Label(self.inTable, text=f"{buflist[8]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l9.pack(side=tk.TOP, fill=tk.X)  
 self.l9e = ttk.Entry(self.inTable, width=15)  
 self.l9e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
 self.inputButton = tk.Button(self.inTable, text="Добавить", fg="black", width=15, font=('', 15), command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=15, y=735)  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", width=15, font=('', 15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=205, y=735)  
  
 if tablename == "deliverycontract":  
 buflist = list\_deliverycontract  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление в договора на поставку")  
 inputTableWin.geometry('410x700')  
 inputTableWin.resizable(False, False)  
  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l0 = tk.Label(self.inTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l0.pack(side=tk.TOP, fill=tk.X)  
 self.l0e = ttk.Entry(self.inTable, width=15)  
 self.l0e.pack(fill=tk.X)  
  
 self.l1 = tk.Label(self.inTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable, width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.inTable, width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.inTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.inTable, width=15)  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.inTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=1, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.inTable, width=15)  
 self.l6e.pack(fill=tk.X)  
  
 self.l8 = tk.Label(self.inTable, text=f"{buflist[8]}", bd=0, justify=CENTER, height=1, font=('', 18))  
 self.l8.pack(side=tk.TOP, fill=tk.X)  
 self.l8e = ttk.Entry(self.inTable, width=15)  
 self.l8e.pack(fill=tk.X)  
  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
 self.inputButton = tk.Button(self.inTable, text="Добавить", fg="black", width=15, font=('', 15), command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=15, y=635)  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", width=15, font=('', 15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=205, y=635)  
  
 if tablename == "naryad":  
 buflist = list\_naryad  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление в наряд")  
 inputTableWin.geometry('400x800')  
 inputTableWin.resizable(False, False)  
  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.inTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable, width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.inTable, width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.inTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.inTable, width=15)  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.inTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.inTable, width=15)  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.inTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.inTable, width=15)  
 self.l7e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.inTable, text="Добавить", fg="black", width=18, font=('', 15), command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=90, y=700)  
  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=90, y=750)  
  
 if tablename == "pl":  
 buflist = list\_pl  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление ПЛ")  
 inputTableWin.geometry('400x850')  
 inputTableWin.resizable(False, False)  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.inTable, text=f"Номер путевого листа", bd=0, justify=CENTER, height=2,  
 font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable, width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text=f"Дата составления ПЛ", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.inTable, width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l22 = tk.Label(self.inTable, text=f"Код ГСМ", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l22.pack(side=tk.TOP, fill=tk.X)  
 self.l22e = ttk.Entry(self.inTable, width=15)  
 self.l22e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"Табельный номер водителя", bd=0, justify=CENTER,  
 height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"Номер водительского удостоверения", bd=0, justify=CENTER, height=2,  
 font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.inTable, text=f"Государственный номер авто", bd=0, justify=CENTER,  
 height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.inTable, width=15)  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.inTable, text=f"Расстояние", bd=0, justify=CENTER,  
 height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.inTable, width=15)  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.inTable, text=f"Единица измерения", bd=0, justify=CENTER,  
 height=2, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.inTable, width=15)  
 self.l7e.pack(fill=tk.X)  
  
 self.l77 = tk.Label(self.inTable, text=f"Остаток ГСМ", bd=0, justify=CENTER,  
 height=2, font=('', 18))  
 self.l77.pack(side=tk.TOP, fill=tk.X)  
 self.l77e = ttk.Entry(self.inTable, width=15)  
 self.l77e.pack(fill=tk.X)  
  
 self.l8 = tk.Label(self.inTable, text=f"Объем полученных ГСМ", bd=0, justify=CENTER,  
 height=2, font=('', 18))  
 self.l8.pack(side=tk.TOP, fill=tk.X)  
 self.l8e = ttk.Entry(self.inTable, width=15)  
 self.l8e.pack(fill=tk.X)  
  
 self.l88 = tk.Label(self.inTable, text=f"Объем потраченных ГСМ", bd=0, justify=CENTER,  
 height=2, font=('', 18))  
 self.l88.pack(side=tk.TOP, fill=tk.X)  
 self.l88e = ttk.Entry(self.inTable, width=15)  
 self.l88e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
 self.inputButton = tk.Button(self.inTable, text="Добавить", fg="black", width=15, font=('', 15), command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=15, y=800)  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", width=15, font=('', 15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=205, y=800)  
  
 if tablename == "ttn":  
 buflist = list\_ttn  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление в ТТН")  
 inputTableWin.geometry('400x600')  
 inputTableWin.resizable(False, False)  
  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.inTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable, width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.inTable, width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.inTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.inTable, width=15)  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.inTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.inTable, width=15)  
 self.l6e.pack(fill=tk.X)  
  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
 self.inputButton = tk.Button(self.inTable, text="Добавить", fg="black", width=15, font=('', 15),command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=15, y=535)  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", width=15, font=('', 15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=205, y=535)  
  
 if tablename == "ksu":  
 buflist = list\_ksu  
 inputTableWin = tk.Toplevel(self)  
 inputTableWin.title("Добавление в КСУ")  
 inputTableWin.geometry('400x700')  
 inputTableWin.resizable(False, False)  
  
 self.inTable = tk.Frame(inputTableWin)  
 self.inTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.inTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.inTable, width=15)  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.inTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.inTable, width=15)  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.inTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.inTable, width=15)  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.inTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.inTable, width=15)  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.inTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.inTable, width=15)  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.inTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.inTable, width=15)  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.inTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.inTable, width=15)  
 self.l7e.pack(fill=tk.X)  
  
 self.l8 = tk.Label(self.inTable, text=f"{buflist[7]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l8.pack(side=tk.TOP, fill=tk.X)  
 self.l8e = ttk.Entry(self.inTable, width=15)  
 self.l8e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.inTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
 self.inputButton = tk.Button(self.inTable, text="Добавить", fg="black", width=15, font=('', 15),  
 command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))  
 self.inputButton.place(x=15, y=635)  
 self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", width=15, font=('', 15),  
 command=inputTableWin.destroy)  
 self.closeB.place(x=205, y=635)  
 def inputTableSQL(self, column\_names, tablename, tablenamerus):  
 if tablename == "typegsm":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "code\_gsm", "name\_gsm", "unit", "mark\_gsm") VALUES   
 ('{value1}','{value2}', '{value3}','{value4}') """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 self.errorWindows()  
  
 if tablename == "vendorgsm":  
 check = False  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "code\_post", "name\_proizv", "addres\_proizv", "code\_gsm") VALUES   
 ('{value1}','{value2}', '{value3}', '{value4}') """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 self.errorWindows()  
  
 if tablename == "companydrivers":  
 check = False  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = self.l6e.get()  
 value7 = self.l7e.get()  
 value8 = self.l8e.get()  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "tab\_number", "drivers\_name", "national\_avto\_num", "date\_of\_hire", "date\_driverlicens", "validity\_day\_drlic", "num\_drivlicens", "category\_drivlicens") VALUES   
 ('{value1}','{value2}', '{value3}', '{value4}', '{value5}', '{value6}', '{value7}', '{value8}') """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 self.errorWindows()  
  
 if tablename == "deliverycontract":  
 check = False  
 value0 = self.l0e.get()  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = float(self.l5e.get())  
 value6 = float(self.l6e.get())  
 value7 = value5 \* value6  
 value8 = float(self.l8e.get())  
 value9 = float(value7\*value8) /100.0  
 value10= value9+value7  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "contract\_number", "date\_contract", "code\_gsm", "code\_post", "untill", "price", "amount", "stoim", "rate\_nds", "price\_nds", "price\_of\_nds") VALUES   
 ('{value0}','{value1}', '{value2}', '{value3}', '{value4}', {value5}, {value6}, {value7}, {value8}, {value9}, {value10} )""")  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 self.errorWindows()  
  
 if tablename == "naryad":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = self.l6e.get()  
 value7 = self.l7e.get()  
 print(value1, value2, value3, value4, value5)  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "nar\_number", "date\_nar","tab\_number", "drivers\_name", "national\_avto\_num","num\_drivlicens", "rabot") VALUES   
 ('{value1}','{value2}', '{value3}', '{value4}', '{value5}', '{value6}', '{value7}') """)  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 self.errorWindows()  
  
 if tablename == "ttn":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = float(self.l6e.get())  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "ttn\_number", "date\_zakl\_ttn", "code\_gsm", "code\_post", "untill", "amount") VALUES   
 ('{value1}','{value2}', '{value3}', '{value4}', '{value5}', {value6} )""")  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 self.errorWindows()  
  
 if tablename == "pl":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = float(self.l6e.get())  
 value7 = self.l7e.get()  
 value8 = float(self.l8e.get())  
 value9 = self.l22e.get()  
 value10=float(self.l77e.get())  
 value11 =float(self.l88e.get())  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "pl\_number", "date\_pl","code\_gsm","tab\_number","num\_drivlicens","national\_avto\_num","probeg","unit","ostatok","amount","potr") VALUES   
 ('{value1}','{value2}', '{value9}','{value3}', '{value4}', '{value5}',{value6},'{value7}',{value10},{value8}, {value11}) """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 self.errorWindows()  
  
 if tablename == "comptechnmeans":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = float(self.l5e.get())  
 value6 = float(self.l6e.get())  
 value7 = float(self.l7e.get())  
 if value5 > 3.5 and value5 < 12:  
 value8 = "310.29.10.41.112"  
 else:  
 value8 = "310.29.10.41.113"  
 value9 = float(self.l9e.get())  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "national\_avto\_num", "auto\_mark", "body\_number","untill","load\_capacity","year\_of\_product","first\_cost","kod\_porc","last\_cost") VALUES   
 ('{value1}','{value2}', '{value3}', '{value4}', {value5}, {value6}, {value7}, '{value8}', {value9}) """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 self.errorWindows()  
  
 if tablename == "ksu":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = self.l6e.get()  
 value7 = self.l7e.get()  
 value8 = float(self.l8e.get())  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""INSERT INTO "{tablename}"(  
 "ksu\_doc", "date\_ksu", "sklad\_number","tanker\_number","code\_gsm","code\_post","unit","ed\_price","start\_bal","amount\_prih\_gsm","amount\_rash\_gsm","end\_bal") VALUES   
 ('{value1}','{value2}', '{value3}', '{value4}', '{value5}', '{value6}', '{value7}', {value8}, 0,0,0,0) """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 self.errorWindows()  
  
 def errorWindows(self):  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 200  
 h = (h // 2) - 200  
 errorWindow = tk.Toplevel(self)  
 errorWindow.title("Ошибка ввода")  
 errorWindow.geometry('300x150+{}+{}'.format(w, h))  
 errorWindow.resizable(False, False)  
  
 self.errorWindowFrame = tk.Frame(errorWindow)  
 self.errorWindowFrame.place(relwidth=1, relheight=1)  
  
 self.errorLabel = tk.Label(self.errorWindowFrame, text="Некорректный данные !",  
 font=('', 14))  
 self.errorLabel.pack(expand=1, pady=35)  
  
 self.repeatButton = tk.Button(self.errorWindowFrame, text="Повторить", width=20, font=('', 12),  
 command=errorWindow.destroy)  
 self.repeatButton.pack(side=tk.BOTTOM, pady=5)  
  
 def serCH(self, tablename):  
  
 val = searhComboboxList[tablename]  
 serTable = tk.Toplevel(self)  
 serTable.title("Поиск")  
 serTable.geometry('300x100')  
 serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(serTable)  
 self.sTable.place(relheight=1, relwidth=1)  
  
 self.combobox = ttk.Combobox(self.sTable, values=val)  
 self.combobox.pack(anchor=NW, padx=6, pady=6)  
  
 self.temp = ttk.Entry(self.sTable, width=15)  
 self.temp.pack(fill=X)  
  
 self.poisk = tk.Button(self.sTable, text="поиск", width=15, command=partial(self.serBD, tablename))  
 self.poisk.pack(fill=X)  
 def serBD(self, tablename):  
 a1 = self.combobox.get()  
 a2 = self.temp.get()  
  
 self.table = tk.Toplevel(self)  
 self.table.title("Искомые значения")  
 screen\_width = self.table.winfo\_screenwidth()  
 self.table.geometry(f'{screen\_width}x800')  
 self.table.resizable(False, False)  
  
  
 self.dtable = tk.Frame(self.table)  
 self.dtable.place(relheight=1, relwidth=1)  
 data = []  
 if a1 in searhListINT:  
 a1 = searhSQLListINT[(tablename,a1)]  
 a2 = float(a2)  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""SELECT \* FROM {tablename} WHERE {a1} = {a2}  
 """)  
 data = (row for row in cursor.fetchall())  
 except Exception as \_ex:  
 self.errorWindows()  
 else:  
 a1 = searhSQLList[(tablename,a1)]  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""SELECT \* FROM {tablename} WHERE {a1} = '{a2}'  
 """)  
 data = (row for row in cursor.fetchall())  
 except Exception as \_ex:  
 self.errorWindows()  
  
 column\_names = searhComboboxList1[tablename]  
  
 self.TBL = ttk.Treeview(self.dtable, height=22, columns=column\_names, show="headings")  
 self.TBL.pack(fill=X)  
  
 total\_width = 0  
 for i in column\_names:  
 self.TBL.heading(f"{i}", text=f"{i}")  
 if i == '№':  
 self.TBL.column(f"{i}", stretch=False)  
 self.TBL.column(f"{i}", width=50)  
 total\_width += 50;  
 else:  
 column\_width = screen\_width // len(column\_names)  
 self.TBL.column(f"{i}", width=column\_width, stretch=True)  
 total\_width += column\_width  
  
 for row in data:  
 self.TBL.insert('', tk.END, values=tuple(row))  
 for i, value in enumerate(row):  
 max\_width = max([len(str(val)) for j, val in enumerate(row)] + [len(column\_names[i])])  
 column\_width = screen\_width // len(column\_names)  
 self.TBL.column(column\_names[i], width=max\_width + 20, anchor=CENTER)  
  
 self.blueLab = tk.Label(self.dtable, bg="#107eaf", height=35)  
 self.blueLab.pack(side=tk.BOTTOM, fill=tk.X)  
 self.closeButton = tk.Button(self.dtable, text="Закрыть", bd=0, justify=CENTER, width=12, font=('', 18),  
 command=self.reboot)  
 self.closeButton.place(x=650, y = 700)  
  
 def upDATE(self, tablename,column\_names, tablenamerus):  
  
 if tablename == "typegsm":  
 selection = self.tree.selection()  
 upValue = []  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")  
 upValue.append(item\_id1)  
  
 if len(upValue) == 0:  
 pass  
 else:  
 buflist = list\_typegsm  
  
 self.serTable = tk.Toplevel(self)  
 self.serTable.title(f"Изменение в {tablename}")  
 self.serTable.geometry('300x500')  
 self.serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(self.serTable)  
 self.sTable.place(relheight=1, relwidth=1)  
  
 self.l1 = tk.Label(self.sTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.sTable, width=15)  
 self.l1e.insert(0, upValue[0][0])  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.sTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.sTable, width=15)  
 self.l2e.insert(0, upValue[0][1])  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.sTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.sTable, width=15)  
 self.l3e.insert(0, upValue[0][2])  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.sTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.sTable, width=15)  
 self.l4e.insert(0, upValue[0][3])  
 self.l4e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.sTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.sTable, text="Изменить", fg="black", width=18, font=('', 15),  
 command=partial(self.upDateSQL,tablename, upValue,column\_names, tablenamerus))  
 self.inputButton.place(x=40, y=400)  
  
 self.closeB = tk.Button(self.sTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=self.serTable.destroy)  
 self.closeB.place(x=40, y=450)  
  
 if tablename == "vendorgsm":  
 selection = self.tree.selection()  
 upValue = []  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")  
 upValue.append(item\_id1)  
 if len(upValue) == 0:  
 pass  
 else:  
 buflist = list\_vendorgsm  
  
 self.serTable = tk.Toplevel(self)  
 self.serTable.title(f"Изменение в {tablename}")  
 self.serTable.geometry('300x500')  
 self.serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(self.serTable)  
 self.sTable.place(relheight=1, relwidth=1)  
  
 self.l1 = tk.Label(self.sTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.sTable, width=15)  
 self.l1e.insert(0, upValue[0][0])  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.sTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.sTable, width=15)  
 self.l2e.insert(0, upValue[0][1])  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.sTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.sTable, width=15)  
 self.l3e.insert(0, upValue[0][2])  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.sTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.sTable, width=15)  
 self.l4e.insert(0, upValue[0][3])  
 self.l4e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.sTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.sTable, text="Изменить", fg="black", width=18, font=('', 15),  
 command=partial(self.upDateSQL, tablename, upValue, column\_names,  
 tablenamerus))  
 self.inputButton.place(x=40, y=400)  
  
 self.closeB = tk.Button(self.sTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=self.serTable.destroy)  
 self.closeB.place(x=40, y=450)  
  
 if tablename == "companydrivers":  
 selection = self.tree.selection()  
 upValue = []  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")  
 upValue.append(item\_id1)  
 buflist = list\_companydrivers  
 if len(upValue) == 0:  
 pass  
 else:  
  
 self.serTable = tk.Toplevel(self)  
 self.serTable.title(f"Изменение в {tablename}")  
 self.serTable.geometry('410x700')  
 self.serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(self.serTable)  
 self.sTable.place(relheight=1, relwidth=1)  
  
 self.l1 = tk.Label(self.sTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.sTable, width=15)  
 self.l1e.insert(0, upValue[0][0])  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.sTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.sTable, width=15)  
 self.l2e.insert(0, upValue[0][1])  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.sTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.sTable, width=15)  
 self.l3e.insert(0, upValue[0][2])  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.sTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.sTable, width=15)  
 self.l4e.insert(0, upValue[0][3])  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.sTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.sTable, width=15)  
 self.l5e.insert(0, upValue[0][4])  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.sTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.sTable, width=15)  
 self.l6e.insert(0, upValue[0][5])  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.sTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.sTable, width=15)  
 self.l7e.insert(0, upValue[0][6])  
 self.l7e.pack(fill=tk.X)  
  
 self.l8 = tk.Label(self.sTable, text=f"{buflist[7]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l8.pack(side=tk.TOP, fill=tk.X)  
 self.l8e = ttk.Entry(self.sTable, width=15)  
 self.l8e.insert(0, upValue[0][7])  
 self.l8e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.sTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.sTable, text="Изменить", fg="black", width=18, font=('', 15),  
 command=partial(self.upDateSQL, tablename, upValue, column\_names,  
 tablenamerus))  
 self.inputButton.place(x=100, y=600)  
  
 self.closeB = tk.Button(self.sTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=self.serTable.destroy)  
 self.closeB.place(x=100, y=650)  
  
 if tablename == "comptechnmeans":  
 selection = self.tree.selection()  
 upValue = []  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")  
 upValue.append(item\_id1)  
 if len(upValue) == 0:  
 pass  
 else:  
 buflist = list\_comptechnmeans  
  
 self.serTable = tk.Toplevel(self)  
 self.serTable.title(f"Изменение в {tablename}")  
 self.serTable.geometry('410x700')  
 self.serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(self.serTable)  
 self.sTable.place(relheight=1, relwidth=1)  
  
 self.l1 = tk.Label(self.sTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.sTable, width=15)  
 self.l1e.insert(0, upValue[0][0])  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.sTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.sTable, width=15)  
 self.l2e.insert(0, upValue[0][1])  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.sTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.sTable, width=15)  
 self.l3e.insert(0, upValue[0][2])  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.sTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.sTable, width=15)  
 self.l4e.insert(0, upValue[0][3])  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.sTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.sTable, width=15)  
 self.l5e.insert(0, upValue[0][4])  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.sTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.sTable, width=15)  
 self.l6e.insert(0, upValue[0][5])  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.sTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.sTable, width=15)  
 self.l7e.insert(0, upValue[0][6])  
 self.l7e.pack(fill=tk.X)  
  
  
 self.fram1 = tk.Frame(self.sTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.sTable, text="Изменить", fg="black", width=18, font=('', 15),  
 command=partial(self.upDateSQL, tablename, upValue, column\_names,  
 tablenamerus))  
 self.inputButton.place(x=100, y=600)  
  
 self.closeB = tk.Button(self.sTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=self.serTable.destroy)  
 self.closeB.place(x=100, y=650)  
  
 if tablename == "deliverycontract":  
 selection = self.tree.selection()  
 upValue = []  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")  
 upValue.append(item\_id1)  
 if len(upValue) == 0:  
 pass  
 else:  
 buflist = list\_deliverycontract  
 self.serTable = tk.Toplevel(self)  
 self.serTable.title(f"Изменение в {tablename}")  
 self.serTable.geometry('410x800')  
 self.serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(self.serTable)  
 self.sTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.sTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.sTable, width=15)  
 self.l1e.insert(0, upValue[0][0])  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.sTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.sTable, width=15)  
 self.l2e.insert(0, upValue[0][1])  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.sTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.sTable, width=15)  
 self.l3e.insert(0, upValue[0][2])  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.sTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.sTable, width=15)  
 self.l4e.insert(0, upValue[0][3])  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.sTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.sTable, width=15)  
 self.l5e.insert(0, upValue[0][4])  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.sTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.sTable, width=15)  
 self.l6e.insert(0, upValue[0][5])  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.sTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.sTable, width=15)  
 self.l7e.insert(0, upValue[0][6])  
 self.l7e.pack(fill=tk.X)  
  
 self.l9 = tk.Label(self.sTable, text=f"{buflist[8]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l9.pack(side=tk.TOP, fill=tk.X)  
 self.l9e = ttk.Entry(self.sTable, width=15)  
 self.l9e.insert(0, upValue[0][8])  
 self.l9e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.sTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.sTable, text="Изменить", fg="black", width=18, font=('', 15),  
 command=partial(self.upDateSQL, tablename, upValue, column\_names,  
 tablenamerus))  
 self.inputButton.place(x=100, y=690)  
  
 self.closeB = tk.Button(self.sTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=self.serTable.destroy)  
 self.closeB.place(x=100, y=740)  
  
 if tablename == "naryad":  
 selection = self.tree.selection()  
 upValue = []  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")  
 upValue.append(item\_id1)  
 if len(upValue) == 0:  
 pass  
 else:  
 buflist = list\_naryad  
 self.serTable = tk.Toplevel(self)  
 self.serTable.title(f"Изменение в {tablename}")  
 self.serTable.geometry('410x800')  
 self.serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(self.serTable)  
 self.sTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.sTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.sTable, width=15)  
 self.l1e.insert(0, upValue[0][0])  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.sTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.sTable, width=15)  
 self.l2e.insert(0, upValue[0][1])  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.sTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.sTable, width=15)  
 self.l3e.insert(0, upValue[0][2])  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.sTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.sTable, width=15)  
 self.l4e.insert(0, upValue[0][3])  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.sTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.sTable, width=15)  
 self.l5e.insert(0, upValue[0][4])  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.sTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.sTable, width=15)  
 self.l6e.insert(0, upValue[0][5])  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.sTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.sTable, width=15)  
 self.l7e.insert(0, upValue[0][6])  
 self.l7e.pack(fill=tk.X)  
  
  
 self.fram1 = tk.Frame(self.sTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.sTable, text="Изменить", fg="black", width=18, font=('', 15),  
 command=partial(self.upDateSQL, tablename, upValue, column\_names,  
 tablenamerus))  
 self.inputButton.place(x=100, y=700)  
  
 self.closeB = tk.Button(self.sTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=self.serTable.destroy)  
 self.closeB.place(x=100, y=750)  
  
 if tablename == "pl":  
 selection = self.tree.selection()  
 upValue = []  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")  
 upValue.append(item\_id1)  
 if len(upValue) == 0:  
 pass  
 else:  
 buflist = list\_pl  
 self.serTable = tk.Toplevel(self)  
 self.serTable.title(f"Изменение в {tablename}")  
 self.serTable.geometry('410x850')  
 self.serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(self.serTable)  
 self.sTable.place(relwidth=1, relheight=1)  
  
 self.l1 = tk.Label(self.sTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.sTable, width=15)  
 self.l1e.insert(0, upValue[0][0])  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.sTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.sTable, width=15)  
 self.l2e.insert(0, upValue[0][1])  
 self.l2e.pack(fill=tk.X)  
  
 self.l22 = tk.Label(self.sTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l22.pack(side=tk.TOP, fill=tk.X)  
 self.l22e = ttk.Entry(self.sTable, width=15)  
 self.l22e.insert(0, upValue[0][2])  
 self.l22e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.sTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.sTable, width=15)  
 self.l3e.insert(0, upValue[0][3])  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.sTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.sTable, width=15)  
 self.l4e.insert(0, upValue[0][4])  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.sTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.sTable, width=15)  
 self.l5e.insert(0, upValue[0][5])  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.sTable, text=f"{buflist[6]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.sTable, width=15)  
 self.l6e.insert(0, upValue[0][6])  
 self.l6e.pack(fill=tk.X)  
  
 self.l7 = tk.Label(self.sTable, text=f"{buflist[7]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l7.pack(side=tk.TOP, fill=tk.X)  
 self.l7e = ttk.Entry(self.sTable, width=15)  
 self.l7e.insert(0, upValue[0][7])  
 self.l7e.pack(fill=tk.X)  
  
 self.l8 = tk.Label(self.sTable, text=f"{buflist[8]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l8.pack(side=tk.TOP, fill=tk.X)  
 self.l8e = ttk.Entry(self.sTable, width=15)  
 self.l8e.insert(0, upValue[0][8])  
 self.l8e.pack(fill=tk.X)  
  
 self.l9 = tk.Label(self.sTable, text=f"{buflist[9]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l9.pack(side=tk.TOP, fill=tk.X)  
 self.l9e = ttk.Entry(self.sTable, width=15)  
 self.l9e.insert(0, upValue[0][9])  
 self.l9e.pack(fill=tk.X)  
  
 self.l10 = tk.Label(self.sTable, text=f"{buflist[10]}", bd=0, justify=CENTER, height=2, font=('', 18))  
 self.l10.pack(side=tk.TOP, fill=tk.X)  
 self.l10e = ttk.Entry(self.sTable, width=15)  
 self.l10e.insert(0, upValue[0][10])  
 self.l10e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.sTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.sTable, text="Изменить", fg="black", width=18, font=('', 15),  
 command=partial(self.upDateSQL, tablename, upValue, column\_names,  
 tablenamerus))  
 self.inputButton.place(x=15, y=800)  
  
 self.closeB = tk.Button(self.sTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=self.serTable.destroy)  
 self.closeB.place(x=205, y=800)  
  
 if tablename == "ttn":  
 selection = self.tree.selection()  
 upValue = []  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")  
 upValue.append(item\_id1)  
 if len(upValue) == 0:  
 pass  
 else:  
 buflist = list\_ttn  
  
 self.serTable = tk.Toplevel(self)  
 self.serTable.title(f"Изменение в {tablename}")  
 self.serTable.geometry('300x600')  
 self.serTable.resizable(False, False)  
  
 self.sTable = tk.Frame(self.serTable)  
 self.sTable.place(relheight=1, relwidth=1)  
  
 self.l1 = tk.Label(self.sTable, text=f"{buflist[0]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l1.pack(side=tk.TOP, fill=tk.X)  
 self.l1e = ttk.Entry(self.sTable, width=15)  
 self.l1e.insert(0, upValue[0][0])  
 self.l1e.pack(fill=tk.X)  
  
 self.l2 = tk.Label(self.sTable, text=f"{buflist[1]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l2.pack(side=tk.TOP, fill=tk.X)  
 self.l2e = ttk.Entry(self.sTable, width=15)  
 self.l2e.insert(0, upValue[0][1])  
 self.l2e.pack(fill=tk.X)  
  
 self.l3 = tk.Label(self.sTable, text=f"{buflist[2]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l3.pack(side=tk.TOP, fill=tk.X)  
 self.l3e = ttk.Entry(self.sTable, width=15)  
 self.l3e.insert(0, upValue[0][2])  
 self.l3e.pack(fill=tk.X)  
  
 self.l4 = tk.Label(self.sTable, text=f"{buflist[3]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l4.pack(side=tk.TOP, fill=tk.X)  
 self.l4e = ttk.Entry(self.sTable, width=15)  
 self.l4e.insert(0, upValue[0][3])  
 self.l4e.pack(fill=tk.X)  
  
 self.l5 = tk.Label(self.sTable, text=f"{buflist[4]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l5.pack(side=tk.TOP, fill=tk.X)  
 self.l5e = ttk.Entry(self.sTable, width=15)  
 self.l5e.insert(0, upValue[0][4])  
 self.l5e.pack(fill=tk.X)  
  
 self.l6 = tk.Label(self.sTable, text=f"{buflist[5]}", bd=0, justify=CENTER, height=3, font=('', 18))  
 self.l6.pack(side=tk.TOP, fill=tk.X)  
 self.l6e = ttk.Entry(self.sTable, width=15)  
 self.l6e.insert(0, upValue[0][5])  
 self.l6e.pack(fill=tk.X)  
  
 self.fram1 = tk.Frame(self.sTable, bg="#107eaf", width=300, height=600)  
 self.fram1.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.inputButton = tk.Button(self.sTable, text="Изменить", fg="black", width=18, font=('', 15),  
 command=partial(self.upDateSQL, tablename, upValue, column\_names,  
 tablenamerus))  
 self.inputButton.place(x=40, y=500)  
  
 self.closeB = tk.Button(self.sTable, text='Закрыть', fg="black", width=18, font=('', 15),  
 command=self.serTable.destroy)  
 self.closeB.place(x=40, y=550)  
 def upDateSQL(self, tablename, upValue,column\_names, tablenamerus):  
 if tablename == "typegsm":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""UPDATE {tablename} SET code\_gsm='{value1}', name\_gsm='{value2}', unit='{value3}', mark\_gsm='{value4}'  
 WHERE code\_gsm='{upValue[0][0]}' AND name\_gsm='{upValue[0][1]}' AND unit='{upValue[0][2]}' AND mark\_gsm='{upValue[0][3]}' """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 pass  
  
 if tablename == "vendorgsm":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""UPDATE {tablename} SET code\_post='{value1}', name\_proizv='{value2}', addres\_proizv='{value3}', code\_gsm='{value4}'  
 WHERE code\_post='{upValue[0][0]}' AND name\_proizv='{upValue[0][1]}' AND addres\_proizv='{upValue[0][2]}' AND code\_gsm='{upValue[0][3]}' """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 pass  
  
 if tablename == "companydrivers":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = self.l6e.get()  
 value7 = self.l7e.get()  
 value8 = self.l8e.get()  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""UPDATE {tablename} SET tab\_number='{value1}', drivers\_name='{value2}', national\_avto\_num='{value3}', date\_of\_hire='{value4}', date\_driverlicens='{value5}',  
 validity\_day\_drlic='{value6}', num\_drivlicens='{value7}', category\_drivlicens='{value8}'  
 WHERE tab\_number='{upValue[0][0]}' AND drivers\_name='{upValue[0][1]}' AND national\_avto\_num='{upValue[0][2]}' AND date\_of\_hire='{upValue[0][3]}' AND date\_driverlicens='{upValue[0][4]}'   
 AND validity\_day\_drlic='{upValue[0][5]}' AND num\_drivlicens='{upValue[0][6]}' AND category\_drivlicens='{upValue[0][7]}'""")  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 pass  
  
 if tablename == "comptechnmeans":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = float(self.l5e.get())  
 value6 = float(self.l6e.get())  
 value7 = float(self.l7e.get())  
  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""UPDATE {tablename} SET national\_avto\_num='{value1}', auto\_mark='{value2}', body\_number='{value3}', untill='{value4}', load\_capacity={value5},  
 year\_of\_product={value6}, first\_cost={value7}  
 WHERE national\_avto\_num='{upValue[0][0]}' AND auto\_mark='{upValue[0][1]}' AND body\_number='{upValue[0][2]}' AND untill='{upValue[0][3]}' AND load\_capacity={upValue[0][4]}   
 AND year\_of\_product={upValue[0][5]} AND first\_cost={upValue[0][6]} """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 pass  
  
 if tablename == "deliverycontract":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = float(self.l6e.get())  
 value7 = float(self.l7e.get())  
 value8 = value6 \* value7  
 value9 = float(self.l9e.get())  
 value10 = float(value8 \* value9) / 100.0  
 value11 = value10 + value8  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""UPDATE {tablename} SET contract\_number='{value1}', date\_contract='{value2}', code\_gsm='{value3}', code\_post='{value4}', untill='{value5}',  
 price={value6}, amount={value7}, stoim={value8}, rate\_nds={value9}, price\_nds={value10}, price\_of\_nds={value11}  
 WHERE contract\_number='{upValue[0][0]}' AND date\_contract='{upValue[0][1]}' AND code\_gsm='{upValue[0][2]}' AND code\_post='{upValue[0][3]}' AND untill='{upValue[0][4]}'  
 AND price={upValue[0][5]} AND amount={upValue[0][6]} AND stoim={upValue[0][7]} AND rate\_nds={upValue[0][8]} AND price\_nds={upValue[0][9]} AND price\_of\_nds={upValue[0][10]}""")  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 pass  
  
 if tablename == "naryad":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = self.l6e.get()  
 value7 = self.l7e.get()  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""UPDATE {tablename} SET nar\_number='{value1}',date\_nar='{value2}', tab\_number='{value3}', drivers\_name='{value4}', national\_avto\_num='{value5}', num\_drivlicens='{value6}', rabot='{value7}'  
 WHERE nar\_number='{upValue[0][0]}' AND date\_nar='{upValue[0][1]}' AND tab\_number='{upValue[0][2]}' AND drivers\_name='{upValue[0][3]}' AND national\_avto\_num='{upValue[0][4]}' AND num\_drivlicens='{upValue[0][5]}' AND rabot='{upValue[0][6]}' """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 pass  
  
 if tablename == "pl":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = float(self.l6e.get())  
 value7 = self.l7e.get()  
 value8 = float(self.l8e.get())  
 value10 = self.l22e.get()  
 value11 = float(self.l9e.get())  
 value12 = float(self.l10e.get())  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(  
 f"""UPDATE {tablename} SET pl\_number='{value1}', date\_pl='{value2}',code\_gsm='{value10}', tab\_number='{value3}', num\_drivlicens='{value4}', national\_avto\_num='{value5}',probeg={value6},unit='{value7}', ostatok = {value8}, amount = {value11}, potr = {value12}  
 WHERE pl\_number='{upValue[0][0]}' AND date\_pl='{upValue[0][1]}' AND code\_gsm='{upValue[0][2]}' AND tab\_number='{upValue[0][3]}' AND num\_drivlicens='{upValue[0][4]}' AND national\_avto\_num='{upValue[0][5]}' AND probeg={upValue[0][6]} AND unit='{upValue[0][7]}' AND ostatok={upValue[0][8]} AND amount={upValue[0][9]} AND potr = {upValue[0][10]} """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 pass  
  
 if tablename == "ttn":  
 value1 = self.l1e.get()  
 value2 = self.l2e.get()  
 value3 = self.l3e.get()  
 value4 = self.l4e.get()  
 value5 = self.l5e.get()  
 value6 = float(self.l6e.get())  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(  
 f"""UPDATE {tablename} SET ttn\_number='{value1}', date\_zakl\_ttn='{value2}', code\_gsm='{value3}', code\_post='{value4}', untill='{value5}',amount={value6}  
 WHERE ttn\_number='{upValue[0][0]}' AND date\_zakl\_ttn='{upValue[0][1]}' AND code\_gsm='{upValue[0][2]}' AND code\_post='{upValue[0][3]}' AND untill='{upValue[0][4]}' AND amount={upValue[0][5]} """)  
 self.refresh(column\_names, tablename, tablenamerus)  
  
 except Exception as \_ex:  
 pass  
  
 def show\_password(self):  
 self.inputPassword.config(show="")  
 def hide\_password(self):  
 self.inputPassword.config(show="\*")  
 def arhbutton(self):  
 win.title('Авторизация')  
 win.geometry('400x150')  
 win.resizable(False, False)  
  
 self.frame = tk.Frame(win)  
 self.frame.place(relwidth=1, relheight=1)  
  
 self.lab\_Login = tk.Label(self.frame, text="Логин", font=10)  
 self.lab\_Login.place(x=40, y=15)  
  
 self.lab\_Password = tk.Label(self.frame, text="Пароль", font=15)  
 self.lab\_Password.place(x=40, y=45)  
  
 self.inputLogin = ttk.Entry(self.frame, width=15)  
 self.inputLogin.place(x=130, y=15)  
  
 self.inputPassword = ttk.Entry(self.frame, width=15, show="\*")  
 self.inputPassword.place(x=130, y=45)  
  
 self.show\_button = tk.Button(self.frame, text="Показать пароль", command=self.show\_password)  
 self.show\_button.place(x=40, y=78)  
  
 self.hide\_button = tk.Button(self.frame, text="Скрыть пароль", fg="black", width=14, font=('', 12),  
 command=self.hide\_password)  
 self.hide\_button.place(x=220, y=78)  
  
 self.connButton = tk.Button(self.frame, text="Войти", fg="black", width=10, font=('', 12),  
 command=self.checkarhbd)  
 self.connButton.pack(side=tk.BOTTOM, pady=10)  
 def checkarhbd(self):  
 global sysAdmin  
 sysAdmin = self.inputLogin.get()  
 if sysAdmin == "sysadmin" and self.inputPassword.get() == "admin":  
 self.destroy()  
 self.frame.destroy()  
 arhBD(win)  
 else:  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 200  
 h = (h // 2) - 200  
 errorWindow = tk.Toplevel(self)  
 errorWindow.title("Ошибка входа")  
 errorWindow.geometry('300x150+{}+{}'.format(w, h))  
 errorWindow.resizable(False, False)  
  
 self.errorWindowFrame = tk.Frame(errorWindow)  
 self.errorWindowFrame.place(relwidth=1, relheight=1)  
  
 self.errorLabel = tk.Label(self.errorWindowFrame,  
 text="Неверный логин или пароль!\nПовторите попытку снова", font=('', 14))  
 self.errorLabel.pack(expand=1, pady=35)  
  
 self.repeatButton = tk.Button(self.errorWindowFrame, text="Повторить", width=20, font=('', 12),  
 command=errorWindow.destroy)  
 self.repeatButton.pack(side=tk.BOTTOM, pady=5)  
  
 def DELButton(self,column\_names, tablename, tablenamerus):  
 if tablename == "typegsm":  
 selection = self.tree.selection()  
 delValue = ''  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")[0]  
 delValue = item\_id1  
 print(item\_id1)  
 if len(delValue) == 0:  
 pass  
 else:  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""DELETE FROM {tablename} WHERE code\_gsm='{delValue}' """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 print("ERORR")  
  
 if tablename == "vendorgsm":  
 selection = self.tree.selection()  
 delValue1 = ''  
 delValue2 = ''  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")[0]  
 item\_id2 = self.tree.item(item, "values")[3]  
 delValue1 = item\_id1  
 delValue2 = item\_id2  
 print(item\_id1,item\_id2)  
 if len(delValue1) == 0:  
 pass  
 else:  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""DELETE FROM {tablename} WHERE code\_post='{delValue1}' AND code\_gsm='{delValue2}' """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 print("ERORR")  
  
 if tablename == "companydrivers":  
 selection = self.tree.selection()  
 delValue1 = ''  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")[0]  
 delValue1 = item\_id1  
 print(item\_id1)  
 if len(delValue1) == 0:  
 pass  
 else:  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(  
 f"""DELETE FROM {tablename} WHERE tab\_number='{delValue1}' """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 print("ERORR")  
  
 if tablename == "comptechnmeans":  
 selection = self.tree.selection()  
 delValue1 = ''  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")[0]  
 delValue1 = item\_id1  
 print(item\_id1)  
 if len(delValue1) == 0:  
 pass  
 else:  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(  
 f"""DELETE FROM {tablename} WHERE national\_avto\_num='{delValue1}' """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 print("ERORR")  
  
 if tablename == "deliverycontract":  
 selection = self.tree.selection()  
 delValue1 = ''  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")[0]  
 delValue1 = item\_id1  
 print(item\_id1)  
 if len(delValue1) == 0:  
 pass  
 else:  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(  
 f"""DELETE FROM {tablename} WHERE contract\_number='{delValue1}' """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 print("ERORR")  
  
 if tablename == "naryad":  
 selection = self.tree.selection()  
 delValue1 = ''  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")[0]  
 item\_id2 = self.tree.item(item, "values")[1]  
 delValue1 = item\_id1  
 delValue2 = item\_id2  
 print(item\_id1, item\_id2)  
 if len(delValue1) == 0:  
 pass  
 else:  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(  
 f"""DELETE FROM {tablename} WHERE nar\_number='{delValue1}' """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 print("ERORR")  
  
 if tablename == "pl":  
 selection = self.tree.selection()  
 delValue1 = ''  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")[0]  
 delValue1 = item\_id1  
 print(item\_id1)  
 if len(delValue1) == 0:  
 pass  
 else:  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(  
 f"""DELETE FROM {tablename} WHERE pl\_number='{delValue1}' """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 print("ERORR")  
  
 if tablename == "ttn":  
 selection = self.tree.selection()  
 delValue1 = ''  
 delValue2 = ''  
 for item in selection:  
 item\_id1 = self.tree.item(item, "values")[0]  
 item\_id2 = self.tree.item(item, "values")[1]  
 delValue1 = item\_id1  
 delValue2 = item\_id2  
 print(item\_id1, item\_id2)  
 if len(delValue1) == 0:  
 pass  
 else:  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(  
 f"""DELETE FROM {tablename} WHERE ttn\_number='{delValue1}' AND date\_zakl\_ttn='{delValue2}' """)  
 check = True  
 if check:  
 self.refresh(column\_names, tablename, tablenamerus)  
 except Exception as \_ex:  
 print("ERORR")  
  
 def otch1(self):  
 win = tk.Toplevel(self)  
 win.title('Введите код ГСМ')  
 win.geometry('300x150')  
 win.resizable(False, False)  
  
 self.frame = tk.Frame(win)  
 self.frame.place(relwidth=1, relheight=1)  
  
 self.lab\_Login = tk.Label(self.frame, text="Код ГСМ", font=10)  
 self.lab\_Login.place(x=15, y=15)  
  
 self.l1 = ttk.Entry(self.frame, width=13)  
 self.l1.place(x=150, y=15)  
  
 self.connButton = tk.Button(self.frame, text="Сформировать", fg="black", width=10, font=('', 12),  
 command=self.otch1SQL)  
 self.connButton.pack(side=tk.BOTTOM, pady=10)  
 def otch1SQL(self):  
 cdgsm = self.l1.get()  
 self.otch1SQL = tk.Toplevel(self)  
 self.otch1SQL.title(f"Отчет по заключенным договорам")  
 screen\_width = self.otch1SQL.winfo\_screenwidth()  
 self.otch1SQL.geometry(f'{screen\_width}x800')  
 self.otch1SQL.rowconfigure(index=0, weight=1)  
 self.otch1SQL.columnconfigure(index=0, weight=1)  
 self.otch1SQL.resizable(False, False)  
  
 self.viewDB\_otch1SQL = tk.Frame(self.otch1SQL)  
 self.viewDB\_otch1SQL.place(relwidth=1, relheight=1)  
  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""SELECT COUNT(\*) AS contract\_count  
 FROM deliverycontract  
 WHERE deliverycontract.code\_gsm = '{cdgsm}';""")  
 result = cursor.fetchone()[0]  
 except Exception as \_ex:  
 pass  
  
 self.txet = tk.Text(self.viewDB\_otch1SQL, width=20, height=3, wrap=WORD)  
 self.txet.insert(1.0, f"Общее количество договоров (шт): {result}")  
 self.txet.tag\_add('title', 1.0, '1.end')  
  
 self.txet.tag\_config('title', justify=CENTER,  
 font=("", 18, ''))  
 self.txet.pack(side=tk.TOP, fill=tk.X)  
  
 data = []  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""SELECT contract\_number, date\_contract, name\_proizv, typegsm.code\_gsm, name\_gsm, deliverycontract.untill, deliverycontract.amount, deliverycontract.stoim  
 FROM deliverycontract, typegsm, vendorgsm  
 WHERE deliverycontract.code\_gsm = '{cdgsm}' AND typegsm.code\_gsm = '{cdgsm}' AND vendorgsm.code\_gsm = '{cdgsm}';""")  
 data = [row for row in cursor.fetchall()]  
 except Exception as \_ex:  
 pass  
  
 colmnsname1 = ['Список договоров','Список ГСМ']  
 self.tree1 = ttk.Treeview(self.viewDB\_otch1SQL,height=0, columns=colmnsname1, show="headings")  
 self.tree1.pack(fill=X)  
  
 self.tree1.heading(f"{colmnsname1[0]}", text=f"{colmnsname1[0]}")  
 self.tree1.column(f"{colmnsname1[0]}", stretch=False,width=575)  
 self.tree1.heading(f"{colmnsname1[1]}", text=f"{colmnsname1[1]}")  
 self.tree1.column(f"{colmnsname1[1]}", stretch=False, width=screen\_width-575)  
  
 colmnsname2 = ['Номер договора','Дата заключения договора','Наименование поставщика','Код ГСМ','Наименование ГСМ','Единица измерения','Объем поставки','Стоимость договора']  
 self.tree2 = ttk.Treeview(self.viewDB\_otch1SQL, columns=colmnsname2,height=35, show="headings")  
 self.tree2.pack(fill=X)  
  
 for i in colmnsname2:  
 self.tree2.heading(f"{i}", text=f"{i}")  
 if i == 'Номер договора':  
 self.tree2.column(f"{i}", stretch=False)  
 self.tree2.column(f"{i}", width=100)  
 if i == 'Дата заключения договора' or i == 'Наименование поставщика':  
 self.tree2.column(f"{i}", stretch=False)  
 self.tree2.column(f"{i}", width=200)  
 else:  
 self.tree2.column(f"{i}", width=130, stretch=True)  
  
 for row in data:  
 self.tree2.insert('', tk.END, values=tuple(row))  
  
 self.blueLab = tk.Label(self.viewDB\_otch1SQL, bg="#107eaf", height=35)  
 self.blueLab.pack(side=tk.BOTTOM, fill=tk.X)  
 self.closeButton = tk.Button(self.viewDB\_otch1SQL, text="Закрыть", bd=0, justify=CENTER, width=12, font=('', 18),  
 command=self.reboot)  
 self.closeButton.place(x=650, y=750)  
  
 def otch2(self):  
 win = tk.Toplevel(self)  
 win.title('Введите период')  
 win.geometry('300x150')  
 win.resizable(False, False)  
  
 self.frame = tk.Frame(win)  
 self.frame.place(relwidth=1, relheight=1)  
  
 self.lab\_Login = tk.Label(self.frame, text="Начало периода", font=10)  
 self.lab\_Login.place(x=15, y=15)  
  
 self.lab\_Password = tk.Label(self.frame, text="Конец периода", font=15)  
 self.lab\_Password.place(x=15, y=45)  
  
 self.inputmes1= ttk.Entry(self.frame, width=10)  
 self.inputmes1.place(x=150, y=15)  
  
 self.inputmes2 = ttk.Entry(self.frame, width=10)  
 self.inputmes2.place(x=150, y=45)  
  
 self.connButton = tk.Button(self.frame, text="Сформировать", fg="black", width=10, font=('', 12),  
 command=self.otch2SQL)  
 self.connButton.pack(side=tk.BOTTOM, pady=10)  
 def otch2SQL(self):  
 mes1 = int(self.inputmes1.get())  
 mes2 = int(self.inputmes2.get())  
  
 self.otch2SQL = tk.Toplevel(self)  
 self.otch2SQL.title(f"Отчет о движении ГСМ на складе")  
 screen\_width = self.otch2SQL.winfo\_screenwidth()  
 self.otch2SQL.geometry(f'{screen\_width}x800')  
 self.otch2SQL.rowconfigure(index=0, weight=1)  
 self.otch2SQL.columnconfigure(index=0, weight=1)  
 self.otch2SQL.resizable(False, False)  
  
 self.viewDB\_otch2SQL = tk.Frame(self.otch2SQL)  
 self.viewDB\_otch2SQL.place(relwidth=1, relheight=1)  
  
 self.txet = tk.Text(self.viewDB\_otch2SQL, width=20, height=3, wrap=WORD)  
 self.txet.insert(1.0, f"Оборотная ведомость за период {mes1} по {mes2}")  
 self.txet.tag\_add('title', 1.0, '1.end')  
  
 self.txet.tag\_config('title', justify=CENTER,  
 font=("", 18, ''))  
 self.txet.pack(side=tk.TOP, fill=tk.X)  
  
  
 data = []  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f""" SELECT  
 typegsm.code\_gsm,  
 typegsm.name\_gsm,  
 typegsm.unit,  
 ksu.start\_bal,  
 (  
 SELECT SUM(amount)  
 FROM prihfile  
 WHERE ksu.code\_gsm = prihfile.code\_gsm  
 ) AS prih\_gsm,  
 (  
 SELECT SUM(amount)  
 FROM rashfile  
 WHERE ksu.code\_gsm = rashfile.code\_gsm  
 ) AS rash\_gsm,  
 ksu.end\_bal  
 FROM  
 typegsm, ksu, prihfile, rashfile  
 WHERE typegsm.code\_gsm = ksu.code\_gsm   
 AND ksu.code\_gsm = prihfile.code\_gsm   
 AND ksu.code\_gsm = rashfile.code\_gsm   
 AND substring(prihfile.date\_zakl\_ttn, 4, 2)::integer >= {mes1}   
 AND substring(rashfile.date\_pl, 4, 2)::integer >= {mes1}  
 AND substring(prihfile.date\_zakl\_ttn, 4, 2)::integer <= {mes2}   
 AND substring(rashfile.date\_pl, 4, 2)::integer <= {mes2};  
  
 """)  
 data = [row for row in cursor.fetchall()]  
 except Exception as \_ex:  
 pass  
  
 colmnsname1 = ['Код ГСМ','Наименование ГСМ','Единица измерения','Остаток на начало периода','Количество прихода','Количество расхода','Остаток на конец периода']  
 self.tree1 = ttk.Treeview(self.viewDB\_otch2SQL, height=37, columns=colmnsname1, show="headings")  
 self.tree1.pack(fill=X)  
  
 self.tree1.heading(f"{colmnsname1[0]}", text=f"{colmnsname1[0]}")  
 self.tree1.column(f"{colmnsname1[0]}", stretch=False, width=210)  
  
 self.tree1.heading(f"{colmnsname1[1]}", text=f"{colmnsname1[1]}")  
 self.tree1.column(f"{colmnsname1[1]}", stretch=False, width=250)  
  
 self.tree1.heading(f"{colmnsname1[2]}", text=f"{colmnsname1[2]}")  
 self.tree1.column(f"{colmnsname1[2]}", stretch=False, width=180)  
  
 self.tree1.heading(f"{colmnsname1[3]}", text=f"{colmnsname1[3]}")  
 self.tree1.column(f"{colmnsname1[3]}", stretch=False, width=200)  
  
 self.tree1.heading(f"{colmnsname1[4]}", text=f"{colmnsname1[4]}")  
 self.tree1.column(f"{colmnsname1[4]}", stretch=False, width=200)  
  
 self.tree1.heading(f"{colmnsname1[5]}", text=f"{colmnsname1[5]}")  
 self.tree1.column(f"{colmnsname1[5]}", stretch=False, width=200)  
  
 self.tree1.heading(f"{colmnsname1[6]}", text=f"{colmnsname1[6]}")  
 self.tree1.column(f"{colmnsname1[6]}", stretch=False, width=200)  
  
 for row in data:  
 self.tree1.insert('', tk.END, values=tuple(row))  
  
 self.blueLab = tk.Label(self.viewDB\_otch2SQL, bg="#107eaf", height=35)  
 self.blueLab.pack(side=tk.BOTTOM, fill=tk.X)  
 self.closeButton = tk.Button(self.viewDB\_otch2SQL, text="Закрыть", bd=0, justify=CENTER, width=12,  
 font=('', 18),  
 command=self.reboot)  
 self.closeButton.place(x=650, y=750)  
  
 def otch3SQL(self):  
 self.otch3SQL = tk.Toplevel(self)  
 self.otch3SQL.title(f"Отчет по водителям")  
 screen\_width = self.otch3SQL.winfo\_screenwidth()  
 self.otch3SQL.geometry(f'{screen\_width}x800')  
 self.otch3SQL.rowconfigure(index=0, weight=1)  
 self.otch3SQL.columnconfigure(index=0, weight=1)  
 self.otch3SQL.resizable(False, False)  
  
 self.viewDB\_otch3SQLL = tk.Frame(self.otch3SQL)  
 self.viewDB\_otch3SQLL.place(relwidth=1, relheight=1)  
  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""SELECT COUNT(\*) AS contract\_count  
 FROM companydrivers  
 """)  
 result = cursor.fetchone()[0]  
 except Exception as \_ex:  
 pass  
  
 self.txet = tk.Text(self.viewDB\_otch3SQLL, width=20, height=3, wrap=WORD)  
 self.txet.insert(1.0, f"Общее количество водителей: {result}")  
 self.txet.tag\_add('title', 1.0, '1.end')  
  
 self.txet.tag\_config('title', justify=CENTER,  
 font=("", 18, ''))  
 self.txet.pack(side=tk.TOP, fill=tk.X)  
  
 data = []  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""  
 SELECT  
 cd.tab\_number,  
 cd.drivers\_name,  
 COUNT(pl.pl\_number) AS koll\_vyd\_pl,  
 COALESCE(SUM(pl.probeg), 0) AS kol\_pr\_km  
 FROM  
 CompanyDrivers cd  
 LEFT JOIN  
 PL pl ON pl.tab\_number = cd.tab\_number  
 GROUP BY  
 cd.tab\_number,  
 cd.drivers\_name;  
  
 """)  
 data = [row for row in cursor.fetchall()]  
 except Exception as \_ex:  
 pass  
  
 colmnsname1 = ['Табельный номер','ФИО водителя','Количество выданых путевых листов','Количество приехавших км']  
 self.tree1 = ttk.Treeview(self.viewDB\_otch3SQLL, height=37, columns=colmnsname1, show="headings")  
 self.tree1.pack(fill=X)  
  
 self.tree1.heading(f"{colmnsname1[0]}", text=f"{colmnsname1[0]}")  
 self.tree1.column(f"{colmnsname1[0]}", stretch=False, width=300)  
  
 self.tree1.heading(f"{colmnsname1[1]}", text=f"{colmnsname1[1]}")  
 self.tree1.column(f"{colmnsname1[1]}", stretch=False, width=450)  
  
 self.tree1.heading(f"{colmnsname1[2]}", text=f"{colmnsname1[2]}")  
 self.tree1.column(f"{colmnsname1[2]}", stretch=False, width=400)  
  
 self.tree1.heading(f"{colmnsname1[3]}", text=f"{colmnsname1[3]}")  
 self.tree1.column(f"{colmnsname1[3]}", stretch=False, width=300)  
  
 for row in data:  
 self.tree1.insert('', tk.END, values=tuple(row))  
  
 self.blueLab = tk.Label(self.viewDB\_otch3SQLL, bg="#107eaf", height=35)  
 self.blueLab.pack(side=tk.BOTTOM, fill=tk.X)  
 self.closeButton = tk.Button(self.viewDB\_otch3SQLL, text="Закрыть", bd=0, justify=CENTER, width=12, font=('', 18),  
 command=self.reboot)  
 self.closeButton.place(x=650, y=750)  
  
 def otch4(self):  
 win = tk.Toplevel(self)  
 win.title('Введите табномер водителя')  
 win.geometry('300x150')  
 win.resizable(False, False)  
  
 self.frame = tk.Frame(win)  
 self.frame.place(relwidth=1, relheight=1)  
  
 self.lab\_Login = tk.Label(self.frame, text="Табельный номер\nводителя", font=10)  
 self.lab\_Login.place(x=15, y=15)  
  
 self.inp2 = ttk.Entry(self.frame, width=10)  
 self.inp2.place(x=150, y=15)  
  
 self.connButton = tk.Button(self.frame, text="Сформировать", fg="black", width=10, font=('', 12),  
 command=self.otch4SQL)  
  
 self.connButton.pack(side=tk.BOTTOM, pady=10)  
 def otch4SQL(self):  
 tbvod = self.inp2.get()  
 self.otch3SQL = tk.Toplevel(self)  
 self.otch3SQL.title(f"Отчет по путевым листам")  
 screen\_width = self.otch3SQL.winfo\_screenwidth()  
 self.otch3SQL.geometry(f'{screen\_width}x800')  
 self.otch3SQL.rowconfigure(index=0, weight=1)  
 self.otch3SQL.columnconfigure(index=0, weight=1)  
 self.otch3SQL.resizable(False, False)  
  
 self.viewDB\_otch3SQLL = tk.Frame(self.otch3SQL)  
 self.viewDB\_otch3SQLL.place(relwidth=1, relheight=1)  
  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""SELECT COUNT(\*) AS contract\_count  
 FROM pl  
 WHERE pl.tab\_number = '{tbvod}';""")  
 result = cursor.fetchone()[0]  
 except Exception as \_ex:  
 pass  
  
 self.txet = tk.Text(self.viewDB\_otch3SQLL, width=20, height=3, wrap=WORD)  
 self.txet.insert(1.0, f"Общее количество выданных ПЛ: {result}")  
 self.txet.tag\_add('title', 1.0, '1.end')  
  
 self.txet.tag\_config('title', justify=CENTER,  
 font=("", 18, ''))  
 self.txet.pack(side=tk.TOP, fill=tk.X)  
  
 data = []  
 try:  
 with conn.cursor() as cursor:  
 cursor.execute(f"""SELECT pl.national\_avto\_num, pl.pl\_number, pl.tab\_number, companydrivers.drivers\_name, pl.code\_gsm, pl.unit, pl.ostatok, pl.amount, pl.potr  
 FROM pl, companydrivers  
 WHERE pl.tab\_number='{tbvod}' AND companydrivers.tab\_number='{tbvod}';  
 """)  
 data = [row for row in cursor.fetchall()]  
 except Exception as \_ex:  
 pass  
  
 colmnsname1 = ['Государственный номер ТС','Номер ПЛ','Табельный номер водителя','ФИО водителя',  
 'Код ГСМ','Единица измерения','Остаток ГСМ','Объем полученных ГСМ','Объем потраченных ГСМ']  
 self.tree1 = ttk.Treeview(self.viewDB\_otch3SQLL, height=37, columns=colmnsname1, show="headings")  
 self.tree1.pack(fill=X)  
  
 self.tree1.heading(f"{colmnsname1[0]}", text=f"{colmnsname1[0]}")  
 self.tree1.column(f"{colmnsname1[0]}", stretch=False, width=200)  
  
 self.tree1.heading(f"{colmnsname1[1]}", text=f"{colmnsname1[1]}")  
 self.tree1.column(f"{colmnsname1[1]}", stretch=False, width=120)  
  
 self.tree1.heading(f"{colmnsname1[2]}", text=f"{colmnsname1[2]}")  
 self.tree1.column(f"{colmnsname1[2]}", stretch=False, width=200)  
  
 self.tree1.heading(f"{colmnsname1[3]}", text=f"{colmnsname1[3]}")  
 self.tree1.column(f"{colmnsname1[3]}", stretch=False, width=200)  
  
 self.tree1.heading(f"{colmnsname1[4]}", text=f"{colmnsname1[4]}")  
 self.tree1.column(f"{colmnsname1[4]}", stretch=False, width=180)  
  
 self.tree1.heading(f"{colmnsname1[5]}", text=f"{colmnsname1[5]}")  
 self.tree1.column(f"{colmnsname1[5]}", stretch=False, width=120)  
  
 self.tree1.heading(f"{colmnsname1[6]}", text=f"{colmnsname1[6]}")  
 self.tree1.column(f"{colmnsname1[6]}", stretch=False, width=120)  
  
 self.tree1.heading(f"{colmnsname1[7]}", text=f"{colmnsname1[7]}")  
 self.tree1.column(f"{colmnsname1[7]}", stretch=False, width=150)  
  
 self.tree1.heading(f"{colmnsname1[8]}", text=f"{colmnsname1[8]}")  
 self.tree1.column(f"{colmnsname1[8]}", stretch=False, width=140)  
  
 for row in data:  
 self.tree1.insert('', tk.END, values=tuple(row))  
  
 self.blueLab = tk.Label(self.viewDB\_otch3SQLL, bg="#107eaf", height=35)  
 self.blueLab.pack(side=tk.BOTTOM, fill=tk.X)  
 self.closeButton = tk.Button(self.viewDB\_otch3SQLL, text="Закрыть", bd=0, justify=CENTER, width=12,  
 font=('', 18),  
 command=self.reboot)  
 self.closeButton.place(x=650, y=750)  
  
 def refresh(self, column\_names, tablename, tablenamerus):  
 self.viewTableDataBases.destroy()  
 self.viewDB(column\_names, tablename, tablenamerus)  
  
class arhBD(tk.Frame):  
 def \_\_init\_\_(self, win):  
 super().\_\_init\_\_(win)  
 self.startarhbd()  
  
 def startarhbd(self):  
 w = win.winfo\_screenwidth()  
 h = win.winfo\_screenheight()  
 w = (w // 2) - 200  
 h = (h // 2) - 200  
  
 win.title('Восстановление БД')  
 win.geometry('500x400+{}+{}'.format(w, h))  
 win.resizable(False, False)  
 self.frame = tk.Frame(win, bg="#4d4f4c")  
 self.frame.place(relwidth=1, relheight=1)  
  
 img = Image.open("logo2.png")  
 self.tkimage = ImageTk.PhotoImage(img)  
 self.l3 = tk.Label(self.frame, image=self.tkimage, bg="#4d4f4c")  
 self.l3.pack(expand=1)  
  
 value\_var = IntVar()  
 value = 10  
 self.progressbar = ttk.Progressbar(orient="horizontal", variable=value\_var, maximum=100)  
 self.progressbar.pack(side=tk.BOTTOM, fill=tk.X)  
  
 self.label = ttk.Label(self.frame, textvariable=value\_var)  
 self.progressbar.start()  
  
 dump\_file\_path = './kurs.dump'  
 os.environ['PGPASSWORD'] = f'{password}'  
 cmd = f'pg\_restore -h {host} -p 5432 -U {user} -d kurs3 {dump\_file\_path}'  
 p = subprocess.Popen(cmd, shell=True)  
 del os.environ['PGPASSWORD']  
  
 while True:  
 self.frame.update()  
 if value\_var.get() == 99:  
 self.progressbar.stop()  
 loginSystem(win)  
 break  
  
if \_\_name\_\_ =="\_\_main\_\_":  
 win = tk.Tk()  
 start = progrload(win)  
 start.pack()  
 win.mainloop()