МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| к.т.н., профессор |  |  |  |  | С.Г.Фомичева |
| должность, уч. степень |  | подпись, дата |  |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| **Разработка базы данных и подключение desktop-клиента логгирования исключений в распределенной информационной системе**  по дисциплине: РАСПРЕДЕЛЕННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 3843 |  |  |  | А.П. Конева |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

1. **Цель работы**

Изучить принципы построения приложений, являющихся источниками данных в распределенных информационных системах, научиться использовать методы проектирования приложений доступа к данным. Освоить механизмы SQLAlchemy ORM для проектирования desktop-клиентов РИС.

2. **Задачи**

1) В соответствии с заявленной в методических рекомендациях функциональностью, разработать клиент-серверную БД и подключить к ней desktop-клиент, являющийся комплексным проектом логгирования исключений, возникающих в ходе эксплуатации различных форм проекта поддержки лабораторных работ по дисциплинам в области информационной безопасности.

2) Desktop-клиент в зависимости от действий пользователя должен уметь переходить в режим записи исключений в базу данных системы AirLogger (сущность UserException) в реальном масштабе времени.

3) Разработать дополнительный функционал проекта в соответствии с индивидуальным вариантом.

**3. Ход работы**

В процессе выполнения лабораторной работы с помощью использования механизмов SQLAlchemy ORM был создан desktop-клиент, являющийся комплексным проектом логгирования исключений, возникающих в ходе эксплуатации различных форм проекта поддержки лабораторных работ по дисциплинам в области информационной безопасности.

На Рис.1 представлена измененная главная форма. Добавлена кнопка «Перейти к БД», которая появляется при активном чекбаттоне «Показать БД».

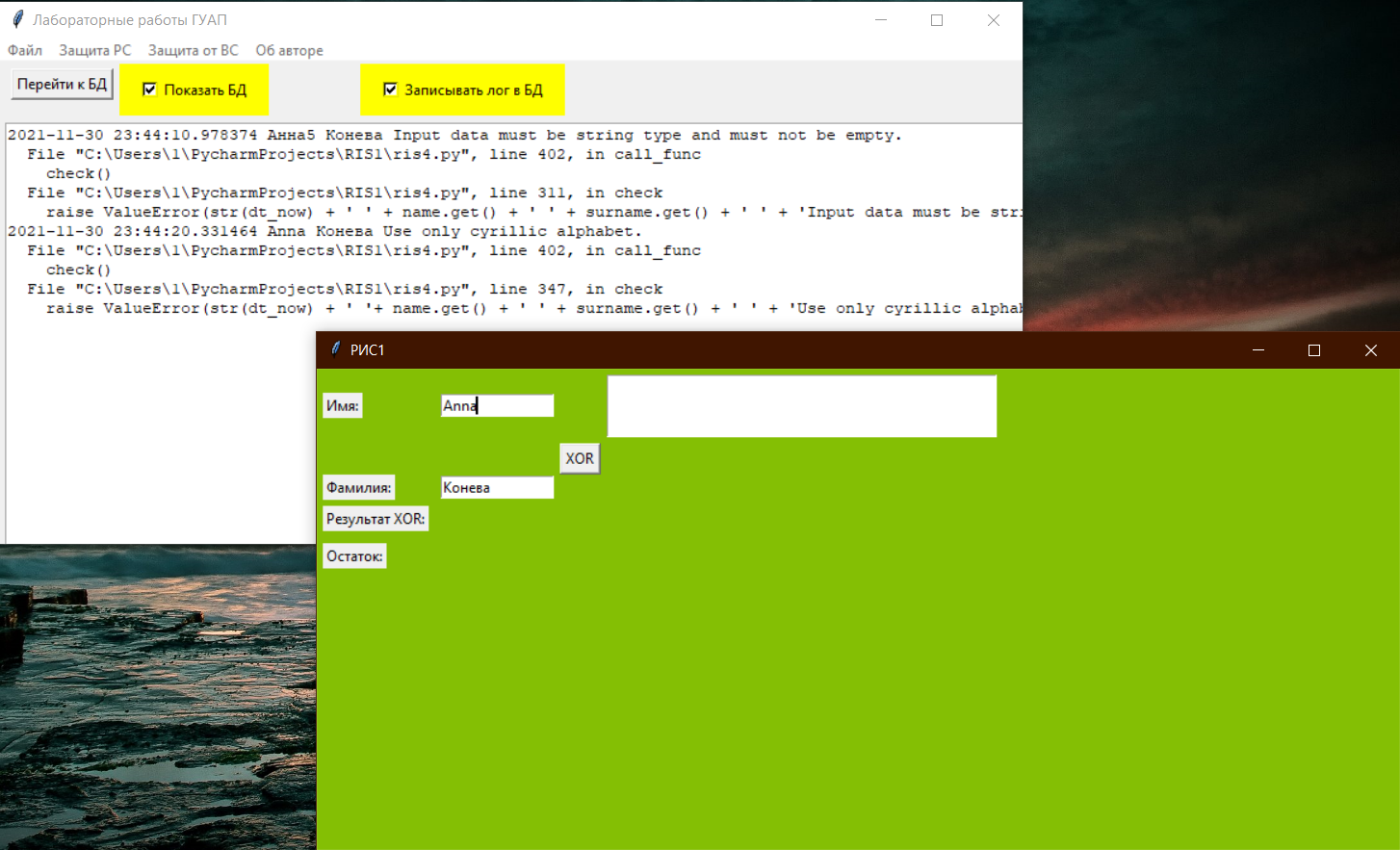


Рис.1 Измененная главная форма «Лабораторные работы ГУАП»

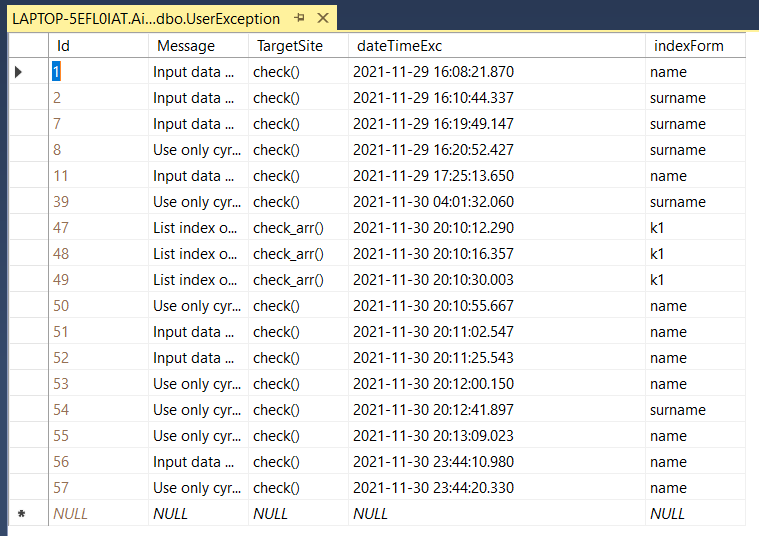
При активном чекбаттоне «Записывать лог в БД» логи, выводящиеся на главную форму, также автоматически записываются в БД AirLogger в сущность UserException. На Рис. 2 показано, что сформированный лог действительно заносится в БД.

Рис.2 Сущность UserException в БД AirLogger

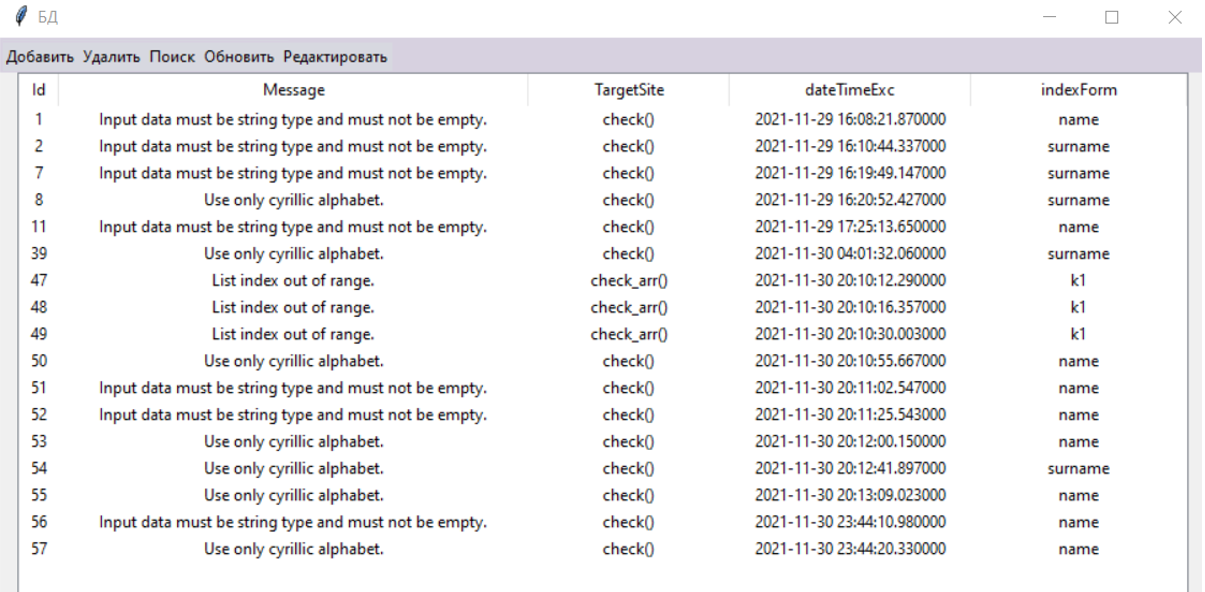
При нажатии на кнопку «Перейти к БД» открывается новая форма, позволяющая выполнять основные операции с БД (Рис.3). В toolbar реализованы кнопки «Добавить», «Удалить», «Поиск», «Обновить», «Редактировать».

Рис.3 Форма «БД»

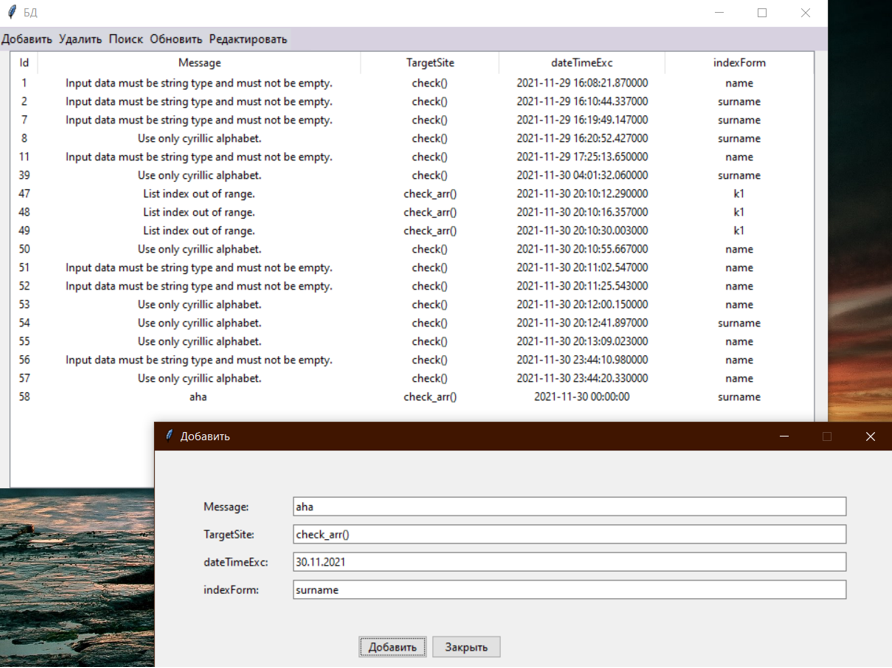
При нажатии на кнопку «Добавить» открывается новая форма «Добавить», куда нужно ввести соответствующие данные для добавления. Рассмотрим результат после нажатия кнопки «Добавить» на форме «Добавить» (Рис.4):

Рис.4 Результат нажатия кнопки «Добавить»

При выделении нужной строки и нажатии на кнопку «Удалить» происходит удаление выбранной строки. Рассмотрим результат удаления строки с Id = 53 после нажатия кнопки «Удалить» (Рис.5):

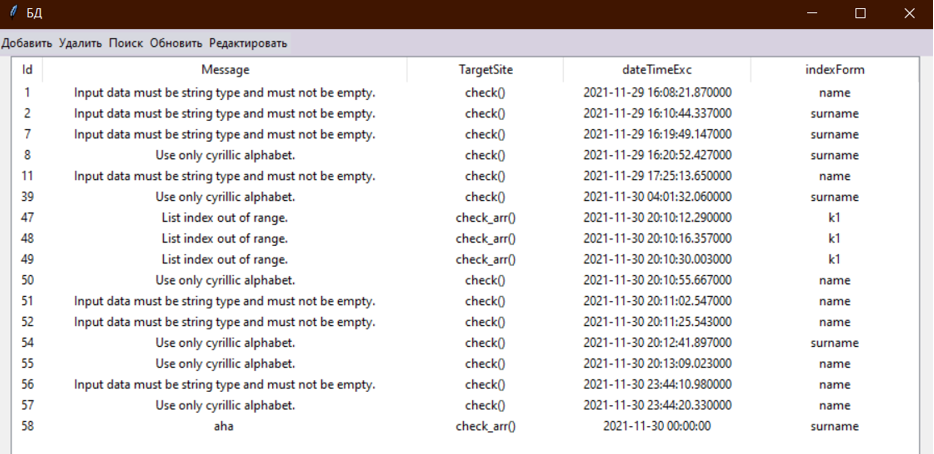


Рис.5 После нажатия кнопки «Удалить»

При выделении нужной строки и нажатии на кнопку «Редактировать» происходит редактирование выбранной строки. Рассмотрим результат удаления строки с Id = 1 после нажатия кнопки «Редактировать» (Рис.6):

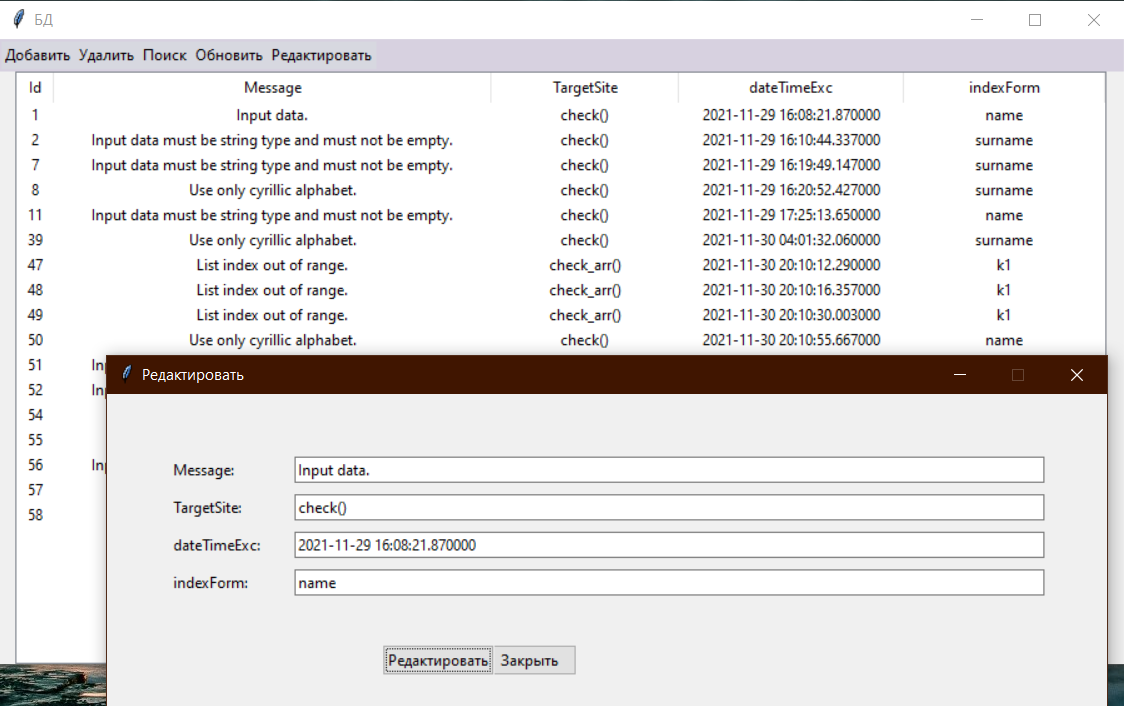


Рис.6 После нажатия кнопки «Редактировать»

На Рис. 7 продемонстрировано успешное добавление, удаление и редактирование соответствующих записей в сущности «UserException».

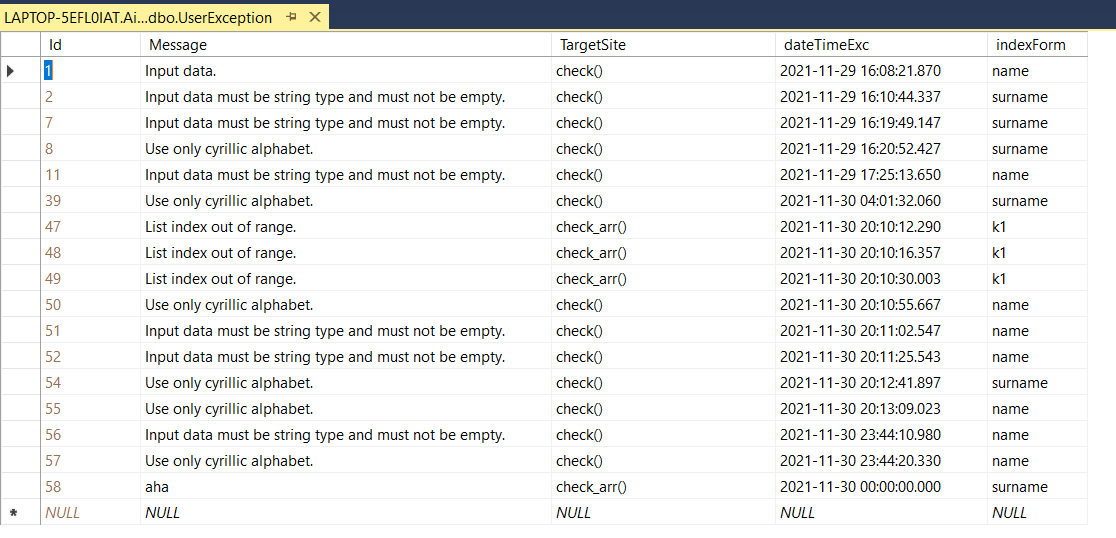


Рис.7 Измененные записи в БД AirLogger в таблице UserException

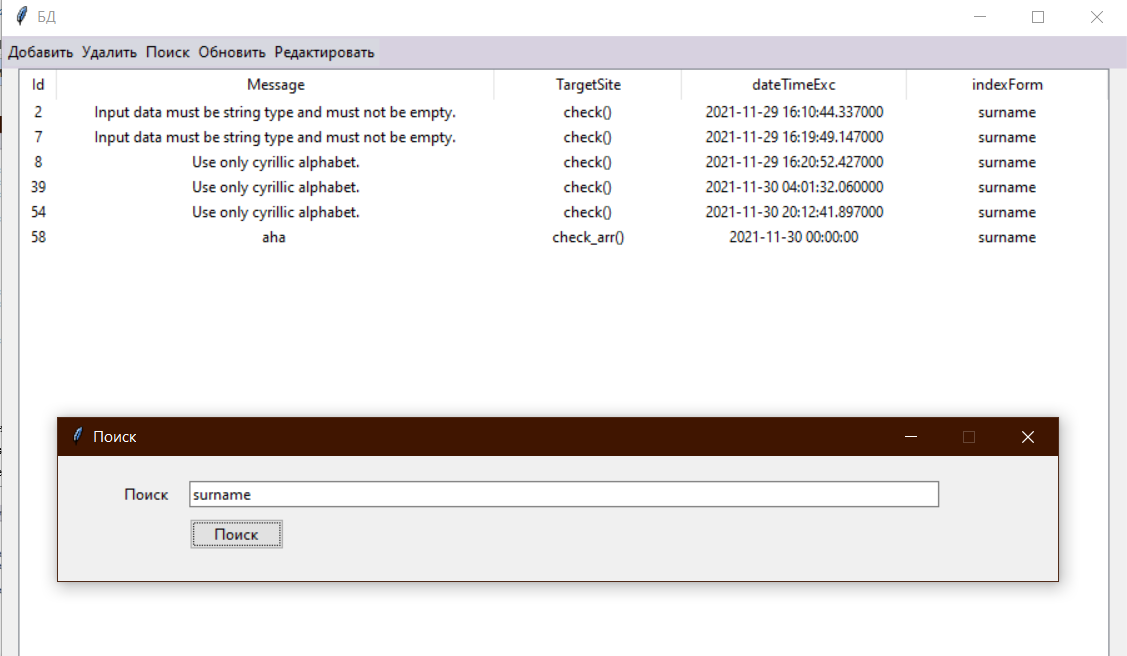
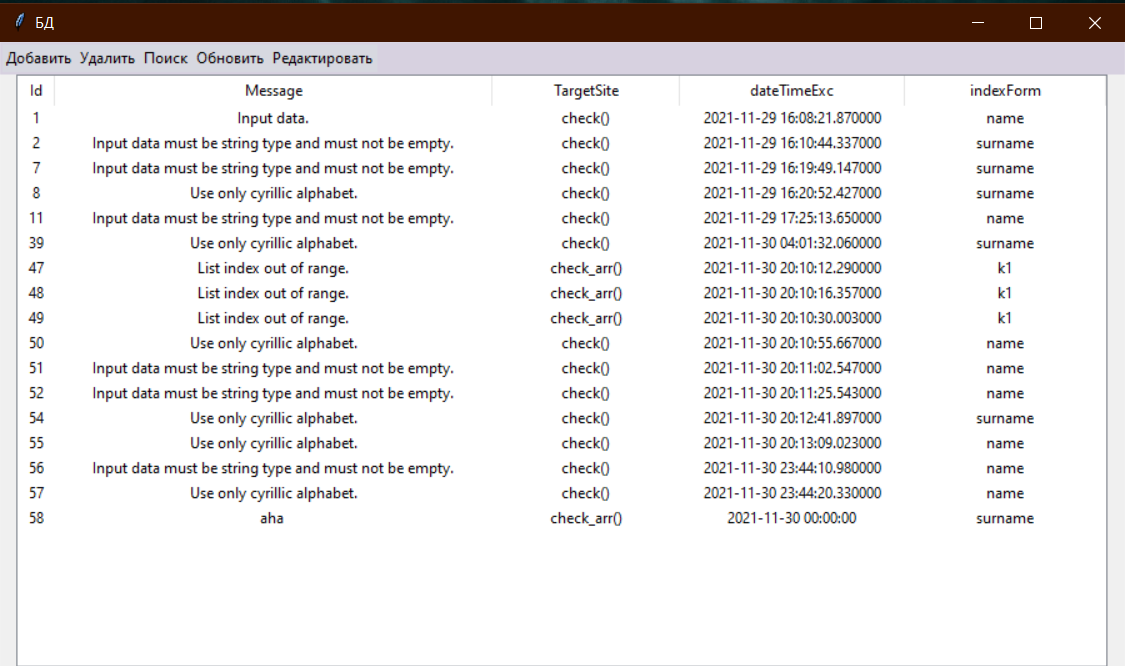
При нажатии на кнопку «Поиск» открывается форма «Поиск», куда нужно ввести данные для поиска. Рассмотрим результат после нажатия кнопки «Поиск» на форме «Поиск (Рис.8):

Рис.8 Результат нажатия кнопки «Поиск»

Для корректного восстановления всех существующих записей в сущности UserException реализовали кнопку «Обновить». Результат нажатия представлен на Рис. 9.

Рис.9 Результат нажатия кнопки «Обновить»

Подключение к БД Airlogger выполнялось с использованием SQLAlchemy ORM

при помощи функции create\_engine:

def \_\_init\_\_(self):  
 self.engine = sqlalchemy.create\_engine('mssql+pyodbc://@LAPTOP-5EFL0IAT/AirLogger?trusted\_connection=yes&driver=SQL+Server+Native+Client+11.0')  
 self.conn = self.engine.raw\_connection()  
 self.c = self.conn.cursor()

При использовании SQLAlchemy ORM взаимодействие с базой данных происходит через объект Session и метод .query и методы объектов сессии .add, .update, .delete, .filter. Например, для удаления строки из БД использовался следующий метод:

session.query(UsExc).filter(UsExc.Id == self.tree.set(selection\_item, '#1')).delete()

Разработали дополнительный функционал проекта в соответствии с индивидуальным вариантом, а именно: реализовали конфигурацию подключения к базе данных с помощью JSON-файла. Для этого создали db.json:

{  
 "ConnectionStrings": {  
 "DefaultConnection": "mssql+pyodbc://@LAPTOP-5EFL0IAT/AirLogger?trusted\_connection=yes&driver=SQL+Server+Native+Client+11.0"  
 }  
}

В основной исполняемый файл добавляем следующий код:

import json  
  
Base = declarative\_base()  
f = open('db.json')  
data = json.load(f)  
URL = data['ConnectionStrings']['DefaultConnection']  
engine = sqlalchemy.create\_engine(URL)  
conn = engine.raw\_connection()

**Дополнительное задание**

*Сделать ограничение на вводимое значение dataTimeExc при добавлении данных в UserException*.

Для этого создадим кнопку «Choose DateTime», при нажатии на которую открывается форма «Calendar», где на календаре можно выбрать дату, а в поле Entry вписать время в формате hh:mm:ss. Если выбранные дата или время не превышают системную дату или время, то выбранная дата и время вносятся в поле значения dateTimeExc(Рис.10-11).

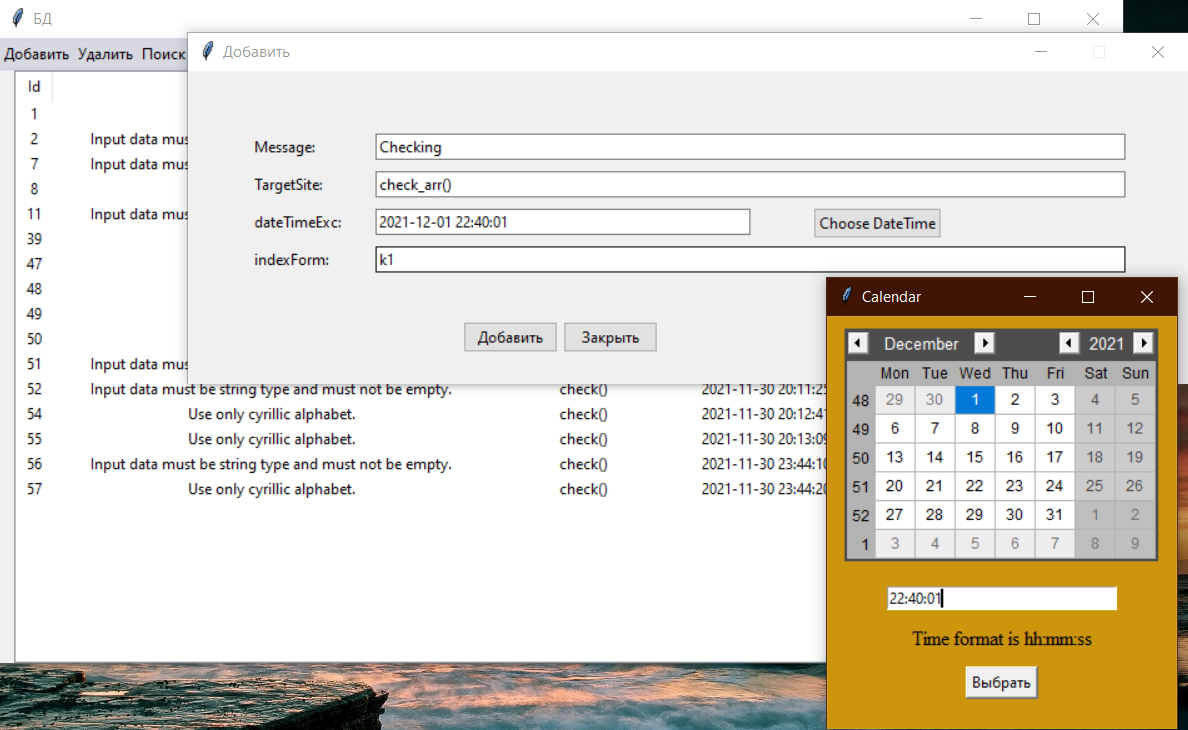


Рис.10 Успешный выбор даты и времени

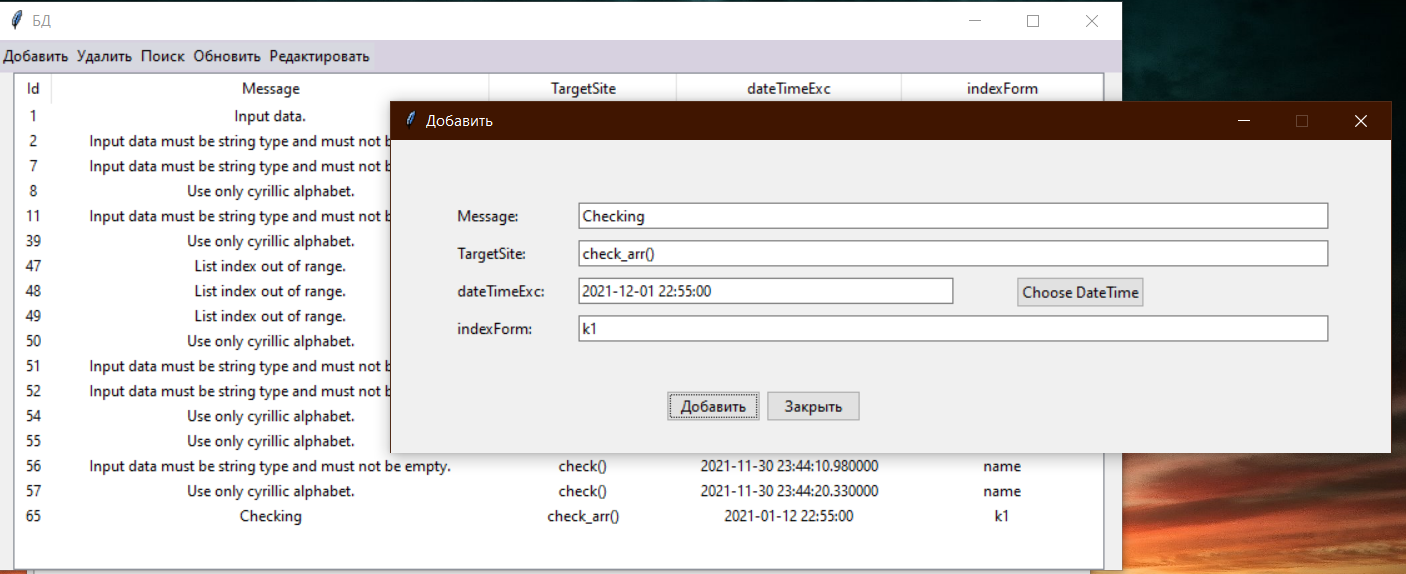


Рис.11 Успешное внесение строки с выбранной датой и временем

В случае, если выбранные дата или время превышают системную дату или время, то появляется Messagebox, сообщающий о данной ошибке (Рис.12).

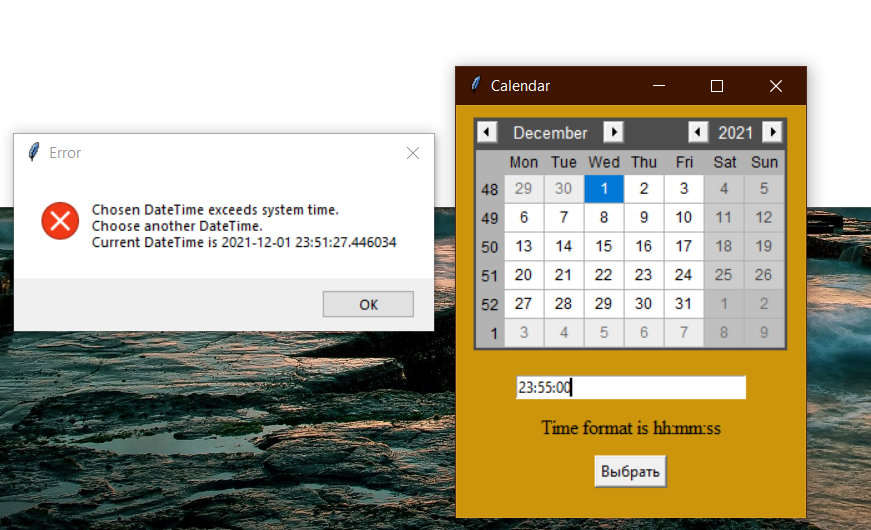


Рис.12 Ошибка выбора даты или времени

**Вывод**

В соответствии с заявленной в методических рекомендациях функциональностью разработали Desktop-клиент. Реализовали автоматическую запись логов в БД в сущность UserException (Рис. 2-3), функции добавления (Рис. 4), удаления (Рис.4), редактирования (рис.6), поиска (Рис.8), Обновления (Рис.9).

Используя механизмы SQLAlchemy ORM, подключили Desktop-клиент к БД AirLogger и организовали взаимодействие с сущностью UserException. Реализовали конфигурацию подключения к базе данных с помощью JSON-файла db.json.

Выполнили дополнительное задание, введя ограничение на вводимое значение даты и времени для dataTimeExc при добавлении данных в UserException(Рис.10-12).

**Листинг программы**

from time import sleep  
import numpy as np  
from threading import Thread  
import tkinter as tk  
from tkinter import ttk  
from tkinter.ttk import Radiobutton  
from tkinter.ttk import Checkbutton  
from tkinter import filedialog  
from tkinter import messagebox  
from tkinter import scrolledtext  
import datetime  
from tkinter import \*  
from tkcalendar import \*  
import re  
import traceback  
import sys  
from time import process\_time\_ns  
import pyodbc  
import sqlalchemy as sqlalchemy  
from sqlalchemy import engine, create\_engine, Table, Column, Integer, String, DateTime  
from sqlalchemy.ext.declarative import declarative\_base  
from sqlalchemy.orm import sessionmaker  
import json  
  
Base = declarative\_base()  
f = open('db.json')  
data = json.load(f)  
URL = data['ConnectionStrings']['DefaultConnection']  
  
engine = sqlalchemy.create\_engine(URL)  
conn = engine.raw\_connection()  
c = conn.cursor() # Создаем курсор для выполнения операций с базой данных  
  
class UsExc(Base):  
 \_\_tablename\_\_ = 'UserException'  
 Id = Column(Integer, primary\_key=True)  
 Message = Column(String)  
 TargetSite = Column(String)  
 dateTimeExc = Column(DateTime)  
 indexForm = Column(String)  
  
def logging(text, method, dt\_now, form):  
 Session = sessionmaker(bind = engine)  
 session = Session()  
  
 c1 = UsExc(Message = text, TargetSite = method, dateTimeExc = dt\_now, indexForm = form )  
  
 session.add(c1)  
 session.commit()  
  
  
def info():  
 messagebox.showinfo("Об авторе", "Конева Анна Павловна\nИнститут инновационных технологий в электромеханике и робототехнике\nГруппа 3843")  
  
window = Tk()  
window.title("Лабораторные работы ГУАП")  
window.geometry('850x400+200+130')  
window.grid\_columnconfigure(0, weight=1)  
  
#чекбаттон  
def chk\_btn():  
 if chk\_show\_db.get() == 1:  
 btn\_show\_bd.place(x=10, y = 5)  
 if chk\_show\_db.get() == 0:  
 btn\_show\_bd.place\_forget() #скрыть кнопку Показать БД  
  
chk\_show\_db = IntVar()  
show\_bd\_checkbutton = Checkbutton(text="Показать БД", variable=chk\_show\_db, onvalue=1, offvalue=0, padx=15, pady=10, bg = 'yellow', var=chk\_show\_db, command = chk\_btn)  
show\_bd\_checkbutton.place(x=100, y = 1)  
chk\_show\_db.set(False)  
  
chk\_wr\_logs = IntVar()  
wr\_logs\_checkbutton = Checkbutton(text="Записывать лог в БД", variable=chk\_wr\_logs, onvalue=1, offvalue=0, padx=15, pady=10, bg = 'yellow', var=chk\_wr\_logs)  
wr\_logs\_checkbutton.place(x=300, y = 1)  
chk\_wr\_logs.set(True)  
  
  
def open\_db():  
 class Main(tk.Frame):  
 def \_\_init\_\_(self, root4):  
 super().\_\_init\_\_(root4)  
 self.init\_main()  
 self.db = db  
 self.view\_logs()  
  
 def init\_main(self):  
 toolbar = tk.Frame(root4, bg='#d7d1e0', bd=2)  
 toolbar.pack(side=tk.TOP, fill=tk.X)  
  
 btn\_open\_add\_dialog = tk.Button(toolbar, text='Добавить', command=self.open\_add\_dialog, bg='#d7d8e0', bd=0, compound=tk.TOP)  
 btn\_open\_add\_dialog.pack(side=tk.LEFT)  
  
 btn\_delete = tk.Button(toolbar, text='Удалить', bg='#d7d8e0', bd=0,compound=tk.TOP, command=self.delete\_log)  
 btn\_delete.pack(side=tk.LEFT)  
  
 btn\_search = tk.Button(toolbar, text='Поиск', bg='#d7d8e0', bd=0,compound=tk.TOP, command=self.open\_search\_dialog)  
 btn\_search.pack(side=tk.LEFT)  
  
 btn\_update = tk.Button(toolbar, text='Обновить', bg='#d7d8e0', bd=0, compound=tk.TOP, command=self.view\_logs)  
 btn\_update.pack(side=tk.LEFT)  
  
 btn\_edit = tk.Button(toolbar, text='Редактировать', bg='#d7d8e0', bd=0, compound=tk.TOP, command=self.open\_edit\_dialog)  
 btn\_edit.pack(side=tk.LEFT)  
  
 self.tree = ttk.Treeview(self, columns=('Id', 'Message', 'TargetSite', 'dateTimeExc', 'indexForm'),  
 height=50, show='headings')  
  
 self.tree.column('Id', width=30, anchor=tk.CENTER)  
 self.tree.column('Message', width=350, anchor=tk.CENTER)  
 self.tree.column('TargetSite', width=150, anchor=tk.CENTER)  
 self.tree.column('dateTimeExc', width=180, anchor=tk.CENTER)  
 self.tree.column('indexForm', width=160, anchor=tk.CENTER)  
  
 self.tree.heading('Id', text='Id')  
 self.tree.heading('Message', text='Message')  
 self.tree.heading('TargetSite', text='TargetSite')  
 self.tree.heading('dateTimeExc', text='dateTimeExc')  
 self.tree.heading('indexForm', text='indexForm')  
 self.tree.pack()  
  
 def view\_logs(self): #для update, подтягивает текущие строки в сущности  
 Session = sessionmaker(bind=engine)  
 session = Session()  
 result = session.query(UsExc.Id, UsExc.Message, UsExc.TargetSite, UsExc.dateTimeExc, UsExc.indexForm) #select  
 for row in self.tree.get\_children():  
 self.tree.delete(row)  
 for row in result:  
 self.tree.insert('', 'end', values=tuple(row))  
  
 def open\_add\_dialog(self):  
 add\_dialog = tk.Toplevel()  
  
 add\_dialog.title('Добавить')  
 add\_dialog.geometry('800x250+300+400')  
 add\_dialog.resizable(False, False)  
  
 label\_Message = tk.Label(add\_dialog, text='Message:')  
 label\_Message.place(x=50, y=50)  
 label\_TargetSite = tk.Label(add\_dialog, text='TargetSite:')  
 label\_TargetSite.place(x=50, y=80)  
 label\_dateTimeExc = tk.Label(add\_dialog, text='dateTimeExc:')  
 label\_dateTimeExc.place(x=50, y=110)  
 label\_indexForm = tk.Label(add\_dialog, text='indexForm:')  
 label\_indexForm.place(x=50, y=140)  
  
 entry\_Message = ttk.Entry(add\_dialog)  
 entry\_Message.place(x=150, y=50, width=600)  
  
 entry\_TargetSite = ttk.Entry(add\_dialog)  
 entry\_TargetSite.place(x=150, y=80, width=600)  
  
 entry\_dateTimeExc = ttk.Entry(add\_dialog)  
 entry\_dateTimeExc.place(x=150, y=110, width=300)  
  
 entry\_indexForm = ttk.Entry(add\_dialog)  
 entry\_indexForm.place(x=150, y=140, width=600)  
  
 def add\_log():  
 Session = sessionmaker(bind=engine)  
 session = Session()  
 adding = UsExc(Message=entry\_Message.get(), TargetSite=entry\_TargetSite.get(),  
 dateTimeExc=entry\_dateTimeExc.get(), indexForm=entry\_indexForm.get())  
 session.add(adding)  
 session.commit()  
 self.view\_logs()  
  
 btn\_cancel = ttk.Button(add\_dialog, text='Закрыть', command=add\_dialog.destroy)  
 btn\_cancel.place(x=300, y=200)  
  
 btn\_ok = ttk.Button(add\_dialog, text='Добавить', command=add\_log)  
 btn\_ok.place(x=220, y=200)  
  
 def calendar():  
 calendar = Tk()  
 calendar.title("Calendar")  
 calendar.geometry("280x330+300+400")  
 calendar.config(bg="#cd950c")  
  
 f = ('Times', 20)  
  
 fone = Frame(calendar)  
 ftwo = Frame(calendar)  
 date\_now = datetime.datetime.now()  
 fone.pack(pady=10)  
 ftwo.pack(pady=10)  
  
 cal = Calendar(  
 fone,  
 selectmode="day",  
 year=date\_now.year,  
 month=date\_now.month,  
 day=date\_now.day  
 )  
 cal.pack(fill="both", expand=True)  
  
 time = Entry(ftwo, width=30)  
 time.pack(side=TOP, fill=X)  
  
 msg = Label(  
 calendar,  
 text="Time format is hh:mm:ss",  
 font=("Times", 12),  
 bg="#cd950c"  
 )  
 msg.pack(side=TOP)  
  
 def calendar\_destroy():  
 calendar.destroy()  
  
 def choose\_datetime():  
 date = cal.selection\_get()  
 dt\_now = datetime.datetime.now()  
 print(dt\_now)  
 h\_m\_s = time.get()  
 chosen\_str = str(date) + ' ' + h\_m\_s # chosen\_date\_in\_str\_format  
 chosen\_dt = datetime.datetime.strptime(chosen\_str, '%Y-%m-%d %H:%M:%S') # chosen\_date\_in\_datetime\_format  
 if chosen\_dt <= dt\_now:  
 entry\_dateTimeExc.insert(0, chosen\_str)  
 calendar\_destroy()  
 else:  
 messagebox.showerror("Error",  
 f"Chosen datetime exceeds system time.\nChoose another datetime.\nCurrent datetime is {date} {dt\_now.time()}")  
  
 btn\_choose = Button(calendar, text="Bыбрать", padx=1, pady=1, command=choose\_datetime)  
 btn\_choose.pack(pady=10)  
  
 msg\_display = Label(calendar, text="", bg="#cd950c")  
 msg\_display.pack(pady=10)  
 calendar.mainloop()  
  
 btn\_choose\_date = ttk.Button(add\_dialog, text='Choose DateTime', command=calendar)  
 btn\_choose\_date.place(x=500, y=109)  
  
  
 def delete\_log(self):  
 Session = sessionmaker(bind=engine)  
 session = Session()  
 for selection\_item in self.tree.selection():  
 session.query(UsExc).filter(UsExc.Id == self.tree.set(selection\_item, '#1')).delete() #delete  
 self.tree.delete(selection\_item)  
 session.commit()  
  
 def open\_search\_dialog(self):  
 search\_dialog = tk.Toplevel()  
 search\_dialog.title('Поиск')  
 search\_dialog.geometry('800x100+300+500')  
 search\_dialog.resizable(False, False)  
  
 label\_search = tk.Label(search\_dialog, text='Поиск')  
 label\_search.place(x=50, y=20)  
  
 entry\_search = ttk.Entry(search\_dialog)  
 entry\_search.place(x=105, y=20, width=600)  
  
 def search\_log():  
 Session = sessionmaker(bind=engine)  
 session = Session()  
 searching = session.query(UsExc.Id, UsExc.Message, UsExc.TargetSite, UsExc.dateTimeExc, UsExc.indexForm).filter(UsExc.indexForm.like(entry\_search.get())).all()  
  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 for row in searching:  
 self.tree.insert('', tk.END, values=tuple(row))  
  
 btn\_search = ttk.Button(search\_dialog, text='Поиск', command = search\_log)  
 btn\_search.place(x=105, y=50)  
  
  
  
 def open\_edit\_dialog(self):  
 edit\_dialog = tk.Toplevel()  
  
 edit\_dialog.title('Редактировать')  
 edit\_dialog.geometry('800x250+300+300')  
 edit\_dialog.resizable(False, False)  
  
 row = ''  
 Session = sessionmaker(bind=engine)  
 session = Session()  
 for selection\_item in self.tree.selection():  
 row = session.query(UsExc.Id, UsExc.Message, UsExc.TargetSite, UsExc.dateTimeExc,  
 UsExc.indexForm).filter(UsExc.Id == self.tree.set(selection\_item, '#1')).one()  
  
 label\_Message = tk.Label(edit\_dialog, text='Message:')  
 label\_Message.place(x=50, y=50)  
 label\_TargetSite = tk.Label(edit\_dialog, text='TargetSite:')  
 label\_TargetSite.place(x=50, y=80)  
 label\_dateTimeExc = tk.Label(edit\_dialog, text='dateTimeExc:')  
 label\_dateTimeExc.place(x=50, y=110)  
 label\_indexForm = tk.Label(edit\_dialog, text='indexForm:')  
 label\_indexForm.place(x=50, y=140)  
  
 entry\_Message = ttk.Entry(edit\_dialog)  
 entry\_Message.place(x=150, y=50, width=600)  
 entry\_Message.insert(0, row[1])  
  
 entry\_TargetSite = ttk.Entry(edit\_dialog)  
 entry\_TargetSite.place(x=150, y=80, width=600)  
 entry\_TargetSite.insert(0, row[2])  
  
 entry\_dateTimeExc = ttk.Entry(edit\_dialog)  
 entry\_dateTimeExc.place(x=150, y=110, width=600)  
 entry\_dateTimeExc.insert(0, row[3])  
  
 entry\_indexForm = ttk.Entry(edit\_dialog)  
 entry\_indexForm.place(x=150, y=140, width=600)  
 entry\_indexForm.insert(0, row[4])  
  
  
 def edit\_log():  
 Session = sessionmaker(bind=engine)  
 session = Session()  
  
 for selection\_item in self.tree.selection():  
 session.query(UsExc).filter(  
 UsExc.Id == self.tree.set(selection\_item, '#1')).update(  
 {UsExc.Message: entry\_Message.get(), UsExc.TargetSite: entry\_TargetSite.get(),  
 UsExc.indexForm: entry\_indexForm.get()}, synchronize\_session = 'fetch')  
 session.commit()  
 self.view\_logs()  
  
  
 btn\_cancel = ttk.Button(edit\_dialog, text='Закрыть', command=edit\_dialog.destroy)  
 btn\_cancel.place(x=300, y=200)  
  
 btn\_ok = ttk.Button(edit\_dialog, text='Редактировать', command=edit\_log)  
 btn\_ok.place(x=220, y=200)  
  
  
  
  
 class DB:  
 def \_\_init\_\_(self):  
 self.engine = sqlalchemy.create\_engine('mssql+pyodbc://@LAPTOP-5EFL0IAT/AirLogger?trusted\_connection=yes&driver=SQL+Server+Native+Client+11.0')  
 self.conn = self.engine.raw\_connection()  
 self.c = self.conn.cursor() # Создаем курсор для выполнения операций с базой данных  
  
 if \_\_name\_\_ == "\_\_main\_\_":  
 root4 = tk.Tk()  
 db = DB()  
 app = Main(root4)  
 app.pack()  
 root4.title("БД")  
 root4.geometry("900x500+300+100")  
 root4.mainloop()  
  
  
#главное окно window  
btn\_show\_bd = Button(window, text="Перейти к БД", command = open\_db)  
btn\_show\_bd.place  
  
logs = scrolledtext.ScrolledText(window, width=500, height=400)  
logs.place(x=5, y = 50)  
  
def open\_laba1():  
 def check():  
 dt\_now = datetime.datetime.now()  
 method = 'check()'  
 check\_n\_on\_number = any(map(str.isdigit, name.get())) #проверка имени на наличие цифр  
 check\_sn\_on\_number = any((map(str.isdigit, surname.get())))  
 name\_get = name.get()  
 surname\_get = surname.get()  
 ng = re.search(r'[^а-яА-Я]', name\_get) #проверка текста на кириллицу  
 sng = re.search(r'[^а-яА-Я]', surname\_get)  
  
 if not name\_get.isalpha(): # если кака-то из строк пустая или содержит цифру  
 if chk\_wr\_logs.get() == 1:  
 text = 'Input data must be string type and must not be empty.'  
 form = 'name'  
 logging(text, method, dt\_now, form)  
 raise ValueError(str(dt\_now) + ' ' + name.get() + ' ' + surname.get() + ' ' + 'Input data must be string type and must not be empty.' +'\n')  
 if not surname\_get.isalpha():  
 if chk\_wr\_logs.get() == 1:  
 text = 'Input data must be string type and must not be empty.'  
 form = 'surname'  
 logging(text, method, dt\_now, form)  
 raise ValueError(str(dt\_now) + ' ' + name.get() + ' ' + surname.get() + ' ' + 'Input data must be string type and must not be empty.' + '\n')  
 if check\_n\_on\_number:  
 if chk\_wr\_logs.get() == 1:  
 text = 'Input data must be string type.'  
 form = 'name'  
 logging(text, method, dt\_now, form)  
 raise ValueError(str(dt\_now) + ' '+ name.get() + ' ' + surname.get() + ' ' + 'Input data must be string type.'+'\n')  
 if check\_sn\_on\_number:  
 if chk\_wr\_logs.get() == 1:  
 text = 'Input data must be string type.'  
 form = 'surname'  
 logging(text, method, dt\_now, form)  
 raise ValueError(str(dt\_now) + ' ' + name.get() + ' ' + surname.get() + ' ' + 'Input data must be string type.' + '\n')  
 if not name\_get: #on null, тип если отсутствует содержимое строки  
 if chk\_wr\_logs.get() == 1:  
 text = 'The string is empty.'  
 form = 'name'  
 logging(text, method, dt\_now, form)  
 raise ValueError(str(dt\_now) + ' ' + name.get() + ' ' + surname.get() + ' ' + 'The string is empty.'+'\n')  
 if not surname\_get:  
 if chk\_wr\_logs.get() == 1:  
 text = 'The string is empty.'  
 form = 'surname'  
 logging(text, method, dt\_now, form)  
 raise ValueError(str(dt\_now) + ' ' + name.get() + ' ' + surname.get() + ' ' + 'The string is empty.' + '\n')  
 if ng is not None:  
 if chk\_wr\_logs.get() == 1:  
 text = 'Use only cyrillic alphabet.'  
 form = 'name'  
 logging(text, method, dt\_now, form)  
 raise ValueError(str(dt\_now) + ' '+ name.get() + ' ' + surname.get() + ' ' + 'Use only cyrillic alphabet.'+'\n')  
 if sng is not None:  
 if chk\_wr\_logs.get() == 1:  
 text = 'Use only cyrillic alphabet.'  
 form = 'surname'  
 logging(text, method, dt\_now, form)  
 raise ValueError(str(dt\_now) + ' ' + name.get() + ' ' + surname.get() + ' ' + 'Use only cyrillic alphabet.' + '\n')  
 string = str(dt\_now) + ' ' + name.get() + ' ' + surname.get() + '\n'  
 logs.insert(INSERT, string)  
  
  
  
 def xor():  
 ost = None  
 a = [ord(i) for i in name.get()]  
 b = [ord(i) for i in surname.get()]  
 la = len(a)  
 lb = len(b)  
 c1 = []  
 global d  
 if la < lb:  
 c = [1 for i in range(la)]  
 for i in range(0, len(a)):  
 c[i] = a[i] ^ b[i]  
 nb = surname.get()[la:]  
 c1 = [chr(c[i]+32) for i in range(len(c))]  
 c1.extend(nb) #добавляет элементы nb к списку c1  
 ost = nb  
 s = c1  
 d = [ord(c) for c in s]  
 else:  
 c = [1 for i in range(lb)]  
 for i in range(0, len(b)):  
 c[i] = a[i] ^ b[i]  
 na = name.get()[lb:]  
 c1 = [chr(c[i] + 32) for i in range(len(c))]  
 c1.extend(na)  
 ost = na  
 if la == lb:  
 c = [1 for i in range(la)]  
 for i in range(0, len(a)):  
 c[i] = a[i] ^ b[i]  
 ost = ' '  
  
 res = Label(root, text=c1)  
 res.grid(column=2, row=4, padx=5, pady=5)  
  
 ostatok = Label(root, text=ost)  
 ostatok.grid(row=5, column=1,padx=5, pady=5)  
  
 dop\_text\_res = ascii(name.get()) + '\n' + ascii(surname.get()) + '\n' + ascii(d)  
 dop\_text.insert(INSERT, dop\_text\_res)  
  
 def call\_func():  
 try: #try и except для исключений  
 check()  
 except ValueError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
  
 return None  
 xor()  
  
  
 global root  
 global dop\_text  
 global surname  
 global name  
 global logs  
 root = Tk()  
 root.title("РИС1")  
 root.geometry('900x400+550+390')  
 root.config(bg='#84BF04')  
  
 lbl\_n = Label(root, text="Имя:")  
 lbl\_sn = Label(root, text="Фамилия:")  
 lbl\_res = Label (root, text='Результат XOR:')  
 lbl\_ost = Label (root, text='Остаток:')  
 lbl\_n.grid(row=0, column=0, sticky="w", padx=5)  
 lbl\_sn.grid(row=2, column=0, sticky="w", padx=5)  
 lbl\_res.grid(row=4, column=0, sticky="w", padx=5, pady=5)  
 lbl\_ost.grid(row=5, column=0, sticky="w", padx=5, pady=5)  
  
 name = Entry(root, width=15)  
 name.grid(column=2, row=0, padx=5, pady=5)  
 surname = Entry(root, width=15)  
 surname.grid(column=2, row=2)  
  
 dop\_text=Text(root, width=40, height=3)  
 dop\_text.grid(column=7, row=0, padx=5, pady=5)  
  
 btn\_xor = Button(root, text="XOR", command=call\_func)  
 btn\_xor.grid(column=3, row=1)  
  
 root.mainloop()  
  
def open\_laba2():  
 arr = []  
 def check\_arr():  
 global dt\_now  
 global k1  
 global k2  
 le = int(spin.get())  
 dt\_now = datetime.datetime.now()  
 k1 = int(txt1.get())  
 k2 = int(txt2.get())  
 method = 'check\_arr()'  
 if k1 is None and k2 is None:  
 if chk\_wr\_logs.get() == 1:  
 text = 'k1 is None.'  
 form = 'k1'  
 logging(text, method, dt\_now, form)  
 raise IndexError(str(dt\_now) + ' ' + 'k1 is None'+ '\n')  
 if k2 is None:  
 if chk\_wr\_logs.get() == 1:  
 text = 'k2 is None.'  
 form = 'k2'  
 logging(text, method, dt\_now, form)  
 raise IndexError(str(dt\_now) + ' ' + 'k2 is None' + '\n')  
 if k1 > k2:  
 if chk\_wr\_logs.get() == 1:  
 text = 'List index out of range.'  
 form = 'k1'  
 logging(text, method, dt\_now, form)  
 raise IndexError(str(dt\_now) + ' ' + 'List index out of range' +'\n')  
  
  
 def random(spin): # число элтов массива указанное в spinbox  
  
 global arrD  
 global l  
 l = int(spin.get())  
 #t\_start = process\_time\_ns()  
 arrA = np.random.randint(-50, 50, int(spin.get()))  
  
 if l < 10:  
 text = 'List index out of range.'  
 form = 'spin'  
 method = 'random(spin)'  
 logging(text, method, dt\_now, form)  
 raise IndexError(str(dt\_now) + ' ' + 'Too little array' + '\n')  
 arrD = np.zeros(l, int)  
 for i in range(0, 10): # меньше 10 элтов нельзя вводить иначе все рушится , надо делать exeption  
 arrD[i] = arrA[i] + i  
 for i in range(10, l):  
 arrD[i] = arrA[i] - i  
 print('Массив А: ', arrA)  
 print('Массив D: ', arrD)  
  
 arr.append(arrD)  
 return (arrA, arrD)  
  
 def emptyArr(spin):  
 arrA = ['' for i in range(int(spin.get()))]  
 arr.append(arrA)  
 print(arrA)  
 return (arrA)  
  
 def ziro\_ex():  
 global t\_start  
 t\_start = process\_time\_ns()  
 min = abs(arrD[0])  
 for i in range(l):  
 if (arrD[i] > 0) and abs(arrD[i]) < min:  
 min = abs(arrD[i])  
 for i in range(l):  
 if arrD[i] == min:  
 arrD[i] = 0  
 print('ziro\_ex: ', arrD)  
 return (arrD)  
  
 def exch():  
 k1= int(txt1.get())-1  
 k2 = int(txt2.get())  
 for i in range(k1, k2):  
 arrD[i] = -(arrD[i])  
 print('exch: ', arrD)  
 return(arrD)  
  
 def call\_f():  
 try: #try и except для исключений  
 check\_arr()  
 except IndexError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
 return None  
 except ValueError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
 return None  
 try:  
 random(spin)  
 except IndexError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
  
 ziro\_ex()  
 exch()  
  
 #threadscalc()  
  
  
 def call\_f\_h():  
 try: #try и except для исключений  
 check\_arr()  
 except IndexError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
 return None  
 except ValueError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
 return None  
 try:  
 ziro\_ex()  
 except ValueError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
 try:  
 exch()  
 except ValueError as e:  
 logs.insert(INSERT, e)  
 exc\_type, exc\_value, exc\_traceback = sys.exc\_info()  
 traces = traceback.extract\_tb(exc\_traceback)  
 for trace\_line in traces.format():  
 logs.insert(INSERT, trace\_line)  
  
  
 textFont1 = ("Arial", 10, "bold italic")  
 textFont2 = ("Arial", 12, "bold")  
 textFont3 = ("Arial", 8, "bold")  
  
 class LabelWidget(Entry):  
 def \_\_init\_\_(self, master, x, y, text):  
 self.text = StringVar(master=master)  
 self.text.set(text)  
 Entry.\_\_init\_\_(self, master=master)  
 self.config(relief="ridge", font=textFont1,  
 justify='center', width=3,  
 textvariable=self.text,  
 state="readonly")  
 self.grid(column=x, row=y)  
  
 class EntryWidget(Entry):  
 def \_\_init\_\_(self, master, x, y, val):  
  
 Entry.\_\_init\_\_(self, master=master)  
 self.value = StringVar(master=master)  
 self.config(textvariable=self.value, width=3,  
 relief="ridge", font=textFont1,  
 justify='center')  
 self.grid(column=x, row=y)  
 self.value.set(val)  
  
 class EntryGrid(Tk):  
 *''' Dialog box with Entry widgets arranged in columns and rows.'''* def \_\_init\_\_(self, root2, colList, rowList, arrD):  
  
 self.root2 = root2  
 self.cols = colList[:]  
 self.colList = colList[:]  
 self.colList.insert(0, "")  
 self.rowList = rowList  
 self.root2.grid()  
 self.make\_header()  
 # print(arr)  
 self.gridDict = {}  
  
 for j in range(len(self.rowList)):  
 for i in range(0, len(arrD[j])):  
 if i > len(arrD[j]):  
 raise IndexError(str(dt\_now) + ' ' + 'List index out of range' + '\n')  
 w = EntryWidget(self.root2, i+1, j + 1, arrD[j][i])  
 self.gridDict[(i, j)] = w.value  
  
 def handler(event, col=i, row=j):  
 return self.\_\_entryhandler(col, row)  
  
 w.bind(sequence="<FocusOut>", func=handler)  
  
 def make\_header(self):  
 self.hdrDict = {}  
 for i, label in enumerate(self.colList):  
 def handler(event, col=i, row=0, text=label):  
 return self.\_\_headerhandler(col, row, text)  
  
 w = LabelWidget(self.root2, i, 0, label)  
 self.hdrDict[(i, 0)] = w  
 w.bind(sequence="<KeyRelease>", func=handler)  
  
 for i, label in enumerate(self.rowList):  
 def handler(event, col=0, row=i + 1, text=label):  
 return self.\_\_headerhandler(col, row, text)  
  
 w = LabelWidget(self.root2, 0, i + 1, label)  
 self.hdrDict[(0, i + 1)] = w  
 w.bind(sequence="<KeyRelease>", func=handler)  
  
 def \_\_entryhandler(self, col, row):  
 s = self.gridDict[(col, row)].get()  
  
 if s.upper().strip() == "EXIT":  
 self.destroy()  
 elif s.upper().strip() == "DEMO":  
 self.demo()  
  
 def demo(self):  
 *''' enter a number into each Entry field '''* for i in range(len(self.cols)):  
 for j in range(len(self.rowList)):  
 sleep(0.25)  
 self.set(i, j, "")  
 self.update\_idletasks()  
 sleep(0.1)  
 self.set(i, j, i + 1 + j)  
 self.update\_idletasks()  
  
 def \_\_headerhandler(self, col, row, text):  
 *''' has no effect when Entry state=readonly '''* self.hdrDict[(col, row)].text.set(text)  
  
 def get(self, x, y):  
 return self.gridDict[(x, y)].get()  
  
 def set(self, x, y, v):  
 self.gridDict[(x, y)].set(v)  
 return v  
  
 def getArrFG(self, rowN, colN):  
 tempArr = []  
 for x in range(colN):  
 tempArr.append(int(self.gridDict[(x+1, len(rowN)-1)].get()))  
  
 arr[-1] = tempArr  
 #threadscalc()  
  
 return arr  
  
  
 if \_\_name\_\_ == "\_\_main\_\_":  
 rows = []  
 root2 = Tk()  
 root2.geometry('780x400')  
 root2.title("РИС2")  
 lbl = Label(root2, text="Размер массива")  
 lbl.place(x=490, y=3)  
 lbl = Label(root2, text="k2")  
 lbl.place(x=680, y=50)  
 lbl = Label(root2, text="k1")  
 lbl.place(x=600, y=50)  
 txt1 = Entry(root2, width=5)  
 txt1.place(x=600, y=70)  
 txt2 = Entry(root2, width=5) # текстовое окошко для ввода сила эт тип из задания  
 txt2.place(x=680, y=70)  
  
 datt = Label(root2, font=("helvetica", 11))  
 datt.place(x=500, y=250)  
  
  
  
 # create a scrollbar widget and set its command to the text widget  
  
 selected = IntVar(master=root2)  
 selected.set(1)  
 rad1 = Radiobutton(root2, text='Случайно', variable=selected, value=1)  
 rad1.place(x=600, y=5)  
 rad2 = Radiobutton(root2, text='Вручную', variable=selected, value=2)  
 rad2.place(x=600, y=25)  
 spinVar = StringVar(master=root2)  
 spinVar.set("10")  
 spin = Spinbox(root2, from\_=10, to=30, width=5, textvariable=spinVar)  
 spin.place(x=500, y=20)  
 table = Frame(root2)  
  
 def update\_time():  
 datt.config(text=f"{datetime.datetime.now()}")  
 root2.after(100, update\_time)  
  
 def task\_interval():  
 tableview()  
 t\_stop = process\_time\_ns()  
 print("Elapsed time during the whole program in seconds:", t\_stop - t\_start)  
 return arr  
  
 def handle():  
 tableItem.getArrFG(rows, int(spinVar.get()))  
  
  
 def tableview():  
 cols = list(range(1, int(spin.get()) + 1))  
 rows.append('D')  
 global tableItem  
 if int(selected.get()) == 1:  
  
 call\_f()  
 tableItem = EntryGrid(table, cols, rows, arr)  
 elif int(selected.get()) == 2:  
 emptyArr(spin)  
 tableItem = EntryGrid(table, cols, rows, arr)  
 table.place(x=0, y=200)  
 # lArr = arrToList(arr)  
 if chk\_state.get() == True:  
 root2.after(2000, task\_interval)  
 return arr  
 btn = Button(root2, text="Создать массив", width=27, height=1, command=lambda: tableview())  
 btn.place(x=500, y=150)  
 btn = Button(root2, text="Выбрать", width=27, height=1,  
 command=lambda: handle())  
 btn.place(x=500, y=200)  
  
 chk\_state = BooleanVar(master=root2)  
 chk\_state\_thread = BooleanVar(master=root2)  
 chk\_state.set(True)  
 chk\_state\_thread.set(True)# задайте проверку состояния чекбокса  
 chk = Checkbutton(root2, text='Включить таймер', var=chk\_state)  
 chk.place(x=580, y=120)  
 chk = Checkbutton(root2, text='Использовать потоки', var=chk\_state\_thread)  
 chk.place(x=580, y=95)  
 update\_time()  
  
  
 def threadscalc():  
 thread1 = Thread(target=ziro\_ex, args=[arrD])  
 thread2 = Thread(target=exch, args=[arrD])  
 thread1.start()  
 thread2.start()  
 thread1.join()  
 thread2.join()  
  
 root2.mainloop()  
  
def save():  
 my\_doc = logs.get("1.0", END) # read from one text box t1  
 fob = filedialog.asksaveasfile(filetypes=[('text file', '\*.txt')],  
 defaultextension='.txt', initialdir='C:\\Users\1\Desktop\7 сем',  
 mode='w')  
 try:  
 fob.write(my\_doc)  
 fob.close()  
 logs.delete('1.0', END) # Delete from position 0 till end  
 logs.update()  
 except:  
 print(" Error initialdir")  
  
def open\_file():  
 file = filedialog.askopenfile()  
 logs.insert(INSERT, file.read())  
  
def create\_file():  
 logs.delete('1.0', END) # Delete from position 0 till end  
 logs.update()  
  
menuBar = Menu(window)  
  
fileMenu = Menu(menuBar, tearoff=0)  
cryptMenu= Menu(fileMenu, tearoff=0)  
cryptMenu.add\_command(label="Эль-Гамаль")  
cryptMenu.add\_command(label="RSA")  
encryptMenu= Menu(fileMenu, tearoff=0)  
encryptMenu.add\_command(label="Эль-Гамаль")  
encryptMenu.add\_command(label="RSA")  
  
fileMenu.add\_command(label="Создать", command = create\_file)  
fileMenu.add\_command(label="Открыть", command = open\_file)  
fileMenu.add\_command(label="Сохранить",command = save)  
fileMenu.add\_separator()  
fileMenu.add\_cascade(label="Шифрование", menu=cryptMenu)  
fileMenu.add\_cascade(label="Дешифрование", menu=encryptMenu)  
fileMenu.add\_separator()  
fileMenu.add\_command(label="Закрыть Alt+X", command=window.destroy)  
  
RSmenu = Menu(menuBar, tearoff=0)  
RSmenu.add\_command(label="РИС1", command=open\_laba1)  
RSmenu.add\_command(label="РИС2", command=open\_laba2)  
RSmenu.add\_command(label="РИС3")  
RSmenu.add\_command(label="РИС4")  
RSmenu.add\_command(label="РИС5")  
  
prMenu = Menu(menuBar, tearoff=0) #protect  
prMenu.add\_command(label="ЗВС1")  
prMenu.add\_command(label="ЗВС2")  
  
menuBar.add\_cascade(label="Файл", menu=fileMenu)  
menuBar.add\_cascade(label="Защита РС", menu=RSmenu) #Raspred Systems  
menuBar.add\_cascade(label="Защита от ВС", menu=prMenu) #protect  
menuBar.add\_cascade(label="Об авторе", command=info)  
  
window.config(menu=menuBar)  
  
  
  
window.mainloop()