РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №6

по дисциплине: Базы данных
тема: «Организация взаимодействия с базой данных через приложение с графическим интерфейсом»

Выполнил: ст. группы ПВ-201
Барышникова Варвара Дмитриевна

Проверил:
Кулешова Екатерина Анатольевна

Белгород 2022 г.

Лабораторная работа №6

Организация взаимодействия с базой данных через приложение с графическим интерфейсом

Цель работы: получение навыков разработки приложений для взаимодействия с базой данных, содержащих графический интерфейс пользователя.

Задание к работе

1. Изучить библиотеку для реализации приложения с графическим интерфейсом на выбранном языке программирования.
2. Разработать приложение с графическим интерфейсом, которое обеспечит подключение к базе данных, разработанной на основе предыдущих лабораторных работ, а также обеспечит выполнение запросов.

Выполнение работы:

Код программы:

```
import tkinter as tk
from tkinter import ttk, RIGHT, BOTTOM, X, Y
from console_app import ServiceBD

from functools import partial
from typing import List, Optional, Tuple, Callable


class MainApp(tk.Frame):
    def __init__(self, root, name_bd):
        super().__init__(root)
        self.scrollbar_table = None
        self.scrollbar_menu = None

        self.__db = ServiceBD(name_bd)
        self.table: Optional[ttk.Treeview] = None
        self.main_workplace = tk.Frame()

        self.__tables = {
            1: 'Клиенты',
            2: 'Статусы клиентов',
            3: 'Типы клиентов',
            4: 'Группы клиентов',
            5: 'Контактные данные',
            6: 'Типы контактных данных',
            7: 'Контактные данные и клиенты',
            8: 'Соц сети',
            9: 'Соц сети контактных данных',
            10: 'Фидбек',
            11: 'Сообщения',
            12: 'Шаблоны сообщений',
            13: 'Занятия',
            14: 'Типы занятий',
```

```
            15: 'Гости',
            16: 'Изделия',
            17: 'Типы фото',
            18: 'Типы изделий',
            19: 'Состояния готовности',
            20: 'Фото изделий',
            21: 'Типы активностей',
            22: 'Активности',
            23: 'Счёты',
            24: 'Сертификаты',
            25: 'Типы оплаты',
            26: 'Платежи',
            27: 'Изделия сертификатом',
            28: 'Гости в августе',
            29: 'Рейтинг топ5',
            30: 'Счеты клиентов по готовности'
        }

        self.__menu_tables = {
            1: (lambda:
                self.__db.select('clients'), ['id', 'name', 'fcs', 'contacts_id',
'client_type_id', 'client_status_id']
                ),

            2: (lambda:
                self.__db.select('client_statuses'), ['id', 'name', 'definition']
                ),

            3: (lambda:
                self.__db.select('client_types'), ['id', 'name', 'definition',
'start_cost']
                ),

            4: (lambda:
                self.__db.select('clients_group'), ['client_group_id', 'client_id',
'is_leader']
                ),

            5: (lambda:
                self.__db.select('contacts'),
                ['id', 'client_id', 'first_name', 'last_name', 'middle_name',
'phone', 'birthday', 'ban_on_spam']
                ),

            6: (lambda:
                self.__db.select('contacts_types'), ['id', 'name']
                ),

            7: (lambda:
                self.__db.select('clients_contacts'), ['contact_id', 'client_id',
'contacts_type_id']
                ),

            8: (lambda:
                self.__db.select('media'), ['id', 'name', 'site', 'is_legal',
'use_phone']
                ),

            9: (lambda:
                self.__db.select('contacts_media'), ['id', 'contacts_id', 'media_id',
'nickname', 'is_main']
                ),
```

```
            10: (lambda:
                 self.__db.select('feedback'), ['id', 'name']
                 ),

            11: (lambda:
                 self.__db.select('messages'),
                 ['id', 'theme', 'text', 'date_sent', 'contacts_media_id',
'feedback_id', 'template_id']
                 ),

            12: (lambda:
                 self.__db.select('message_templates'), ['id', 'theme',
'template_str']
                 ),

            13: (lambda:
                 self.__db.select('classes'), ['id', 'class_type_id', 'date', 'time',
'next_date', 'comment']
                 ),

            14: (lambda:
                 self.__db.select('class_types'), ['id', 'name', 'definition',
'start_cost']
                 ),

            15: (lambda:
                 self.__db.select('guests'), ['id', 'client_id', 'class_id',
'contacts_id']
                 ),

            16: (lambda:
                 self.__db.select('products'),
                 ['id', 'name', 'product_type_id', 'first_class_id',
'ready_state_id', 'definition', 'cost']
                 ),

            17: (lambda:
                 self.__db.select('photo_types'), ['id', 'name']
                 ),

            18: (lambda:
                 self.__db.select('product_types'), ['id', 'name', 'start_cost',
'definition']
                 ),

            19: (lambda:
                 self.__db.select('ready_states'), ['id', 'name', 'comment']
                 ),

            20: (lambda:
                 self.__db.select('product_photos'), ['id', 'class_id', 'product_id',
'photo_path', 'photo_type_id']
                 ),

            21: (lambda:
                 self.__db.select('activity_types'), ['id', 'name']
                 ),

            22: (lambda:
                 self.__db.select('activities'),
                 ['id', 'name', 'client_id', 'activity_type_id', 'comment',
```

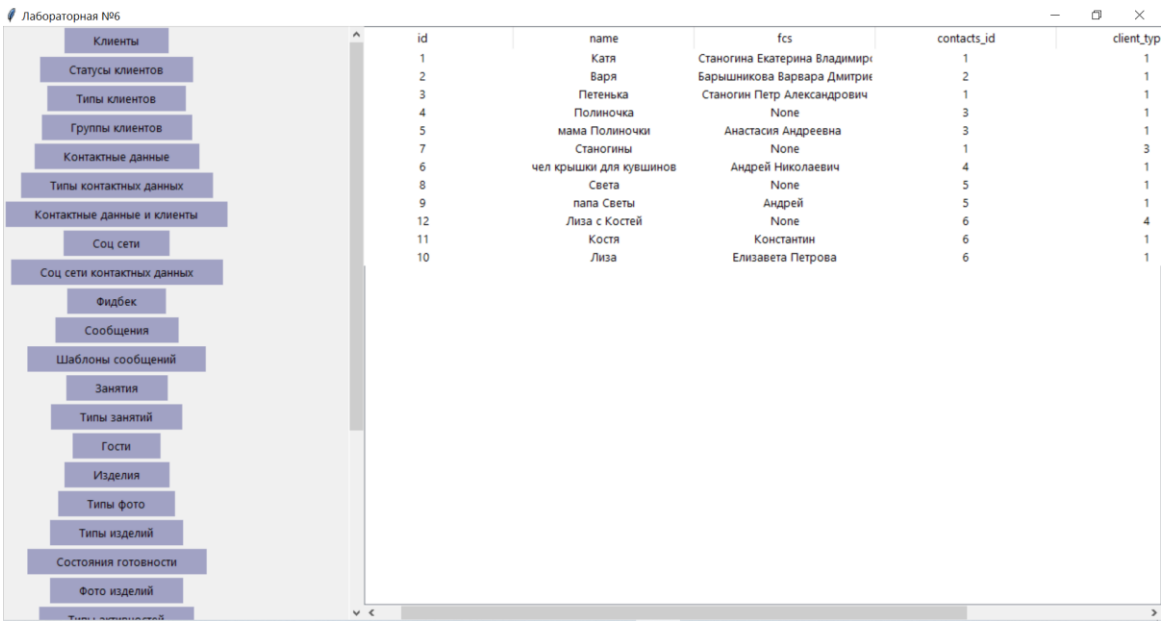
```
'product_id']
                ),

            23: (lambda:
                 self.__db.select('cheques'), ['id', 'client_id', 'product_id',
'class_id', 'name', 'sum', 'comment']
                ),

            24: (lambda:
                 self.__db.select('certificates'),
                 ['id', 'client_buyer_id', 'client_recipient_id', 'purchase_date',
'receiving_date', 'cost',
                  'cost_used',
                  'class_type_id', 'comment']
                ),

            25: (lambda:
                 self.__db.select('paid_type'), ['id', 'name']
                 ),

            26: (lambda:
                 self.__db.select('payments'), ['id', 'cheque_id', 'sum_paid',
'paid_type_id', 'certificate_id', 'date']
                 ),

            27: (lambda:
                 self.__db.select('products',
                                  fields=['products.id as "product_id"',
                                          'products.name as "product_name"',
                                          'payments.certificate_id as
"certificate_id"'],
                                  joins=[('cheques', 'cheques', 'products.id =
cheques.product_id '),
                                         ('payments', 'payments', 'payments.cheque_id
= cheques.id '),
                                          ('paid_type', 'paid_type',
'payments.paid_type_id = paid_type.id ')],
                                  group_by=['products.id',
'payments.certificate_id'],
                                  order_by='products.id',
                                  where=['(paid_type.name LIKE \'Сертификат%\')'],
                                  ), ['product_id', 'product_name',
'certificate_id']),

            28: (lambda:
                 self.__db.execute_select(f'SELECT clients.id as "client_id", '
                                          'clients.name as "guest_name"'
                                          'FROM guests '
                                          'JOIN clients on clients.id =
guests.client_id '
                                          'JOIN classes on guests.class_id =
classes.id '
                                          'WHERE (classes.date BETWEEN \'2022-07-
31\'::date AND \'2022-09-01\'::date) '

                                          'GROUP BY clients.id, clients.name '

                                          'ORDER BY clients.id'
                                          ';'
                                          ), ['client_id', 'guest_name']),

            29: (lambda:
```

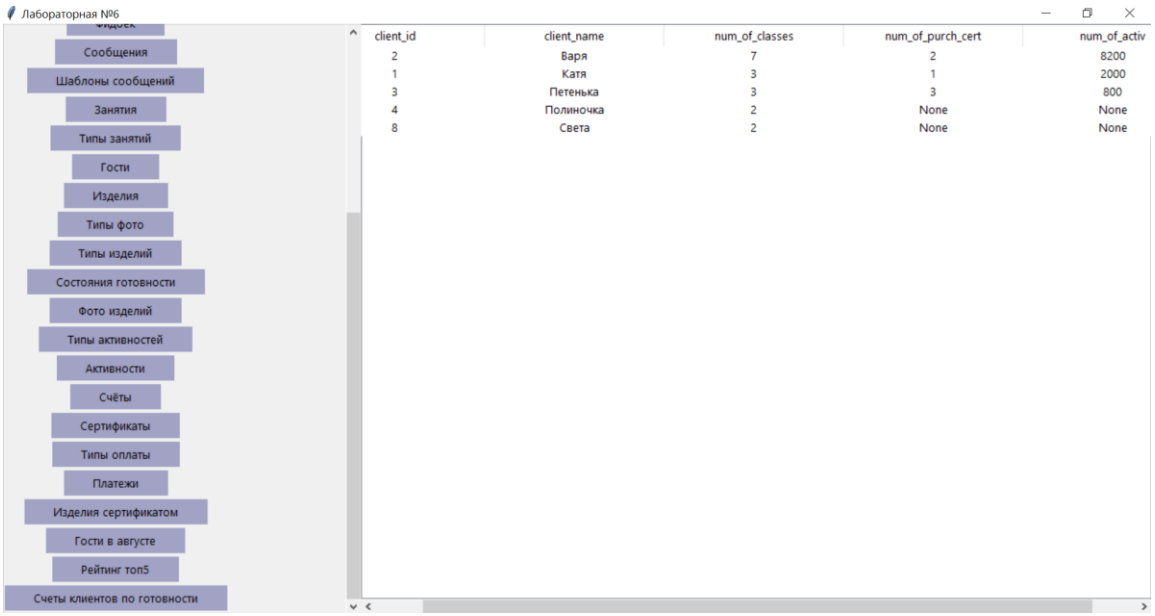
```
                self.__db.execute_select('SELECT *'
                                         'FROM (SELECT client_id,'
                                         'client_name,'
                                         'SUM(num_of_activities) as
"num_of_activities"'
                                         'FROM (SELECT clients.id           as
"client_id",'
                                         'clients.name         as "client_name",'
                                         'COUNT(DISTINCT g.id) as
"num_of_activities"'
                                         'FROM clients '
                                         'JOIN guests g on clients.id = g.client_id
'
                                         'GROUP BY clients.id '
                                         'UNION ALL '
                                         'SELECT cl.id, '
                                         'cl.name, '
                                         'COUNT(DISTINCT cert.id)'
                                         'FROM certificates cert'
                                         ' JOIN clients cl on cert.client_buyer_id =
cl.id'
                                         ' GROUP BY cl.id'
                                         ' UNION ALL'
                                         ' SELECT cl.id,'
                                         ' cl.name,'
                                         ' COUNT(DISTINCT act.id)'
                                         'FROM activities act'
                                         ' JOIN clients cl on act.client_id = cl.id'
                                         ' GROUP BY cl.id) AS
clients_dif_activities'
                                         ' GROUP BY client_id, client_name'
                                         ' ORDER BY num_of_activities DESC'
                                         ' LIMIT 5) AS top_activity'

                                         ' LEFT JOIN (SELECT ch.client_id as
"client_who_paid_id",'
                                         ' SUM(p.sum_paid)'
                                         'FROM cheques ch'
                                         ' JOIN payments p on ch.id = p.cheque_id'
                                         ' WHERE (p.date BETWEEN \'2021-12-
31\'::date AND \'2023-01-01\'::date)'
                                         'GROUP BY ch.client_id) AS payment ON
payment.client_who_paid_id = '
                                         'top_activity.client_id '
                                         ),
                ['client_id', 'client_name', 'num_of_classes', 'num_of_purch_cert',
'num_of_activ']
                ),

            30: (lambda:
                 self.__db.execute_select(
                     'SELECT clients.id as "client_id",'
                     'clients.name as "client_name",'
                     'SUM(case when clients.id = c.client_id and rs.id = 4 then c.sum
end) as "изделие готово",'
                     'SUM(case when clients.id = c.client_id and rs.id = 3 then c.sum
end) as "ждёт второй обжиг", '
                     'SUM(case when clients.id = c.client_id and rs.id = 2 then c.sum
end) as "готово к покраске", '
                     'SUM(case when clients.id = c.client_id and rs.id = 1 then c.sum
end) as "ещё сохнет"'
                     ''
```

```
                    'FROM clients'
                    ' JOIN cheques c on clients.id = c.client_id'
                    ' JOIN products p on c.product_id = p.id'
                    ' JOIN ready_states rs on p.ready_state_id = rs.id'
                    ' GROUP BY clients.id'
                    ' ORDER BY clients.id'
                ),
                ['client_id', 'client_name', 'изделие готово', 'ждёт второй обжиг',
'готово к покраске', 'ещё сохнет']
                )
        }

    def create_button_panel(self):
        self.main_workplace = tk.Frame()
        toolbar = tk.Frame(self.main_workplace, bg='#a1a2c4', bd=0)

        if self.scrollbar_menu is not None:
            self.scrollbar_menu.destroy()

        canvas = tk.Canvas(toolbar)
        self.scrollbar_menu = ttk.Scrollbar(toolbar, orient="vertical",
command=canvas.yview)
        scrollable_frame = tk.Frame(canvas)

        scrollable_frame.bind(
            "<Configure>",
            lambda e: canvas.configure(
                scrollregion=canvas.bbox("all")
            )
        )

        canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
        canvas.configure(yscrollcommand=self.scrollbar_menu.set, height=800)

        for option, name in self.__tables.items():
            tk.Button(
                scrollable_frame,
                text=name,
                command=partial(self.__show_result, self.__menu_tables[option]),
                bg='#a1a2c4',
                bd=0,
                padx=30,
                pady=2
            ).pack(side=tk.TOP, pady=2)

        self.scrollbar_menu.pack(fill=Y, side=RIGHT)
        self.main_workplace.pack(side=tk.TOP, fill=tk.X)
        canvas.pack(side='left', fill='both')
        toolbar.pack(side=tk.LEFT, fill=tk.X)

    def __show_result(self, params: Tuple[Callable, List[str]]):
        func, columns = params

        if self.table is not None:
            self.table.destroy()
        self.table = ttk.Treeview(self.main_workplace, columns=columns,
show='headings')

        for column in columns:
            self.table.column(column, anchor='center')
            self.table.heading(column, text=column, anchor='center')
        self.table.pack(side='top')
```

```
        for row in func():
            self.table.insert('', 'end', values=row)

        if self.scrollbar_table is not None:
            self.scrollbar_table.destroy()

        self.scrollbar_table = ttk.Scrollbar(self.main_workplace,
orient="horizontal", command=self.table.xview)
        self.table.configure(xscrollcommand=self.scrollbar_table.set)
        self.scrollbar_table.pack(fill=X, side=BOTTOM)

        self.table.pack(expand=tk.YES, fill=tk.BOTH)


if __name__ == '__main__':
    root = tk.Tk()

    app = MainApp(root, "len_ceramic")
    app.create_button_panel()
    app.pack()

    root.title("Лабораторная №6")
    root.geometry("500x600")
    root.mainloop()
```

Скриншоты работы программы:





Вывод: В процессе выполнения лабораторной работы были получены навыки разработки приложения для взаимодействия с базой данных, с использованием графического интерфейса.