ГУАП

КАФЕДРА № 33

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | А.А. Бардовский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| РАЗРАБОТКА БАЗЫ ДАННЫХ И ПОДКЛЮЧЕНИЕ DESKTOP-КЛИЕНТА ЛОГГИРОВАНИЯ ИСКЛЮЧЕНИЙ В РАСПРЕДЕЛЕННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЕ |
| по курсу: Технологии и методы программирования |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 3136 |  |  |  | В.А. Панчук |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023 г.

Оглавление

[1. Цель лабораторной работы 3](#_Toc151641279)

[2. Постановка задачи 3](#_Toc151641280)

[3. Основное задание 3](#_Toc151641281)

[3.1. Листинг программы 6](#_Toc151641282)

[3.2. Примеры работы программы 10](#_Toc151641283)

[4. Дополнительное задание 11](#_Toc151641284)

[4.1. Листинг программы 11](#_Toc151641285)

[5. Вывод 12](#_Toc151641286)

# Цель лабораторной работы

Изучить принципы построения приложений, являющихся источниками данных в распределенных информационных системах, научиться использовать методы проектирования приложений доступа к данным, базируясь на принципах Code-First. Освоить механизмы Entity Framework для проектирования desktop-клиентов РИС.

# Постановка задачи

В первой части лабораторной работы нужно было реализовать базу данных в соответствии с ER-диаграммой, представленной в личном кабинете. К созданной базе данных необходимо подключить desktop-клиент, в котором должна быть реализована возможность записи исключений в базу данных.

Вторая часть лабораторной работы включала в себя реализацию глобального уровня логирования всех операций, производимых с базой данных. Должна быть возможность настройки уровня логирования производимых операций.

В форме должен быть реализован CRUD функционал с выводом таблиц базы данных в элемент dataGridView.

Для работы с базой данных была создана отдельная форма с необходимым функционалом.

Форма включает в себя три кнопки, два блока с текстом и элемент dataGridView. Это позволяет реализовать следующие функции:

• Чтение данных из базы данных;

• Запись исключений в базу данных с пользовательским текстом;

• Изменение записей базы данных;

• Удаление записей базы данных;

• Вывод результатов логирования в блок с текстом.

# Основное задание

В соответствии с ER- диаграммой (файл ER\_AirLogger.pdf) с заявленной в методических рекомендациях функциональностью, разработать клиентсерверную 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.

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.

На рисунке 1 и 2 изображены блок схемы алгоритма работы формы.

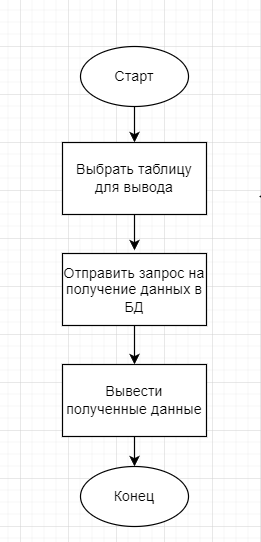
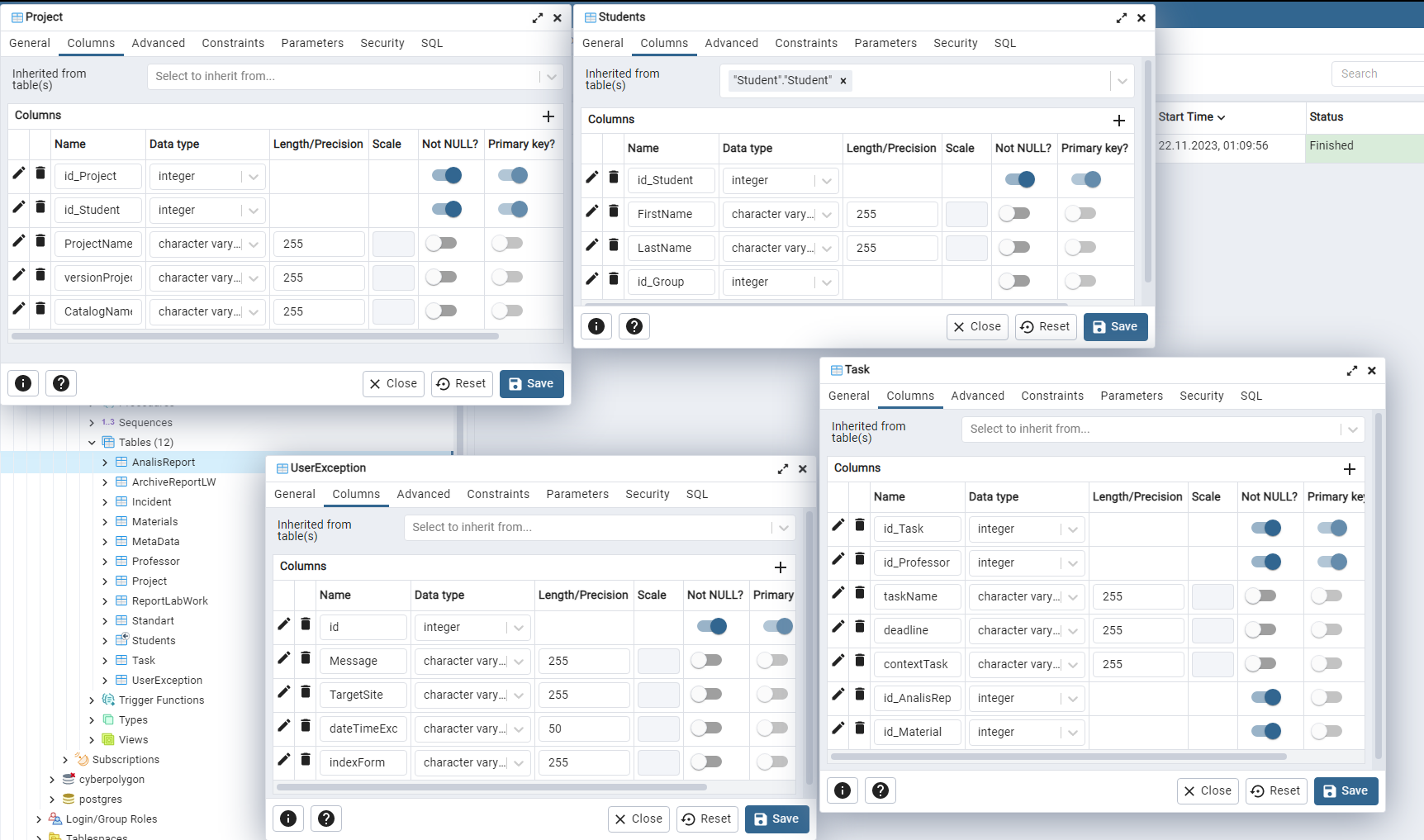


Рисунок − Блок-схема вывода из бд



Рисунок − Блок-схема ввода в бд

Для создания базы данных использовался PostgreSQL. Пример нескольких созданных таблиц из базы данных представлены на рисунке 3.



## Листинг программы

Программа работы с DB:

using Microsoft.EntityFrameworkCore;

namespace AirLoggerPGLib

{

public class DataBase : DbContext

{

//public DbSet<Student> Students { get; set; } = null!;

public DbSet<UserException> UserException { get; set; } = null!;

public DbSet<Students> Students { get; set; } = null!;

public DbSet<Professor> Professor { get; set; } = null!;

public DbSet<Project> Project { get; set; } = null!;

public DataBase(DbContextOptions<DataBase> options)

:base(options)

{

Database.EnsureCreated();

}

public void AddLogInDB(string m, string t, string d, string i)

{

UserException Exception1 = new UserException { Message = m, TargetSite = t, dateTimeExc = d, indexForm = i };

UserException.Add(Exception1);

SaveChanges();

}

}

}

Программа списка таблиц из AirLogger:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AirLoggerPGLib

{

public class UserException

{

[Key]

public int id { get; set; }

public string? Message { get; set; }

public string? TargetSite { get; set; }

public string? indexForm { get; set; }

public string? dateTimeExc { get; set; }

}

public class Students

{

[Key]

public int? id\_Student { get; set; }

public string? FirstName { get; set; }

public string? LastName { get; set; }

public int? id\_Group { get; set; }

}

public class Professor

{

[Key]

public int? id\_Professor { get; set; }

public string? FirstName { get; set; }

public string? LastName { get; set; }

public int? id\_Department { get; set; }

}

public class Project

{

[Key]

public int? id\_Project { get; set; }

public int? id\_Student { get; set; }

public string? ProjectName { get; set; }

public string? versionProject { get; set; }

public string? CatalogName { get; set; }

}

}

Часть кода вывода данных из бд(рисунок 3):

private void button6\_Click(object sender, EventArgs e)

{

try

{

dataGridView1.Rows.Clear();

dataGridView1.Refresh();

using (DataBase db = new DataBase(options))

{

switch (comboBox1.SelectedIndex)

{

case 0:

{

var users = db.UserException.ToList();

foreach (UserException u in users)

{

//richTextBox1.Text += ($"{u.id\_Student}.{u.FirstName} - {u.LastName} - {u.id\_Group}" + "\n\r");

this.dataGridView1.Rows.Add(u.id, u.Message, u.TargetSite, u.dateTimeExc, u.indexForm);

}

break;

}

case 1:

{

var students = db.Students.ToList();

foreach (Students s in students)

{

this.dataGridView1.Rows.Add(s.id\_Student, s.FirstName,

s.LastName, s.id\_Group);

}

}

break;

case 2:

{

var projects = db.Project.ToList();

foreach (Project p in projects)

{

this.dataGridView1.Rows.Add(p.id\_Project, p.id\_Student,

p.ProjectName, p.versionProject, p.CatalogName);

}

break;

}

case 3:

{

var professors = db.Professor.ToList();

foreach (Professor p in professors)

{

this.dataGridView1.Rows.Add(p.id\_Professor, p.FirstName,

p.LastName, p.id\_Department);

}

break;

}

default: { break; }

}

}

}

Ввод данных в бд через пробрасывание исключения(рисунок 4):

catch (DivideByZeroException error)

{

using (DataBase db = new DataBase(options))

{

db.AddLogInDB(Convert.ToString(error.Message), Convert.ToString(error.TargetSite), Convert.ToString(Program.f1.dateLabel) + ' ' + Convert.ToString(Program.f1.timeLabel), GetType().FullName);

}

}

Ввод данных в бд руками(рисунок 5-6):

private void button1\_Click(object sender, EventArgs e)

{

using (DataBase db = new DataBase(options))

{

db.AddLogInDB(textBox2.Text, textBox3.Text, textBox4.Text, textBox5.Text);

}

Dispose();

}

## Примеры работы программы

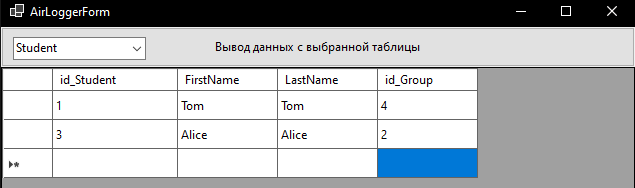


Рисунок 3 – Пример работы вывода данных из БД

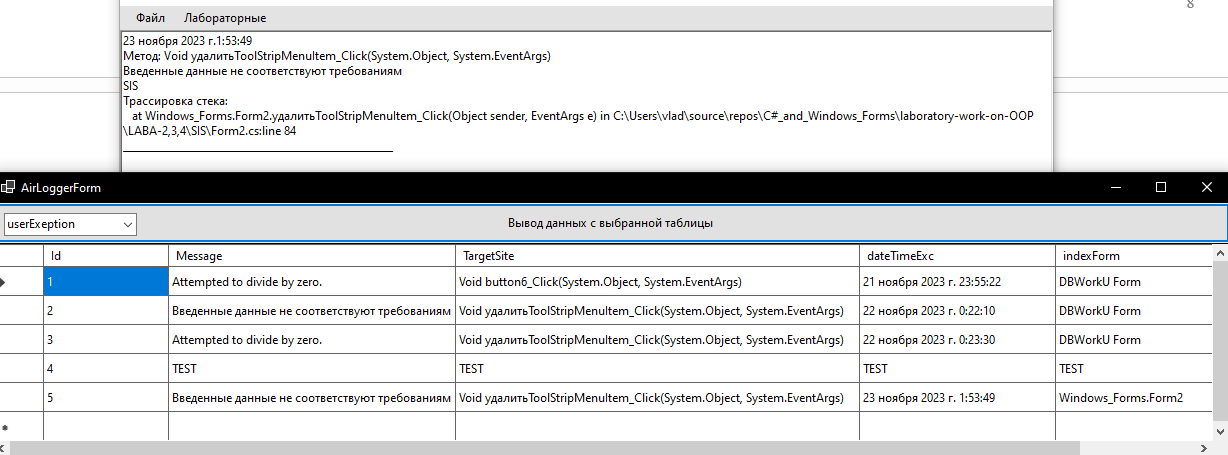


Рисунок – Пример работы ввода данных в БД через прокидывание исключения

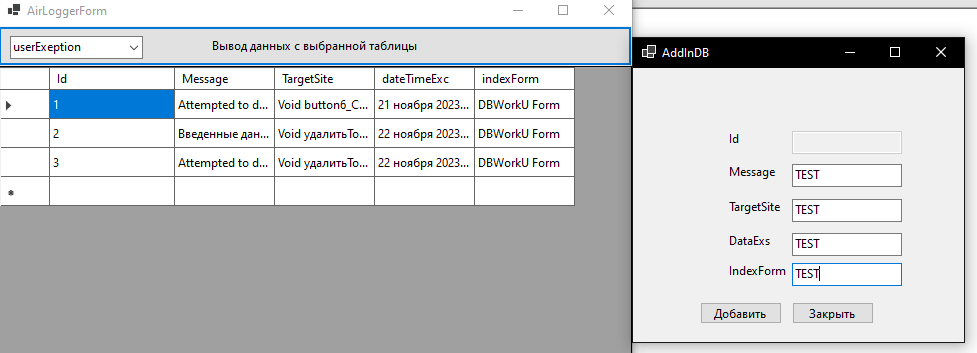


Рисунок − До нажатия кнопки добавления в БД

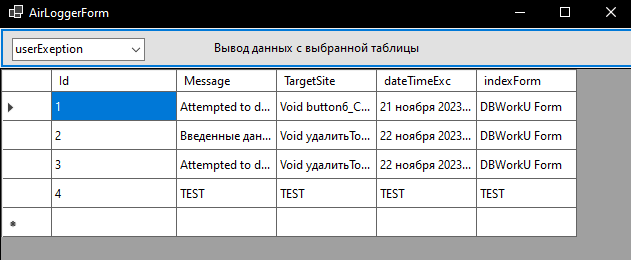


Рисунок − После нажатия кнопки добавления в БД

# Дополнительное задание

Реализовать конфигурацию подключений к базе данных с помощью JSON файла.

## Листинг программы

var builder = new ConfigurationBuilder();

// установка пути к текущему каталогу

builder.SetBasePath(Directory.GetCurrentDirectory());

// получаем конфигурацию из файла

builder.AddJsonFile("airlogger.json");

// создаем конфигурацию

var config = builder.Build();

// получаем строку подключения

string connectionString = config.GetConnectionString("DefaultConnection");

var optionsBuilder = new DbContextOptionsBuilder<DataBase>();

options = optionsBuilder

.UseNpgsql(connectionString)

.Options;

# Вывод

Изучили принципы построения приложений, являющихся источниками данных в распределенных информационных системах, научились использовать методы проектирования приложений доступа к данным, базируясь на принципах Code-First. Освоили механизмы Entity Framework для проектирования desktop-клиентов.