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».

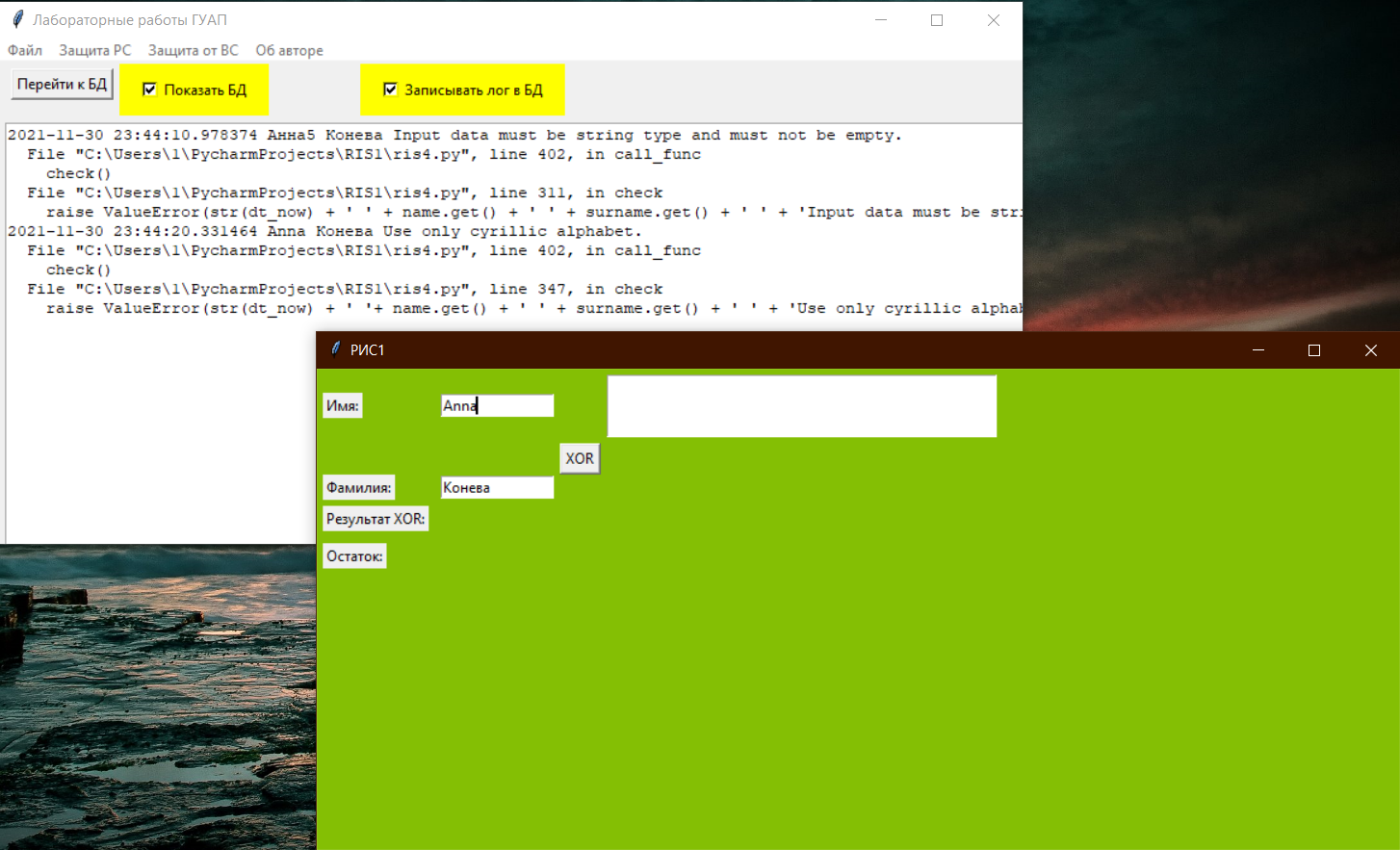


Рис.1 Измененная главная форма «Лабораторные работы ГУАП»

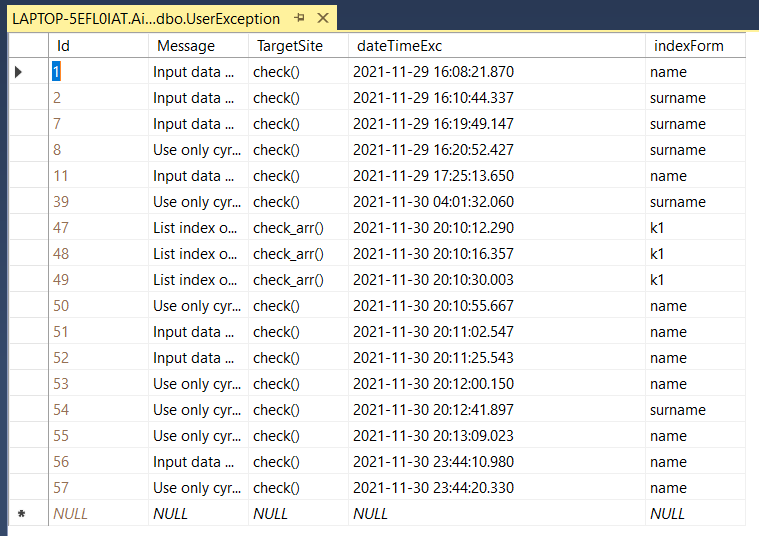
При активном чекбаттоне «Записывать лог в БД» логи, выводящиеся на главную форму, также автоматически записываются в БД AirLogger в сущность UserException. На Рис. 2 показано, что сформированный лог действительно заносится в БД.

Рис.2 Сущность UserException в БД AirLogger

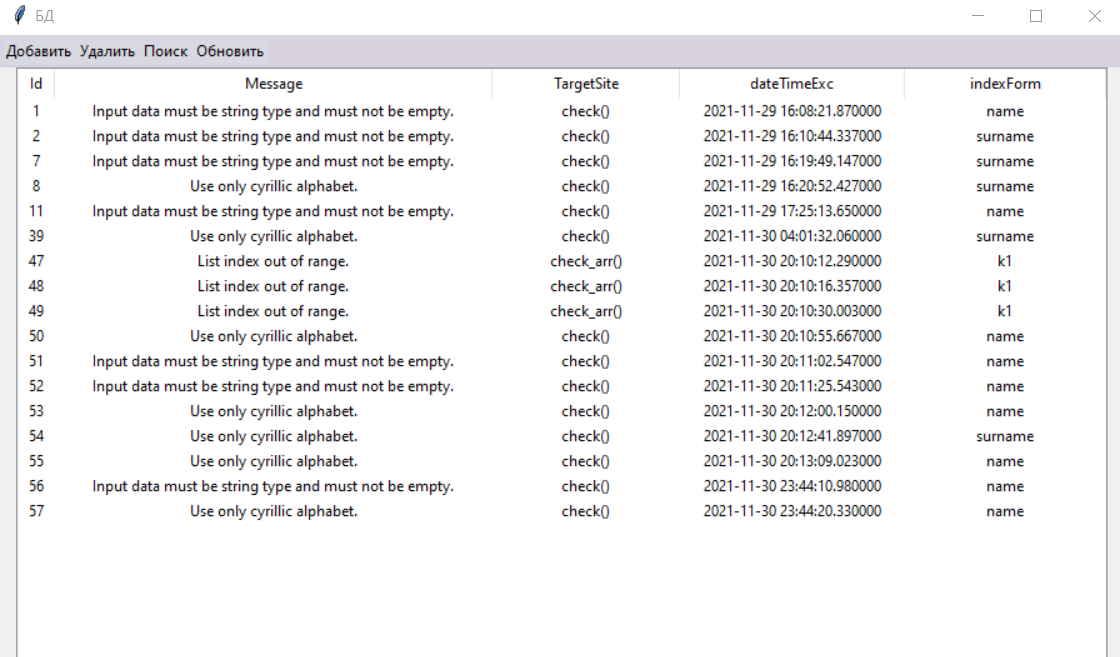
При нажатии на кнопку «Перейти к БД» открывается новая форма, позволяющая выполнять основные операции с БД (Рис.3). В toolbar реализованы кнопки Добавить, Удалить, Поиск, Обновить.

Рис.3 Форма «БД»

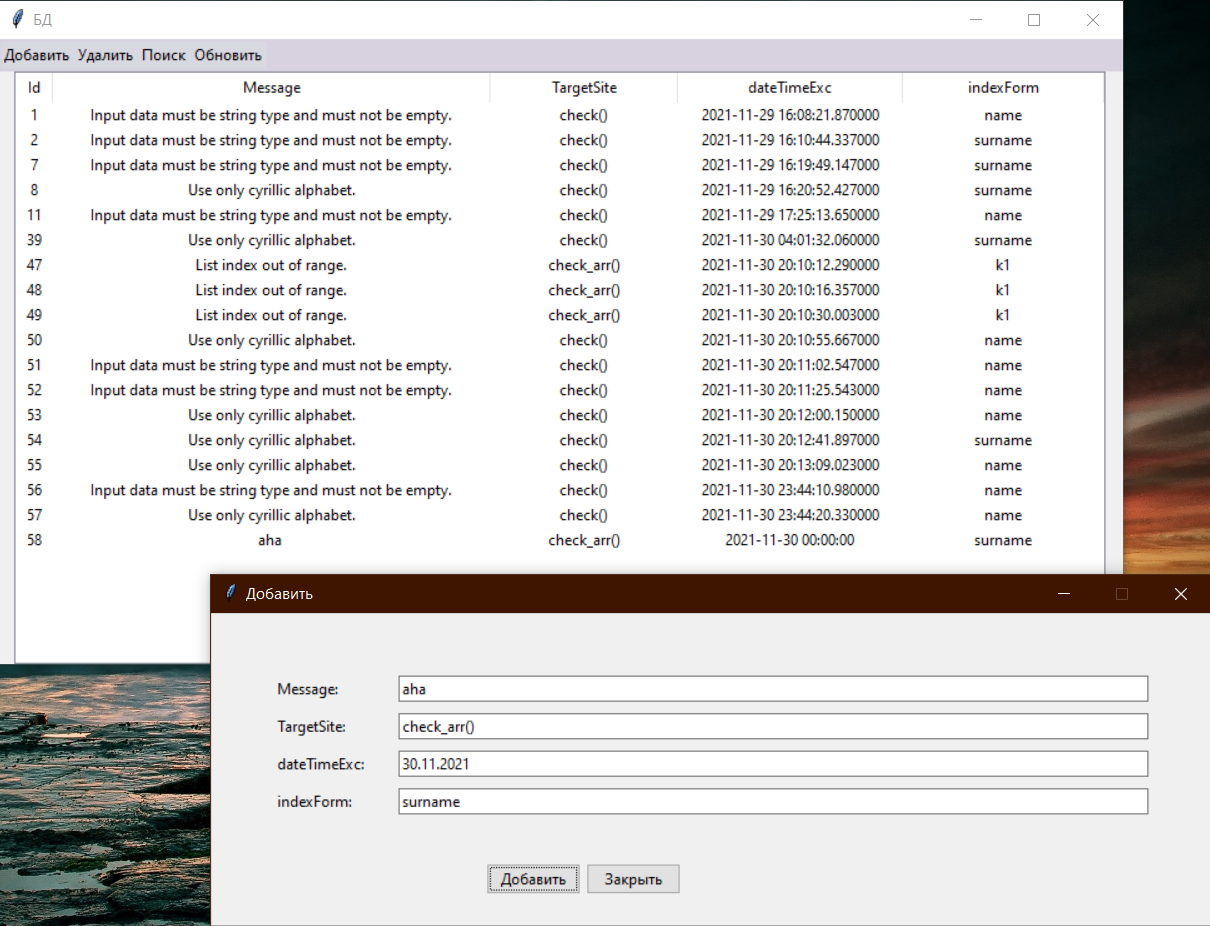
При нажатии на кнопку «Добавить» открывается новая форма «Добавить», куда нужно ввести соответствующие данные для добавления. Рассмотрим результат после нажатия кнопки «Добавить» на форме «Добавить» (Рис.4):

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):

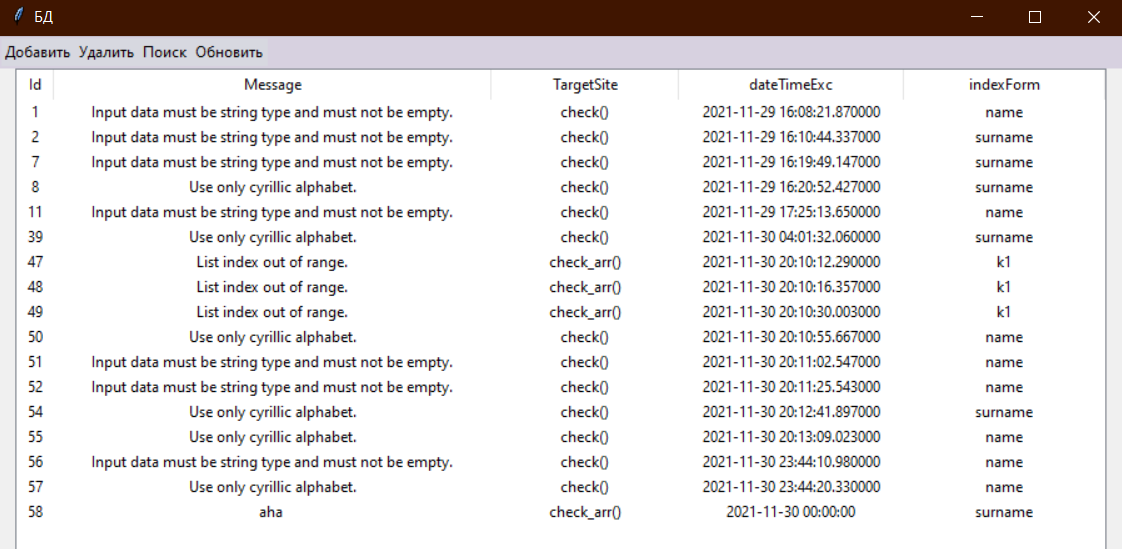
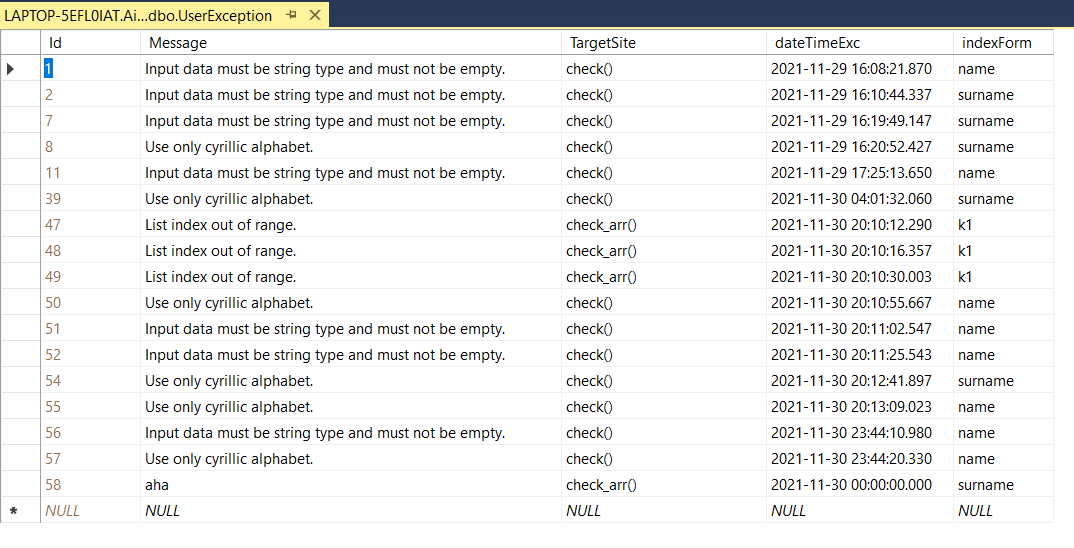


Рис.5 После нажатия кнопки «Удалить»

На рис. 6 продемонстрировано успешное добавление и удаление соответствующих записей в сущности «UserException».

Рис.6 Измененные записи в БД AirLogger в таблице UserException

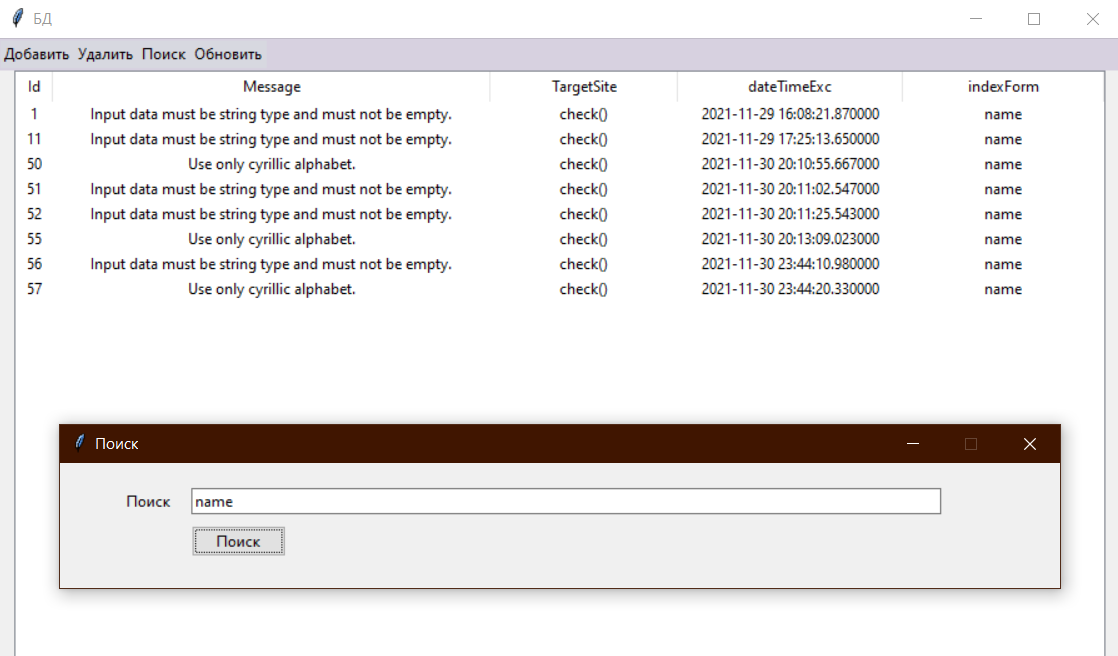
При нажатии на кнопку «Поиск» открывается форма «Поиск», куда нужно ввести данные для поиска. Рассмотрим результат после нажатия кнопки «Поиск» на форме «Поиск (Рис.7):

Рис.7 Результат нажатия кнопки «Поиск»

Для корректного восстановления всех существующих записей в сущности UserException реализовали кнопку «Обновить». Результат нажатия представлен на Рис. 8.

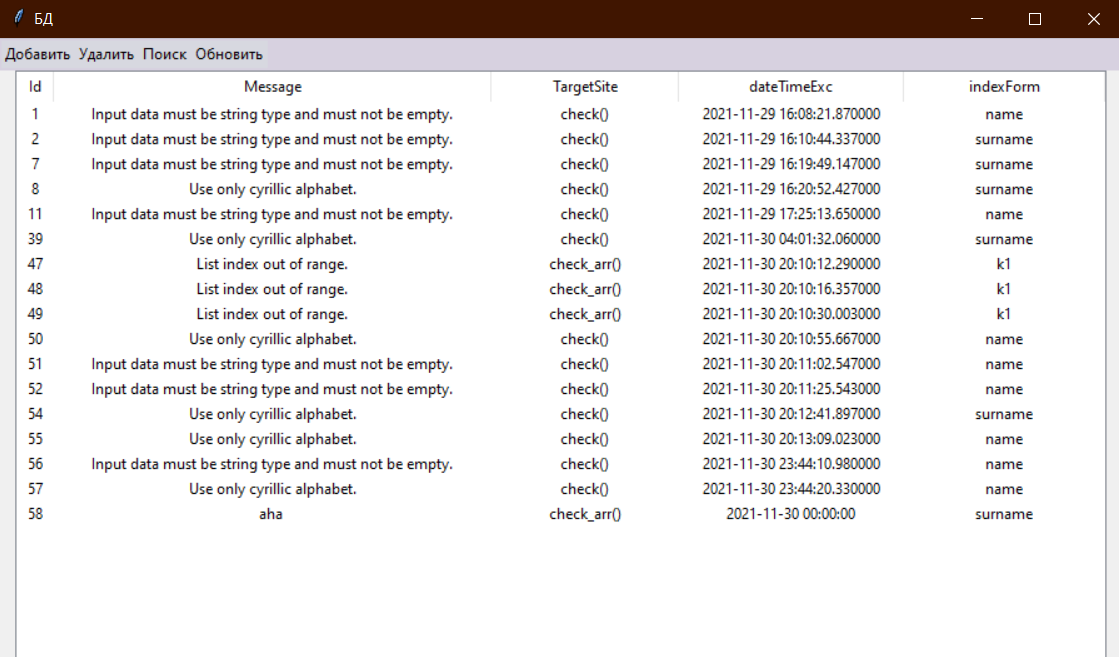


Рис.8 Результат нажатия кнопки «Обновить»

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()

**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.5), поиска (Рис.7), Обновления (Рис.8).

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.

**Листинг программы**

import tkinter as tk  
from tkinter import ttk  
from tkinter.ttk import Checkbutton  
from tkinter import scrolledtext  
import datetime  
from tkinter import \*  
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

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=600)  
  
 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 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()