



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 *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.

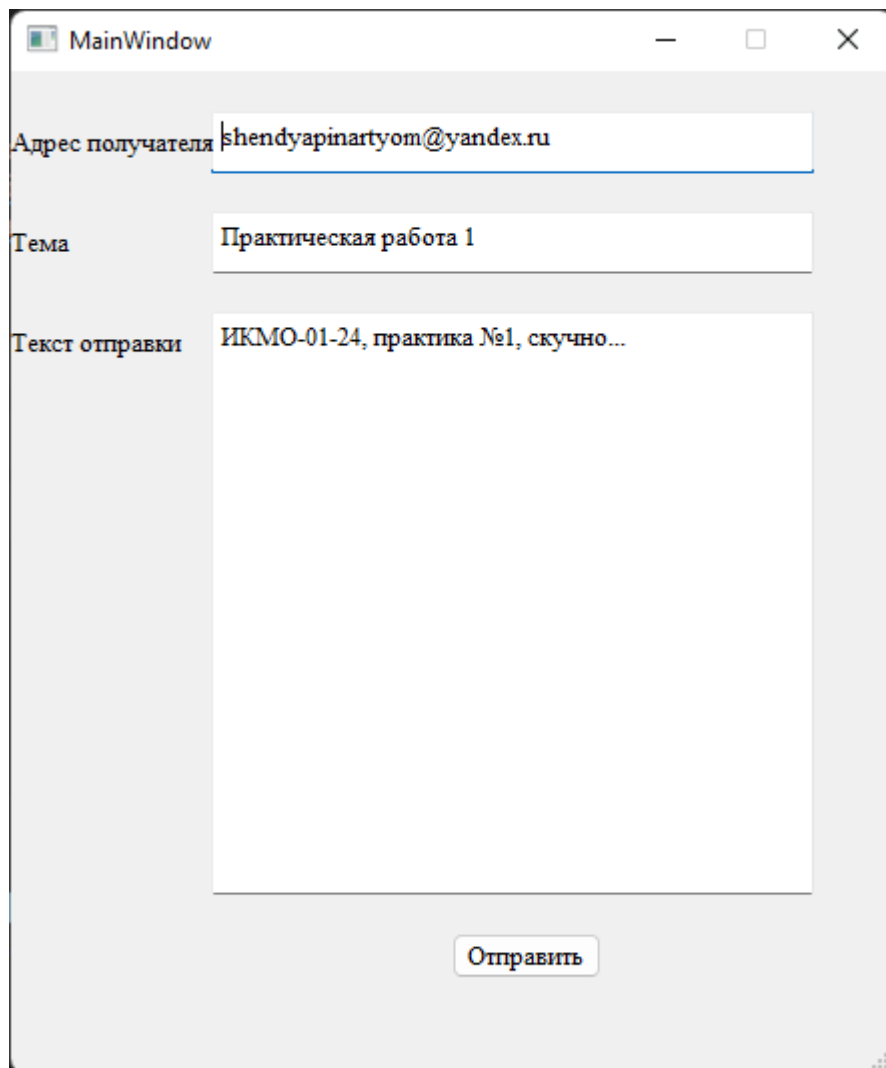


Рисунок 1.1 – Окно с данными

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
uis.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)}")
```

Практическая работа 1



terrifyingant@yandex.ru terrifyingant@yandex.ru Сегодня в 0:10
я >

← Ответить → Переслать Удалить ... Ещё

ИКМО-01-24, практика №1, скучно...

← Ответить

Рисунок 1.2 – результат работы

Практическая работа №2

The image shows a screenshot of a software application window titled "MainWindow". The window contains a form for sending an email. The form has the following fields and controls:

- Сервер**: A dropdown menu with "yandex" selected.
- Адрес отправителя**: A text input field containing "temifyingant@yandex.ru".
- Пароль**: A text input field filled with ten dots, indicating a password.
- Адрес получателя**: A text input field containing "shendyapinaryom@yandex.ru".
- Выбрать файл**: A button labeled "Открыть".
- Список файлов**: A text area containing the file path "D:/рабочий стол/Учеба/мага/2 курс/ИППОриПИС/Практика 02/requirements.txt".
- Тема**: A text input field containing "Практическая работа 2".
- Текст отправки**: A large text area containing the text "ИКМО-01-24, практика №2, скучно...".
- Отправить**: A button at the bottom right of the form.

Рисунок 2.1 – заполненная форма

Практическая работа 2



terrifyingant@yandex.ru terrifyingant@yandex.ru Сегодня в 0:14
Я >

← Ответить → Переслать Удалить ... Ещё



requirements.
txt

ИКМО-01-24, практика №2, скучно...

← Ответить

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_f
iles_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

MainWindow

Сервер: yandex

Адрес отправителя: tenifyingant@yandex.ru

Пароль:

Адрес получателя:

Выбрать CSV файл:

Тема: Практическая работа 3

Текст отправки: ИКМО-01-24, практика №3, скучно...

Рисунок 3.1 – заполненные формы

Листинг 3.1 – содержание файла csv для отправки

```
shendyarinartyom@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;  
  
shendyarinartyom@yandex.ru; D:\рабочий стол\Учеба\мага\2  
курс\ИППОриПИС\Практика 03\requirements.txt;
```

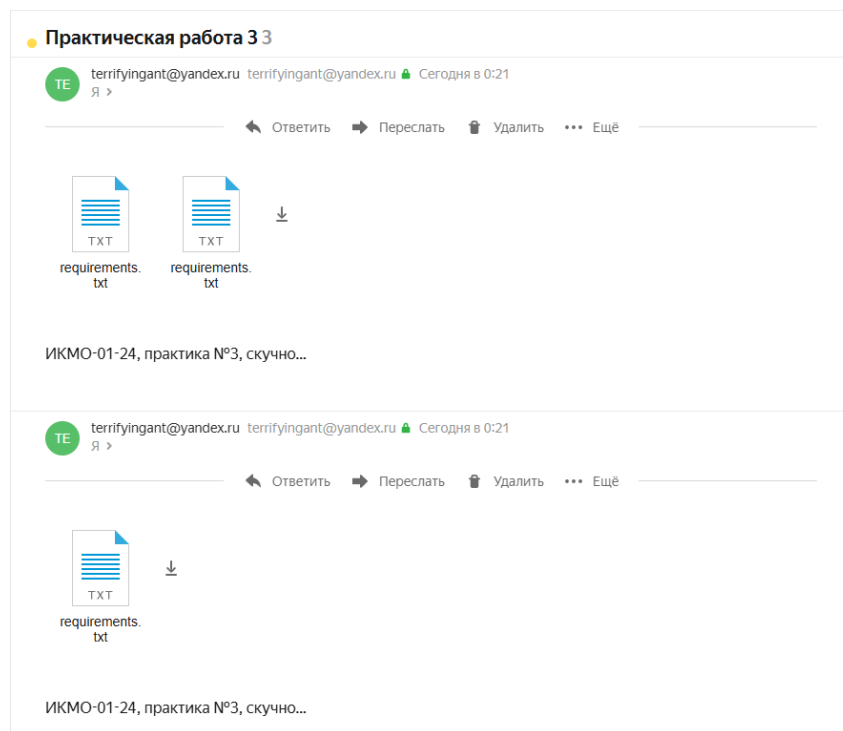


Рисунок 3.2 – результат работы программы

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

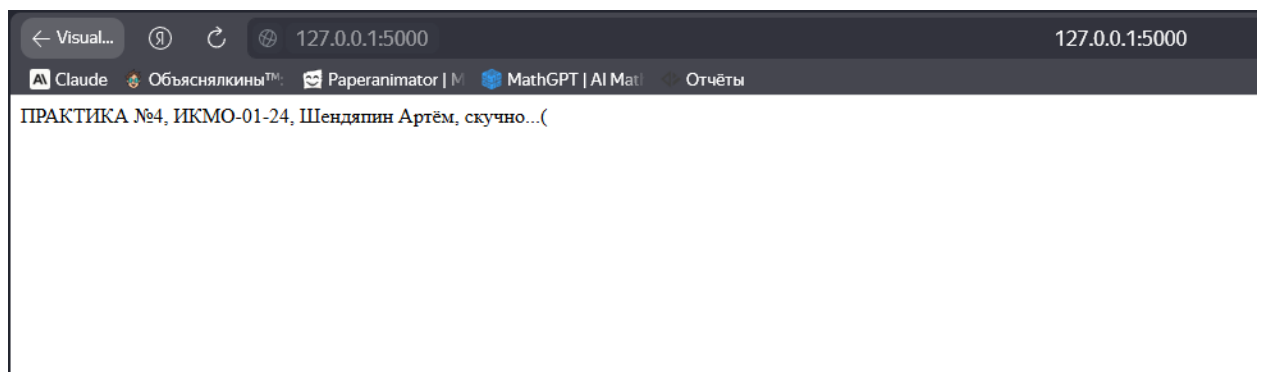


Рисунок 4.1 – запущенный на Flask сервер

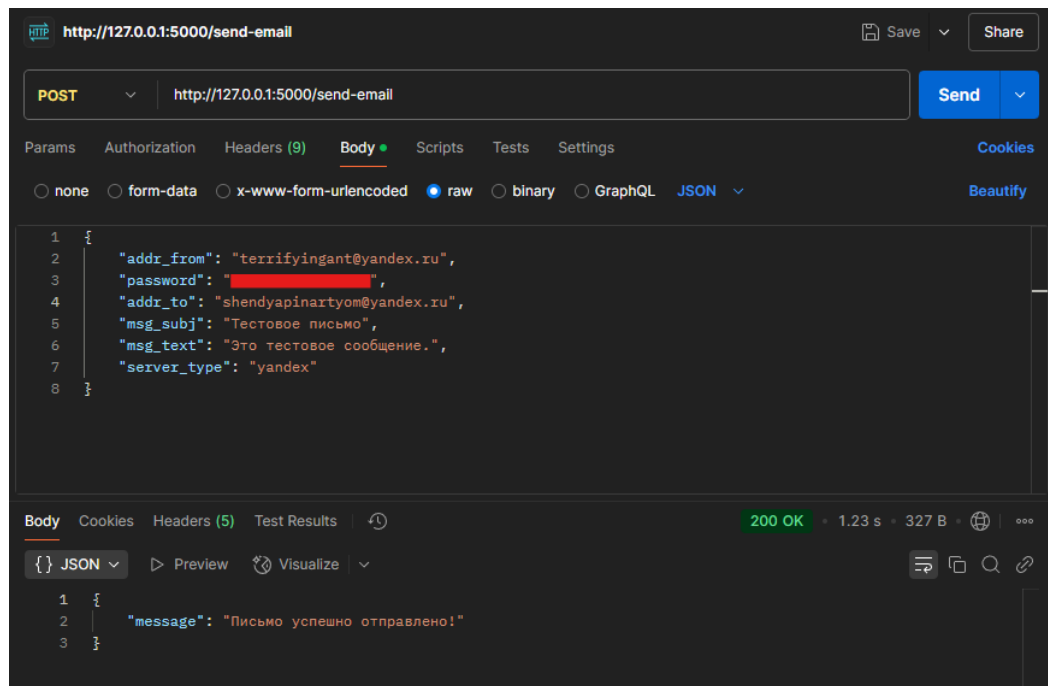


Рисунок 4.2 – проверка работоспособности сервера по отправке сообщений

Тестовое письмо

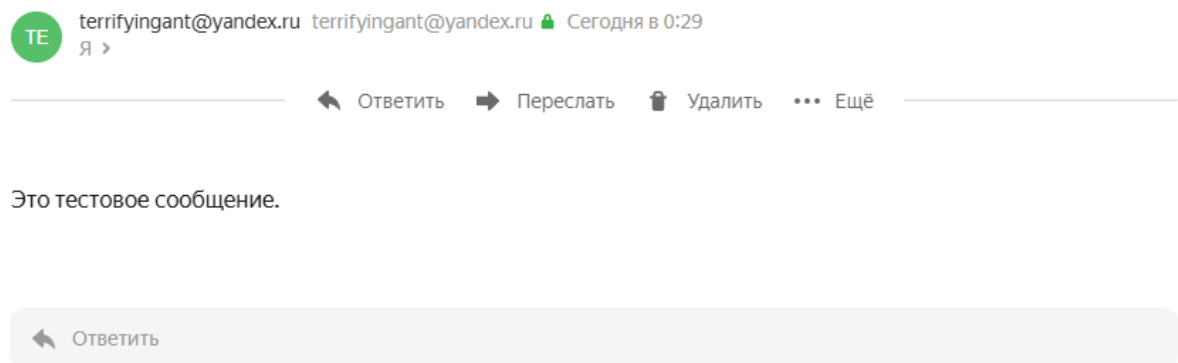


Рисунок 4.3 – результат отправки сообщения

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

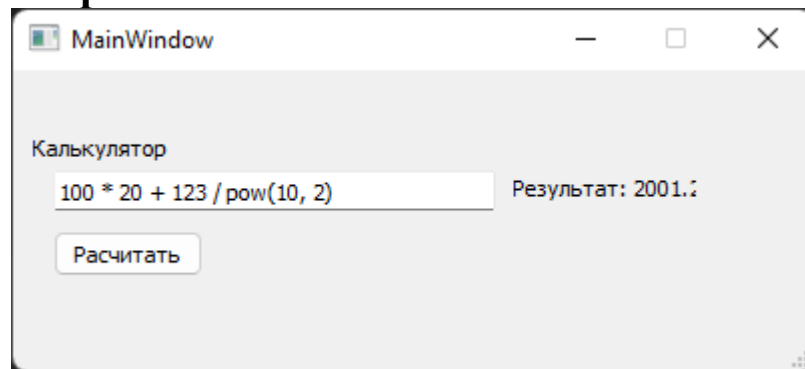


Рисунок 5.1 – проверка работоспособности калькулятора

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(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

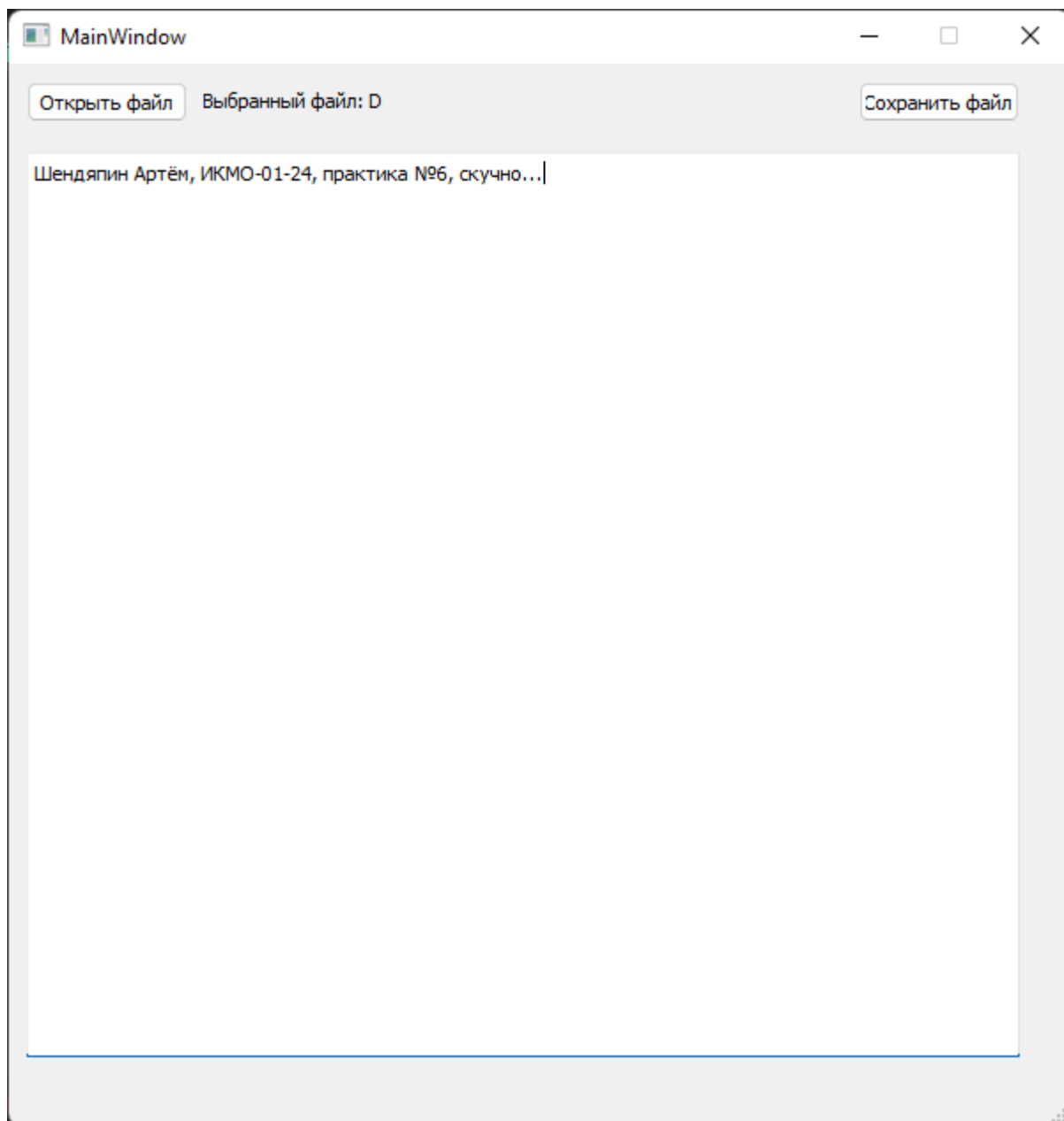


Рисунок 6.1 – форма с открытым файлом

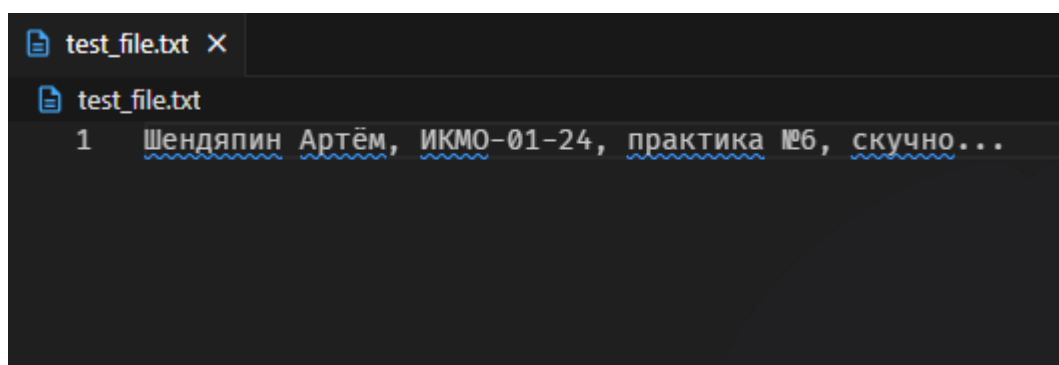


Рисунок 6.2 файл, до пересохранения с новым текстом

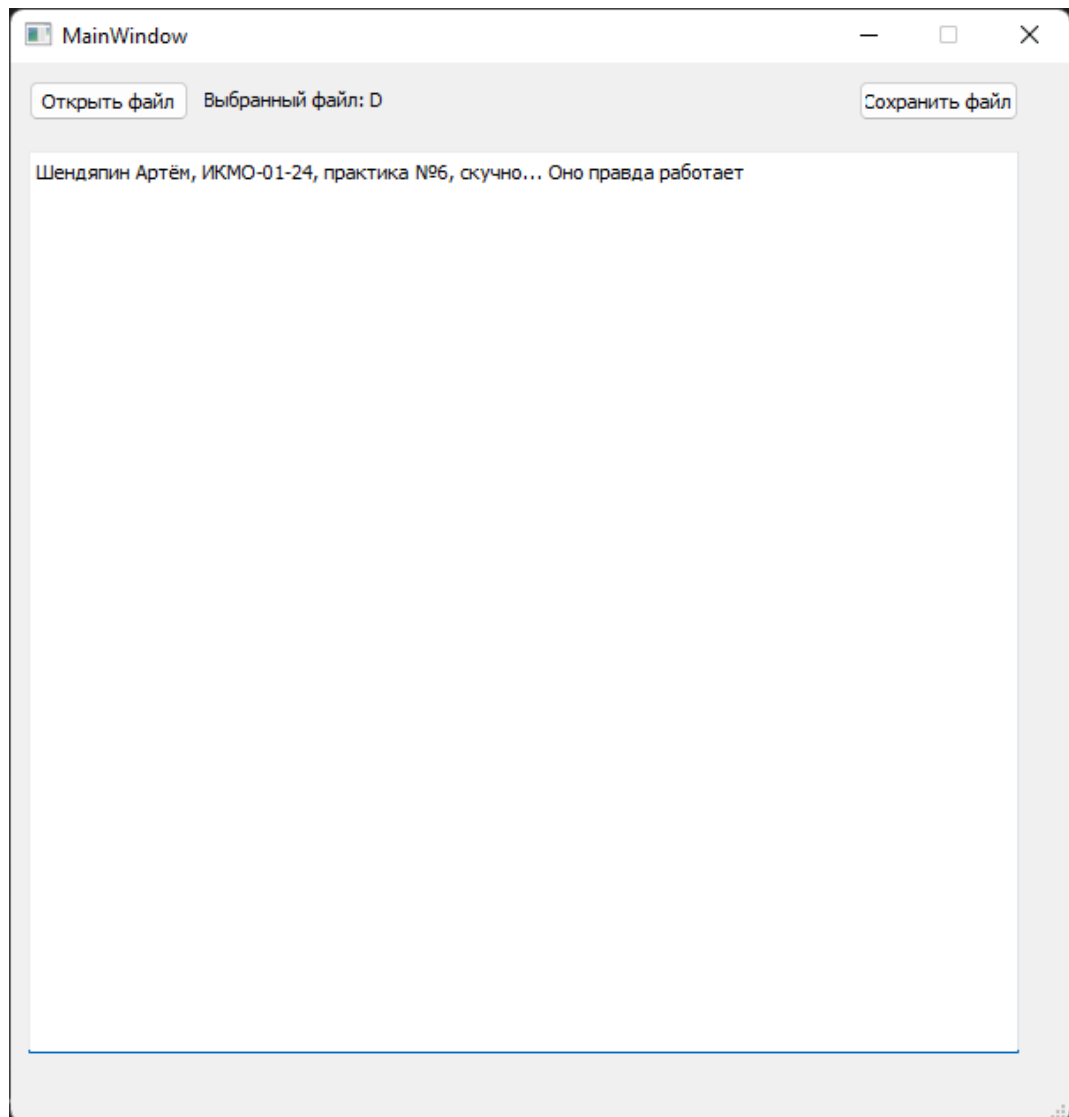


Рисунок 6.3 – измененный текст

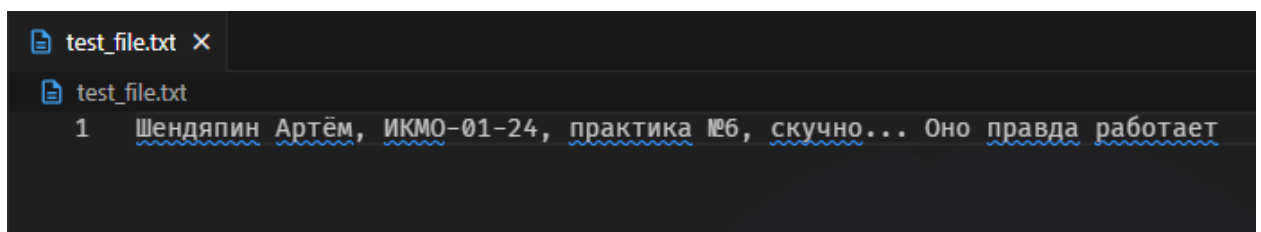


Рисунок 6.4 – файл после сохранения

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 ruqt 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 ruqt, без лишних трат времени на восстановление\внесение данных в базу данных, в случае, если что-то пошло не так.

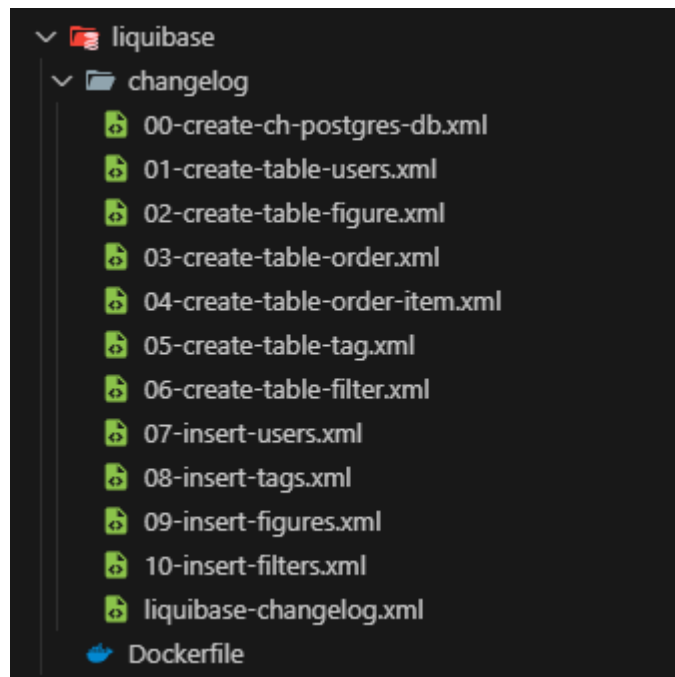


Рисунок 7.1 – changelog для Liquibase

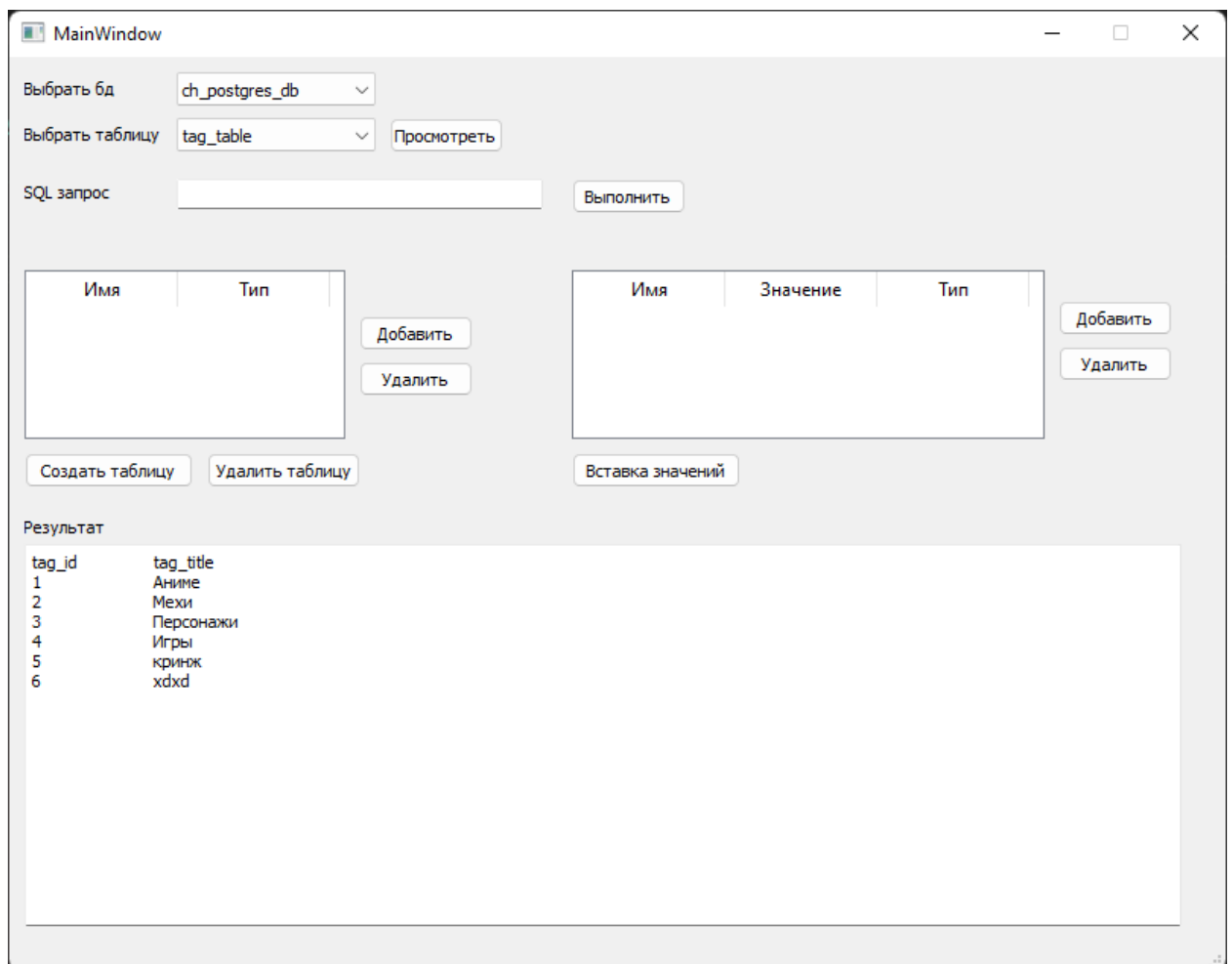


Рисунок 7.2 – окно для взаимодействия с базой данных

MainWindow

Выбрать бд:

Выбрать таблицу:

SQL запрос:

Имя	Тип
-----	-----

	Имя	Значение	Тип
1	tag_id	7	
2	tag_title	Тэг для отчета	

Результат

tag_id	tag_title
1	Аниме
2	Мехи
3	Персонажи
4	Игры
5	кринж
6	хдхд
7	Тэг для отчета

Рисунок 7.3 – добавление значения в таблицу

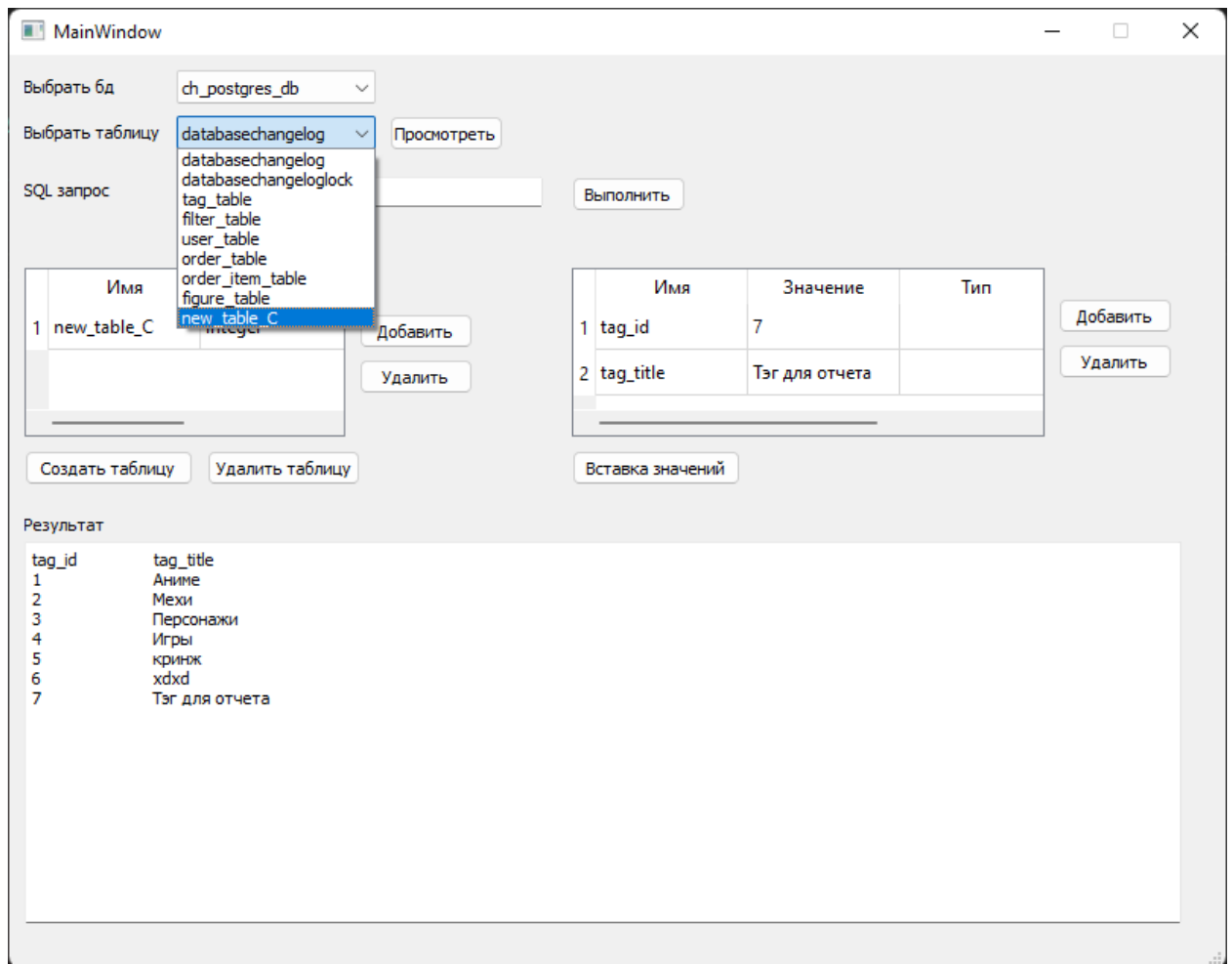


Рисунок 7.4 – добавление новой таблицы

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]

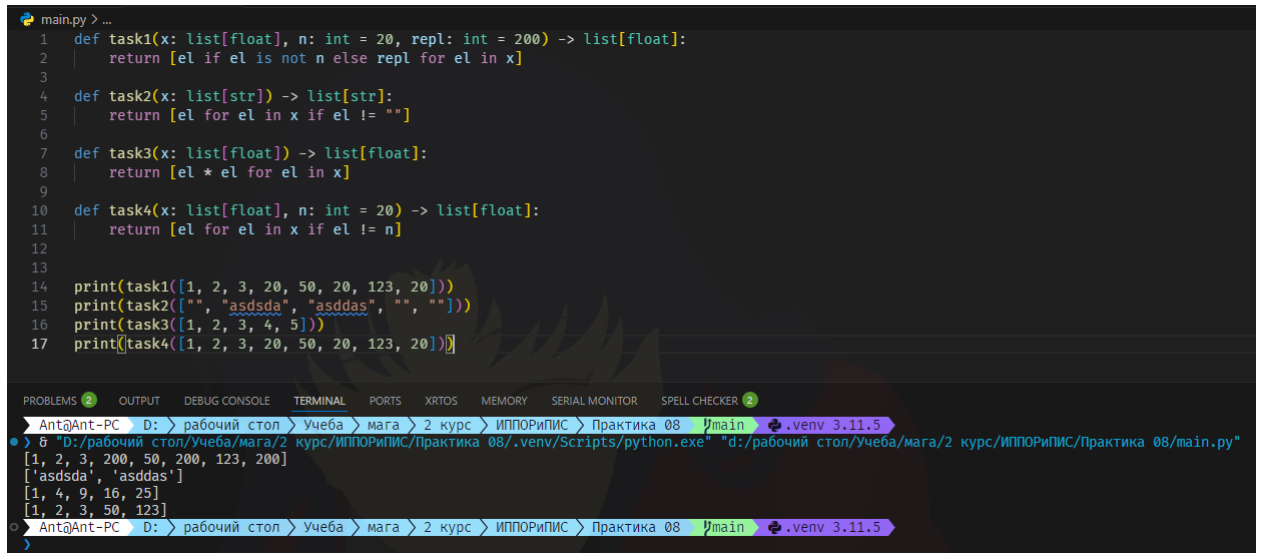
def task2(x: list[str]) -> list[str]:
    return [el for el in x if el != ""]

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]))
```



The screenshot shows a Python IDE with a dark theme. The editor window displays a script named `main.py` with the following code:

```
1 def task1(x: list[float], n: int = 20, repl: int = 200) -> list[float]:  
2     return [el if el is not n else repl for el in x]  
3  
4 def task2(x: list[str]) -> list[str]:  
5     return [el for el in x if el != ""]  
6  
7 def task3(x: list[float]) -> list[float]:  
8     return [el * el for el in x]  
9  
10 def task4(x: list[float], n: int = 20) -> list[float]:  
11     return [el for el in x if el != n]  
12  
13  
14 print(task1([1, 2, 3, 20, 50, 20, 123, 20]))  
15 print(task2(["", "asdsda", "asddas", "", ""]))  
16 print(task3([1, 2, 3, 4, 5]))  
17 print(task4([1, 2, 3, 20, 50, 20, 123, 20]))
```

The terminal window at the bottom shows the output of the script:

```
Ant@Ant-PC D:\> рабочий стол > Учеба > мага > 2 курс > ИППОРИПС > Практика 08 > main .venv 3.11.5  
> 0 "D:\рабочий стол\учеба\мага\2 курс\ИППОРИПС\Практика 08\.venv\Scripts\python.exe" "D:\рабочий стол\учеба\мага\2 курс\ИППОРИПС\Практика 08/main.py"  
[1, 2, 3, 200, 50, 200, 123, 200]  
['asdsda', 'asddas']  
[1, 4, 9, 16, 25]  
[1, 2, 3, 50, 123]
```

Рисунок 8.1 – результат запуска программы