```
import tkinter as tk      # importa il modulo Python GUI Tkinter con sinonimo tk
import psycopg2
import sys
sys.path.append(r'/home/pi/dispensa/others')
import UpdateAlexaCatalog
import UpdateAlexaDB


class DB():

    def __init__(self, barcode, prodotto, categoria, quantita, command):
        self.barcode = barcode
        self.prodotto = prodotto
        self.categoria = categoria
        self.quantita = quantita
        self.command = command

    def command(self):
        try:
            conn = psycopg2.connect(
                host = "raspberrypi.lan",
                database = "postgres",
                user = "postgres",
                password = "<YourPassword>")
            cur = conn.cursor()
            if self.command=="ins":
                #print("ins")
                if self.barcode=="":
                    avviso="Attenzione! Non puoi inserire un prodotto senza barcode"
                    print("avviso")
                    winavv.winres(winavv("AVVISO", avviso, "yellow", "blue", 50, 10000))
```

```
                    return
                command_text = "insert into products values ('{}','{}','{}','0');".format(self.barcode,
self.prodotto, self.categoria)
            if self.command=="upd":
                #print("upd")
                command_text = "update products set quantity = '{}' where barcode =
'{}';".format(self.quantita, self.barcode)
            if self.command=="sel":
                command_text = "select * from products where barcode = '{}';".format(self.barcode)
            cur.execute(command_text)
            #print(command_text)
            if self.command=="sel":
                record = cur.fetchone()
                #print(record)
                if record == None: #Ã¨ il caso della selezione per aggiornare un prodotto non esistente
                    avviso="Attenzione! Prodotto {} non esistente. Usa il modulo 'Nuovo prodotto' per
definirlo".format(self.prodotto)
                    print("avviso")
                    winavv.winres(winavv("AVVISO", avviso, "yellow", "blue", 50, 10000))
                    return["null", "null", 0]
                prodotto = record[1]
                categoria = record[2]
                quantita=int(record[3])
                return[prodotto, categoria, quantita]
            conn.commit()
            avviso="AttivitÃ  sul prodotto '{}' eseguita con successo! QuantitÃ  totale:
{}".format(self.prodotto, self.quantita)
            if __name__ == '__main__':
                winavv.winres(winavv("OK", avviso, "white", "black", 100, 3000))
```

```
        except (Exception, psycopg2.DatabaseError) as error:
            print(error)
            # Creo una nuova finestra dal titolo "errore"

            winavv.winres(winavv("ERRORE", error, "white", "black", 100, 10000))

        finally:
            if conn is not None:
                conn.close()

class winavv:

    def __init__(self, titolo, testo, bg,fg,pady,wafter):
        self.titolo=titolo
        self.testo=testo
        self.bg=bg
        self.fg=fg
        self.pady=pady
        self.wafter=wafter

    def winres(self):
        win_avv = tk.Tk()        # crea la finestra con la label di default (tk)
        win_avv.title(self.titolo)
        win_avv.geometry('+150+450')
        frm_avv = tk.Frame(master=win_avv, relief=tk.RAISED, borderwidth=3)
        lbl_avv = tk.Label(master=frm_avv,
                           text="{}".format(self.testo),
                           bg=self.bg,
                           fg=self.fg,
                           font=("Verdana", 18)
```

```
                        )
        lbl_avv.grid(row=0, column=0, sticky="e") #l'etichetta viene posizionata piÃ¹ a destra possibile
        # Impacchetta il frame dentro la finestra
        frm_avv.pack(fill=tk.X, padx=50, pady=self.pady) #con fill la finestra diventa responsiva
orizzontalmente (X)
        win_avv.after(self.wafter, lambda: win_avv.destroy()) # Chiude la finestra dopo 3 secondi

class Modulo_generico:
    def __init__(self, titolo):
        self.titolo = titolo

    def modulo(self):
        def pulisci():
            print("pulito")
            ent_barcode.delete(0, tk.END)
            if self.titolo =="Inserisci" or self.titolo == "Preleva":
                ent_quantita.delete(0, tk.END)
                ent_quantita.insert(tk.END, 1) #inserisce il valore di default pari a '1'

            if self.titolo=="Nuovo prodotto":
                ent_prodotto.delete(0, tk.END)
                ent_categoria.delete(0, tk.END)

            ent_barcode.focus_set() #il cursore lampeggia dentro il campo del barcode

        def salva():
            print("salvato")
            barcode = ent_barcode.get()
            if self.titolo=="Nuovo prodotto":
                prodotto = ent_prodotto.get()
                categoria = ent_categoria.get()
```

```
                quantita = 0
                command = "ins"
            if self.titolo =="Inserisci" or self.titolo == "Preleva":
                command="sel"
                delta=int(ent_quantita.get())
                print("if titolo inserisci")
                try:
                    (prodotto, categoria, quantita) = DB.command(DB(barcode,"", "", "", command))
                    print(prodotto)

                except (Exception) as error:
                    print(error)
                    return(error)


                if self.titolo=="Inserisci":
                    quantita=quantita + delta
                    command="upd"

                if self.titolo=="Preleva":
                    quantita=quantita - delta
                    print(quantita)
                    if quantita==0:
                        avviso="Attenzione! Ultimo prodotto {}. Inserirlo in lista della
spesa!".format(prodotto)
                        print(avviso)
                        winavv.winres(winavv("AVVISO", avviso, "red", "white", 50, 10000))
                    command="upd"
            if prodotto != "null":
                print(prodotto)
                DB.command(DB(barcode, prodotto, categoria, quantita, command))
```

```
        #aggiorno Alexa
        if self.titolo=="Nuovo prodotto" and barcode != "":
            UpdateAlexaCatalog.NewCatalog()
        UpdateAlexaDB.NewDB()

    pulisci()

winpro = tk.Toplevel()
winpro.title(self.titolo)

winpro.geometry('+1+0')


# Crea un nuovo frame `frm_modulo` per contenere le widget
# Label e Entry per imputare le descrizioni dei prodotti (barcode, nome prodotto, categoria).
frm_modulo = tk.Frame(master=winpro, relief=tk.SUNKEN, borderwidth=3)
# Impacchetta il frame dentro la finestra
frm_modulo.pack()

# Crea le Label e Entry widget per "Barcode"
lbl_barcode = tk.Label(master=frm_modulo,
                       text="Barcode: ",
                       bg="white",
                       fg="black",
                       font=("Verdana", 18)
                       )
ent_barcode = tk.Entry(master=frm_modulo,
                       bg="white",
                       fg="black",
                       font=("Verdana", 18),
                       width=25)
```

```
        # Si usa il grid geometry manager per definirne le posizioni
        # Il widget Label va nella prima colonna, Entry nella seconda
        # entrambi nella prima riga della griglia
        lbl_barcode.grid(row=0, column=0, sticky="e") #l'etichetta viene posizionata piÃ¹ a destra possibile
        ent_barcode.focus_set() #il cursore lampeggia dentro il campo del barcode
        ent_barcode.grid(row=0, column=1)

# Crea le Label e Entry widget per "QuantitÃ "
        lbl_quantita = tk.Label(master=frm_modulo,
                                text="QuantitÃ : ",
                                bg="white",
                                fg="black",
                                font=("Verdana", 18)
                                )
        ent_quantita = tk.Entry(master=frm_modulo,
                                bg="white",
                                fg="black",
                                font=("Verdana", 18),
                                width=25)
        ent_quantita.insert(tk.END, 1) #inserisce il valore di default pari a '1'

        # Si usa il grid geometry manager per definirne le posizioni
        # Il widget Label va nella prima colonna, Entry nella seconda
        # entrambi nella seconda riga della griglia
        lbl_quantita.grid(row=1, column=0, sticky="e") #l'etichetta viene posizionata piÃ¹ a destra possibile
        ent_quantita.grid(row=1, column=1)

        if self.titolo == "Nuovo prodotto":

            # Crea le Label e Entry widget per "Nome prodotto"
            lbl_prodotto = tk.Label(master=frm_modulo,
```

```
                            text="Nome prodotto: ",
                            bg="white",
                            fg="black",
                            font=("Verdana", 18)
                            )
    ent_prodotto = tk.Entry(master=frm_modulo,
                            bg="white",
                            fg="black",
                            font=("Verdana", 18),
                            width=25)
    # i relativi widget vengono posizionati sulla seconda riga
    lbl_prodotto.grid(row=1, column=0, sticky="e")
    ent_prodotto.grid(row=1, column=1)

    # Crea le Label e Entry widget per "Categoria"
    lbl_categoria = tk.Label(master=frm_modulo,
                             text="Categoria: ",
                             bg="white",
                             fg="black",
                             font=("Verdana", 18)
                             )
    ent_categoria = tk.Entry(master=frm_modulo,
                             bg="white",
                             fg="black",
                             font=("Verdana", 18),
                             width=25)
    # i relativi widget vengono posizionati sulla terza riga
    lbl_categoria.grid(row=2, column=0, sticky="e")
    ent_categoria.grid(row=2, column=1)

# Crea `frm_pulsanti` per contenere
```

```
        # i pulsanti Salva e Pulisci. Questo frame riempie
        # l'intera finestra orizzontalmente con
        # 5 pixels di padding orizzontale e verticale.
        frm_pulsanti = tk.Frame(master=winpro, relief=tk.RAISED, borderwidth=3)
        frm_pulsanti.pack(fill=tk.X, ipadx=5, ipady=5) #con fill la finestra diventa responsiva
orizzontalmente (X)

        # Crea il pulsante "Salva" e lo impacchetta
        # a destra del frame `frm_pulsanti`
        btn_salva = tk.Button(master=frm_pulsanti,
                              text="Salva",
                              bg="white",
                              fg="black",
                              font=("Verdana", 18),
                              command=salva)
        btn_salva.pack(side=tk.RIGHT, padx=10, ipadx=10)

        # Crea il pulsante "Pulisci" e lo impaccherra
        # sul lato destro del frame `frm_pulsanti`
        btn_pulisci = tk.Button(master=frm_pulsanti,
                                text="Pulisci",
                                bg="white",
                                fg="black",
                                font=("Verdana", 18),
                                command=pulisci)
        btn_pulisci.pack(side=tk.RIGHT, ipadx=10)

        def autosave():


            if self.titolo =="Inserisci" or self.titolo == "Preleva":
```

```
                #print("autosave")
                ent_barcode.focus_set()
                barcode = ent_barcode.get()
                if barcode != "":
                    print("barcode {} salvato automaticamente".format(barcode))
                    salva()

            winpro.after(3000, lambda: autosave()) #


        ##############
        #autosave()
        winpro.after(3000, lambda: autosave()) #
        ##############8076809540421

def main():

    # Creo una nuova finestra dal titolo "Gestione magazzino"
    window = tk.Tk()        # crea la finestra con la label di default (tk)
    window.title("Gestione magazzino")

    # Creo un nuovo frame per i pulsanti principali
    frm_pulsanti_princ = tk.Frame()

    # Impacchetta il frame dentro la finestra
    frm_pulsanti_princ.pack(fill=tk.X, padx=100, pady=100) #con fill la finestra diventa responsiva
orizzontalmente (X)

    # Crea il pulsante "Inserisci" e lo impacchetta
    # a destra del frame `frm_pulsanti_princ`
```

```
btn_inserisci = tk.Button(
    master=frm_pulsanti_princ,
    text="Inserisci",
    width=10,
    height=1,
    bg="white",
    fg="black",
    font=("Verdana", 32),
    command=lambda: inserisci(),
    relief=tk.RAISED
    )
btn_inserisci.pack(side=tk.LEFT, padx=5)

# Crea il pulsante "Preleva" e lo impaccherra
# sul lato destro del frame `frm_pulsanti_princ`

btn_preleva = tk.Button(
    master=frm_pulsanti_princ,
    text="Preleva",
    width=10,
    height=1,
    bg="white",
    fg="black",
    font=("Verdana", 32),
    command=lambda: preleva(),
    relief=tk.RAISED
    )
btn_preleva.pack(side=tk.LEFT, padx=20)

# Crea il pulsante "Nuovo prodotto" e lo impacchetta
# sul lato destro del frame `frm_pulsanti_princ`
```

```
btn_nuovop = tk.Button(
    master=frm_pulsanti_princ,
    text="Nuovo prodotto",
    width=16,
    height=1,
    bg="white",
    fg="black",
    font=("Verdana", 20),
    relief=tk.RAISED,
    command=lambda: nuovo_prod()
    )
btn_nuovop.pack(side=tk.RIGHT, padx=50, ipady = 10)

def nuovo_prod():

    btn_nuovop["relief"]=tk.SUNKEN
    btn_inserisci["relief"]=tk.RAISED
    btn_preleva["relief"]=tk.RAISED
    btn_nuovop["state"]="disabled"
    btn_inserisci["state"]="normal"
    btn_preleva["state"]="normal"
    Modulo_generico.modulo(Modulo_generico("Nuovo prodotto"))

def inserisci():

    btn_nuovop["relief"]=tk.RAISED
    btn_inserisci["relief"]=tk.SUNKEN
    btn_preleva["relief"]=tk.RAISED
    btn_nuovop["state"]="normal"
    btn_inserisci["state"]="disabled"
```

```
        btn_preleva["state"]="normal"
        Modulo_generico.modulo(Modulo_generico("Inserisci"))

    def preleva():

        btn_nuovop["relief"]=tk.RAISED
        btn_inserisci["relief"]=tk.RAISED
        btn_preleva["relief"]=tk.SUNKEN
        btn_nuovop["state"]="normal"
        btn_inserisci["state"]="normal"
        btn_preleva["state"]="disabled"
        Modulo_generico.modulo(Modulo_generico("Preleva"))

    # Avvia l'applicazione
    window.mainloop()
if __name__ == '__main__':
    main()
```