|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра практической и прикладной информатики (ППИ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ**

по дисциплине «Инструментальное программное обеспечение разработки и проектирования информационных систем»

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИКМО-01-24. Шендяпин А.В.* | (подпись) | |
| Преподаватель | *Мельников Д.А.* | (подпись) | |
|  |  | |  | |

Москва 2025 г.

## **Практическая работа №1**

В процессе выполнения практической работы была создана форма в QTDesigner. После этого она была подключена к коду на python, где получилось реализовать отправку сообщений на почту.

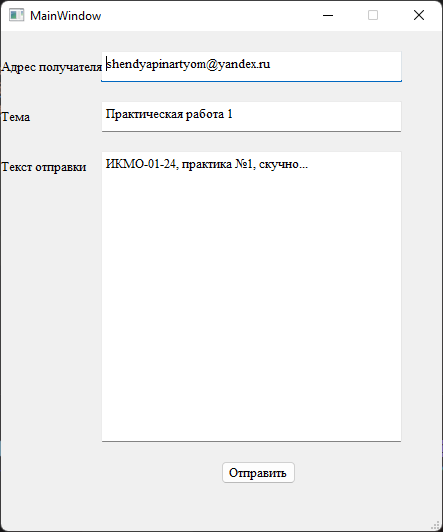


Рисунок 1.1 – Окно с данными

*Листинг 1.1 – код для работы с формой*

from PyQt5 import QtWidgets, uic

import smtplib

import os

from dotenv import load\_dotenv

from email.message import EmailMessage

class MainWindow(QtWidgets.QMainWindow):

def \_\_init\_\_(self):

super(MainWindow, self).\_\_init\_\_()

*Листинг 1.1 – продолжение*

# Загружаем UI из файла

uic.loadUi('main.ui', self)

# Инициализация виджетов

self.msg\_addr = self.findChild(QtWidgets.QTextEdit, 'msg\_addr')

self.msg\_header = self.findChild(QtWidgets.QTextEdit, 'msg\_header')

self.msg\_text = self.findChild(QtWidgets.QPlainTextEdit, 'msg\_text')

self.pushButton = self.findChild(QtWidgets.QPushButton, 'pushButton')

self.pushButton.clicked.connect(self.on\_send\_button\_clicked)

def set\_address(self, text):

self.msg\_addr.setText(text

def set\_header(self, text):

self.msg\_header.setText(text)

def set\_message(self, text):

self.msg\_text.setPlainText(text)

def get\_address(self):

return self.msg\_addr.toPlainText()

def get\_header(self):

return self.msg\_header.toPlainText()

def get\_message(self):

return self.msg\_text.toPlainText()

def on\_send\_button\_clicked(self):

address = self.get\_address()

header = self.get\_header()

message = self.get\_message()

print(f"Адрес: {address}")

print(f"Тема: {header}")

print(f"Сообщение: {message}")

send\_email(address, header, message)

*Листинг 1.3 – продолжение*

def send\_email(addr\_to, msg\_subj, msg\_text):

load\_dotenv()

msg = EmailMessage()

msg['From'] = os.getenv("LOGIN")

msg['To'] = addr\_to

msg['Subject'] = msg\_subj

msg.set\_content(msg\_text)

try:

server = smtplib.SMTP('smtp.yandex.ru', 587)

server.ehlo()

server.starttls()

server.ehlo()

server.login(os.getenv("LOGIN"), os.getenv("PASSWORD"))

server.set\_debuglevel(1)

server.sendmail(os.getenv("LOGIN"), [addr\_to], msg.as\_string())

server.quit()

except Exception as e:

raise RuntimeError(f"Ошибка при отправке письма: {str(e)}")

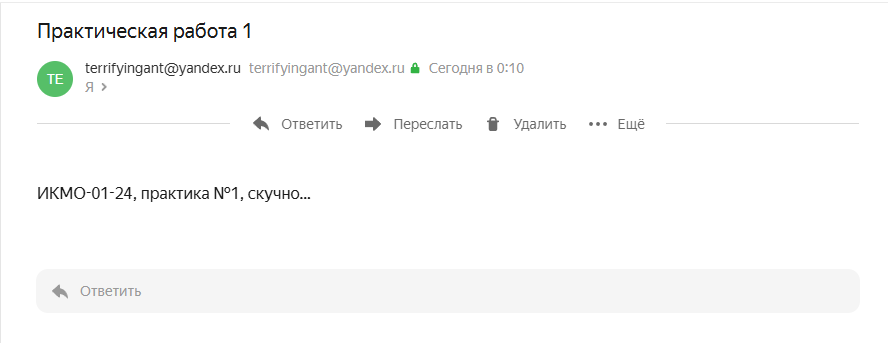


Рисунок 1.2 – результат работы

## **Практическая работа №2**

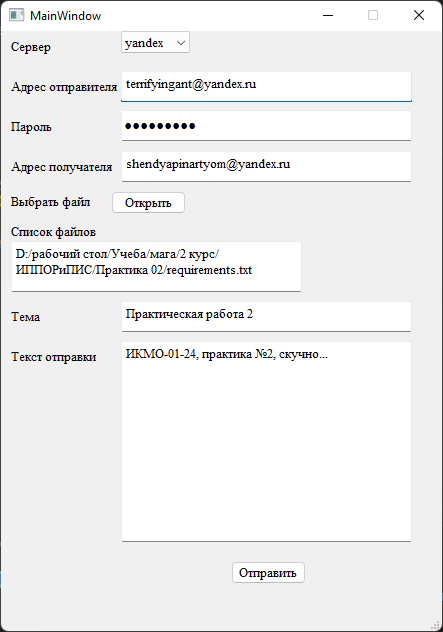


Рисунок 2.1 – заполненная форма

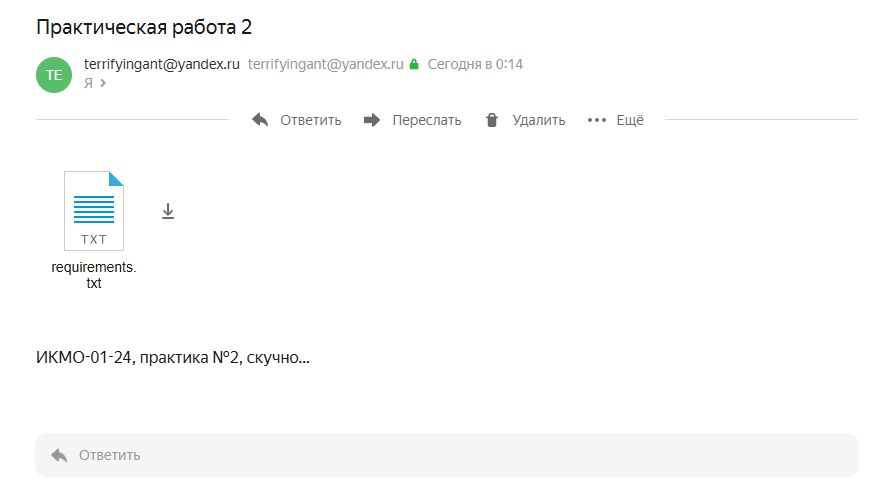
****

Рисунок 2.2 – результат отправки сообщения

*Листинг 2.1 –основной код формы*

from PyQt5 import QtWidgets, uic

import smtplib

import os

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

from PyQt5.QtWidgets import QFileDialog

class MainWindow(QtWidgets.QMainWindow):

    def \_\_init\_\_(self):

        super(MainWindow, self).\_\_init\_\_()

        uic.loadUi('main.ui', self)

        self.msg\_addr = self.findChild(QtWidgets.QTextEdit, 'msg\_addr')  # Адрес получателя

        self.msg\_header = self.findChild(QtWidgets.QTextEdit, 'msg\_header')  # Тема

        self.msg\_text = self.findChild(QtWidgets.QPlainTextEdit, 'msg\_text')  # Текст сообщения

        self.pushButton = self.findChild(QtWidgets.QPushButton, 'pushButton')  # Кнопка "Отправить"

*Листинг 2.1 – продолжение*

        self.server\_type = self.findChild(QtWidgets.QComboBox, 'server\_type')  # Выбор сервера

        self.msg\_addr\_from = self.findChild(QtWidgets.QTextEdit, 'msg\_addr\_from')  # Адрес отправителя

        self.from\_password = self.findChild(QtWidgets.QLineEdit, 'from\_password')  # Пароль отправителя (QLineEdit)

        self.push\_button\_choose\_files = self.findChild(QtWidgets.QPushButton, 'push\_button\_choose\_files')  # Кнопка выбора файлов

        self.msg\_header\_2 = self.findChild(QtWidgets.QTextEdit, 'msg\_header\_2')  # Поле для отображения выбранных файлов

        self.from\_password.setEchoMode(QtWidgets.QLineEdit.Password)  # Отображение пароля как звездочек

        self.pushButton.clicked.connect(self.on\_send\_button\_clicked)

        self.push\_button\_choose\_files.clicked.connect(self.on\_choose\_files\_clicked)

        self.selected\_files = []

    def set\_address(self, text):

        self.msg\_addr.setText(text)

    def set\_header(self, text):

        self.msg\_header.setText(text)

    def set\_message(self, text):

        self.msg\_text.setPlainText(text)

    def get\_address(self):

        return self.msg\_addr.toPlainText()

    def get\_header(self):

        return self.msg\_header.toPlainText()

    def get\_message(self):

        return self.msg\_text.toPlainText()

    def on\_choose\_files\_clicked(self):

        options = QFileDialog.Options()

        files, \_ = QFileDialog.getOpenFileNames(self, "Выберите файлы", "", "All Files (\*);;", options=options)

        if files:

*Листинг 2.1 – продолжение*

            self.selected\_files = files

            self.msg\_header\_2.setText("\n".join(files))  # Отображаем список выбранных файлов

    def on\_send\_button\_clicked(self):

        """Обработчик нажатия на кнопку 'Отправить'."""

        addr\_to = self.get\_address()

        addr\_from = self.msg\_addr\_from.toPlainText()

        password = self.from\_password.text()  # Получаем пароль из QLineEdit

        msg\_subj = self.get\_header()

        msg\_text = self.get\_message()

        if not addr\_to or not addr\_from or not password or not msg\_subj or not msg\_text:

            QtWidgets.QMessageBox.warning(self, "Ошибка", "Заполните все поля!")

            return

        server\_type = self.server\_type.currentText()

        smtp\_server, smtp\_port = self.get\_smtp\_settings(server\_type)

        try:

            send\_email(addr\_from, password, addr\_to, msg\_subj, msg\_text, self.selected\_files, smtp\_server, smtp\_port)

            QtWidgets.QMessageBox.information(self, "Успех", "Письмо успешно отправлено!")

        except Exception as e:

            QtWidgets.QMessageBox.critical(self, "Ошибка", f"Не удалось отправить письмо: {str(e)}")

    def get\_smtp\_settings(self, server\_type):

        if server\_type == "yandex":

            return "smtp.yandex.ru", 587

        elif server\_type == "mail":

*Листинг 2.1 – продолжение*

            return "smtp.mail.ru", 587

        elif server\_type == "gmail":

            return "smtp.gmail.com", 587

        else:

            raise ValueError("Неизвестный тип сервера")

def send\_email(addr\_from, password, addr\_to, msg\_subj, msg\_text, attachments, smtp\_server, smtp\_port):

    msg = MIMEMultipart()

    msg['From'] = addr\_from

    msg['To'] = addr\_to

    msg['Subject'] = msg\_subj

    msg.attach(MIMEText(msg\_text, 'plain'))

    for file\_path in attachments:

        try:

            with open(file\_path, "rb") as f:

                file\_data = f.read()

                file\_name = os.path.basename(file\_path)

                mime\_part = MIMEBase("application", "octet-stream")

                mime\_part.set\_payload(file\_data)

                encoders.encode\_base64(mime\_part)

                mime\_part.add\_header("Content-Disposition", f"attachment; filename={file\_name}")

                msg.attach(mime\_part)

        except Exception as e:

            print(f"Не удалось прикрепить файл {file\_path}: {str(e)}")

    try:

        server = smtplib.SMTP(smtp\_server, smtp\_port)

        server.ehlo()

        server.starttls()

        server.login(addr\_from, password)

        server.sendmail(addr\_from, [addr\_to], msg.as\_string())

        server.quit()

    except Exception as e:        raise RuntimeError(f"Ошибка при отправке письма: {str(e)}")

## **Практическая работа №3**

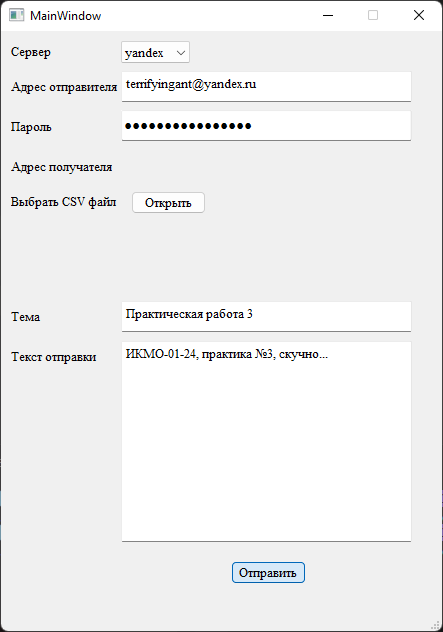
**

Рисунок 3.1 – заполненные формы

*Листинг 3.1 – содержание файла csv для отправки*

shendyapinartyom@yandex.ru; D:\рабочий стол\Учеба\мага\2 курс\ИППОРиПИС\Практика 03\requirements.txt; D:\рабочий стол\Учеба\мага\2 курс\ИППОРиПИС\Практика 03\requirements.txt;

shendyapinartyom@yandex.ru; D:\рабочий стол\Учеба\мага\2 курс\ИППОРиПИС\Практика 03\requirements.txt;

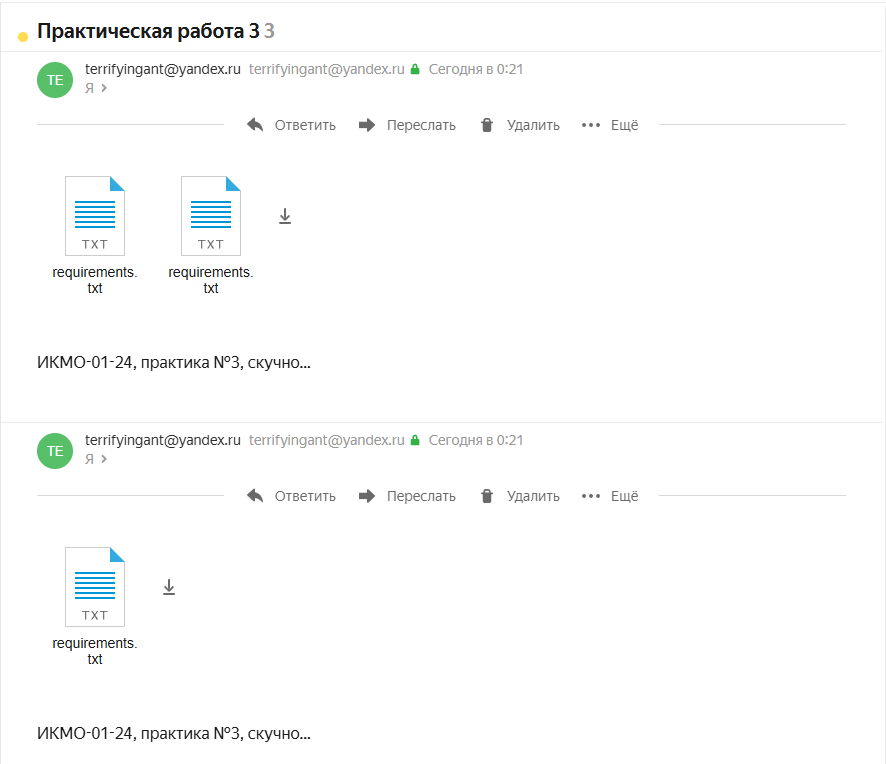


Рисунок 3.2 – результат работы программы

Код достаточно большой, чтобы вставлять его в отчет. Его можно найти на github по ссылке - <https://github.com/TerrifyingAnt/mag-python-pracs/tree/main/Практика%2003>.

## **Практическая работа №4**

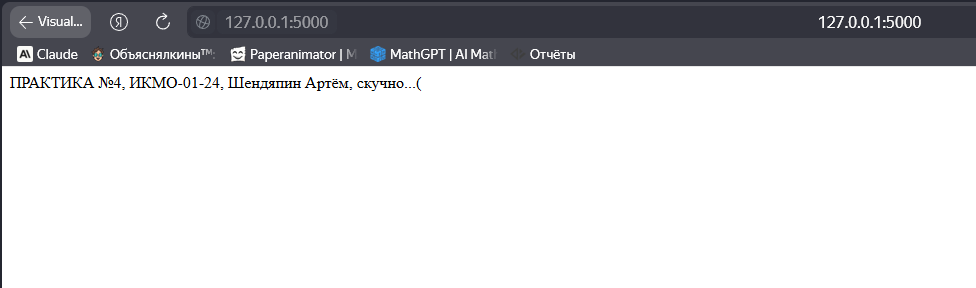
**

Рисунок 4.1 – запущенный на Flusk сервер

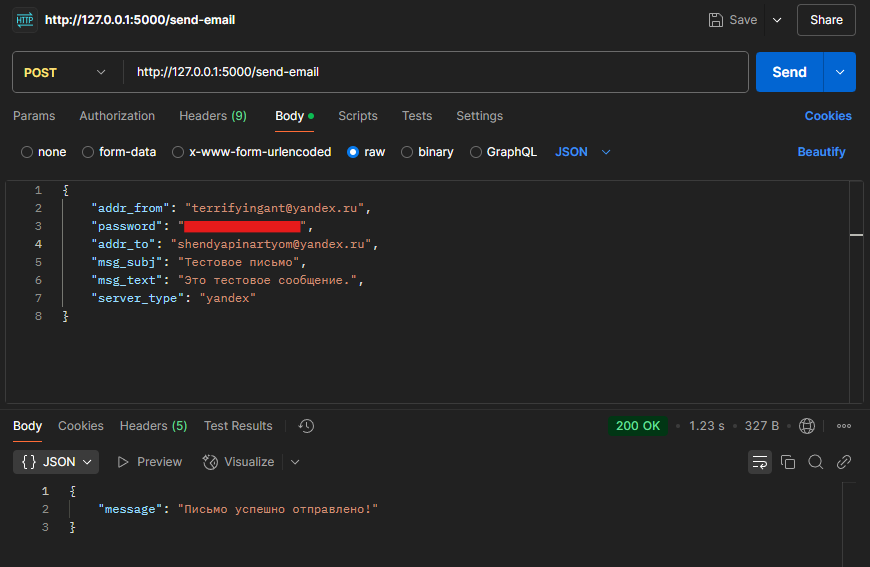
**

Рисунок 4.2 – проверка работоспособности сервера по отправке сообщений

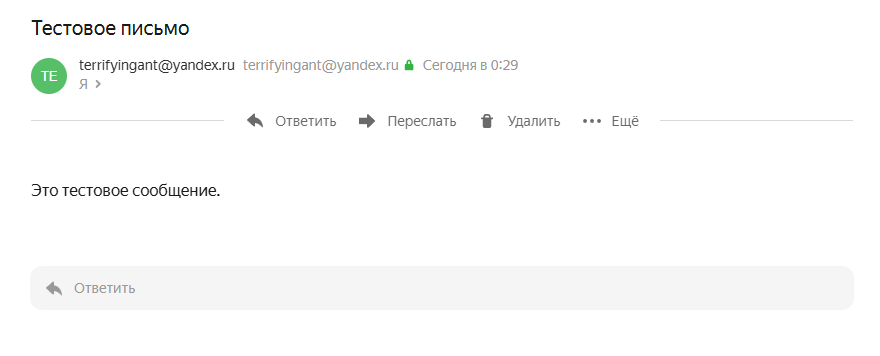


Рисунок 4.3 – результат отправки сообщения

*Листинг 4.1 – код для запуска сервера*

from flask import Flask, request, jsonify

from utils import \*

app = Flask(\_\_name\_\_)

@app.route('/send-email', methods=['POST'])

def send\_email\_handler():

    """Обработчик POST-запроса для отправки письма."""

    try:

        # Получаем данные из JSON-тела запроса

*Листинг 4.1 – продолжение*

        data = request.json

        addr\_from = data.get('addr\_from')

        password = data.get('password')

        addr\_to = data.get('addr\_to')

        msg\_subj = data.get('msg\_subj')

        msg\_text = data.get('msg\_text')

        attachments = data.get('attachments', [])

        server\_type = data.get('server\_type')

        # Проверяем обязательные поля

        if not all([addr\_from, password, addr\_to, msg\_subj, msg\_text, server\_type]):

            return jsonify({"error": "Необходимо заполнить все обязательные поля"}), 400

        # Получаем настройки SMTP-сервера

        if server\_type == "yandex":

            smtp\_server, smtp\_port = "smtp.yandex.ru", 587

        elif server\_type == "mail":

            smtp\_server, smtp\_port = "smtp.mail.ru", 587

        elif server\_type == "gmail":

            smtp\_server, smtp\_port = "smtp.gmail.com", 587

        else:

            return jsonify({"error": "Неизвестный тип сервера"}), 400

        # Отправляем письмо

        result = send\_email(addr\_from, password, addr\_to, msg\_subj, msg\_text, attachments, smtp\_server, smtp\_port)

        return jsonify({"message": result}), 200

    except Exception as e:

        return jsonify({"error": str(e)}), 500

*Листинг 4.1 – продолжение*

@app.route('/', methods=['GET'])

def base\_route\_handler():

    return "ПРАКТИКА №4, ИКМО-01-24, Шендяпин Артём, скучно...("

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

## **Практическая работа №5**

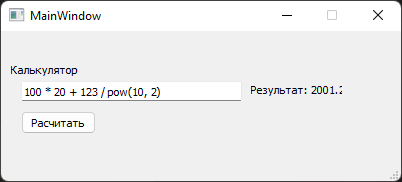


Рисунок 5.1 – проверка работоспособности калькулятора

*Листинг 5.1 – код работы калькулятора*

from PyQt5 import QtWidgets, uic

from PyQt5.QtWidgets import QMessageBox

def safe\_eval(expression):

    try:

        allowed\_names = {

            'abs': abs,

            'pow': pow,

            'round': round

        }

        result = eval(expression, {"\_\_builtins\_\_": None}, allowed\_names)

        return str(result)

    except Exception as e:

        return f"Ошибка: {str(e)}"

*Листинг 5.1 – продолжение*

class MainWindow(QtWidgets.QMainWindow):

    def \_\_init\_\_(self):

        super(MainWindow, self).\_\_init\_\_()

        uic.loadUi("main.ui", self)

        self.pushButton.clicked.connect(self.calculate)

    def calculate(self):

        expression = self.lineEdit.text()

        # Проверяем, что поле не пустое

        if not expression.strip():

            self.label\_2.setText("Результат: ")

            QMessageBox.warning(self, "Ошибка", "Поле ввода пустое!")

            return

        result = safe\_eval(expression)

        self.label\_2.setText(f"Результат: {result}")

## **Практическая работа №6**

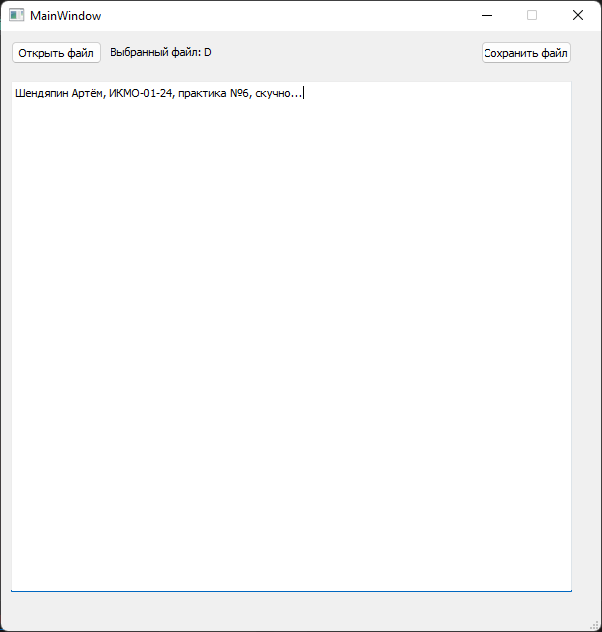


Рисунок 6.1 – форма с открытым файлом

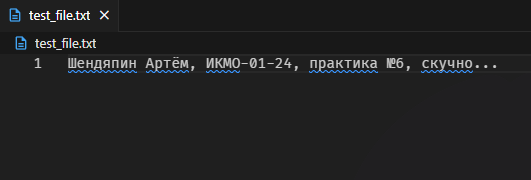


Рисунок 6.2 файл, до пересохранения с новым текстом

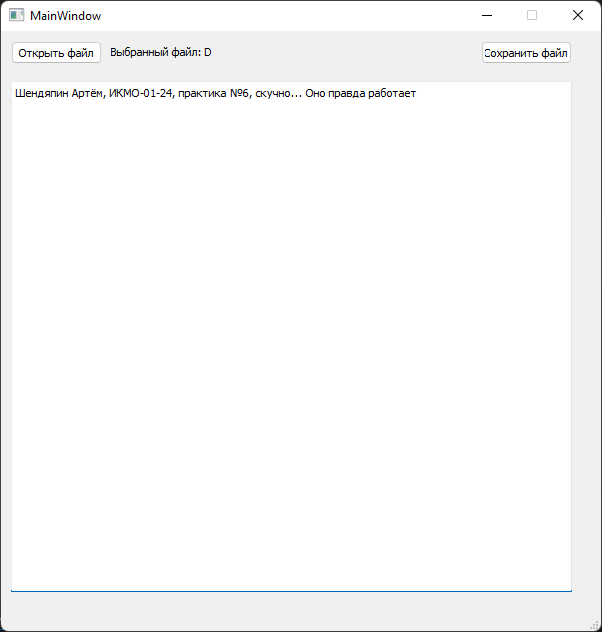


Рисунок 6.3 – измененный текст

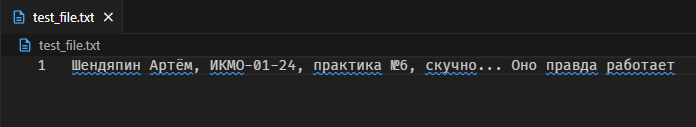


Рисунок 6.4 – файл после сохранения

*Листинг 6.1 – код для работы с текстовыми документами*

import sys

from PyQt5 import QtWidgets, uic

from PyQt5.QtWidgets import QFileDialog, QMessageBox

class TextEditor(QtWidgets.QMainWindow):

    def \_\_init\_\_(self):

*Листинг 6.1 - продолжение*

        super(TextEditor, self).\_\_init\_\_()

        # Загрузка интерфейса из XML (.ui файла)

        uic.loadUi("main.ui", self)  # Убедитесь, что путь к файлу правильный

        # Разрешаем редактирование текстового поля

        self.textBrowser.setReadOnly(False)

        # Подключение кнопок к обработчикам событий

        self.pushButton.clicked.connect(self.open\_file)  # Открыть файл

        self.pushButton\_2.clicked.connect(self.save\_file)  # Сохранить файл

        # Переменная для хранения пути к текущему файлу

        self.current\_file\_path = None

    def open\_file(self):

        # Открываем диалоговое окно для выбора файла

        options = QFileDialog.Options()

        file\_path, \_ = QFileDialog.getOpenFileName(

            self,

            "Открыть текстовый файл",

            "",

            "Text Files (\*.txt);;All Files (\*)",

            options=options

        )

        if file\_path:

            try:

                # Читаем содержимое файла

                with open(file\_path, "r", encoding="utf-8") as file:

*Листинг 6.1 - продолжение*

                    content = file.read()

                # Отображаем содержимое в текстовом поле

                self.textBrowser.setPlainText(content)

                # Обновляем метку с выбранным файлом

                self.label.setText(f"Выбранный файл: {file\_path}")

                # Сохраняем путь к файлу

                self.current\_file\_path = file\_path

            except Exception as e:

                QMessageBox.critical(self, "Ошибка", f"Не удалось открыть файл:\n{str(e)}")

    def save\_file(self):

        # Если файл уже был открыт, сохраняем изменения в тот же файл

        if self.current\_file\_path:

            try:

                # Получаем текст из текстового поля

                content = self.textBrowser.toPlainText()

                # Сохраняем текст в файл

                with open(self.current\_file\_path, "w", encoding="utf-8") as file:

                    file.write(content)

                QMessageBox.information(self, "Успех", "Файл успешно сохранен!")

            except Exception as e:

                QMessageBox.critical(self, "Ошибка", f"Не удалось сохранить файл:\n{str(e)}")

        else:

            # Если файл не был открыт, предлагаем выбрать место для сохранения

*Листинг 6.1 - продолжение*

            options = QFileDialog.Options()

            file\_path, \_ = QFileDialog.getSaveFileName(

                self,

                "Сохранить текстовый файл",

                "",

                "Text Files (\*.txt);;All Files (\*)",

                options=options

            )

            if file\_path:

                try:

                    # Получаем текст из текстового поля

                    content = self.textBrowser.toPlainText()

                    # Сохраняем текст в файл

                    with open(file\_path, "w", encoding="utf-8") as file:

                        file.write(content)

                    # Обновляем метку с выбранным файлом

                    self.label.setText(f"Выбранный файл: {file\_path}")

                    # Сохраняем путь к файлу

                    self.current\_file\_path = file\_path

                    QMessageBox.information(self, "Успех", "Файл успешно сохранен!")

                except Exception as e:

                    QMessageBox.critical(self, "Ошибка", f"Не удалось сохранить файл:\n{str(e)}")

## **Практическая работа №7**

Для того, чтобы было удобно тестировать форму на pyqt и не переживать за сохранность данных в базе данных, было принято решение развернуть 2 докер-контейнера – один с PostgreSQL, второй с Liquibase. Был написан Docker-compose, который выглядит следующим образом:

*Листинг 7.1 – docker-compose*

version: '3.8'

services:

 postgres:

    restart: always

    container\_name: py\_prac\_7\_db

    environment:

      POSTGRES\_USER: postgres

      POSTGRES\_PASSWORD: postgres

      POSTGRES\_DB: ch\_postgres\_db

    ports:

      - '6543:5432'

    volumes:

      - ./postgres/container\_data:/var/lib/postgresql/data

    build:

      dockerfile: postgres/Dockerfile

 liquibase:

    container\_name: py\_prac\_7\_liquibase

    restart: on-failure

    depends\_on:

      - postgres

    environment:

      LIQUIBASE\_SEARCH\_PATH: ./changelog

    build:

      dockerfile: liquibase/Dockerfile

    command: liquibase --url="jdbc:postgresql://postgres:5432/ch\_postgres\_db" --changeLogFile=liquibase-changelog.xml --username=postgres --password=postgres update

networks:

  my\_network:

    driver: bridge

Liquibase – это невероятно удобная утилита для миграции баз данных. В данной практической работе она была настроена таким образом, что она производит все необходимые миграции, создает все базы данных и вносит в них всю информацию после того, как запускается Postgresql. Это позволило сосредоточиться на выполнении работы, связанной с кодом на pyqt, без лишних трат времени на восстановление\внесение данных в базу данных, в случае, если что-то пошло не так.

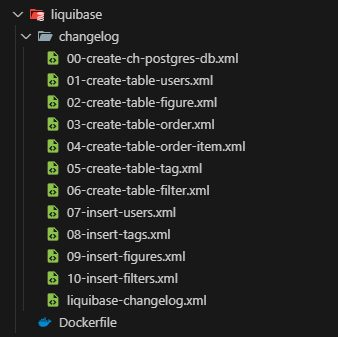


Рисунок 7.1 – changelog для Liquibase

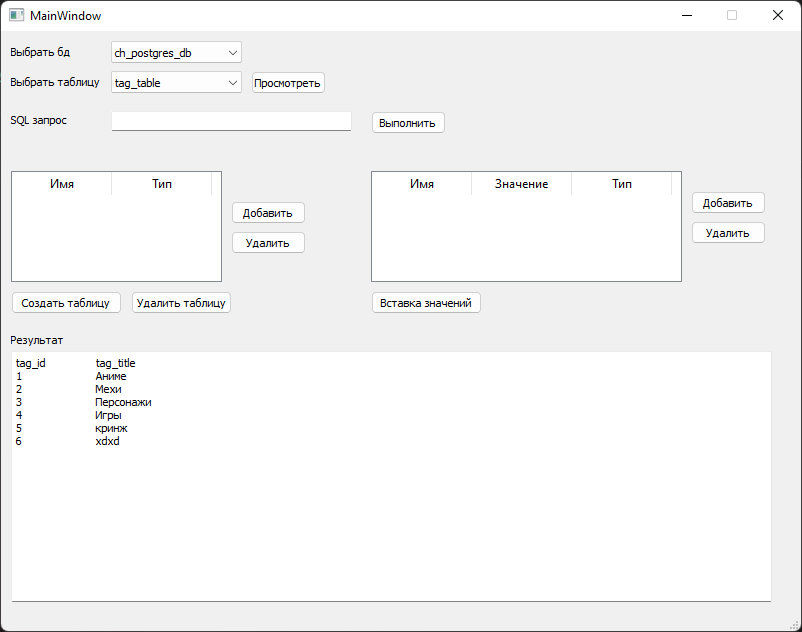


Рисунок 7.2 – окно для взаимодействия с базой данных

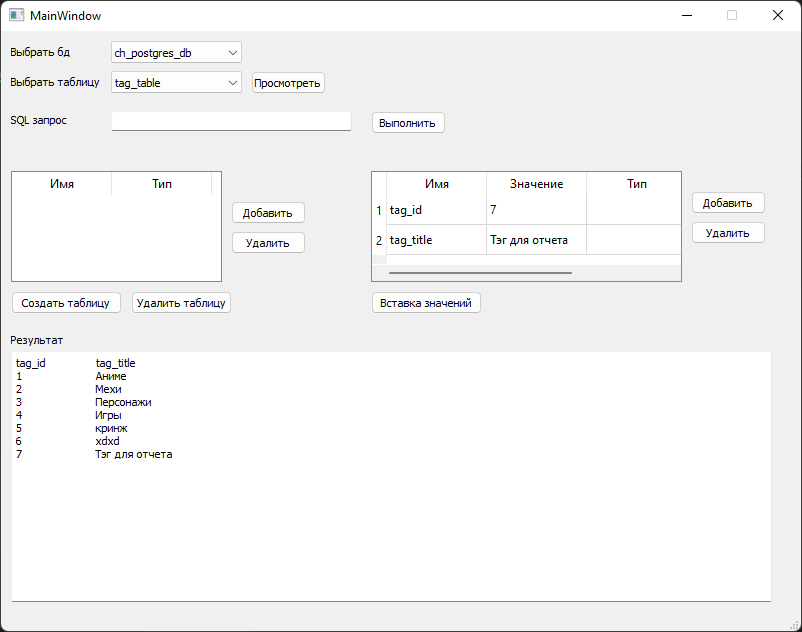


Рисунок 7.3 – добавление значения в таблицу

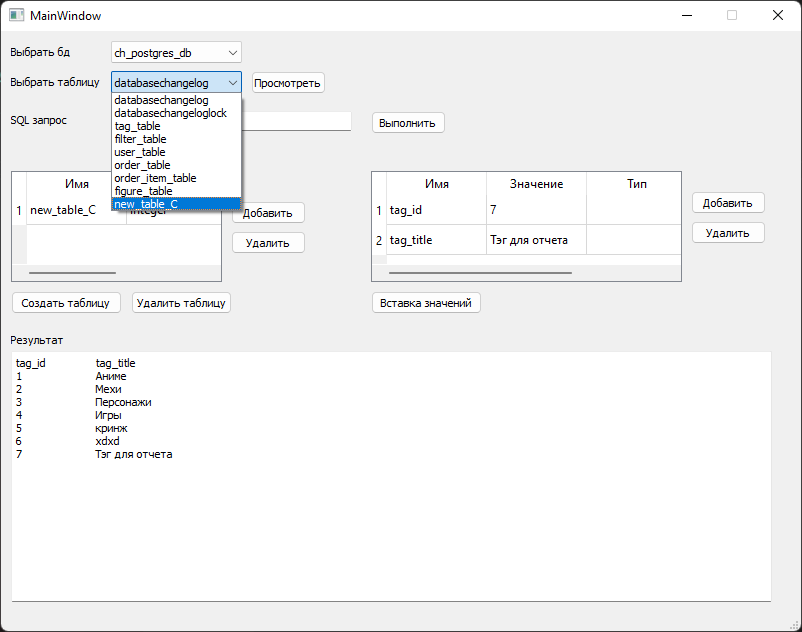


Рисунок 7.4 – добавление новой таблицы

Листинг для данной практической работы большой и в отчет не помещается. Его в удобном формате можно найти по ссылке на github - <https://github.com/TerrifyingAnt/mag-python-pracs/tree/main/Практика%2007>

## **Практическая работа №8**

Листинг 8.1 – код программы

def task1(x: list[float], n: int = 20, repl: int = 200) -> list[float]:

    return [el if el is not n else repl for el in x]

def task2(x: list[str]) -> list[str]:

    return [el for el in x if el != ""]

def task3(x: list[float]) -> list[float]:

    return [el \* el for el in x]

def task4(x: list[float], n: int = 20) -> list[float]:

    return [el for el in x if el != n]

Листинг 8.1 – продолжение

print(task1([1, 2, 3, 20, 50, 20, 123, 20]))

print(task2(["", "asdsda", "asddas", "", ""]))

print(task3([1, 2, 3, 4, 5]))

print(task4([1, 2, 3, 20, 50, 20, 123, 20]))

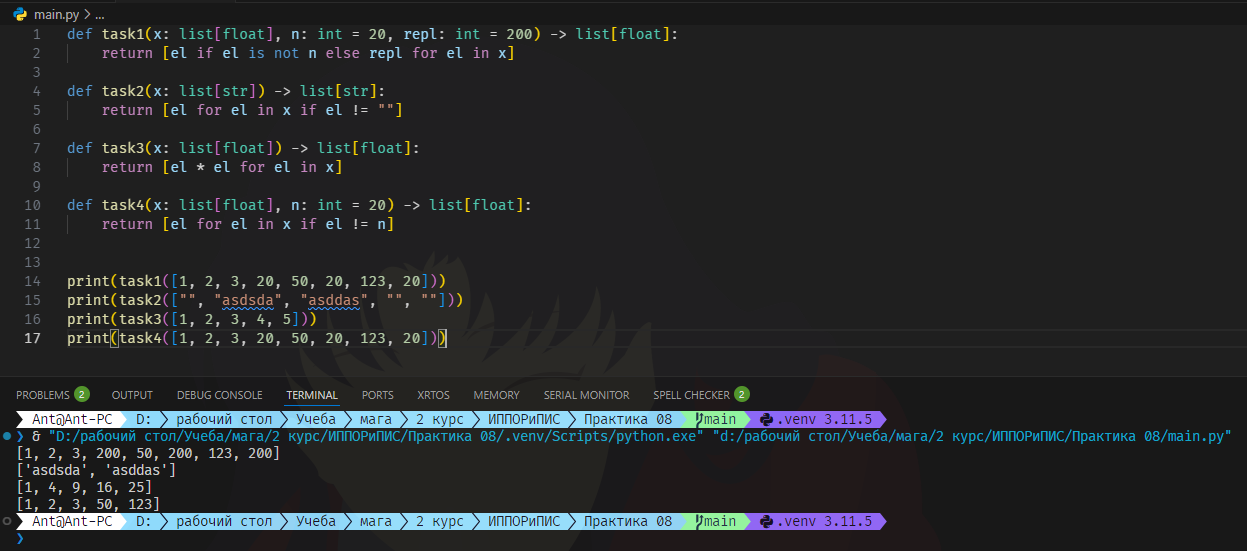


Рисунок 8.1 – результат запуска программы