UNIWERSYTET ROLNICZY IM. HUGONA KOŁŁĄTAJA W KRAKOWIE

Wydział Inżynierii Produkcji i Energetyki
Rok akademicki 2022/2023

Inżynieria Oprogramowania

Temat: **Aplikacja do małego sklepu stacjonarnego**

Kierunek studiów: IM

Autor: Szymon Kierepka, Bartosz Mucha, Szymon Flaga,
15.06.2023

Ten kod tworzy bazę danych SQLite z czterema tabelami (**tabela1**, **tabela2**, **tabela3**, **tabela4**). Który umożliwia użytkownikowi wykonywanie różnych operacji na bazie danych, takich jak dodawanie, usuwanie, edytowanie, wyświetlanie, wyszukiwanie, sortowanie, filtrowanie oraz eksportowanie i importowanie danych z pliku CSV. (widok tabelek i interfejsu na samym dole)

Kod sql tworzący bazę danych

```
import sqlite3

# Utworzenie połączenia z bazą danych
conn = sqlite3.connect('baza_danych.db')
c = conn.cursor()

# Utworzenie tabel
c.execute('''CREATE TABLE IF NOT EXISTS tabela1
             (id INTEGER PRIMARY KEY,
             nazwa TEXT,
             cena REAL)''')

c.execute('''CREATE TABLE IF NOT EXISTS tabela2
             (id INTEGER PRIMARY KEY,
             marka TEXT,
             ilosc INTEGER)''')

c.execute('''CREATE TABLE IF NOT EXISTS tabela3
             (id INTEGER PRIMARY KEY,
             kategoria TEXT,
             opis TEXT)''')

c.execute('''CREATE TABLE IF NOT EXISTS tabela4
             (id INTEGER PRIMARY KEY,
             producent TEXT,
             rok INTEGER)''')
```

```
# Wypełnienie tabel danymi
c.execute("INSERT INTO tabela1 (id, nazwa, cena) VALUES (1, 'Produkt 1', 10.99)")
c.execute("INSERT INTO tabela1 (id, nazwa, cena) VALUES (2, 'Produkt 2', 15.99)")
c.execute("INSERT INTO tabela1 (id, nazwa, cena) VALUES (3, 'Produkt 3', 20.50)")

c.execute("INSERT INTO tabela2 (id, marka, ilosc) VALUES (1, 'Marka A', 50)")
c.execute("INSERT INTO tabela2 (id, marka, ilosc) VALUES (2, 'Marka B', 100)")
c.execute("INSERT INTO tabela2 (id, marka, ilosc) VALUES (3, 'Marka C', 75)")

c.execute("INSERT INTO tabela3 (id, kategoria, opis) VALUES (1, 'Kategoria A', 'Opis kategorii A')")
c.execute("INSERT INTO tabela3 (id, kategoria, opis) VALUES (2, 'Kategoria B', 'Opis kategorii B')")
c.execute("INSERT INTO tabela3 (id, kategoria, opis) VALUES (3, 'Kategoria C', 'Opis kategorii C')")

c.execute("INSERT INTO tabela4 (id, producent, rok) VALUES (1, 'Producent X', 2019)")
c.execute("INSERT INTO tabela4 (id, producent, rok) VALUES (2, 'Producent Y', 2020)")
c.execute("INSERT INTO tabela4 (id, producent, rok) VALUES (3, 'Producent Z', 2021)")

# Zatwierdzenie zmian i zamknięcie połączenia
conn.commit()
conn.close()
```

Kod źródłowy

```
import sqlite3
from tkinter import *
import csv

# Funkcja dodająca nowy rekord do bazy danych
def dodaj_rekord():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
```

```
    c.execute("INSERT INTO tabela1 (nazwa, cena) VALUES (?, ?)", (pole1.get(), pole2.get()))
    conn.commit()
    conn.close()

# Funkcja usuwająca rekord z bazy danych
def usun_rekord():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    c.execute("DELETE FROM tabela1 WHERE id=?", (pole_id.get(),))
    conn.commit()
    conn.close()

# Funkcja edytująca rekord w bazie danych
def edytuj_rekord():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    c.execute("UPDATE tabela1 SET nazwa=?, cena=? WHERE id=?", (pole1.get(), pole2.get(), pole_id.get()))
    conn.commit()
    conn.close()

# Funkcja wyświetlająca rekordy z bazy danych
def wyswietl_rekordy():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    c.execute("SELECT * FROM tabela1")
    rekordy = c.fetchall()
    for rekord in rekordy:
        print(rekord)
    conn.close()

# Funkcja wyszukująca rekordy w bazie danych
def wyszukaj_rekordy():
```

```
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    c.execute("SELECT * FROM tabela1 WHERE kolumna1=?", (pole1.get(),))
    rekordy = c.fetchall()
    for rekord in rekordy:
        print(rekord)
    conn.close()

# Funkcja sortująca rekordy w bazie danych
def sortuj_rekordy():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    c.execute("SELECT * FROM tabela1 ORDER BY nazwa")
    rekordy = c.fetchall()
    for rekord in rekordy:
        print(rekord)
    conn.close()

# Funkcja filtrowania rekordów w bazie danych
def filtrowanie_rekordy():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    c.execute("SELECT * FROM tabela1 WHERE nazwa LIKE ?", (f"%{pole1.get()}%",))
    rekordy = c.fetchall()
    for rekord in rekordy:
        print(rekord)
    conn.close()

# Funkcja eksportująca rekordy z bazy danych do pliku CSV
def eksportuj_do_csv():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
```

```
    c.execute("SELECT * FROM tabela1")
    rekordy = c.fetchall()
    with open('eksport.csv', 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(['nazwa', 'cena'])
        writer.writerows(rekordy)
    conn.close()

# Funkcja importująca rekordy z pliku CSV do bazy danych
def importuj_z_csv():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    with open('import.csv', 'r') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            c.execute("INSERT INTO tabela1 (nazwa, cena) VALUES (?, ?)", (row[0], row[1]))
    conn.commit()
    conn.close()

# Tworzenie interfejsu użytkownika przy użyciu biblioteki Tkinter
root = Tk()

pole1 = Entry(root)
pole1.pack()

pole2 = Entry(root)
pole2.pack()

pole_id = Entry(root)
pole_id.pack()

przycisk_dodaj = Button(root, text="Dodaj rekord", command=dodaj_rekord)
```

```
przycisk_dodaj.pack()

przycisk_usun = Button(root, text="Usuń rekord", command=usun_rekord)
przycisk_usun.pack()

przycisk_edytuj = Button(root, text="Edytuj rekord", command=edytuj_rekord)
przycisk_edytuj.pack()

przycisk_wyswietl = Button(root, text="Wyświetl rekordy", command=wyswietl_rekordy)
przycisk_wyswietl.pack()

przycisk_wyszukaj = Button(root, text="Wyszukaj rekordy", command=wyszukaj_rekordy)
przycisk_wyszukaj.pack()

przycisk_sortuj = Button(root, text="Sortuj rekordy", command=sortuj_rekordy)
przycisk_sortuj.pack()

przycisk_filtruj = Button(root, text="Filtruj rekordy", command=filtrowanie_rekordy)
przycisk_filtruj.pack()

przycisk_eksportuj = Button(root, text="Eksportuj do CSV", command=eksportuj_do_csv)
przycisk_eksportuj.pack()

przycisk_importuj = Button(root, text="Importuj z CSV", command=importuj_z_csv)
przycisk_importuj.pack()

root.mainloop()

# Funkcja tworząca tabele w bazie danych
def utworz_tabele():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
```

```
    # Tabela 1
    c.execute('''CREATE TABLE IF NOT EXISTS tabela1
                 (id INTEGER PRIMARY KEY AUTOINCREMENT,
                 nazwa TEXT,
                 cena TEXT)''')

    # Tabela 2
    c.execute('''CREATE TABLE IF NOT EXISTS tabela2
                 (id INTEGER PRIMARY KEY AUTOINCREMENT,
                 marka TEXT,
                 ilosc TEXT)''')

    # Tabela 3
    c.execute('''CREATE TABLE IF NOT EXISTS tabela3
                 (id INTEGER PRIMARY KEY AUTOINCREMENT,
                 kategoria TEXT,
                 opis TEXT)''')

    # Tabela 4
     c.execute('''CREATE TABLE IF NOT EXISTS tabela4
                 (id INTEGER PRIMARY KEY AUTOINCREMENT,
                 producent TEXT,
                 rok TEXT)''')

    conn.commit()
    conn.close()

# Funkcja wypełniająca tabele danymi
def wypelnij_tabele():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
```

```
# Wypełnienie tabel danymi
c.execute("INSERT INTO tabela1 (id, nazwa, cena) VALUES (1, 'Produkt 1', 10.99)")
c.execute("INSERT INTO tabela1 (id, nazwa, cena) VALUES (2, 'Produkt 2', 15.99)")
c.execute("INSERT INTO tabela1 (id, nazwa, cena) VALUES (3, 'Produkt 3', 20.50)")

c.execute("INSERT INTO tabela2 (id, marka, ilosc) VALUES (1, 'Marka A', 50)")
c.execute("INSERT INTO tabela2 (id, marka, ilosc) VALUES (2, 'Marka B', 100)")
c.execute("INSERT INTO tabela2 (id, marka, ilosc) VALUES (3, 'Marka C', 75)")

c.execute("INSERT INTO tabela3 (id, kategoria, opis) VALUES (1, 'Kategoria A', 'Opis kategorii A')")
c.execute("INSERT INTO tabela3 (id, kategoria, opis) VALUES (2, 'Kategoria B', 'Opis kategorii B')")
c.execute("INSERT INTO tabela3 (id, kategoria, opis) VALUES (3, 'Kategoria C', 'Opis kategorii C')")

c.execute("INSERT INTO tabela4 (id, producent, rok) VALUES (1, 'Producent X', 2019)")
c.execute("INSERT INTO tabela4 (id, producent, rok) VALUES (2, 'Producent Y', 2020)")
c.execute("INSERT INTO tabela4 (id, producent, rok) VALUES (3, 'Producent Z', 2021)")

# Zatwierdzenie zmian i zamknięcie połączenia
conn.commit()
conn.close()

# Funkcja sortująca rekordy w bazie danych według kategorii dla każdej tabeli
def sortuj_rekordy_kategoria():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()

    # Tabela 1
    c.execute("SELECT * FROM tabela1 ORDER BY cena")
    rekordy_tabela1 = c.fetchall()
    print("Tabela 1 - Posortowana według kategorii:")
    for rekord in rekordy_tabela1:
        print(rekord)
```

```
    # Tabela 2
    c.execute("SELECT * FROM tabela2 ORDER BY ilosc")
    rekordy_tabela2 = c.fetchall()
    print("Tabela 2 - Posortowana według kategorii:")
    for rekord in rekordy_tabela2:
        print(rekord)

    # Tabela 3
    c.execute("SELECT * FROM tabela3 ORDER BY opis")
    rekordy_tabela3 = c.fetchall()
    print("Tabela 3 - Posortowana według kategorii:")
    for rekord in rekordy_tabela3:
        print(rekord)

    # Tabela 4
    c.execute("SELECT * FROM tabela4 ORDER BY rok")
    rekordy_tabela4 = c.fetchall()
    print("Tabela 4 - Posortowana według kategorii:")
    for rekord in rekordy_tabela4:
        print(rekord)

    conn.close()

# Tworzenie interfejsu użytkownika przy użyciu biblioteki Tkinter
root = Tk()

utworz_tabele()
wypelnij_tabele()

przycisk_sortuj_kategorie = Button(root, text="Sortuj rekordy według kategorii",
command=sortuj_rekordy_kategoria)
przycisk_sortuj_kategorie.pack()
```

```
root.mainloop()
```

Testy:

Test dodawania rekordu:

```
import sqlite3

# Utworzenie połączenia z bazą danych
conn = sqlite3.connect('baza_danych.db')
c = conn.cursor()

# Dodanie nowego rekordu do tabela1
c.execute("INSERT INTO tabela1 (nazwa, cena) VALUES ('Nowy produkt', 25.99)")
conn.commit()

# Sprawdzenie czy rekord został dodany
c.execute("SELECT * FROM tabela1 WHERE nazwa='Nowy produkt' AND cena=25.99")
record_exists = c.fetchone() is not None
assert record_exists, "Nowy rekord nie został dodany do tabela1."

conn.close()
```

Test usuwania rekordu:

```
import sqlite3
from tkinter import *

# Funkcja usuwająca rekord z bazy danych
def usun_rekord():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    c.execute("DELETE FROM tabela1 WHERE nazwa=?", (pole1.get(),))
    conn.commit()
    conn.close()
```

```
# Test usuwania rekordu
def test_usuwania_rekordu():
    pole1.insert(0, "Nazwa produktu do usunięcia")
    usun_rekord()
    wyswietl_rekordy()

# Tworzenie interfejsu użytkownika przy użyciu biblioteki Tkinter
root = Tk()

pole1 = Entry(root)
pole1.pack()

przycisk_usun = Button(root, text="Usuń rekord", command=test_usuwania_rekordu)
przycisk_usun.pack()

root.mainloop()
```

Test wyświetlania rekordów:

```
import sqlite3
from tkinter import *

# Funkcja wyświetlająca rekordy z bazy danych
def wyswietl_rekordy():
    conn = sqlite3.connect('baza_danych.db')
    c = conn.cursor()
    c.execute("SELECT * FROM tabela1")
    rekordy = c.fetchall()
    for rekord in rekordy:
        print(rekord)
    conn.close()

# Test wyświetlania rekordów
def test_wyswietlania_rekordow():
    wyswietl_rekordy()
```

```
# Tworzenie interfejsu użytkownika przy użyciu biblioteki Tkinter
root = Tk()

przycisk_wyswietl = Button(root, text="Wyświetl rekordy", command=test_wyswietlania_rekordow)
przycisk_wyswietl.pack()

root.mainloop()
```

Test edytowania rekordów:

```
import sqlite3

# Utworzenie połączenia z bazą danych
conn = sqlite3.connect('baza_danych.db')
c = conn.cursor()

# Edycja rekordu w tabela1
c.execute("UPDATE tabela1 SET nazwa='Produkt X', cena=30.0 WHERE id=1")
conn.commit()

# Sprawdzenie czy rekord został zmieniony
c.execute("SELECT * FROM tabela1 WHERE id=1 AND nazwa='Produkt X' AND cena=30.0")
record_exists = c.fetchone() is not None
```

Test tworzenia tabel:

```
import sqlite3

# Utworzenie połączenia z bazą danych
conn = sqlite3.connect('baza_danych.db')
c = conn.cursor()

# Sprawdzenie czy tabela1 została utworzona
c.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='tabela1'")
table_exists = c.fetchone() is not None
assert table_exists, "Tabela tabela1 nie została utworzona."
```

```
# Sprawdzenie czy tabela2 została utworzona
c.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='tabela2'")
table_exists = c.fetchone() is not None
assert table_exists, "Tabela tabela2 nie została utworzona."

# Sprawdzenie czy tabela3 została utworzona
c.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='tabela3'")
table_exists = c.fetchone() is not None
assert table_exists, "Tabela tabela3 nie została utworzona."

# Sprawdzenie czy tabela4 została utworzona
c.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='tabela4'")
table_exists = c.fetchone() is not None
assert table_exists, "Tabela tabela4 nie została utworzona."
conn.close()
```

Widok Tabelek

id	nazwa	cena
1	Produkt 1	10.99
2	Produkt 2	15.99
3	Produkt 3	20.5

id	marka	ilosc
1	Marka A	50
2	Marka B	100
3	Marka C	75

id	kategoria	opis
1	Kategoria A	Opis kategorii A
2	Kategoria B	Opis kategorii B
3	Kategoria C	Opis kategorii C

id	producent	rok
1	Producent X	2019
2	Producent Y	2020
3	Producent Z	2021

Widok Interfejsu