**ПРИЛОЖЕНИЕ В. ТЕКСТ ПРОГРАММЫ**

import tkinter as tk

from tkinter import \*

from tkinter import ttk

from PIL import Image, ImageTk

import psycopg2

from functools import partial

from materials import \*

conn = psycopg2.connect(

user="postgres",

password="SQLGD2525",

host="127.0.0.1",

port=5432,

database="ARM"

)

conn.autocommit = True

class progrload(tk.Frame):

def \_\_init\_\_(self, win):

super().\_\_init\_\_(win)

self.progbar()

def progbar(self):

win.title('АРМ: загрузка')

win.geometry('500x290+{}+{}'.format(w, h))

win.resizable(False, False)

self.frame = tk.Frame(win)

self.frame.place(relwidth=1, relheight=1)

value\_var = IntVar()

value = 10

self.loadLabel = tk.Label(self.frame, text="Загрузка! Пожалуйста, подождите!", font=("", 18))

self.loadLabel.pack(pady=20)

img = Image.open("logo.png")

self.tkimage = ImageTk.PhotoImage(img)

self.l3 = tk.Label(self.frame, image=self.tkimage)

self.l3.pack()

self.progressbar = ttk.Progressbar(orient="horizontal", variable=value\_var, maximum=100, length=460)

self.progressbar.place(x=20, y=240)

self.label = ttk.Label(self.frame, textvariable=value\_var)

self.progressbar.start()

while True:

self.frame.update()

if value\_var.get() == 1:

self.progressbar.stop()

self.frame.destroy()

break

class loginSystem(tk.Frame):

def \_\_init\_\_(self, logWin):

super().\_\_init\_\_(logWin)

self.loginSystem()

def loginSystem(self):

progrload(win)

win.title('Авторизация')

win.geometry('260x140+{}+{}'.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="Введите логин", font=10)

self.lab\_Login.place(x=58, y=0)

self.lab\_Password = tk.Label(self.frame, text="Введите пароль", font=15)

self.lab\_Password.place(x=52, y=46)

self.inputLogin = ttk.Entry(self.frame, width=40)

self.inputLogin.place(x=8, y=28)

self.inputPassword = ttk.Entry(self.frame, width=40, show="\*")

self.inputPassword.place(x=8, y=74)

self.connButton = tk.Button(self.frame, text="Войти",fg="black", bg="orange", width=10, font=('', 12), command=self.checkLogin)

self.connButton.place(x=80, y=105)

def checkLogin(self):

Polzovatel = self.inputLogin.get()

if Polzovatel == "admin" and self.inputPassword.get() == "admin":

self.destroy()

self.frame.destroy()

Main(win)

else:

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",fg="black", bg="orange", width=20, font=('', 12), command=errorWindow.destroy)

self.repeatButton.pack(side=tk.BOTTOM, pady=5)

class Main(tk.Frame):

def \_\_init\_\_(self, win):

super().\_\_init\_\_(win)

self.startMain()

def check\_connect(self):

if conn:

return

else:

self.errorWindows("Ошибка соединения", "Невозможно подключиться к БД")

exit(0)

def startMain(self):

win.title('АРМ бухгалтер по распределению заработной платы')

win.geometry('800x600+{}+{}'.format(w, h))

win.resizable(False, False)

self.frameMain = tk.Frame(win)

self.frameMain.place(relwidth=1, relheight=1)

self.mainLabel = tk.Label(self.frameMain, text="ГЛАВНОЕ МЕНЮ", font=('', 26))

self.mainLabel.place(x=255, y=52)

self.spiskiButton = tk.Button(self.frameMain, text="Справочные документы",fg="black", bg="orange", width=40, font=('', 18), command=self.spiskiApp)

self.spiskiButton.place(x=115, y=127)

self.docButton = tk.Button(self.frameMain, text="Оперативные документы",fg="black", bg="orange", width=40, font=('', 18), command=self.docApp)

self.docButton.place(x=115, y=202)

self.othetButton = tk.Button(self.frameMain, text="Отчётные документы",fg="black", bg="orange", width=40, font=('', 18), command=self.othWindowSp)

self.othetButton.place(x=115, y=277)

self.rebornButton = tk.Button(self.frameMain, text="Восстановление базы данных",fg="black", bg="orange", width=40, font=('', 18), command=self.razarch)

self.rebornButton.place(x=115, y=352)

self.spravkaApp = tk.Button(self.frameMain, text="Справка",fg="black", bg="orange", width=40, font=('', 18), command=self.spravka)

self.spravkaApp.place(x=115, y=427)

self.closeApp = tk.Button(self.frameMain, text="Выход",fg="black", bg="orange", width=40, font=('', 18), command=self.closeApp)

self.closeApp.place(x=115, y=502)

def spiskiApp(self):

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.spiskiLabel = tk.Label(self.spiskiFrame, text="МЕНЮ СПРАВОЧНЫХ ДОКУМЕНТОВ", font=('', 26))

self.spiskiLabel.place(x=88, y=53)

self.spiskButton1 = tk.Button(self.spiskiFrame, text="Список бригад",fg="black", bg="orange", width=40, font=('', 18), command=partial(self.viewDB, spisok\_brigad, "SB", "Список бригад"))

self.spiskButton1.place(x=115, y=128)

self.spiskButton2 = tk.Button(self.spiskiFrame, text="Положение по премированию",fg="black", bg="orange", width=40, font=('', 18), command=partial(self.viewDB, spisok\_premii, "PP", "Положение по премированию"))

self.spiskButton2.place(x=115, y=203)

self.spiskButton3 = tk.Button(self.spiskiFrame, text="Список сотрудников",fg="black", bg="orange", width=40, font=('', 18), command=partial(self.viewDB, spisok\_sotrudnikov, "SSB", "Список сотрудников"))

self.spiskButton3.place(x=115, y=278)

self.spiskButton4 = tk.Button(self.spiskiFrame, text="Список разрядов",fg="black", bg="orange", width=40, font=('', 18), command=partial(self.viewDB, spisok\_razradow, "SR", "Список разрядов"))

self.spiskButton4.place(x=115, y=353)

self.spiskButton4 = tk.Button(self.spiskiFrame, text="Список сеток",fg="black", bg="orange", width=40, font=('', 18), command=partial(self.viewDB, spisok\_setok, "SS", "Список сеток"))

self.spiskButton4.place(x=115, y=427)

self.closeA = tk.Button(self.spiskiFrame, text='Назад', fg="black", bg="orange", width=40, font=('', 18), command=spiskiAppWindow.destroy)

self.closeA.place(x=115, y=503)

def docApp(self):

docAppWindow = tk.Toplevel(self)

docAppWindow.title('Оперативные документы')

docAppWindow.geometry('800x600+{}+{}'.format(w, h))

docAppWindow.resizable(False, False)

self.docFrame = tk.Frame(docAppWindow)

self.docFrame.place(relwidth=1, relheight=1)

self.docLabel = tk.Label(self.docFrame, text="МЕНЮ ОПЕРАТИВНЫХ ДОКУМЕНТОВ", font=('', 26))

self.docLabel.place(x=75, y=128)

self.docButton1 = tk.Button(self.docFrame, text='Наряд на сдельную работу', fg="black", bg="orange", width=40, font=('', 18), command=partial(self.viewDB, oper\_narad, "NSR", "Наряд на сдельную работу"))

self.docButton1.place(x=115, y=203)

self.docButton2 = tk.Button(self.docFrame, text='Табель учёта фактически отработанного времени', fg="black", bg="orange", width=40, font=('', 18), command=partial(self.viewDB, oper\_tabel, "TFOW", "Табель учёта фиктически отработанного времени"))

self.docButton2.place(x=115, y=277)

self.docButton3 = tk.Button(self.docFrame, text='Протокол совета бригад', fg="black", bg="orange", width=40, font=('', 18), command=partial(self.viewDB, oper\_protokol, "PSB", "Протокол совета бригад"))

self.docButton3.place(x=115, y=352)

self.closeB = tk.Button(self.docFrame, text='Закрыть', fg="black", bg="orange", width=40, font=('', 18), command=docAppWindow.destroy)

self.closeB.place(x=115, y=427)

def othWindowSp(self):

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.othetLabel = tk.Label(self.othWindow,text="МЕНЮ ОТЧЁТНЫХ ДОКУМЕНТОВ",font=('', 26))

self.othetLabel.place(x=113, y=128)

self.otButton1 = tk.Button(self.othWindow, text="Сводная ведомость ЗП бригад",fg="black", bg="orange", width=48, font=('', 18), command=partial(self.tempO, "O1"))

self.otButton1.place(x=60, y=203)

self.otButton2 = tk.Button(self.othWindow, text="Сводная ведомость премии бригад",fg="black", bg="orange", width=48, font=('', 18), command=partial(self.tempO, "O2"))

self.otButton2.place(x=60, y=277)

self.otButton3 = tk.Button(self.othWindow, text='Ведомость распределения ЗП и премии работникам бригад', fg="black", bg="orange", width=48, font=('', 18), command=partial(self.tempO, "O3"))

self.otButton3.place(x=60, y=352)

self.closeC = tk.Button(self.othWindow, text='Закрыть', fg="black", bg="orange", width=48, font=('', 18), command=othWind.destroy)

self.closeC.place(x=60, y=427)

def spravka(self):

self.viewspravka = tk.Toplevel(self)

self.viewspravka.title('Справка')

self.viewspravka.geometry('800x480+{}+{}'.format(w, h))

self.viewspravka.resizable(False, False)

self.frameSpravka = tk.Frame(self.viewspravka)

self.frameSpravka.place(relwidth=1, relheight=1)

self.armName = tk.Label(self.frameSpravka, text="Название АРМ: ", font=("", 12))

self.armName.grid(row=0, column=0, pady=10, sticky='e')

self.\_version = tk.Label(self.frameSpravka, text=" Бухгалтер по распределению заработной платы работниам бригад", font=("", 12))

self.\_version.grid(row=0, column=1, columnspan=2, pady=10, sticky='w')

self.version = tk.Label(self.frameSpravka, text="Версия программы: ", font=("", 12))

self.version.grid(row=2, column=0, pady=10, sticky='e')

self.\_version = tk.Label(self.frameSpravka, text=" 1.0", font=("", 12))

self.\_version.grid(row=2, column=1, columnspan=2, pady=10, sticky='w')

self.razrab = tk.Label(self.frameSpravka, text="Информация о разработчике: ", font=("", 12), justify="right")

self.razrab.grid(row=4, column=0, pady=10, sticky='e')

self.\_razrab = tk.Label(self.frameSpravka, text=" Абоимов И.В. - студент БрГТУ, ФЭИС, АСОИ, группа АС-59", font=("", 12))

self.\_razrab.grid(row=4, column=1, columnspan=2, pady=10, sticky='w')

self.opisanie = tk.Label(self.frameSpravka, text="Описание: ", font=("", 12), justify="right")

self.opisanie.grid(row=6, column=0, rowspan=4, sticky='e')

self.opisanie1 = tk.Label(self.frameSpravka, text=" Данная АРМ была разработана с целью сокращения", font=("", 12))

self.opisanie1.grid(row=7, column=1, columnspan=2, sticky='w')

self.opisanie2 = tk.Label(self.frameSpravka, text=" трудозатрат по ведению информации и отчётных", font=("", 12))

self.opisanie2.grid(row=8, column=1, columnspan=2, sticky='w')

self.opisanie3 = tk.Label(self.frameSpravka, text=" документов при решении комплекса задач по", font=("", 12))

self.opisanie3.grid(row=9, column=1, columnspan=2, sticky='w')

self.opisanie4 = tk.Label(self.frameSpravka, text=" распределению заработной платы работникам бригад", font=("", 12))

self.opisanie4.grid(row=10, column=1, columnspan=2, sticky='w')

self.hollow = tk.Label(self.frameSpravka, text='.')

self.hollow.grid(row=11, column=0, sticky='w', pady=5)

self.drugaya = tk.Label(self.frameSpravka, text="Другая справочная информация: ", font=("", 12), justify="right")

self.drugaya.grid(row=13, column=0, sticky='e')

self.drugaya1 = tk.Label(self.frameSpravka, text=" Данная программа разработана для решения", font=("", 12))

self.drugaya1.grid(row=12, column=1, columnspan=2, sticky='w')

self.drugaya2 = tk.Label(self.frameSpravka, text=" проблемы расчёта и распределения заработной", font=("", 12))

self.drugaya2.grid(row=13, column=1, columnspan=2, sticky='w')

self.drugaya3 = tk.Label(self.frameSpravka, text=" платы и премии работникам бригад", font=("", 12))

self.drugaya3.grid(row=14, column=1, columnspan=2, sticky='w')

self.closeApp = tk.Button(self.frameSpravka, text="Назад",fg="black", bg="orange", width=40, font=('', 18), command=self.viewspravka.destroy)

self.closeApp.place(x=115, y=380)

def tempO(self, ot):

self.topO = tk.Toplevel(self)

self.topO.title("Выбор отчёта")

self.topO.geometry("260x140+{}+{}".format(w, h))

self.topO.resizable(False, False)

self.Oframe = tk.Frame(self.topO)

self.lab\_brigade = tk.Label(self.topO, text="Бригада", font=10)

self.lab\_brigade.place(x=89, y=0)

self.lab\_mounth = tk.Label(self.topO, text="Месяц", font=15)

self.lab\_mounth.place(x=97, y=46)

self.inputbrigade = ttk.Entry(self.topO, width=40)

self.inputbrigade.place(x=8, y=28)

self.inputmounth = ttk.Entry(self.topO, width=40)

self.inputmounth.place(x=8, y=74)

self.late = tk.Button(self.topO, text="Расчёт",fg="black", bg="orange", width=10, font=('', 12), command=partial(self.checkdata, ot))

self.late.place(x=80, y=105)

def checkdata(self, ot):

try:

brigade = self.inputbrigade.get()

mounth = self.inputmounth.get()

self.topO.destroy()

if ot == "O1":

self.create\_O1(brigade, mounth)

elif ot == "O2":

self.create\_O2(brigade, mounth)

elif ot == "O3":

self.create\_O3(brigade, mounth)

except Exception as \_ex:

self.errorWindows("Ошибка ввода.", "Неккоретные данные")

return

def create\_O1(self, brigade, mounth):

nsr\_data = []

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "NSR" WHERE brigade\_code='{brigade}' AND mounth='{mounth}' """)

nsr\_data = [row for row in cursor.fetchall()]

if len(nsr\_data) == 0:

raise Exception

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В НСР нет необходимых данных", "Продолжить")

return

pp\_data = []

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "PP" WHERE brigade\_code='{brigade}' """)

pp\_data = [row for row in cursor.fetchall()]

if len(pp\_data) == 0:

raise Exception

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В ПП нет необходимых данных", "Продолжить")

return

svod\_zp = 0

data = []

for pp in pp\_data:

temp = [f'{pp[1]}', 0, '', 0]

for nsr in nsr\_data:

temp[1] = nsr[5]

temp[2] = nsr[6]

if (nsr[4] == pp[1]):

temp[3] = temp[3] + nsr[12]

svod\_zp = svod\_zp + temp[3]

data.append(tuple(temp))

svod\_zp = round(svod\_zp \* 100) / 100

if svod\_zp < 0:

self.errorWindows("Ошибка расчёта", "Не удалось произвести расчёт", "Продолжить")

return

self.topO1table = tk.Toplevel(self)

self.topO1table.title(f"Сводная ведомость распределения заработной платы бригады")

screen\_width = self.topO1table.winfo\_screenwidth()

screen\_height = self.topO1table.winfo\_screenheight()

self.topO1table.geometry(f'{screen\_width}x{screen\_height}')

self.topO1table.rowconfigure(index=0, weight=1)

self.topO1table.columnconfigure(index=0, weight=1)

self.topO1table.resizable(False, False)

self.\_viewO1 = tk.Frame(self.topO1table)

self.\_viewO1.place(relwidth=1, relheight=1)

self.tableLableO1 = tk.Label(self.\_viewO1, text='СВОДНАЯ ВЕДОМОСТЬ ЗП БРИГАДЫ', font=('', 26))

self.tableLableO1.pack(pady=10)

column\_names = ot\_sved\_zp

self.treeO1 = ttk.Treeview(self.\_viewO1, height=25, columns=column\_names, show="headings")

self.treeO1.pack(fill=X)

total\_width = 0

for i in column\_names:

self.treeO1.heading(f"{i}", text=f"{i}")

if i == '№':

self.treeO1.column(f"{i}", stretch=False)

self.treeO1.column(f"{i}", width=50)

total\_width += 50

else:

column\_width = screen\_width // len(column\_names)

self.treeO1.column(f"{i}", width=column\_width, stretch=True)

total\_width += column\_width

for row in data:

self.treeO1.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.treeO1.column(column\_names[i], width=max\_width + 20, anchor=CENTER)

self.svod\_zp\_L = tk.Label(self.\_viewO1, text=f"Итоговая Заработная плата = {svod\_zp} рублей", bd=0, justify=RIGHT, font=('', 18))

self.svod\_zp\_L.place(x=10, y=610)

self.closeO1 = tk.Button(self.\_viewO1, text='Закрыть', fg="black", bg="orange", width=56, font=('', 18), command=self.topO1table.destroy)

self.closeO1.place(x=379, y=700)

def create\_O2(self, brigade, mounth):

nsr\_data = []

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "NSR" WHERE brigade\_code='{brigade}' AND mounth='{mounth}' """)

nsr\_data = [row for row in cursor.fetchall()]

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В НСР нет необходимых данных", "Продолжить")

return

pp\_data = []

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "PP" WHERE brigade\_code='{brigade}' """)

pp\_data = [row for row in cursor.fetchall()]

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В НСР нет необходимых данных", "Продолжить")

return

svod\_prem = 0

data = []

for pp in pp\_data:

temp = [f'{pp[1]}', 0, '', 0, 0, 0]

for nsr in nsr\_data:

temp[1] = nsr[5]

temp[2] = nsr[6]

if (nsr[4] == pp[1]):

temp[3] = temp[3] + nsr[12]

temp[4] = pp[3]

temp[5] = round(float(temp[4]) \* float(temp[3])) / 100

svod\_prem = svod\_prem + temp[5]

data.append(tuple(temp))

svod\_prem = round(svod\_prem \* 100) / 100

if svod\_prem < 0:

self.errorWindows("Ошибка расчёта", "Не удалось произвести расчёт", "Продолжить")

return

self.topO2table = tk.Toplevel(self)

self.topO2table.title(f"Сводная ведомость распределения заработной платы бригады")

screen\_width = self.topO2table.winfo\_screenwidth()

screen\_height = self.topO2table.winfo\_screenheight()

self.topO2table.geometry(f'{screen\_width}x{screen\_height}')

self.topO2table.rowconfigure(index=0, weight=1)

self.topO2table.columnconfigure(index=0, weight=1)

self.topO2table.resizable(False, False)

self.\_viewO2 = tk.Frame(self.topO2table)

self.\_viewO2.place(relwidth=1, relheight=1)

self.tableLableO2 = tk.Label(self.\_viewO2, text='СВОДНАЯ ВЕДОМОСТЬ ПРЕМИИ БРИГАДЫ', font=('', 26))

self.tableLableO2.pack(pady=10)

column\_names = ot\_sved\_prem

self.treeO2 = ttk.Treeview(self.\_viewO2, height=25, columns=column\_names, show="headings")

self.treeO2.pack(fill=X)

total\_width = 0

for i in column\_names:

self.treeO2.heading(f"{i}", text=f"{i}")

if i == '№':

self.treeO2.column(f"{i}", stretch=False)

self.treeO2.column(f"{i}", width=50)

total\_width += 50

else:

column\_width = screen\_width // len(column\_names)

self.treeO2.column(f"{i}", width=column\_width, stretch=True)

total\_width += column\_width

for row in data:

self.treeO2.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.treeO2.column(column\_names[i], width=max\_width + 20, anchor=CENTER)

self.svod\_prem\_L = tk.Label(self.\_viewO2, text=f"Итоговая Премия = {svod\_prem} рублей",bd=0, justify=RIGHT, font=('', 18))

self.svod\_prem\_L.place(x=10, y=610)

self.closeO2 = tk.Button(self.\_viewO2, text='Закрыть', fg="black", bg="orange", width=56, font=('', 18), command=self.topO2table.destroy)

self.closeO2.place(x=379, y=700)

def create\_O3(self, brigade, mounth):

psb\_data = []

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "PSB" WHERE brigade\_code='{brigade}' AND mounth='{mounth}' """)

psb\_data = [row for row in cursor.fetchall()]

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В ПСБ нет необходимых данных", "Продолжить")

return

nsr\_data = []

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "NSR" WHERE brigade\_code='{brigade}' AND mounth='{mounth}' """)

nsr\_data = [row for row in cursor.fetchall()]

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В НСР нет необходимых данных", "Продолжить")

return

pp\_data = []

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "PP" WHERE brigade\_code='{brigade}' """)

pp\_data = [row for row in cursor.fetchall()]

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В ПП нет необходимых данных", "Продолжить")

return

sum\_prem = 0

data = []

for pp in pp\_data:

for nsr in nsr\_data:

if (nsr[4] == pp[1]):

sum\_prem = sum\_prem + round(nsr[12] \* pp[3]) / 100

sum\_prem = round(sum\_prem \* 100) / 100

sum\_nsr = 0

for nsr in nsr\_data:

sum\_nsr = sum\_nsr + nsr[12]

Sb = 300

Br = 22.5

Tr = 8.4

S0 = float(Sb) / Br / Tr

tdata = []

for psb in psb\_data:

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "SS" WHERE grid='{psb[6]}' """)

grid\_r = [row for row in cursor.fetchall()]

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В СС нет необходимых данных", "Продолжить")

return

grid\_ratio = grid\_r[0][1]

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "SR" WHERE rank={psb[5]} """)

rank\_r = [row for row in cursor.fetchall()]

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В СР нет необходимых данных", "Продолжить")

return

rank\_ratio = rank\_r[0][1]

Tar = round((S0 \* grid\_ratio \* rank\_ratio \* psb[8]) \* 100) / 100

temp = [f'{psb[2]}', f'{psb[3]}', f'{psb[4]}', psb[5], f'{psb[6]}', psb[7], '-', psb[8], 0, Tar, 0, 0]

tdata.append(temp)

sum\_Tar = 0

for datum in tdata:

sum\_Tar = sum\_Tar + datum[9]

oPrir = round((sum\_nsr - sum\_Tar) \* 100) / 100

p = 0

for datum in tdata:

p = p + datum[5] \* datum[9]

p = oPrir / p

sum\_Prir = 0

if oPrir < 0:

self.errorWindows("Ошибка расчёта.", "Сумма по приработкам < 0", "Продолжить")

return

elif oPrir > 0:

for datum in tdata:

datum[8] = p \* datum[5] \* datum[9]

datum[8] = round(datum[8] \* 100) / 100

sum\_Prir = sum\_Prir + datum[8]

elif oPrir == 0:

for datum in tdata:

datum[8] = 0

sum\_Prir = round(sum\_Prir \* 100) / 100

if oPrir - sum\_Prir > 0:

max\_ktu = 0

for i in range(0, len(tdata)):

if tdata[i][5] > max\_ktu:

max\_ktu = tdata[i][5]

best\_men = []

for i in range(0, len(tdata)):

if tdata[i][5] == max\_ktu:

best\_men.append(i)

out = True

while out:

for i in best\_men:

tdata[i][8] = tdata[i][8] + 0.01

tdata[i][8] = round(tdata[i][8] \* 100) / 100

sum\_Prir = round((sum\_Prir + 0.01) \* 100) /100

if oPrir - sum\_Prir == 0:

out = False

break

elif oPrir - sum\_Prir < 0:

min\_ktu = 9999

for i in range(0, len(tdata)):

if tdata[i][5] < min\_ktu:

min\_ktu = tdata[i][5]

worse\_men = []

for i in range(0, len(tdata)):

if tdata[i][5] == min\_ktu:

worse\_men.append(i)

out = True

while out:

for i in worse\_men:

tdata[i][8] = tdata[i][8] - 0.01

tdata[i][8] = round(tdata[i][8] \* 100) / 100

sum\_Prir = round((sum\_Prir - 0.01) \* 100) /100

if oPrir - sum\_Prir == 0:

out = False

break

all\_zp = 0

all\_prem = 0

for datum in tdata:

datum[10] = datum[8] + datum[9]

datum[10] = round(datum[10] \* 100) / 100

pp = []

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "PP" WHERE brigade\_code='{brigade}' AND profession\_code='{datum[2]}' """)

pp = [row for row in cursor.fetchall()]

except Exception as \_ex:

self.errorWindows("Ошибка обращения.", "В ПП нет необходимых данных", "Продолжить") return

datum[11] = round(datum[10] \* pp[0][3])

datum[11] = datum[11] / 100

all\_zp = all\_zp + datum[10]

all\_prem = all\_prem + datum[11]

all\_zp = round(all\_zp \* 100) / 100

all\_prem = round(all\_prem \* 100) / 100

if all\_zp - sum\_nsr != 0:

self.errorWindows("Ошибка расчёта", "Не удалось произвести расчёт", "Продолжить")

return

data = []

for datum in tdata:

data.append(tuple(datum))

self.topO3table = tk.Toplevel(self)

self.topO3table.title(f"Сводная ведомость распределения заработной платы бригады")

screen\_width = self.topO3table.winfo\_screenwidth()

screen\_height = self.topO3table.winfo\_screenheight()

self.topO3table.geometry(f'{screen\_width}x{screen\_height}')

self.topO3table.rowconfigure(index=0, weight=1)

self.topO3table.columnconfigure(index=0, weight=1)

self.topO3table.resizable(False, False)

self.\_viewO3 = tk.Frame(self.topO3table)

self.\_viewO3.place(relwidth=1, relheight=1)

self.tableLableO3 = tk.Label(self.\_viewO3, text='ВЕДОМОСТЬ РАСПРЕДЕЛЕНИЯ ЗП И ПРЕМИИ СОТРУДНИКАМ БРИГАД', font=('', 26))

self.tableLableO3.pack(pady=10)

column\_names = ot\_ved

self.treeO3 = ttk.Treeview(self.\_viewO3, height=21, columns=column\_names, show="headings")

self.treeO3.pack(fill=X)

total\_width = 0

for i in column\_names:

self.treeO3.heading(f"{i}", text=f"{i}")

if i == '№':

self.treeO3.column(f"{i}", stretch=False)

self.treeO3.column(f"{i}", width=50)

total\_width += 50

else:

column\_width = screen\_width // len(column\_names)

self.treeO3.column(f"{i}", width=column\_width, stretch=True)

total\_width += column\_width

for row in data:

self.treeO3.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.treeO3.column(column\_names[i], width=max\_width + 20, anchor=CENTER)

self.itog\_zp = tk.Label(self.\_viewO3, text=f"Итоговая Заработная плата = {all\_zp} рублей",bd=0, justify=RIGHT, font=('', 18))

self.itog\_zp.place(x=10, y=560)

self.itog\_prem = tk.Label(self.\_viewO3, text=f"Итоговая Премия = {all\_prem} рублей",

bd=0, justify=RIGHT, font=('', 18))

self.itog\_prem.place(x=10, y=610)

self.closeO3 = tk.Button(self.\_viewO3, text='Закрыть', fg="black", bg="orange", width=56, font=('', 18), command=self.topO3table.destroy)

self.closeO3.place(x=379, y=700)

def viewDB(self, column\_names, tablename, tablenamerus):

self.viewTableDataBases = tk.Toplevel(self)

self.viewTableDataBases.title(f"{tablenamerus}")

screen\_width = self.viewTableDataBases.winfo\_screenwidth()

screen\_height = self.viewTableDataBases.winfo\_screenheight()

self.viewTableDataBases.geometry(f'{screen\_width}x{screen\_height}')

self.viewTableDataBases.rowconfigure(index=0, weight=1)

self.viewTableDataBases.columnconfigure(index=0, weight=1)

self.viewTableDataBases.resizable(False, False)

self.\_viewDB = tk.Frame(self.viewTableDataBases)

self.\_viewDB.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:

self.viewTableDataBases.destroy()

self.errorWindows("Ошибка соединения", "Таблица не загрузилась")

return

tabletitle = ''

if tablename == 'SB':

tabletitle = 'СПИСОК БРИГАД'

elif tablename == 'SSB':

tabletitle = 'СПИСОК СОТРУДНИКОВ БРИГАДЫ'

elif tablename == 'PP':

tabletitle = 'ПОЛОЖЕНИЕ ПО ПРЕМИРОВАНИЮ'

elif tablename == 'SS':

tabletitle = 'СПИСОК СЕТОК'

elif tablename == 'SR':

tabletitle = 'СПИСОК РАЗРЯДОВ'

elif tablename == 'NSR':

tabletitle = 'НАРЯД НА СДЕЛЬНУЮ РАБОТУ'

elif tablename == 'PSB':

tabletitle = 'ПРОТОКОЛ СОВЕТА БРИГАД'

elif tablename == 'TFOW':

tabletitle = 'ТАБЕЛЬ УЧЁТА ФАКТИЧЕСКИ ОТРАБОТАННОГО ВРЕМЕНИ'

self.tableLable = tk.Label(self.\_viewDB, text=f'{tabletitle}', font=('', 26))

self.tableLable.pack(pady=10)

self.tree = ttk.Treeview(self.\_viewDB, height=30, columns=column\_names, show="headings")

self.tree.pack(fill=X, pady=5)

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

data.reverse()

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)

self.inputButton = tk.Button(self.\_viewDB, text="Добавить",fg="black", bg="orange", width=15, font=('', 18), command=partial(self.inputTableWindows, column\_names, tablename, tablenamerus))

self.inputButton.place(x=100, y=740)

if tablename == "SR":

self.inputButton["state"] = "disable"

self.inputButton["bg"] = "gray"

self.changeButton = tk.Button(self.\_viewDB, text="Изменить",fg="black", bg="orange", width=15, font=('', 18), command=partial(self.inputTableWindows, column\_names, tablename, tablenamerus, 'CHANGE'))

self.changeButton.place(x=379, y=740)

if tablename == "SR" or tablename == "SS"\

or tablename == "NSR" or tablename == "TFOW" or tablename == "PSB":

self.changeButton["state"] = "disable"

self.changeButton["bg"] = "gray"

self.deleteButton = tk.Button(self.\_viewDB, text="Удаление",fg="black", bg="orange", width=15, font=('', 18), command=partial(self.DELButton, column\_names, tablename, tablenamerus))

self.deleteButton.place(x=658, y=740)

if tablename == "SR" or tablename == "SS":

self.deleteButton["state"] = "disable"

self.deleteButton["bg"] = "gray"

self.searchButton = tk.Button(self.\_viewDB, text="Поиск",fg="black", bg="orange", width=15, font=('', 18), command=partial(self.serCH, tablename))

self.searchButton.place(x=938, y=740)

if tablename == "SR":

self.searchButton["state"] = "disable"

self.searchButton["bg"] = "gray"

self.closeButton = tk.Button(self.\_viewDB, text="Закрыть",fg="black", bg="orange", width=15, font=('', 18), command=self.viewTableDataBases.destroy)

self.closeButton.place(x=1215, y=740)

def inputTableWindows(self, column\_names, tablename, tablenamerus, mode="INPUT"):

selection = ''

if mode == 'CHANGE':

selection = self.tree.selection()

if len(selection) == 0:

self.errorWindows("Ошибка", "Запись не выбрана")

return

self.inputTableWin = tk.Toplevel(self)

self.inputTableWin.resizable(False, False)

self.inTable = tk.Frame(self.inputTableWin)

self.inTable.place(relwidth=1, relheight=1)

if tablename == "SS":

self.inputTableWin.title("Добавление в CC")

self.inputTableWin.geometry('400x302')

buflist = spisok\_setok

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)

elif tablename == "PP":

if mode == "CHANGE":

self.inputTableWin.title("Изменение записи ПП")

else:

self.inputTableWin.title("Добавление в ПП")

self.inputTableWin.geometry('400x460')

buflist = spisok\_premii

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)

if mode == 'CHANGE':

t = self.tree.item(selection[0], "values")[0]

self.l1e.insert(0, f"{t}")

t = self.tree.item(selection[0], "values")[1]

self.l2e.insert(0, f"{t}")

t = self.tree.item(selection[0], "values")[2]

self.l3e.insert(0, f"{t}")

t = self.tree.item(selection[0], "values")[3]

self.l4e.insert(0, f"{t}")

elif tablename == "SB":

if mode == "CHANGE":

self.inputTableWin.title("Изменение записи Сб")

else:

self.inputTableWin.title("Добавление в СБ")

self.inputTableWin.geometry('400x375')

buflist = spisok\_brigad

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)

if mode == 'CHANGE':

t = self.tree.item(selection[0], "values")[0]

self.l1e.insert(0, f"{t}")

t = self.tree.item(selection[0], "values")[1]

self.l2e.insert(0, f"{t}")

t = self.tree.item(selection[0], "values")[2]

self.l3e.insert(0, f"{t}")

elif tablename == "SSB":

if mode == "CHANGE":

self.inputTableWin.title("Изменение записи ССБ")

else:

self.inputTableWin.title("Добавление в ССБ")

self.inputTableWin.geometry('400x605')

buflist = spisok\_sotrudnikov

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)

if mode == 'CHANGE':

t = self.tree.item(selection[0], "values")[0]

self.l1e.insert(0, f"{t}")

t = self.tree.item(selection[0], "values")[1]

self.l2e.insert(0, f"{t}")

t = self.tree.item(selection[0], "values")[2]

self.l3e.insert(0, f"{t}")

elif tablename == "NSR":

self.inputTableWin.title("Добавление в НСР")

self.inputTableWin.geometry('400x740')

buflist = oper\_narad

self.l1 = tk.Label(self.inTable, text=f"{buflist[0]}",

bd=0, justify=CENTER, height=1, 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=1, 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=1, 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=1, 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=1, 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=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.l7 = tk.Label(self.inTable, text=f"{buflist[6]}",

bd=0, justify=CENTER, height=1, 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=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.l9 = tk.Label(self.inTable, text=f"{buflist[8]}",

bd=0, justify=CENTER, height=1, 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.l10 = tk.Label(self.inTable, text=f"{buflist[10]}",

bd=0, justify=CENTER, height=1, font=('', 18))

self.l10.pack(side=tk.TOP, fill=tk.X)

self.l10e = ttk.Entry(self.inTable, width=15)

self.l10e.pack(fill=tk.X)

self.l11 = tk.Label(self.inTable, text=f"{buflist[11]}",

bd=0, justify=CENTER, height=1, font=('', 18))

self.l11.pack(side=tk.TOP, fill=tk.X)

self.l11e = ttk.Entry(self.inTable, width=15)

self.l11e.pack(fill=tk.X)

self.l12 = tk.Label(self.inTable, text=f"{buflist[13]}",

bd=0, justify=CENTER, height=1, font=('', 18))

self.l12.pack(side=tk.TOP, fill=tk.X)

self.l12e = ttk.Entry(self.inTable, width=15)

self.l12e.pack(fill=tk.X)

elif tablename == "PSB":

self.inputTableWin.title("Добавление в ПСБ")

self.inputTableWin.geometry('400x590')

buflist = oper\_protokol

self.l1 = tk.Label(self.inTable, text=f"{buflist[0]}",

bd=0, justify=CENTER, height=1, 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=1, 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=1, 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=1, 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=1, 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=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.l7 = tk.Label(self.inTable, text=f"{buflist[6]}",

bd=0, justify=CENTER, height=1, 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=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.l9 = tk.Label(self.inTable, text=f"{buflist[8]}",

bd=0, justify=CENTER, height=1, 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)

elif tablename == "TFOW":

self.inputTableWin.title("Добавление в ТФОВ")

self.inputTableWin.geometry('400x640')

buflist = oper\_tabel

self.l1 = tk.Label(self.inTable, text=f"{buflist[0]}",

bd=0, justify=CENTER, height=1, 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=1, 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=1, 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=1, 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=1, 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=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.l7 = tk.Label(self.inTable, text=f"{buflist[6]}",

bd=0, justify=CENTER, height=1, 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=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.l9 = tk.Label(self.inTable, text=f"{buflist[8]}",

bd=0, justify=CENTER, height=1, 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.l10 = tk.Label(self.inTable, text=f"{buflist[9]}",

bd=0, justify=CENTER, height=1, font=('', 18))

self.l10.pack(side=tk.TOP, fill=tk.X)

self.l10e = ttk.Entry(self.inTable, width=15)

self.l10e.pack(fill=tk.X)

if mode == 'CHANGE':

self.changeButton = tk.Button(self.inTable, text="Изменить",fg="black", bg="orange", width=18, font=('', 15), command=partial(self.change, selection, column\_names, tablename, tablenamerus))

self.changeButton.pack(pady=20)

else:

self.inputButton = tk.Button(self.inTable, text="Добавить",fg="black", bg="orange", width=18, font=('', 15), command=partial(self.inputTableSQL, column\_names, tablename, tablenamerus))

self.inputButton.pack(pady=20)

self.closeB = tk.Button(self.inTable, text='Закрыть', fg="black", bg="orange", width=18, font=('', 15), command=self.inputTableWin.destroy)

self.closeB.pack()

def inputTableSQL(self, column\_names, tablename, tablenamerus):

if tablename == "SS":

try:

value1 = self.l1e.get()

value2 = float(self.l2e.get())

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("grid", "grid\_ratio") VALUES ('{value1}', {value2}) """)

self.errorWindows("Добавление в СС", "Добавление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

self.inputTableWin.destroy()

except Exception as \_ex:

self.errorWindows("Ошибка добавления", "Не удалось добавить запись")

return

if tablename == "PP":

try:

value1 = self.l1e.get()

value2 = self.l2e.get()

value3 = int(self.l3e.get())

value4 = int(self.l4e.get())

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("brigade\_code", "profession\_code", "percent\_per\_type\_null", "percent\_per\_type\_one") VALUES ('{value1}', '{value2}', {value3}, {value4}) """)

self.errorWindows("Добавление в ПП", "Добавление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

self.inputTableWin.destroy()

except Exception as \_ex:

self.errorWindows("Ошибка добавления", "Не удалось добавить запись")

return

if tablename == "SB":

try:

value1 = self.l1e.get()

value2 = self.l2e.get()

value3 = int(self.l3e.get())

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("brigade\_code", "sn\_in\_brigadier", "person\_quantity") VALUES ('{value1}', '{value2}', {value3}) """)

self.errorWindows("Добавление в СБ", "Добавление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

self.inputTableWin.destroy()

except Exception as \_ex:

self.errorWindows("Ошибка добавления", "Не удалось добавить запись")

return

if tablename == "SSB":

try:

value1 = self.l1e.get()

value2 = self.l2e.get()

value3 = self.l3e.get()

value4 = self.l4e.get()

value5 = int(self.l5e.get())

value6 = self.l6e.get()

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("brigade\_code", "service\_number", "profession\_code", "sn\_in", "rank", "grid") VALUES ('{value1}', '{value2}', '{value3}', '{value4}', {value5}, '{value6}') """)

self.errorWindows("Добавление в ССБ", "Добавление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

self.inputTableWin.destroy()

except Exception as \_ex:

self.errorWindows("Ошибка добавления", "Не удалось добавить запись")

return

if tablename == 'NSR':

try:

value1 = int(self.l1e.get())

value2 = self.l2e.get()

value3 = self.l3e.get()

value4 = self.l4e.get()

value5 = self.l5e.get()

value6 = int(self.l6e.get())

value7 = self.l7e.get()

value8 = int(self.l8e.get())

value9 = float(self.l9e.get())

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "SS" WHERE grid='{value7}' """)

grid\_r = [row for row in cursor.fetchall()]

if len(grid\_r) == 0:

raise Exception

except Exception as \_ex:

self.errorWindows("Ошибка сетки", "Указана несуществующая сетка")

return

grid\_ratio = grid\_r[0][1]

try:

with conn.cursor() as cursor:

cursor.execute(f"""SELECT \* FROM "SR" WHERE rank={value6} """)

rank\_r = [row for row in cursor.fetchall()]

if len(rank\_r) == 0:

raise Exception

except Exception as \_ex:

self.errorWindows("Ошибка разряда", "Указан несуществующий разряд")

return

rank\_ratio = rank\_r[0][1]

try:

value10 = round(grid\_ratio \* rank\_ratio \* value9 \* 100) / 100

value11 = int(self.l10e.get())

value12 = int(self.l11e.get())

if value12 > value11:

raise Exception

value13 = round(float(value10 \* value11) \* 100 / value8) / 100

value14 = self.l12e.get()

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("number", "brigade\_code", "detail", "operation\_number", "profession\_code", "rank", "grid", "meter", "hours", "price", "out\_detail\_quantity", "in\_detail\_quantity", "total\_sum", "mounth") VALUES ({value1}, '{value2}', '{value3}', '{value4}', '{value5}', {value6}, '{value7}', {value8}, {value9}, {value10}, {value11}, {value12}, {value13}, '{value14}') """)

self.errorWindows("Добавление в НСР", "Добавление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

self.inputTableWin.destroy()

except Exception as \_ex:

self.errorWindows("Ошибка добавления", "Не удалось добавить данные")

return

if tablename == "PSB":

try:

value1 = self.l1e.get()

value2 = self.l2e.get()

value3 = self.l3e.get()

value4 = self.l4e.get()

value5 = self.l5e.get()

value6 = int(self.l6e.get())

value7 = self.l7e.get()

value8 = float(self.l8e.get())

value9 = float(self.l9e.get())

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("mounth", "brigade\_code", "service\_number", "sn\_in" ,"profession\_code", "rank", "grid", "KTU", "work\_time\_hours") VALUES ('{value1}', '{value2}', '{value3}', '{value4}', '{value5}', {value6}, '{value7}', {value8}, {value9}) """)

self.errorWindows("Добавление в ПСБ", "Добавление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

self.inputTableWin.destroy()

except Exception as \_ex:

self.errorWindows("Ошибка добавления", "Не удалось добавить запись")

return

if tablename == "TFOW":

try:

value1 = self.l1e.get()

value2 = self.l2e.get()

value3 = self.l3e.get()

value4 = self.l4e.get()

value5 = self.l5e.get()

value6 = int(self.l6e.get())

value7 = self.l7e.get()

value8 = self.l8e.get()

value9 = self.l9e.get()

value10 = float(self.l10e.get())

work\_day\_time = 8.4

value11 = value10 / work\_day\_time

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("mounth", "brigade\_code", "service\_number", "sn\_in" ,"profession\_code", "rank", "grid", "oklad", "graph\_code", "work\_time\_hours", "work\_time\_days") VALUES ('{value1}', '{value2}', '{value3}', '{value4}', '{value5}', {value6}, '{value7}', '{value8}', '{value9}', {value10}, {value11}) """)

self.errorWindows("Добавление в ТФОВ", "Добавление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

self.inputTableWin.destroy()

def serCH(self, tablename):

val = s\_searhComboboxList[tablename]

self.serTable = tk.Toplevel(self)

self.serTable.title("Поиск")

self.serTable.geometry('300x160')

self.serTable.resizable(False, False)

self.sTable = tk.Frame(self.serTable)

self.sTable.place(relheight=1, relwidth=1)

self.serPole = tk.Label(self.sTable, text='Выберите поле', font=("", 12))

self.serPole.pack(anchor=N, pady=3)

self.combobox = ttk.Combobox(self.sTable, values=val, width=40)

self.combobox.pack(anchor=N, pady=3)

self.serPole = tk.Label(self.sTable, text='Введите искомое значение', font=("", 12))

self.serPole.pack(pady=3)

self.temp = ttk.Entry(self.sTable, width=15)

self.temp.pack(fill=X, pady=3)

self.poisk = tk.Button(self.sTable, text="поиск",fg="black", bg="orange", width=15, font=('', 12), command=partial(self.serBD, tablename))

self.poisk.place(x=75, y=118)

def serBD(self, tablename):

a1 = self.combobox.get()

a2 = self.temp.get()

if a1 == '' or a2 == '':

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

self.table = tk.Toplevel(self)

self.table.title("Искомые значения")

screen\_width = self.table.winfo\_screenwidth()

self.table.geometry(f'{screen\_width}x600')

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 = int(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("Ошибка поиска", "Не удалось произвести операцию поиска")

self.table.destroy()

column\_names = searhComboboxList1[tablename]

self.tableLable = tk.Label(self.dtable, text=f'РЕЗУЛЬТАТЫ ПОИСКА', font=('', 26))

self.tableLable.pack(pady=10)

self.TBL = ttk.Treeview(self.dtable, height=22, columns=column\_names, show="headings")

self.TBL.pack(fill=X)я

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.closeButton = tk.Button(self.dtable, text="Закрыть",justify=CENTER, fg="black", bg="orange", width=40, font=('', 20), command=self.out\_ser)

self.closeButton.place(x=460, y=540)

def refresh(self, column\_names, tablename, tablenamerus):

self.viewTableDataBases.destroy()

self.viewDB(column\_names, tablename, tablenamerus)

def errorWindows(self, msg1, msg2, msg3="Повторить"):

errorWindow = tk.Toplevel(self)

errorWindow.title(f"{msg1}")

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=f"{msg2}",font=('', 14))

self.errorLabel.pack(expand=1, pady=35)

self.repeatButton = tk.Button(self.errorWindowFrame, text=f'{msg3}', fg="black", bg="orange", width=20, font=('', 12), command=errorWindow.destroy)

self.repeatButton.pack(side=tk.BOTTOM, pady=10)

def DELButton(self, column\_names, tablename, tablenamerus):

selection = self.tree.selection()

if len(selection) == 0:

self.errorWindows("Ошибка", "Запись не выбрана")

return

if tablename == "PP":

for item in selection:

delValue1 = self.tree.item(item, "values")[0]

delValue2 = self.tree.item(item, "values")[1]

try:

with conn.cursor() as cursor:

cursor.execute(f"""DELETE FROM public."PP" WHERE brigade\_code='{delValue1}' AND profession\_code='{delValue2}' """)

self.errorWindows("Удаление из ПП", "Удаление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

self.errorWindows("Ошибка удаления", "Невозможно удалить запись")

if tablename == "SB":

for item in selection:

delValue1 = self.tree.item(item, "values")[0]

try:

with conn.cursor() as cursor:

cursor.execute(f"""DELETE FROM public."SB" WHERE brigade\_code='{delValue1}' """)

self.errorWindows("Удаление из СБ", "Удаление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

self.errorWindows("Ошибка удаления", "Невозможно удалить запись")

if tablename == "SSB":

for item in selection:

delValue1 = self.tree.item(item, "values")[0]

delValue2 = self.tree.item(item, "values")[1]

delValue3 = self.tree.item(item, "values")[2]

try:

with conn.cursor() as cursor:

cursor.execute(f"""DELETE FROM public."SSB" WHERE brigade\_code='{delValue1}' AND service\_number='{delValue2}' AND profession\_code='{delValue3}' """)

self.errorWindows("Удаление из ССБ", "Удаление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

self.errorWindows("Ошибка удаления", "Невозможно удалить запись")

if tablename == "NSR":

for item in selection:

delValue1 = self.tree.item(item, "values")[0]

delValue2 = self.tree.item(item, "values")[1]

delValue3 = self.tree.item(item, "values")[4]

try:

with conn.cursor() as cursor:

cursor.execute(f"""DELETE FROM public."NSR" WHERE number='{delValue1}' AND brigade\_code='{delValue2}' AND profession\_code='{delValue3}' """)

self.errorWindows("Удаление из НСР", "Удаление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

self.errorWindows("Ошибка удаления", "Невозможно удалить запись")

if tablename == "PSB":

for item in selection:

delValue1 = self.tree.item(item, "values")[0]

delValue2 = self.tree.item(item, "values")[1]

delValue3 = self.tree.item(item, "values")[2]

delValue4 = self.tree.item(item, "values")[4]

try:

with conn.cursor() as cursor:

cursor.execute(f"""DELETE FROM public."PSB" WHERE mounth='{delValue1}' AND brigade\_code='{delValue2}' AND service\_number='{delValue3}' AND profession\_code='{delValue4}' """)

self.errorWindows("Удаление из ПСБ", "Удаление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

self.errorWindows("Ошибка удаления", "Невозможно удалить запись")

if tablename == "TFOW":

for item in selection:

delValue1 = self.tree.item(item, "values")[0]

delValue2 = self.tree.item(item, "values")[1]

delValue3 = self.tree.item(item, "values")[2]

delValue4 = self.tree.item(item, "values")[4]

try:

with conn.cursor() as cursor:

cursor.execute(f"""DELETE FROM public."TFOW" WHERE mounth='{delValue1}' AND brigade\_code='{delValue2}' AND service\_number='{delValue3}' AND profession\_code=' {delValue4}' """)

self.errorWindows("Удаление из ТФОВ", "Удаление прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

self.errorWindows("Ошибка удаления", "Невозможно удалить запись")

def change(self, selection, column\_names, tablename, tablenamerus,):

if tablename == "SSB":

old\_data = [0, 0, 0, 0, 0, 0]

try:

value1 = self.l1e.get()

value2 = self.l2e.get()

value3 = self.l3e.get()

value4 = self.l4e.get()

value5 = int(self.l5e.get())

value6 = self.l6e.get()

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

old\_data[0] = self.tree.item(selection[0], "values")[0]

old\_data[1] = self.tree.item(selection[0], "values")[1]

old\_data[2] = self.tree.item(selection[0], "values")[2]

old\_data[3] = self.tree.item(selection[0], "values")[3]

old\_data[4] = self.tree.item(selection[0], "values")[4]

old\_data[5] = self.tree.item(selection[0], "values")[5]

with conn.cursor() as cursor:

cursor.execute(f"""DELETE FROM public."SSB" WHERE brigade\_code='{old\_data[0]}' AND service\_number='{old\_data[1]}' AND profession\_code='{old\_data[2]}' """)

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("brigade\_code", "service\_number", "profession\_code", "sn\_in", "rank", "grid") VALUES ('{value1}', '{value2}', '{value3}', '{value4}', {value5}, '{value6}') """)

self.errorWindows("Изменение в ССБ", "Изменение прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("brigade\_code","service\_number", "profession\_code", "sn\_in", "rank", "grid") VALUES ('{old\_data[0]}', '{old\_data[1]}', '{old\_data[2]}', '{old\_data[3]}', {old\_data[4]}, '{old\_data[5]}') """)

self.errorWindows("Ошибка изменения", "Невозможно изменить данные!")

self.refresh(column\_names, tablename, tablenamerus)

elif tablename == "SB":

old\_data = [0, 0, 0]

try:

value1 = self.l1e.get()

value2 = self.l2e.get()

value3 = int(self.l3e.get())

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

old\_data[0] = self.tree.item(selection[0], "values")[0]

old\_data[1] = self.tree.item(selection[0], "values")[1]

old\_data[2] = self.tree.item(selection[0], "values")[2]

with conn.cursor() as cursor:

cursor.execute(f"""DELETE FROM public."SB" WHERE brigade\_code='{old\_data[0]}' """)

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"( "brigade\_code", "sn\_in\_brigadier", "person\_quantity") VALUES ('{value1}', '{value2}', {value3}) """)

self.errorWindows("Изменение в СБ", "Изменение прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"( "brigade\_code", "sn\_in\_ brigadier", "person\_quantity") VALUES ('{old\_data[0]}', '{old\_data[1]}', {old\_data[2]}) """)

self.errorWindows("Ошибка изменения", "Невозможно изменить данные!")

self.refresh(column\_names, tablename, tablenamerus)

elif tablename == "PP":

old\_data = [0, 0, 0, 0]

try:

value1 = self.l1e.get()

value2 = self.l2e.get()

value3 = int(self.l3e.get())

value4 = int(self.l4e.get())

except Exception as \_ex:

self.errorWindows("Ошибка ввода", "Некорректные данные")

return

old\_data[0] = self.tree.item(selection[0], "values")[0]

old\_data[1] = self.tree.item(selection[0], "values")[1]

old\_data[2] = self.tree.item(selection[0], "values")[2]

old\_data[3] = self.tree.item(selection[0], "values")[3]

with conn.cursor() as cursor:

cursor.execute(

f"""DELETE FROM public."PP" WHERE brigade\_code='{old\_data[0]}' AND profession\_code='{old\_data[1]}' """)

try:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("brigade\_code", "profession\_code", "percent\_per\_type\_null", "percent\_per\_type\_one") VALUES ('{value1}', '{value2}', {value3}, {value4}) """)

self.errorWindows("Изменение в СБ", "Изменение прошло успешно", "Продолжить")

self.refresh(column\_names, tablename, tablenamerus)

except Exception as \_ex:

with conn.cursor() as cursor:

cursor.execute(f"""INSERT INTO "{tablename}"("brigade\_code", "profession\_code", "percent\_per\_type\_null", "percent\_per\_type\_one") VALUES

('{old\_data[0]}', '{old\_data[1]}', {old\_data[2]}, {old\_data[3]}) """)

self.errorWindows("Ошибка изменения", "Невозможно изменить данные!")

self.refresh(column\_names, tablename, tablenamerus)

def razarch(self):

self.errorWindows("Разархивация", "Восстановление прошло успешно", "Продолжить")

def closeApp(self):

clWin = tk.Toplevel(self)

clWin.title("Выход из АРМ")

clWin.geometry('320x150+{}+{}'.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?", font=('', 14))

self.textCloseWindow.place(x=15, y=30)

self.yesButton = tk.Button(self.closeWindow, text="Да", fg="black", bg="red", width=10, font=('', 12), command=self.rebot)

self.yesButton.place(x=35, y=85)

self.noButton = tk.Button(self.closeWindow, text="Нет", fg="black", bg="green", width=10, font=('', 12), command=clWin.destroy)

self.noButton.place(x=185, y=85)

if \_\_name\_\_ == "\_\_main\_\_":

win = tk.Tk()

start = loginSystem(win)

start.pack()

win.mainloop()