Badania Operacyjne 2			
Zakupy w Auchan – struktury danych, implementacja GUI, baza danych			
EAIiIB	Automatyka i Robotyka	Grupa 1, środa 08:00	III Rok
L.p.	**Skład zespołu laboratoryjnego**		
1	Gabriela Bergiel		
2	Adam Barnaś		
3	Karolina Chochrek		

Spis treści

Struktury danych

Zaimplementowaliśmy najważniejsze struktury danych, które będą potrzebne do łączenia się z bazą danych na późniejszych etapach projektu.

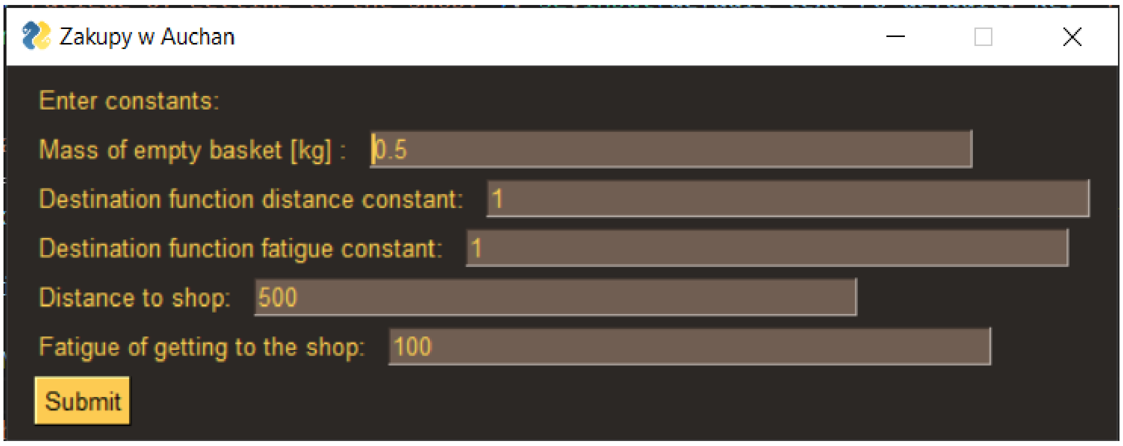
```
import sqlite3
from sqlite3 import Error

class Database:
    def __init__(self):
        self.database = "database/data.db"
        self.conn = self.create_connection(self.database)


    def create_connection(self, db_file):
        """ create a database connection to the SQLite database
            specified by db_file
        :param db_file: database file
        :return: Connection object or None
        """
        conn = None
        try:
            conn = sqlite3.connect(db_file)
        except Error as e:
            print(e)

        return conn
    def create_user(self, user, avatar=None):
        """
        Create a new project into the projects table
        :param conn:
        :param project:
        :return: project id
        """
        userfull = [user[0], user[1], 0, 0, avatar]
        sql = ''' INSERT INTO tabela(login,password,total_distance,XP,avatar)
                  VALUES(?,?,?,?,?) '''
        cur = self.conn.cursor()
        cur.execute(sql, userfull)
        self.conn.commit()
        return None

    def update_userinfo(self, user, distance, XP):
        """
        Create a new project into the projects table
        :param conn:
        :param project:
        :return: project id
        """
        changes = (distance, XP, user[0], user[1])
        sql = ''' UPDATE tabela
                          SET total_distance = ?,
                              XP = ?
                          WHERE login = ? AND password = ? '''
        cur = self.conn.cursor()
        cur.execute(sql, changes)
        self.conn.commit()
        return None

    def check_userinfo(self, user):
        sql = '''SELECT * FROM tabela WHERE login = ? and password = ?'''
        cur = self.conn.cursor()
        cur.execute(sql, user)
        nodes = cur.fetchall()
        if len(nodes) > 0:
            pass
        else:
            self.create_user(user)

    def get_userdata(self, user):
        sql = '''SELECT * FROM tabela WHERE login = ? AND password = ?'''
        cur = self.conn.cursor()
        cur.execute(sql, user)
        dane = cur.fetchall()
        return dane
```

```python
    def delete_user(self, user):
        """
        Delete a task by task id
        :param conn:  Connection to the SQLite database
        :param id: id of the task
        :return:
        """
        sql = 'DELETE FROM tabela WHERE login = ? AND password = ?'
        cur = self.conn.cursor()
        cur.execute(sql, user)
        self.conn.commit()

    def delete_all(self):
        """
        Delete a task by task id
        :param conn:  Connection to the SQLite database
        :param id: id of the task
        :return:
        """
        sql = 'DELETE FROM tabela'
        cur = self.conn.cursor()
        cur.execute(sql)
        self.conn.commit()



class Trasa:
    def __init__(self):
        self.database = "database/trasa.db"
        self.conn = self.create_connection(self.database)

    def create_connection(self, db_file):
        """ create a database connection to the SQLite database
            specified by db_file
        :param db_file: database file
        :return: Connection object or None
        """
        conn = None
        try:
            conn = sqlite3.connect(db_file)
        except Error as e:
            print(e)

        return conn

    def create_node(self, coords):
        node = [coords[0], coords[1], 0]
        sql = ''' INSERT INTO tabela(lon, lat, visit)
                  VALUES(?,?,?) '''
        cur = self.conn.cursor()
        cur.execute(sql, node)
        self.conn.commit()

    def update_node(self, coords):
        min_lat, min_lon, max_lat, max_lon = coords
        sql = ''' UPDATE tabela
                          SET visit = 1
                          WHERE visit = 0 AND lon > %s AND lon < %s AND lat >
%s AND lat < %s '''%(min_lon, max_lon, min_lat, max_lat)

    def get_nodes(self, coords):
        min_lat, min_lon, max_lat, max_lon = coords
        sql = '''SELECT * FROM tabela WHERE lon > %s AND lon < %s AND lat > %s
AND lat < %s '''%(min_lon, max_lon, min_lat, max_lat)
        cur = self.conn.cursor()
        cur.execute(sql)
        nodes = cur.fetchall()
        nodesy = []
        for node in nodes:
            node = (node[0], node[1])
            nodesy.append(node)
        return nodesy

    def get_visted(self, coords):
        min_lat, min_lon, max_lat, max_lon = coords
        sql = '''SELECT * FROM tabela WHERE visit = 1 AND lon > %s AND lon <
%s AND lat > %s AND lat < %s '''%(min_lon, max_lon, min_lat, max_lat)
        cur = self.conn.cursor()
        cur.execute(sql)
        nodes = cur.fetchall()
        nodesy = []
        for node in nodes:
            node = (node[0], node[1])
            nodesy.append(node)
        return nodesy

    def get_not_visted(self, coords):
        min_lat, min_lon, max_lat, max_lon = coords
        sql = '''SELECT * FROM tabela WHERE visit = 0 AND lon > %s AND lon <
%s AND lat > %s AND lat < %s '''%(min_lon, max_lon, min_lat, max_lat)
        cur = self.conn.cursor()
        cur.execute(sql)
        nodes = cur.fetchall()
        nodesy = []
        for node in nodes:
```

```python
            node = (node[0], node[1])
            nodesy.append(node)
        return nodesy

    def delete_trip(self):
        """
        Delete a task by task id
        :param conn:  Connection to the SQLite database
        :param id: id of the task
        :return:
        """
        sql = 'DELETE FROM tabela'
        cur = self.conn.cursor()
        cur.execute(sql)
        self.conn.commit()

class Miejsca:
    def __init__(self):
        self.database = "database/miejsca.db"
        self.conn = self.create_connection(self.database)

    def create_connection(self, db_file):
        """ create a database connection to the SQLite database
            specified by db_file
        :param db_file: database file
        :return: Connection object or None
        """
        conn = None
        try:
            conn = sqlite3.connect(db_file)
        except Error as e:
            print(e)
        return conn

    def convertToBinaryData(self, filename):
        # Convert digital data to binary format
        with open(filename, 'rb') as file:
            blobData = file.read()
        return blobData

    def create_place(self, place, obrazek):
        # place = [nazwa, lon, lat, opis]
        binob = self.convertToBinaryData(obrazek)
        calosc = [place[1], place[2], place[0], place[3], binob]
        sql = ''' INSERT INTO tabela(lon, lat, nazwa, opis, obrazek)
                  VALUES(?,?,?,?,?) '''
        cur = self.conn.cursor()
        cur.execute(sql, calosc)
        self.conn.commit()

    def get_places(self, coords, style):
        min_lon, min_lat, max_lon, max_lat = coords
        sql = '''SELECT * FROM tabela WHERE lon > ? AND lon < ? AND lat > ?
AND lat < ? '''
        cur = self.conn.cursor()
        cur.execute(sql, (min_lon, max_lon, min_lat, max_lat))
        places = cur.fetchall()
        miejsca = []
        for place in places:
            place = (place[0], (place[1], place[2]))
            miejsca.append(place)
        return miejsca



def main():
    baza = Database()
    baza.delete_all()
    user = ('nick', "haslo")
    # baza.create_user(user)
    baza.check_userinfo(user)
    print(baza.get_userdata(user))
    # baza.delete_user(user)

    # trasa = Trasa()
    # trasa.create_node((23, 23))
    # trasa.create_node((25, 25))
    # print(trasa.get_not_visted((10, 12, 40, 40)))
    # trasa.delete_trip()

if __name__ == '__main__':
    main()
```

Implementacja GUI

Do zaimplementowania GUI użyliśmy biblioteki PySimpleGUI, która jest przydatna do tworzenia aplikacji okienkowych.


Zakupy w Auchan
Enter constants:
Mass of empty basket [kg] : 0.5
Destination function distance constant: 1
Destination function fatigue constant: 1
Distance to shop: 500
Fatigue of getting to the shop: 100
Submit

Rys. Widok okna GUI.

```
import PySimpleGUI as sg

sg.theme('DarkAmber')   # Add a touch of color

def input_constants(mo_default = 0.5, c_l_default = 1, c_f_default = 1,
L0_default = 500, F0_default = 100):
    #inputs: constants for algorithm, returns dict with given constants or
default values

    layout = [  [sg.Text('Enter constants: ')],
                [sg.Text(f'Mass of empty basket [kg] :'),
sg.Input(default_text=mo_default, key="m0")],
                [sg.Text(f'Destination function distance constant:'),
sg.Input(default_text=c_l_default, key="c_l")],
                [sg.Text(f'Destination function fatigue constant:'),
sg.Input(default_text=c_f_default, key="c_f")],
                [sg.Text(f'Distance to
shop:'),  sg.Input(default_text=L0_default, key="L0")],
                [sg.Text(f'Fatigue of getting to the shop:'),
sg.Input(default_text=F0_default, key="F0")],
                [sg.Button("Submit")]
                ]

    window = sg.Window("Zakupy w Auchan", layout)
    m0, c_l, c_f, L0, F0 = mo_default, c_l_default, c_f_default, L0_default
,F0_default
    const_dict = {"m0": mo_default, "c_l": c_l_default, "c_f": c_f_default,
"L0": L0_default , "F0": F0_default}
    while True:
        event, values = window.read()

        if event == sg.WIN_CLOSED:
            break
        elif event == "Submit":
            const_dict = {k: float(values[k])  for (k, v) in values.items()}


    window.close()
    return const_dict


def outputs(LS, DL):
    layout = [  [sg.Text]

    ]

# TEST: You can call the function with your default value
cd = input_constants()
# print(cd)
```

Baza danych

Zaimplementowaliśmy bazę danych, która zawiera przykładowe produkty, które klient może kupić.

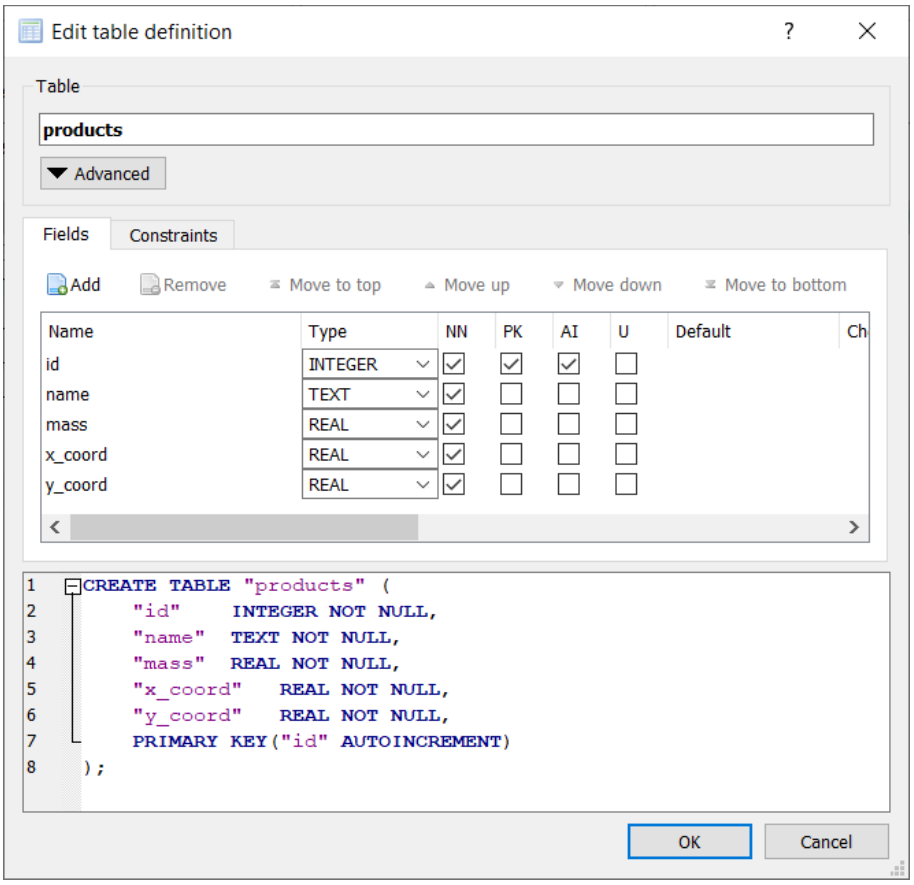
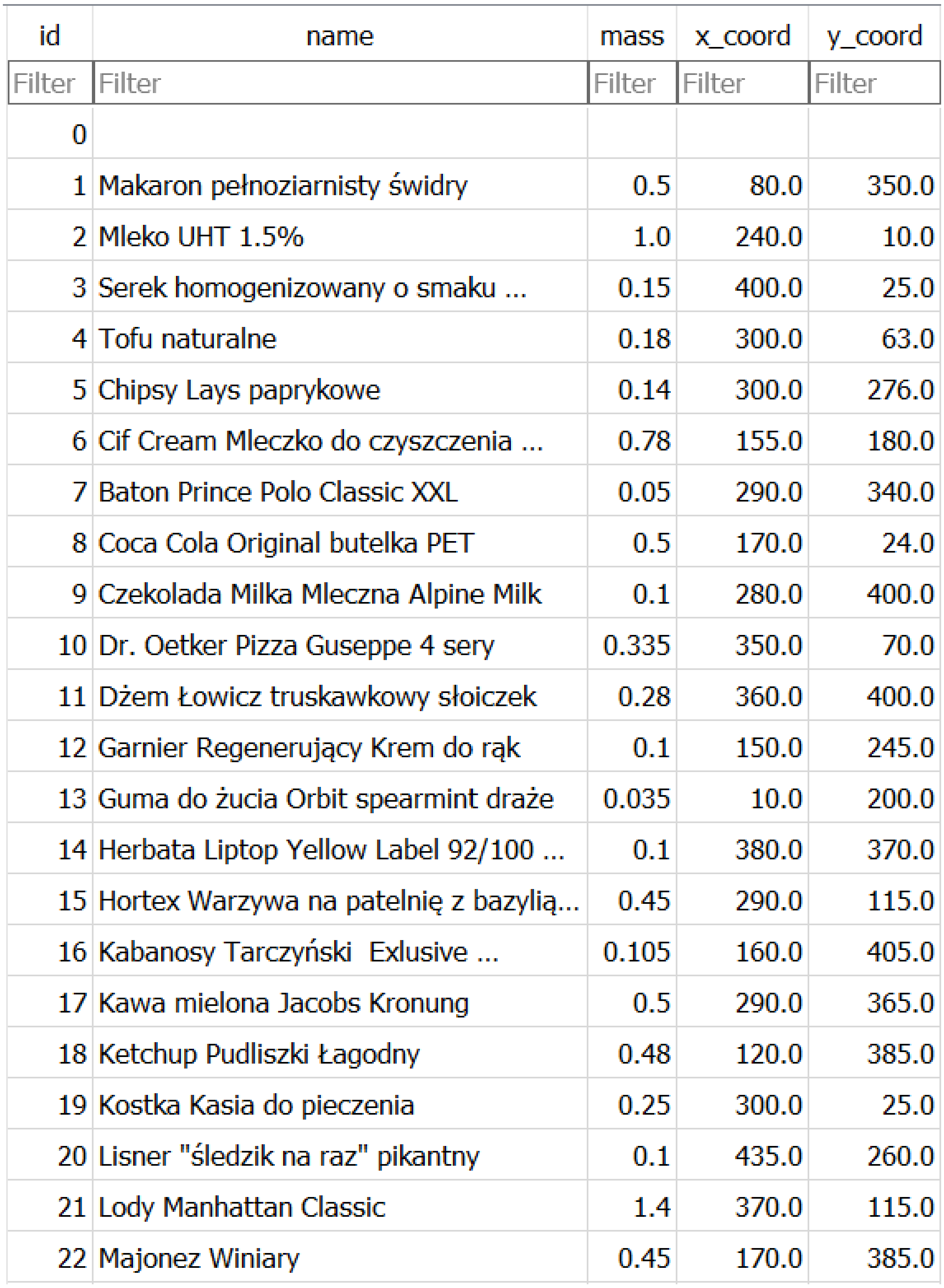


Table: **products**

Name	Type	NN	PK	AI	U	Default	Ch
id	INTEGER	✓	✓	✓			
name	TEXT	✓					
mass	REAL	✓					
x_coord	REAL	✓					
y_coord	REAL	✓					

```
CREATE TABLE "products" (
    "id"    INTEGER NOT NULL,
    "name"  TEXT NOT NULL,
    "mass"  REAL NOT NULL,
    "x_coord"   REAL NOT NULL,
    "y_coord"   REAL NOT NULL,
    PRIMARY KEY("id" AUTOINCREMENT)
);
```

Rys. 2. Inicjalizacja bazy danych w programie „DB Browser for SQLite”.



id	name	mass	x_coord	y_coord
Filter	Filter	Filter	Filter	Filter
0				
1	Makaron pełnoziarnisty świdry	0.5	80.0	350.0
2	Mleko UHT 1.5%	1.0	240.0	10.0
3	Serek homogenizowany o smaku ...	0.15	400.0	25.0
4	Tofu naturalne	0.18	300.0	63.0
5	Chipsy Lays paprykowe	0.14	300.0	276.0
6	Cif Cream Mleczko do czyszczenia ...	0.78	155.0	180.0
7	Baton Prince Polo Classic XXL	0.05	290.0	340.0
8	Coca Cola Original butelka PET	0.5	170.0	24.0
9	Czekolada Milka Mleczna Alpine Milk	0.1	280.0	400.0
10	Dr. Oetker Pizza Guseppe 4 sery	0.335	350.0	70.0
11	Dżem Łowicz truskawkowy słoiczek	0.28	360.0	400.0
12	Garnier Regenerujący Krem do rąk	0.1	150.0	245.0
13	Guma do żucia Orbit spearmint draże	0.035	10.0	200.0
14	Herbata Liptop Yellow Label 92/100 ...	0.1	380.0	370.0
15	Hortex Warzywa na patelnię z bazylią...	0.45	290.0	115.0
16	Kabanosy Tarczyński Exlusive ...	0.105	160.0	405.0
17	Kawa mielona Jacobs Kronung	0.5	290.0	365.0
18	Ketchup Pudliszki Łagodny	0.48	120.0	385.0
19	Kostka Kasia do pieczenia	0.25	300.0	25.0
20	Lisner "śledzik na raz" pikantny	0.1	435.0	260.0
21	Lody Manhattan Classic	1.4	370.0	115.0
22	Majonez Winiary	0.45	170.0	385.0

Rys. 3. Pierwsze przykładowe 22 rekordy w bazie danych.