Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Базы данных

ОТЧЕТ
по лабораторной работе № 2
на тему
«Создание приложения для базы данных»
ВАРИАНТ №18 – Магазин продуктов

Студент:

Преподаватель:

МИНСК 2024

1. ЦЕЛЬ РАБОТЫ

Создать прикладную программу для работы с базой данных и выполняющую заданные транзакции, а также реализовать механизм работы с базой данных (добавление новых данных в таблицу, удаление, обновление).

2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1) Определиться с выбором операционной системы, среды разработки и языка программирования для написания программы для работы с базой данных.

2) Правила выполнения задания:

- Используя выбранный язык программирования, написать код для выполнения заданных транзакций;
- Писать запрос в приложении нельзя! Нужно реализовать интерфейс вывода запросов из 4 и 5 лабораторной работы;
- Проверить функциональность моего приложения с помощью различных тестов.

3) Оформить отчет.

3. ТРЕБОВАНИЯ К СИСТЕМЕ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

В первую очередь начнем работу с подборки конфигурации для приложения. Для создания прикладной программы для работы с базой данных была выбрана следующая программная структура:

- Операционная система – Windows 10;
- Среда разработки - PyCharm Community Edition 2023.2.4;
- Язык программирования – Python;
- Версия Python – 3.8.2.

В связи с данной программной структурой определены следующие минимальные требования для запуска данного приложения:

1) Операционная система: Поддерживаются следующие операционные системы:
   - Windows 7 и выше;
   - macOS 10.11 и выше;
   - Linux совместимый с Python 3.
2) Процессор: Минимум 1.0 ГГц или выше для обеспечения достаточной производительности в процессе работы приложения.
3) Оперативная память: Рекомендуется иметь не менее 2 ГБ оперативной памяти для плавного функционирования приложения.
4) Объем памяти: Рекомендуется иметь не менее 100 МБ свободного места на диске для надежного хранения временных файлов и данных приложения.

Так же, исходя из программной структуры, были определены требования окружения и программное обеспечение, необходимые для корректной работы приложения:

1) Версия Python: Рекомендуется использовать версию Python 3.6 или более позднюю.
2) Библиотеки: В данной программе используются такие библиотеки как psycopg2 и tkinter. Данные модули обязаны быть установлены на вашем устройстве перед запуском приложения.
3) Доступ к базе данных PostgreSQL: Приложение работает с базой данных PostgreSQL. Убедитесь, что имеется доступ к PostgreSQL серверу с помощью учетных данных, указанных в параметрах подключения.
4) Доступ к интернету: Приложение не требует постоянного подключения к интернету для своей работы. Однако, для установки зависимостей из сети или получения обновлений может потребоваться временное соединение.
5) Разрешения пользователя: Пользователь, запускающий приложение, должен иметь достаточные разрешения для доступа к файлам и

ресурсам, необходимым для работы приложения, а также для выполнения операций записи и чтения в базу данных.

4. ОПЕРАЦИОННЫЕ АЛГОРИТМЫ И КЛЮЧЕВЫЕ ФУНКЦИОНАЛЬНЫЕ АСПЕКТЫ

Для активации программы можно воспользоваться двумя способами.

1) Запуск исполняемого файла main.exe.

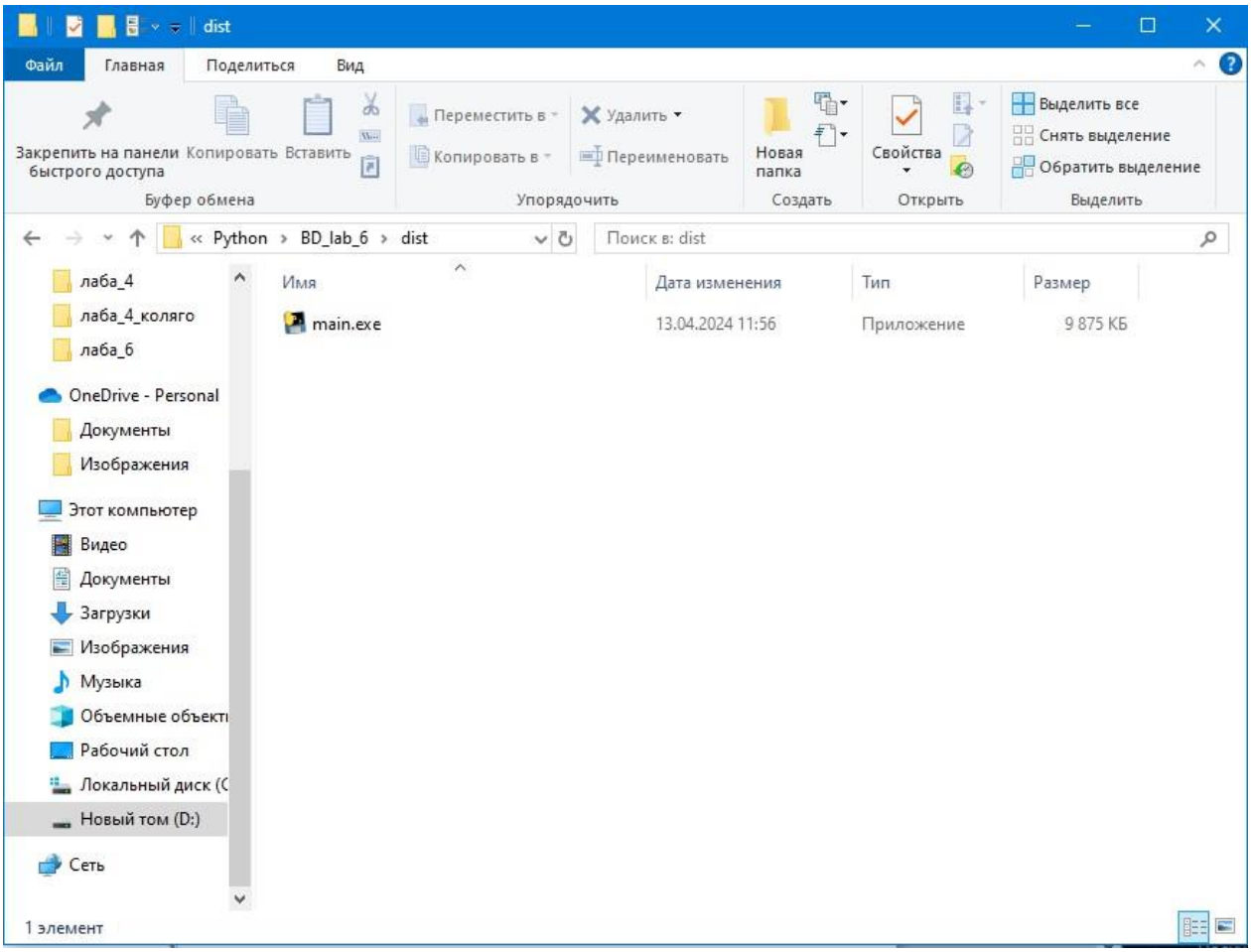

Рисунок 4.1 - Использование файла main.exe

2) Непосредственно из самой среды разработки.

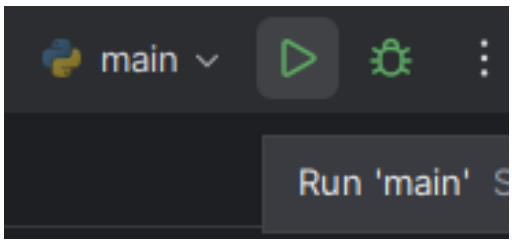

Рисунок 4.2 - Использование компилятора из среды разработки

Каждый из способов в конечном итоге откроют окно, в котором сразу же будет отображено главное меню программы.

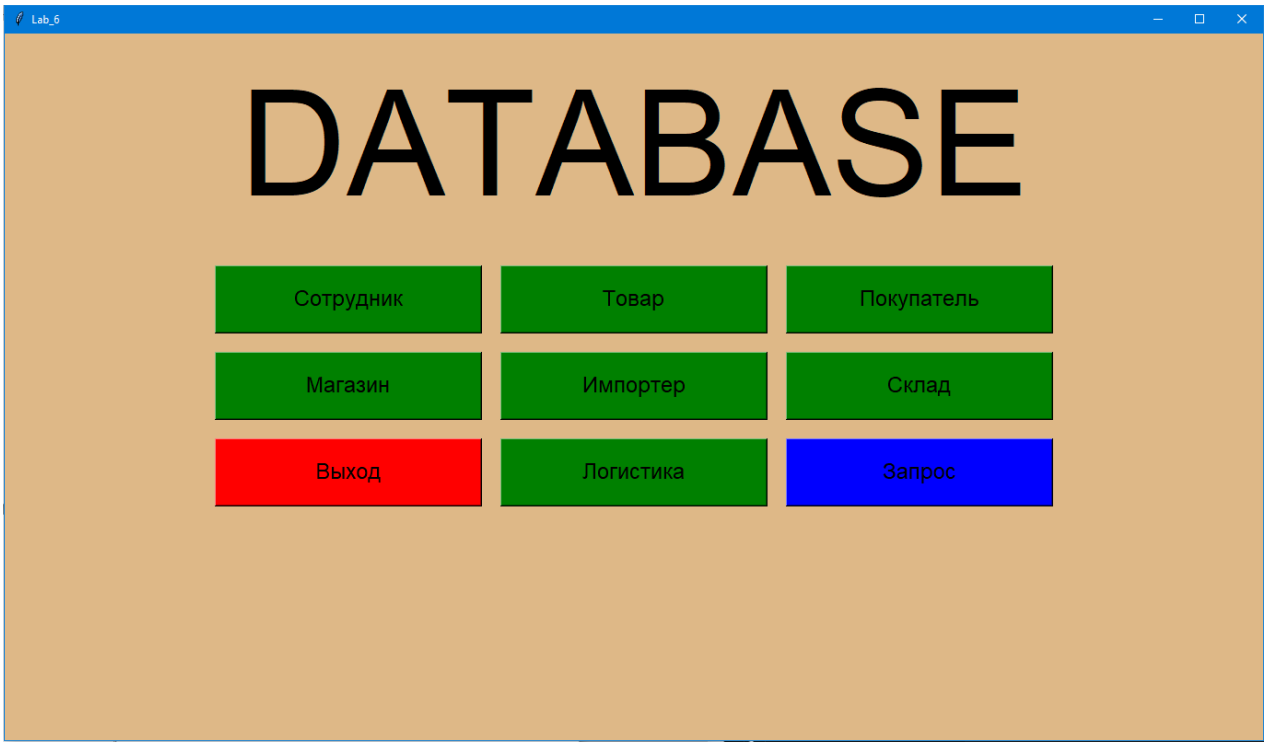

Рисунок 4.3 – Главное меню программы

В главном меню представлено 9 кнопок, с каждой из которых пользователь может взаимодействовать. Перечень всех кнопок указан ниже.

- Кнопка «Сотрудник» - позволяет взаимодействовать с таблицей Сотрудник;
- Кнопка «Товар» - позволяет взаимодействовать с таблицей Товар;
- Кнопка «Покупатель» - позволяет взаимодействовать с таблицей Покупатель;
- Кнопка «Магазин» - позволяет взаимодействовать с таблицей Магазин;
- Кнопка «Импортер» - позволяет взаимодействовать с таблицей Импортер;
- Кнопка «Склад» - позволяет взаимодействовать с таблицей Склад;
- Кнопка «Логистика» - позволяет взаимодействовать с таблицей Логистика;
- Кнопка «Запрос» - позволяет ввести запрос;
- Кнопка «Выход» - позволяет выйти из программы.

При нажатии на кнопку, которая отвечает за взаимодействие с таблицей базы данных, появляется еще одно окно. В этом окне пользователю предлагаются 3 кнопки, для быстрой работы с раннее выбранной таблицей.

Для более наглядно примера нажмем на кнопку «Сотрудник» и в следствии с этим программа будет работать с данной таблицей.

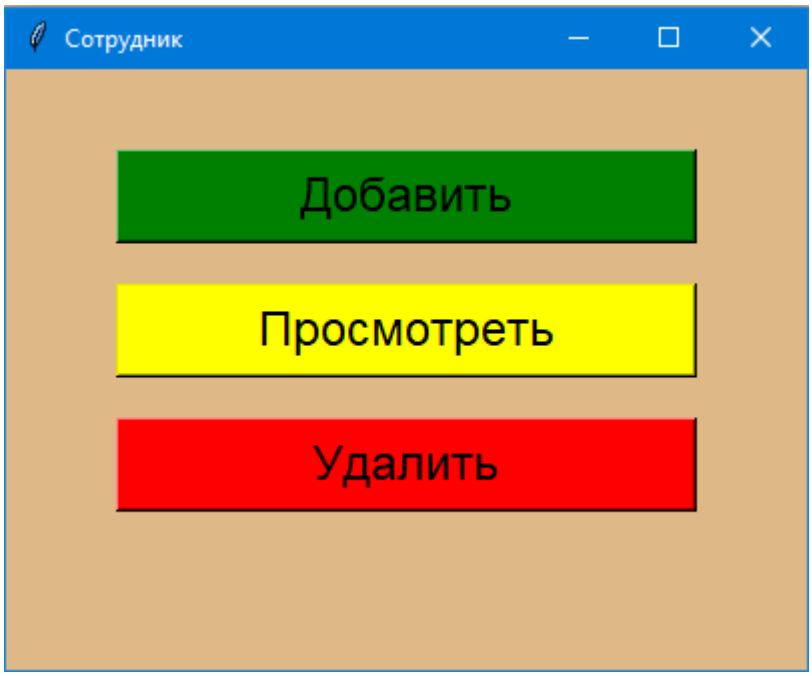

Рисунок 4.4 – Окно взаимодействия с кнопкой «Сотрудник»

Каждая из кнопок позволяет пользователю выполнить следующие действия:

- Кнопка «Добавить» - позволяет запись в выбранную таблицу;
- Кнопка «Просмотреть» - позволяет полностью просмотреть все записи в выбранной таблице;
- Кнопка «Удалить» - позволяет удалить запрос из выбранной таблицы.

Нажмем на кнопку «Добавить». После этого появляется еще одно окно. В этом окне выводятся все те поля, которые пользователю необходимо заполнить.

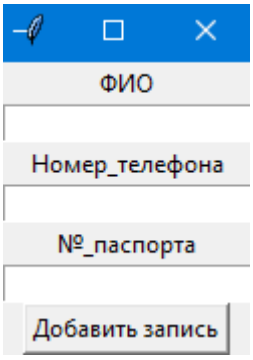

Рисунок 4.5 – Окно добавления записи в таблицу

Если все поля были заполнены корректно, тогда запись будет успешно добавлена в таблицу.

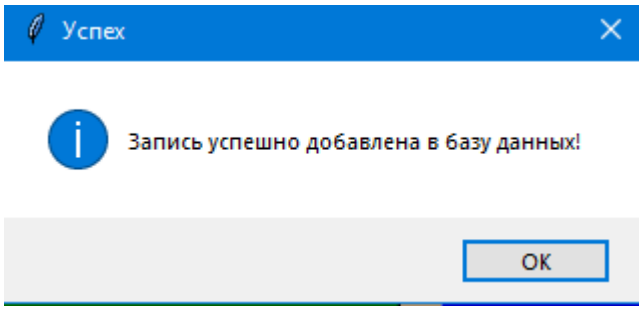

Рисунок 4.6 – Окно успешного добавления записи в таблицу

Теперь воспользуемся кнопкой «Просмотреть». Ее функционал заключается в том, что при нажатии на нее откроется окно со всеми запросами в выбранной таблице.

сотрудник	id_сотрудника	ФИО	Магазин_id	Номер_телефона	№_паспорта
	1	Иванов Петр Иванович	None	+375251234567	7977246A001PB9
	2	Петров Алексей Сергеевич	None	+375292345678	8642103B002LM8
	3	Сидорова Ольга Николаевна	None	+375333456789	9571364C003QC7
	4	Козлова Елена Владимировна	None	+375444567890	2437568D004RB6
	5	Смирнов Игорь Дмитриевич	None	+375255678901	5134679E005VD5
	6	Федорова Анна Александровна	None	+375296789012	9724805F006ZC4
	7	Морозов Павел Игоревич	None	+375337890123	7429081G007XY3
	8	Николаев Денис Александрович	None	+375448901234	8406952H008WT2
	9	Иванова Мария Степановна	None	+375259012345	3165720I009UT1
	10	Сергеев Александр Викторович	None	+375290123456	6049257J010SR0

Рисунок 4.7 – Окно просмотра записей таблицы

В данном окне присутствуют несколько столбцов.

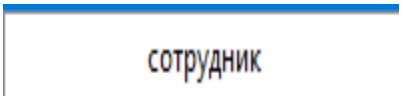

Рисунок 4.8 – 1 столбец – название таблицы

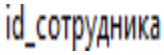

Рисунок 4.9 – 2 столбец – id запроса

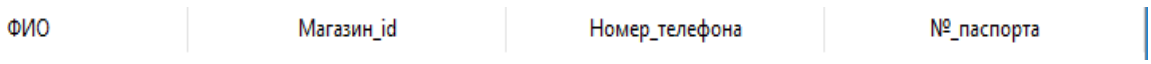

Рисунок 4.10 – 3 и последующий столбцы – поля таблицы

Так же можем убедиться, что ранее вписанный запрос был успешно добавлен в данную таблицу.

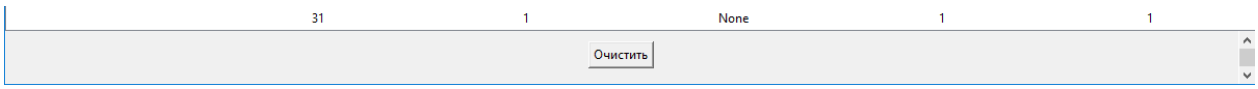

Рисунок 4.11 – Ранее добавленный запрос

Еще в открывшемся окне после нажатия кнопки «Просмотреть» имеется кнопка «Очистить». При ее нажатии происходит полное удаление всех записей из таблицы.

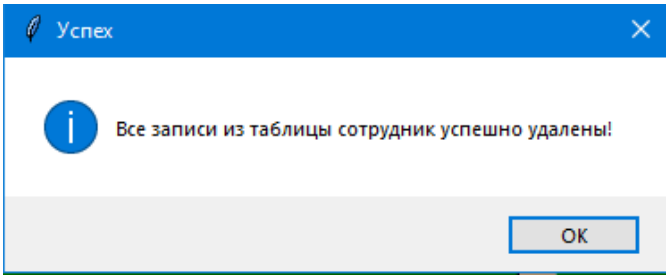

Рисунок 4.12 – Успешное удаление всех записей из таблицы

И, при последующем просмотре таблицы можно убедиться в том, что все данные были стерты.

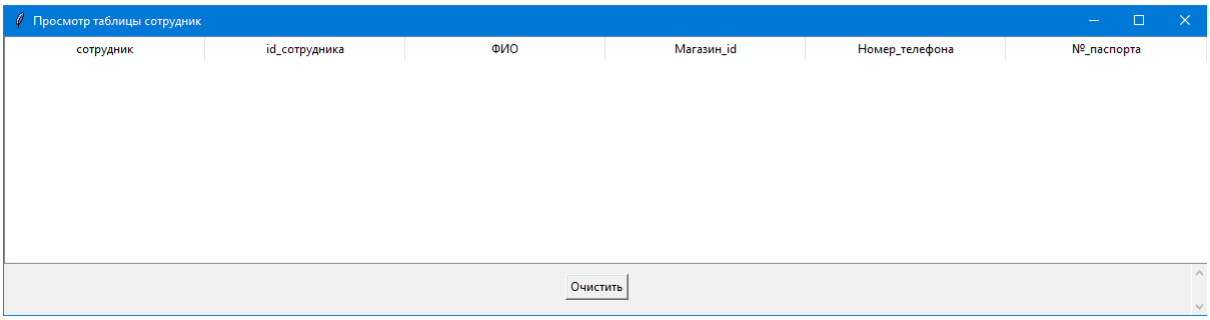

Рисунок 4.13 – Вид таблицы после нажатия на кнопку «Очистить»

Возвращаемся в окно взаимодействия с таблицей. В данном окне присутствует еще одна кнопка «Удалить». При ее нажатии открывается окно, в котором пользователю предлагается удалить запись исходя из ее id.

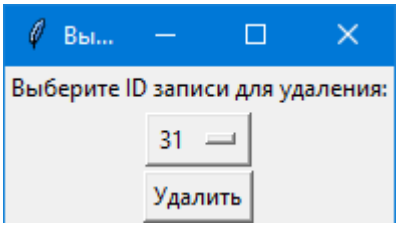

Рисунок 4.14 – Окно выбора записи для удаления

При успешном удалении появится окно, которое показано на рисунке 4.15.

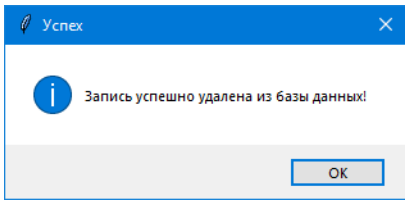

Рисунок 4.15 – Успешное удаление записи из таблицы

Вернемся в главное меню. При нажатии на кнопку «Запрос» появляется окно, в котором пользователь может ввести команду, синтаксис которой должен быть аналогичен запросам, которые используются в приложении pgAdmin4.

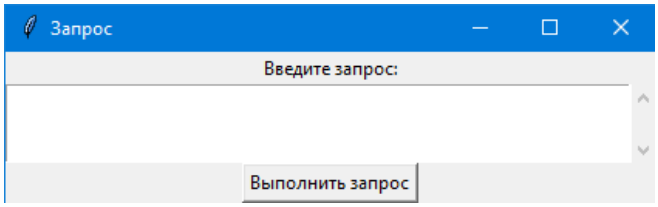

Рисунок 4.16 – Окно запроса

В данном окне требуется заполнить поле и затем нажать на кнопку «Выполнить запрос». Если запрос будет введен корректно, тогда после его выполнения, тогда будут выведена та информация, которая была написана пользователем в запросе.

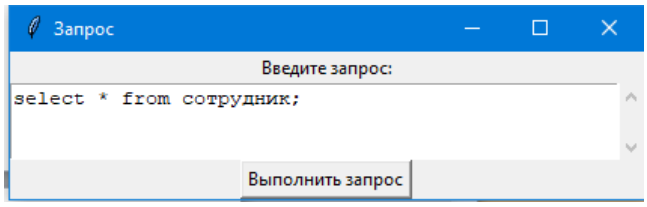

Рисунок 4.17 – Пример корректного запроса

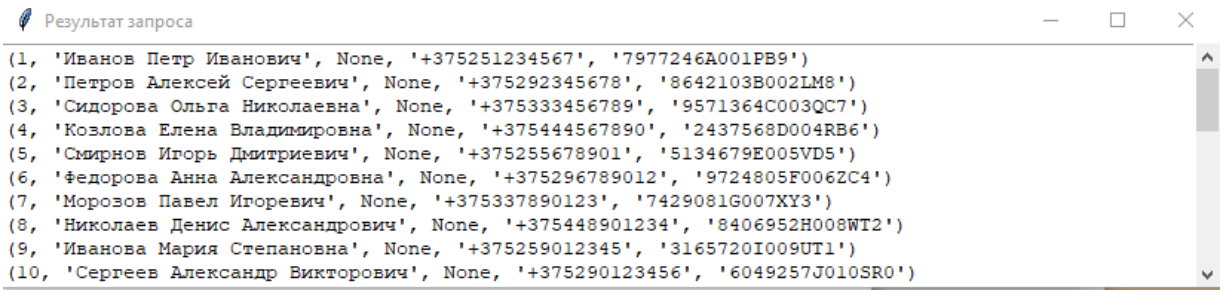

Рисунок 4.18 – Результат ранее веденного запроса

Приложение будет работать до тех пор, пока пользователь не нажмет на кнопку «Выход», либо пока не закроет окно главного меню.

5. ЛИСТИНГ КОДА

main.py

```
import tkinter as tk
from view_db import ViewWindow
from add_db import AddWindow
from delete_db import DelWindow
from request_db import ReqWindow


def create_operation_window(entity):
    operation_window = tk.Toplevel()
    operation_window.title(entity)
    operation_window.geometry("400x300")
    operation_window.configure(bg="burlywood")

    button_frame = tk.Frame(operation_window, bg="burlywood")
    button_frame.pack(pady=30)

    operations = ["Добавить", "Просмотреть", "Удалить"]
    for operation in operations:
        button = tk.Button(button_frame, text=operation, font=("Arial",
18), width=20, height=1)
        if operation == "Добавить":
            button.config(command=lambda:       AddWindow(entity.lower()),
bg="green")
        elif operation == "Просмотреть":
            button.config(command=lambda:      ViewWindow(entity.lower()),
bg="yellow")
        elif operation == "Удалить":
            button.config(command=lambda:       DelWindow(entity.lower()),
bg="red")
        button.pack(pady=10)


root = tk.Tk()
root.title("Lab_6")
root.geometry("1366x768")
root.configure(bg="burlywood")

label = tk.Label(root, text="DATABASE", font=("Arial", 120), pady=30,
bg="burlywood")
label.pack()

button_frame = tk.Frame(root, bg="burlywood")
button_frame.pack()

entities = ["Сотрудник", "Товар", "Покупатель", "Магазин", "Импортер",
"Склад", "Выход", "Логистика", "Запрос"]
positions = [(0, 0), (0, 1), (0, 2),
             (1, 0), (1, 1), (1, 2),
             (2, 0), (2, 1), (2, 2)]

for entity, pos in zip(entities, positions):
    if entity == "Выход":
        button = tk.Button(button_frame, text=entity, font=("Arial",
18), width=20, height=2, command=root.quit,
```

```
                                bg="red")
            elif entity == "Запрос":
                button = tk.Button(button_frame, text=entity, font=("Arial",
18), width=20, height=2,
                                command=lambda: ReqWindow(), bg="blue")
            else:
                button = tk.Button(button_frame, text=entity, font=("Arial",
18), width=20, height=2,
                                command=lambda                          e=entity:
create_operation_window(e), bg="green")
            button.grid(row=pos[0], column=pos[1], padx=10, pady=10)

        root.mainloop()

        add_db.py

        import tkinter as tk
        import psycopg2
        from tkinter import messagebox

        conn_params = {
            "host": "localhost",
            "port": "5432",
            "database": "postgres",
            "user": "postgres",
            "password": ""
        }

        class AddWindow:
            def __init__(self, entity):
                self.entity = entity
                self.window = tk.Toplevel()
                self.window.title(f"Добавление записи в {entity}")
                self.fields = {}
                self.create_input_fields()

            def create_input_fields(self):
                connection = psycopg2.connect(**conn_params)
                cursor = connection.cursor()
                cursor.execute("SELECT                column_name                FROM
information_schema.columns WHERE table_name = %s", (self.entity,))
                columns_info = cursor.fetchall()
                connection.close()

                fields_info = {info[0]: info[0] for info in columns_info if "id"
not in info[0].lower()}
                for field, label_text in fields_info.items():
                    label = tk.Label(self.window, text=label_text)
                    label.pack()
                    entry = tk.Entry(self.window)
                    entry.pack()
                    self.fields[field] = entry

                button = tk.Button(self.window, text="Добавить запись",
command=self.add_record_to_database)
                button.pack()

            def add_record_to_database(self):
                values = {field: entry.get() for field, entry in
self.fields.items()}
                try:
                    connection = psycopg2.connect(**conn_params)
                    cursor = connection.cursor()
```

```
                    cursor.execute(f"SELECT       MAX(id_{self.entity}a)       FROM
{self.entity}")
                    max_id = cursor.fetchone()[0]
                    next_id = max_id + 1 if max_id is not None else 1
                    values[f"id_{self.entity}a"] = next_id
                    columns = ', '.join(values.keys())
                    placeholders = ', '.join(['%s' for _ in values])
                    sql   =   f"INSERT   INTO   {self.entity}   ({columns})   VALUES
({placeholders})"
                    cursor.execute(sql, tuple(values.values()))
                    connection.commit()
                    connection.close()
                    messagebox.showinfo("Успех",  "Запись  успешно  добавлена  в
базу данных!")
                    self.window.destroy()
                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось добавить запись:
{e}")
```

view_db.py

```
import tkinter as tk
from tkinter import ttk, messagebox
import psycopg2

class ViewWindow(tk.Toplevel):
    def __init__(self, entity):
        super().__init__()
        self.entity = entity
        self.title(f"Просмотр таблицы {entity}")

        self.tree = ttk.Treeview(self)
        self.tree.pack(expand=True, fill=tk.BOTH)

        scrollbar      =      ttk.Scrollbar(self,       orient="vertical",
command=self.tree.yview)
        scrollbar.pack(side="right", fill="y")

        self.tree.configure(yscrollcommand=scrollbar.set)

        self.get_data(entity)

        clear_button = tk.Button(self, text="Очистить", command=lambda:
self.clear_table(entity))
        clear_button.pack(pady=10)

    def get_data(self, entity):
        try:
            conn = psycopg2.connect(
                host="localhost",
                port="5432",
                database="postgres",
                user="postgres",
                password=""
            )
            cursor = conn.cursor()

            cursor.execute(f"SELECT * FROM {entity}")

            columns = [desc[0] for desc in cursor.description]

            self.tree["columns"] = columns
            self.tree.heading("#0", text=entity)
```

```
                    for col in columns:
                        self.tree.heading(col, text=col)
                        self.tree.column(col, anchor=tk.CENTER)

                    self.tree.delete(*self.tree.get_children())

                    for row in cursor.fetchall():
                        self.tree.insert("", "end", values=row)

                    cursor.close()
                    conn.close()

                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось получить данные
из таблицы: {e}")

            def clear_table(self, entity):
                try:
                    conn = psycopg2.connect(
                        host="localhost",
                        port="5432",
                        database="postgres",
                        user="postgres",
                        password=""
                    )
                    cursor = conn.cursor()

                    cursor.execute(f"DELETE FROM {entity}")

                    conn.commit()

                    self.tree.delete(*self.tree.get_children())

                    messagebox.showinfo("Успех", f"Все записи из таблицы
{entity} успешно удалены!")

                except Exception as e:
                    messagebox.showerror("Ошибка", f"Не удалось очистить
таблицу: {e}")

                finally:
                    cursor.close()
                    conn.close()
```

delete_db.py

```
import tkinter as tk
import psycopg2
from tkinter import messagebox

# Параметры подключения к базе данных PostgreSQL
conn_params = {
    "host": "localhost",
    "port": "5432",
    "database": "postgres",
    "user": "postgres",
    "password": ""
}

class DelWindow:
    def __init__(self, entity):
        self.entity = entity
        self.delete_window = tk.Toplevel()
```

```
            self.delete_window.title("Выберите запись для удаления")

            try:
                # Получаем доступные ID записей из базы данных
                conn = psycopg2.connect(**conn_params)
                cur = conn.cursor()

                # Формируем название столбца ID в соответствии с таблицей
                id_column = f"id_{entity.lower()}a"
                cur.execute(f"SELECT {id_column} FROM {entity}")
                record_ids = [row[0] for row in cur.fetchall()]
                conn.close()

                if not record_ids:
                    messagebox.showerror("Ошибка", "Нет доступных записей
для удаления.")
                    self.delete_window.destroy()
                    return

                # Создаем метку с инструкцией
                label = tk.Label(self.delete_window, text="Выберите ID
записи для удаления:")
                label.pack()

                # Устанавливаем начальное значение для выбранного ID
                self.selected_id = tk.StringVar(self.delete_window)
                self.selected_id.set(record_ids[0])

                # Создаем выпадающее меню для выбора ID
                id_menu = tk.OptionMenu(self.delete_window,
self.selected_id, *record_ids)
                id_menu.pack()

                # Создаем кнопку для удаления записи
                delete_button = tk.Button(self.delete_window,
text="Удалить", command=self.delete_selected_record)
                delete_button.pack()
            except Exception as e:
                messagebox.showerror("Ошибка", f"Ошибка при выполнении
запроса: {e}")

        def delete_selected_record(self):
            try:
                selected_record_id = self.selected_id.get()
                if not selected_record_id:
                    raise ValueError("Не выбран ID записи")

                # Удаляем запись из базы данных
                conn = psycopg2.connect(**conn_params)
                cur = conn.cursor()
                id_column = f"id_{self.entity.lower()}a"
                cur.execute(f"DELETE FROM {self.entity} WHERE {id_column} =
%s", (selected_record_id,))
                conn.commit()
                conn.close()

                messagebox.showinfo("Успех", "Запись успешно удалена из базы
данных!")
                self.delete_window.destroy()
            except ValueError as ve:
                messagebox.showerror("Ошибка", str(ve))
            except Exception as e:
```

```
                messagebox.showerror("Ошибка", f"Не удалось удалить запись:
{e}")
```

request_db.py

```
import tkinter as tk
from tkinter import scrolledtext
import psycopg2
from tkinter import messagebox

conn_params = {
    "host": "localhost",
    "port": "5432",
    "database": "postgres",
    "user": "postgres",
    "password": ""
}

class ReqWindow:
    def __init__(self):
        self.window = tk.Toplevel()
        self.window.title("Запрос")

        self.create_widgets()

    def create_widgets(self):
        label = tk.Label(self.window, text="Введите запрос:")
        label.pack()

        self.query_entry = scrolledtext.ScrolledText(self.window,
width=50, height=2)
        self.query_entry.pack()

        execute_button = tk.Button(self.window, text="Выполнить запрос",
command=self.execute_query)
        execute_button.pack()

    def execute_query(self):
        query = self.query_entry.get("1.0", tk.END)
        try:
            connection = psycopg2.connect(**conn_params)
            cursor = connection.cursor()
            cursor.execute(query)
            records = cursor.fetchall()
            connection.close()

            self.show_result_window(records)
        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при выполнении
запроса: {e}")

    def show_result_window(self, records):
        result_window = tk.Toplevel()
        result_window.title("Результат запроса")

        result_text = scrolledtext.ScrolledText(result_window,
width=100, height=10)
        result_text.pack()

        for record in records:
            result_text.insert(tk.END, str(record) + "\n")
```

6. ВЫВОД

Была создана прикладная программа для работы с базой данных, которая выполняет заданные транзакции, а также был реализован механизм работы с базой данных.