МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

ОТЧЕТ

по лабораторной работе № 7
на тему: «Разработка приложения, взаимодействующего с базой данных»
по дисциплине: «Программирование на языке Python»
Вариант № 7

Выполнила: Карпикова С.П. Шифр: 170580
Институт приборостроения, автоматизации и информационных технологий
Направление: 09.03.04 «Программная инженерия»
Группа: 71-ПГ
Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «____» _____ 2019 г.

Орел, 2019 г.

Задание:

Разработать приложение (можно web-приложение), взаимодействующее с базой данных. Приложение должно иметь удобный графический интерфейс. Базу данных разработать в соответствии с темой своего варианта (База данных «Лекарства»). База данных должна состоять из 2-3 связанных таблиц; одна таблица основная.

Функционал приложения:

- добавление информации в основную таблица;

- удалении информации из основной таблицы;

- отображение информации из основной таблицы.

Добавление и отображение информации должно быть реализовано в читаемой для пользователя форме (внешние ключи не отображать, вместо них отображать пользователю понятную информацию).

Решение:

```
from pymysql import *

class LibMedicaments(object):

    def __init__(self, **kwargs):
        self.__dict__.update(kwargs)

    def connect(self):
        self.connection = connect(
            host='127.0.0.1',
            user='root',
            password='stivka1855275',
            db='medicaments',
            charset='utf8mb4',
            cursorclass=cursors.DictCursor
        )

    def close_connection(self):
        self.connection.close()

    def get_medicaments(self):
        sql = '''SELECT
                    vendore_code,
                    medicament.name,
                    cost,
                    description,
                    CONCAT(producer.name, ' / ', producer.country) as producer,
                    category.name as category
                FROM medicaments.medicament
                INNER JOIN medicamenthasproducer on medicament.vendore_code =
medicamenthasproducer.id_medicament
```

```
            INNER JOIN producer on producer.id = medicamenthasproducer.id_producer
            INNER JOIN medicamenthascategory on medicament.vendore_code =
medicamenthascategory.id_medicament
            INNER JOIN category on category.id = medicamenthascategory.id_category;'''

        cursor = self.connection.cursor()
        cursor.execute(sql)
        return cursor.fetchall()

    def get_producers(self):
        sql = "SELECT id, CONCAT(producer.name, ' / ', producer.country) as producer FROM medicaments.producer;"

        cursor = self.connection.cursor()
        cursor.execute(sql)
        return cursor.fetchall()

    def get_categories(self):
        sql = "SELECT id, name FROM medicaments.category;"

        cursor = self.connection.cursor()
        cursor.execute(sql)
        return cursor.fetchall()
    def delete_medicament(self, vendore_code):
        sql = 'DELETE FROM medicaments.medicament where vendore_code = %s'

        cursor = self.connection.cursor()
        cursor.execute(sql, int(vendore_code))
        self.connection.commit()
        return cursor

    def insert_medicament(self, vendore_code, name, description, cost, id_category, id_producer):
        sql = '''INSERT INTO medicament (
                    vendore_code,
                    name,
                    description,
                    cost
                )
                VALUES (%s, %s, %s, %s);'''
        cursor = self.connection.cursor()
        cursor.execute(sql, (vendore_code, name, description, cost))
        self.connection.commit()

        sql = '''INSERT INTO medicamenthascategory (
                    id_medicament,
                    id_category
                )
                VALUES (%s, %s);'''

        cursor = self.connection.cursor()
        cursor.execute(sql, (vendore_code, id_category))
        self.connection.commit()

        sql = '''INSERT INTO medicamenthasproducer (
                    id_medicament,
                    id_producer
                )
                VALUES (%s, %s);'''

        cursor = self.connection.cursor()
        cursor.execute(sql, (vendore_code, id_producer))
        self.connection.commit()
        return cursor
```

```
from tkinter import *
from tkinter.ttk import *
from tkinter import messagebox as mb
from tkinter import filedialog
from Medicament import Medicament
from Db import LibMedicaments

class MainFrame(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.medicaments = []
        self.categories = []
        self.producers = []
        self.file_name = ""

        self.db = LibMedicaments()
        self.db.connect()

        self.initUI()
        self.bindEvents()

    def initUI(self):
        self.parent.title("Лекарства")
        self.pack(fill=BOTH, expand=True)

        self.frame_add_1 = Frame(self)
        self.frame_add_1.pack(fill=X)

        lbl_vendor_code = Label(self.frame_add_1, text="Артикул", width=8)
        lbl_vendor_code.pack(side=LEFT, padx=5, pady=5)

        self.entry_vendor_code = Entry(self.frame_add_1, width=15)
        self.entry_vendor_code.pack(side=LEFT, padx=5)

        lbl_name = Label(self.frame_add_1, text="Название", width=10)
        lbl_name.pack(side=LEFT, anchor=N, padx=5, pady=5)

        self.entry_name = Entry(self.frame_add_1, width=25)
        self.entry_name.pack(side=LEFT, padx=5)

        lbl_cost = Label(self.frame_add_1, text="Стоимость", width=10)
        lbl_cost.pack(side=LEFT, anchor=N, padx=5, pady=5)

        self.entry_cost = Entry(self.frame_add_1, width=15)
        self.entry_cost.pack(side=LEFT, padx=5)

        values = []
        self.producers = self.db.get_producers()
        print(self.producers)
        for row in self.producers:
            values.append(row['producer'])

        self.entry_producer = Combobox(self.frame_add_1, state='readonly', width=25)
        self.entry_producer['values'] = tuple(values)
        self.entry_producer.current(0)
        self.entry_producer.pack(side=LEFT, padx=5, pady=5)

        values = []
        self.categories = self.db.get_categories()
        for row in self.categories:
            values.append(row['name'])
```

```
self.entry_category = Combobox(self.frame_add_1, state='readonly', width=25)
self.entry_category['values'] = tuple(values)
self.entry_category.current(0)
self.entry_category.pack(side=LEFT, padx=5, pady=5)

self.frame_add_2 = Frame(self)
self.frame_add_2.pack(fill=X)

lbl_desc = Label(self.frame_add_2, text="Описание", width=10)
lbl_desc.pack(side=LEFT, anchor=N, padx=5, pady=5)

self.txt_desc = Text(self.frame_add_2, width=25, height=3)
self.txt_desc.pack(fill=BOTH, pady=5, padx=5)

self.btn_add = Button(self.frame_add_2, text="Добавить", width=15)
self.btn_add.pack()

self.frame_remove = Frame(self)
self.frame_remove.pack(fill=X)

lbl_remove = Label(self.frame_remove, text="Артикул", width=8)
lbl_remove.pack(side=LEFT, padx=5, pady=5)

self.entry_remove = Entry(self.frame_remove, width=15)
self.entry_remove.pack(side=LEFT, padx=5)

self.btn_remove = Button(self.frame_remove, text="Удалить", width=15)
self.btn_remove.pack(side=LEFT, padx=5, pady=5)

self.frame_table = Frame(self)
self.frame_table.pack(fill=X)

self.table = Treeview(self.frame_table)

self.table["columns"] = ("1", "2", "3", "4", "5")

self.table.column("#0", width=5, minwidth=5)
self.table.column("1", width=40, minwidth=40)
self.table.column("2", width=5, minwidth=5)
self.table.column("3", width=250, minwidth=250)
self.table.column("4", width=5, minwidth=5)
self.table.column("5", width=5, minwidth=5)

self.table.heading("#0", text="Артикул", anchor=W)
self.table.heading("1", text="Название", anchor=W)
self.table.heading("2", text="Стоимость", anchor=W)
self.table.heading("3", text="Описание", anchor=W)
self.table.heading("4", text="Производитель", anchor=W)
self.table.heading("5", text="Категория", anchor=W)

self.table.pack(side=TOP, fill=X)

self.main_frame = Frame(self)
self.main_frame.pack(fill=X)

lbl1 = Label(self.main_frame, text="Меню", width=6)
lbl1.pack(side=LEFT, padx=5, pady=5)

self.v = StringVar()
self.v.trace('w', self.on_combo_change)
```

```
    self.combo = Combobox(self.main_frame, state='readonly', width=25, textvar=self.v)
    self.combo['values'] = (
    "Добавить лекарство", "Удалить лекарство", "Вывод списка лекарств")
    self.combo.current(0)
    self.combo.pack(side=LEFT, padx=5, pady=5)

    self.mainmenu = Menu(self.parent)

    self.parent.config(menu=self.mainmenu)

    self.filemenu = Menu(self.mainmenu, tearoff=0)
    self.filemenu.add_command(label="Выход", command=self.ask_exit)

    helpmenu = Menu(self.mainmenu, tearoff=0)
    helpmenu.add_command(label="О программе", command=self.about)



def bindEvents(self):
    self.btn_add.bind('<Button-1>', self.add_new_medicament)
    self.btn_remove.bind('<Button-1>', self.delete_medicament)

def hide(self):
    self.frame_add_1.pack_forget()
    self.frame_add_2.pack_forget()
    self.frame_remove.pack_forget()
    self.frame_table.pack_forget()

def on_combo_change(self, index, value, op):
    self.hide()
    current_state = self.combo.get()
    if current_state == "Добавить лекарство":
        self.frame_add_1.pack(fill=X)
        self.frame_add_2.pack(fill=X)

    elif current_state == "Удалить лекарство":
        self.frame_remove.pack(fill=X)

    else:
        self.frame_table.pack(fill=X)
        self.get_medicaments()


def add_new_medicament(self, event):
    vendor_code = self.entry_vendor_code.get()
    name = self.entry_name.get()
    cost = self.entry_cost.get()
    description = self.txt_desc.get("1.0", END)
    id_category = [x for x in self.categories if x['name'] == self.entry_category.get()][0]['id']
    id_producer = [x for x in self.producers if x['producer'] == self.entry_producer.get()][0]['id']

    if len(vendor_code) == 0 or len(name) == 0 or len(cost) == 0 or len(description) == 0:
        mb.showerror("Ошибка", "Все поля должны быть заполнены!")
        return

    if not cost.isdigit():
        mb.showerror("Ошибка", "Стоимость должна иметь численное значение!")
        return

    self.db.insert_medicament(vendor_code, name, description, cost, id_category, id_producer)

    self.clear_text()
```

```
    mb.askokcancel("Успешно", "Информация добавлена!")
    self.filemenu.entryconfig(3, state=NORMAL)
    if self.file_name:
        self.filemenu.entryconfig(2, state=NORMAL)

def get_medicaments(self):
    self.table.delete(*self.table.get_children())


    for row in self.db.get_medicaments():
        print(row)
        self.table.insert('', 'end',
            text=row['vendore_code'],
             values=(
                row['name'],
                row['cost'],
                row['description'],
                row['producer'],
                row['category']
             )
        )

   # print(self.medicaments)

    # for item in self.medicaments:
    #     item.insert(self.table)

def get_without_prescription(self):
    self.table.delete(*self.table.get_children())

    for item in self.medicaments:
        item.without_prescription(self.table)

def clear_text(self):
    self.entry_vendor_code.delete(0, 'end')
    self.entry_name.delete(0, 'end')
    self.entry_cost.delete(0, 'end')
    self.txt_desc.delete("1.0", END)
    self.entry_remove.delete(0, 'end')

def delete_medicament(self, event):
    vendor_code = self.entry_remove.get()

    if len(vendor_code) == 0:
        mb.showerror("Ошибка", "Заполните поле!")
        return

    self.db.delete_medicament(vendor_code)
    self.clear_text()
    mb.askokcancel("Успешно", "Удалено!")
    self.filemenu.entryconfig(2, state=NORMAL)

def create(self):
    self.ask_save()
    self.file_name = ""
    self.clear_text()
    self.medicaments.clear()
    self.table.delete(*self.table.get_children())
    self.combo.current(0)
    self.filemenu.entryconfig(2, state=DISABLED)
    self.filemenu.entryconfig(3, state=DISABLED)
```

```
    def ask_exit(self):
        answer = mb.askyesno("Выход", "Вы дейстельно хотите выйти?")
        if answer == True:
            self.quit()

    def about(self):
        mb.askokcancel("о программе", "Сделала Сонька Карпикова из 71 ПГ!")


def main():
    root = Tk()
    root.geometry("1000x500")
    app = MainFrame(root)
    root.mainloop()


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