Студент группы ПОКС-22 Петецкий А.А.

Практическое занятие №16

Тема: Разработка многооконного приложения для работы с однотабличной БД в IDE PyCharm Community.

Цель: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, работы с БД в IDE PyCharm Community.

Постановка задачи:

Приложение БЮРО ПО ТРУДОУСТРОЙСТВУ для некоторой организации. БД должна содержать таблицу Работник со следующей структурой записи: фамилия, имя, отчество, возраст, пол, профессия, стаж работы. БД должна обеспечивать получение информации о работнике по стажу работы.

Текст программы:

```
# вариант 17
# Приложение БЮРО ПО ТРУДОУСТРОЙСТВУ для некоторой организации.
# БД должна содержать таблицу Работник со следующей структурой записи:
# фамилия, имя, отчество, возраст, пол, профессия, стаж работы.
# БД должна обеспечивать получение информации о работнике по стажу работы.

# Для генерирования 100 рандомных работников запустите generator.py
# Для редактирования данных необходимо выбрать нужную строчку, а после этого выбрать
"Редактировать"
# Для удаления данных необходимо выбрать нужную строчку, а после этого выбрать "Удалить"

import tkinter as tk
from tkinter import ttk
import sqlite3 as sq
import re


class Main(tk.Frame):
    """Класс для главного окна"""

    def __init__(self, root):
        super().__init__(root)
        self.init_main()
        self.db = db
        self.view_records()

    def init_main(self):
        toolbar = tk.Frame(bg='#e3a874', bd=4)
        toolbar.pack(side=tk.TOP, fill=tk.X)

        self.add_img = tk.PhotoImage(file="plus.png")
        self.btn_open_dialog = tk.Button(toolbar, text='Добавить работника',
command=self.open_dialog, bg='#e6710b',
                                         bd=0,
                                         compound=tk.TOP, image=self.add_img)
        self.btn_open_dialog.pack(side=tk.LEFT)

        self.update_img = tk.PhotoImage(file="edit.png")
        btn_edit_dialog = tk.Button(toolbar, text="Редактировать",
command=self.open_update_dialog, bg='#e6710b',
                                    bd=0, compound=tk.TOP, image=self.update_img)
        btn_edit_dialog.pack(side=tk.LEFT)

        self.delete_img = tk.PhotoImage(file="delete.png")
```

```
        btn_delete = tk.Button(toolbar, text="Удалить запись", command=self.open_delete,
bg='#e6710b',
                               bd=0, compound=tk.TOP, image=self.delete_img)
        btn_delete.pack(side=tk.LEFT)

        self.search_img = tk.PhotoImage(file="search.png")
        btn_search = tk.Button(toolbar, text="Поиск записи", command=self.open_search_dialog,
bg='#e6710b',
                               bd=0, compound=tk.TOP, image=self.search_img)
        btn_search.pack(side=tk.LEFT)

        self.refresh_img = tk.PhotoImage(file="refresh.png")
        btn_refresh = tk.Button(toolbar, text="Обновить экран", command=self.view_records,
bg='#e6710b',
                                bd=0, compound=tk.TOP, image=self.refresh_img)
        btn_refresh.pack(side=tk.LEFT)

        self.tree = ttk.Treeview(self, columns=(
            'user_id', 'surname', 'name', 'patronymic', 'old', 'sex', 'profession',
'experience'), height=17,
                                 show='headings')

        self.tree.column('user_id', width=50, anchor=tk.CENTER)
        self.tree.column('surname', width=140, anchor=tk.CENTER)
        self.tree.column('name', width=140, anchor=tk.CENTER)
        self.tree.column('patronymic', width=140, anchor=tk.CENTER)
        self.tree.column('old', width=100, anchor=tk.CENTER)
        self.tree.column('sex', width=100, anchor=tk.CENTER)
        self.tree.column('profession', width=140, anchor=tk.CENTER)
        self.tree.column('experience', width=100, anchor=tk.CENTER)

        self.tree.heading('user_id', text='ID')
        self.tree.heading('surname', text='Фамилия')
        self.tree.heading('name', text='Имя')
        self.tree.heading('patronymic', text='Отчество')
        self.tree.heading('old', text='Возраст')
        self.tree.heading('sex', text='Пол')
        self.tree.heading('profession', text='Профессия')
        self.tree.heading('experience', text='Стаж работы')

        self.tree.pack()

    def records(self, surname, name, patronymic, old, sex, profession, experience):
        is_valid, msg = self.clean_data(surname, name, patronymic, old, profession, experience)
        if is_valid:
            self.db.insert_data(surname, name, patronymic, old, sex, profession, experience)
            self.view_records()
        else:
            self.open_error(msg)

    def update_record(self, surname, name, patronymic, old, sex, profession, experience):
        is_valid, msg = self.clean_data(surname, name, patronymic, old, profession, experience)
        if is_valid:
            try:
                self.db.cur.execute(
                    """UPDATE Rabotnik SET surname=?, name=?, patronymic=?, old=?, sex=?,
profession=?, experience=? WHERE user_id=?""",
                    (surname, name, patronymic, old, sex, profession, experience,
                     self.tree.set(self.tree.selection()[0], '#1')))
                self.db.con.commit()
                self.view_records()
            except IndexError:
                self.open_error('Вы не выбрали запись')
        else:
```

```
            self.open_error(msg)

    def view_records(self):
        self.db.cur.execute("""SELECT * FROM Rabotnik""")
        [self.tree.delete(i) for i in self.tree.get_children()]
        [self.tree.insert('', 'end', values=row) for row in self.db.cur.fetchall()]

    def delete_records(self):
        for selection_item in self.tree.selection():
            self.db.cur.execute("""DELETE FROM Rabotnik WHERE user_id=?""",
(self.tree.set(selection_item, '#1'),))
        self.db.con.commit()
        self.view_records()

    def search_records(self, experience):
        experience = (experience,)
        self.db.cur.execute("""SELECT * FROM Rabotnik WHERE experience>=?""", experience)
        [self.tree.delete(i) for i in self.tree.get_children()]
        [self.tree.insert('', 'end', values=row) for row in self.db.cur.fetchall()]

    def clean_data(self, surname, name, patronymic, old, profession, experience):
        if re.findall(r'[^a-zA-ZА-ЯёЁа-я]', surname):
            return False, 'Неверно указана фамилия\n(Проверьте на наличие лишних пробелов)'
        if re.findall(r'^\Z', surname):
            return False, 'Укажите фамилию'

        if re.findall(r'[^a-zA-ZА-ЯёЁа-я]', name):
            return False, 'Неверно указано имя\n(Проверьте на наличие лишних пробелов)'
        if re.findall(r'^\Z', name):
            return False, 'Укажите имя'

        if re.findall(r'[^a-zA-ZА-ЯёЁа-я]', patronymic):
            return False, 'Неверно указано отчество\n(Проверьте на наличие лишних пробелов)'
        if re.findall(r'^\Z', patronymic):
            return False, 'Укажите отчество'

        if not (old.isdigit() and int(old) >= 14 and int(old) <= 80):
            return False, 'Неверно указан возраст\n(Возраст должен быть >=14 и <=80 лет)'
        if re.findall(r'^\Z', old):
            return False, 'Укажите возраст'

        if re.findall(r'[^a-zA-ZА-ЯёЁа-я]', profession):
            return False, 'Неверно указана профессия\n(Проверьте на наличие лишних пробелов)'
        if re.findall(r'^\Z', profession):
            return False, 'Укажите профессию'

        if not (experience.isdigit() and int(experience) >= 1 and int(experience) <= int(old) -
14):
            return False, 'Неверно указан стаж работы\n(Стаж работы не может быть > (Возраст-
14))'
        if re.findall(r'^\Z', experience):
            return False, 'Укажите стаж работы'

        return True, ''

    def open_error(self, msg: str):
        Error(root, app, msg)

    def open_delete(self):
        Delete(root, app)

    def open_dialog(self):
        Child(root, app)
```

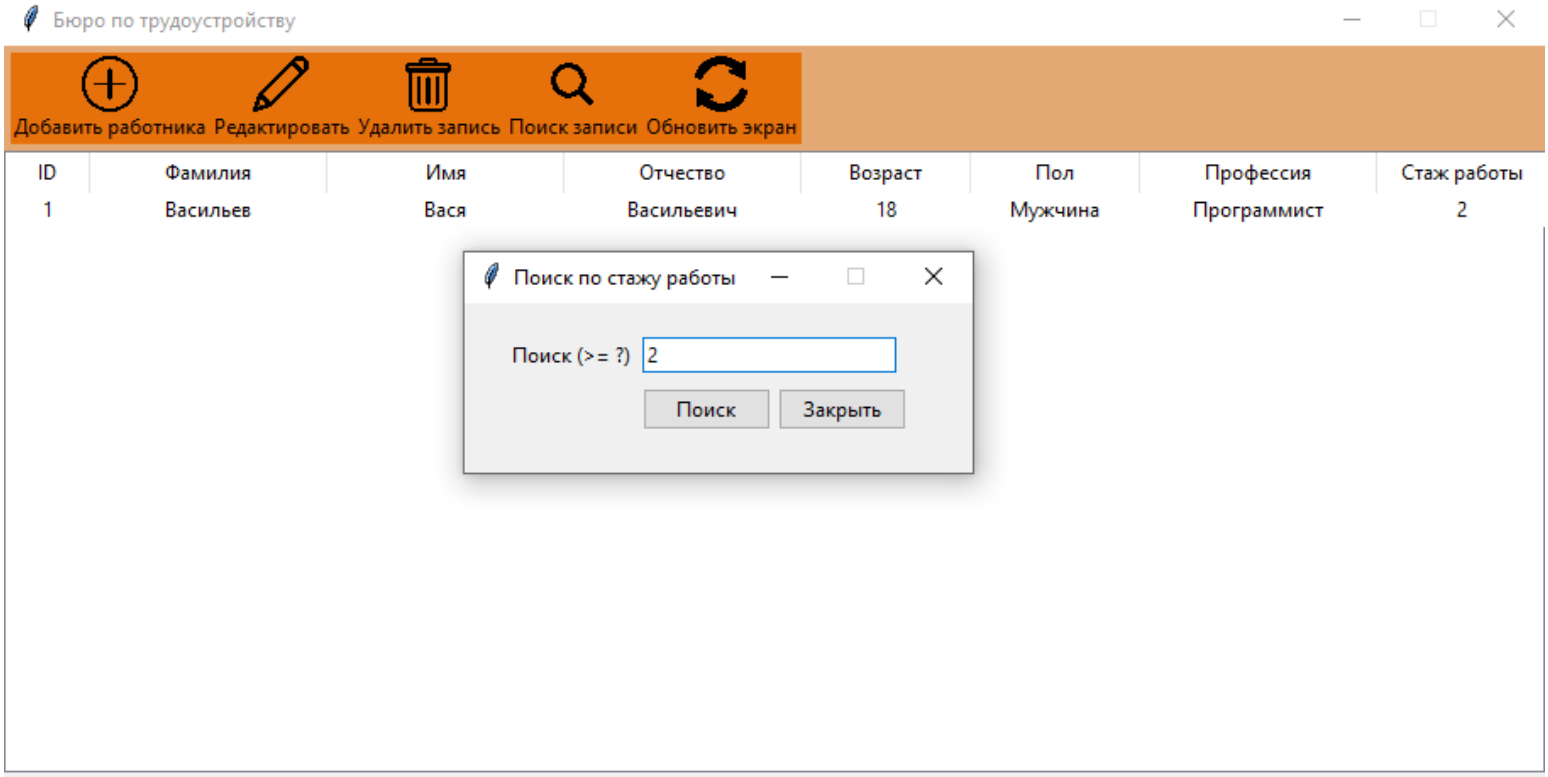
```
    def open_update_dialog(self):
        Update()

    def open_search_dialog(self):
        Search()


class Child(tk.Toplevel):
    """Класс окна добавления работников"""

    def __init__(self, root, app):
        super().__init__(root)
        self.init_child()
        self.view = app

    def init_child(self):
        self.title('Добавить работника')
        self.geometry('400x260+400+300')
        self.resizable(False, False)

        label_surname = tk.Label(self, text='Фамилия')
        label_surname.place(x=50, y=25)
        self.entry_surname = ttk.Entry(self)
        self.entry_surname.place(x=145, y=25)

        label_name = tk.Label(self, text='Имя')
        label_name.place(x=50, y=50)
        self.entry_name = ttk.Entry(self)
        self.entry_name.place(x=145, y=50)

        label_patronymic = tk.Label(self, text='Отчество')
        label_patronymic.place(x=50, y=75)
        self.entry_patronymic = ttk.Entry(self)
        self.entry_patronymic.place(x=145, y=75)

        label_old = tk.Label(self, text='Возраст')
        label_old.place(x=50, y=100)
        self.entry_old = ttk.Entry(self)
        self.entry_old.place(x=145, y=100)

        label_sex = tk.Label(self, text='Пол')
        label_sex.place(x=50, y=125)
        self.combobox = ttk.Combobox(self, values=[u'Мужчина', u'Женщина'], state="readonly")
        self.combobox.current(0)
        self.combobox.place(x=145, y=125)

        label_profession = tk.Label(self, text='Профессия')
        label_profession.place(x=50, y=150)
        self.entry_profession = ttk.Entry(self)
        self.entry_profession.place(x=145, y=150)

        label_experience = tk.Label(self, text='Стаж работы')
        label_experience.place(x=50, y=175)
        self.entry_experience = ttk.Entry(self)
        self.entry_experience.place(x=145, y=175)

        btn_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)
        btn_cancel.place(x=300, y=220)

        self.btn_ok = ttk.Button(self, text='Добавить')
        self.btn_ok.place(x=220, y=220)
        self.btn_ok.bind('<Button-1>', lambda event: self.view.records(self.entry_surname.get(),
                                                                       self.entry_name.get(),
```

```
self.entry_patronymic.get(),
                                                    self.entry_old.get(),
                                                    self.combobox.get(),

self.entry_profession.get(),

self.entry_experience.get()))

        self.grab_set()
        self.focus_set()


class Update(Child):
    """Класс окна редактирования данных о работниках"""

    def __init__(self):
        super().__init__(root, app)
        self.init_edit()
        self.view = app

    def init_edit(self):
        self.title("Редактировать запись")
        btn_edit = ttk.Button(self, text="Редактировать")
        btn_edit.place(x=205, y=220)
        btn_edit.bind('<Button-1>', lambda event:
self.view.update_record(self.entry_surname.get(),
                                                        self.entry_name.get(),

self.entry_patronymic.get(),
                                                        self.entry_old.get(),
                                                        self.combobox.get(),

self.entry_profession.get(),

self.entry_experience.get()))
        self.btn_ok.destroy()

        self.combobox = ttk.Combobox(self, values=[u'Мужчина', u'Женщина'], state="readonly")
        self.combobox.place(x=145, y=125)


class Search(tk.Toplevel):
    """Класс окна поиска работников по стажу работы"""

    def __init__(self):
        super().__init__()
        self.init_search()
        self.view = app

    def init_search(self):
        self.title("Поиск по стажу работы")
        self.geometry("300x100+400+300")
        self.resizable(False, False)

        label_search = tk.Label(self, text="Поиск (>= ?)")
        label_search.place(x=25, y=20)

        self.entry_search = ttk.Entry(self)
        self.entry_search.place(x=105, y=20, width=150)

        btn_cancel = ttk.Button(self, text="Закрыть", command=self.destroy)
        btn_cancel.place(x=185, y=50)

        btn_search = ttk.Button(self, text="Поиск")
```

```
        btn_search.place(x=105, y=50)
        btn_search.bind('<Button-1>', lambda event:
self.view.search_records(self.entry_search.get()))
        btn_search.bind('<Button-1>', lambda event: self.destroy(), add='+')

        self.grab_set()
        self.focus_set()


class Error(tk.Toplevel):
    def __init__(self, root, app, msg: str) -> None:
        super().__init__(root)
        self.init_error(msg)
        self.view = app

    def init_error(self, msg):
        self.title("Ошибка!")
        self.geometry("300x100+450+350")
        self.resizable(False, False)

        label_error = tk.Label(self, text=msg)
        label_error.place(x=28, y=20)

        btn_cancel = ttk.Button(self, text="Закрыть", command=self.destroy)
        btn_cancel.place(x=185, y=65)

        self.grab_set()
        self.focus_set()


class Delete(tk.Toplevel):
    def __init__(self, root, app):
        super().__init__(root)
        self.init_delete()
        self.view = app

    def init_delete(self):
        self.title("Предупреждение!")
        self.geometry("300x100+450+350")
        self.resizable(False, False)

        label_error = tk.Label(self, text='Вы уверены, что хотите удалить запись?')
        label_error.place(x=28, y=20)

        btn_yes = ttk.Button(self, text="Удалить")
        btn_yes.place(x=100, y=60)
        btn_yes.bind('<Button-1>', lambda event: self.view.delete_records())
        btn_yes.bind('<Button-1>', lambda event: self.destroy(), add='+')

        btn_cancel = ttk.Button(self, text="Закрыть", command=self.destroy)
        btn_cancel.place(x=185, y=60)

        self.grab_set()
        self.focus_set()


class DB:
    """Класс базы данных"""

    def __init__(self):
        with sq.connect('Rabotnik.db') as self.con:
            self.cur = self.con.cursor()
            self.cur.execute("""CREATE TABLE IF NOT EXISTS Rabotnik (
                user_id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
                surname TEXT NOT NULL,
                name TEXT NOT NULL,
                patronymic TEXT NOT NULL,
                old INTEGER,
                sex INTEGER NOT NULL,
                profession NOT NULL,
                experience INTEGER
                )""")

    def insert_data(self, surname, name, patronymic, old, sex, profession, experience):
        self.cur.execute(
            """INSERT INTO Rabotnik(surname, name, patronymic, old, sex, profession, experience)
VALUES (?, ?, ?, ?, ?, ?, ?)""",
            (surname, name, patronymic, old, sex, profession, experience))
        self.con.commit()


if __name__ == "__main__":
    root = tk.Tk()
    db = DB()
    app = Main(root)
    app.pack()
    root.title("Бюро по трудоустройству")
    root.geometry("910x450+300+250")
    root.resizable(False, False)
    root.mainloop()
```

Протокол работы программы:



Вывод:

В процессе выполнения практического занятия я закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, работы с БД в IDE PyCharm Community.