КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп’ютерних наук та кібернетики

Кафедра теорії та технологій програмування інформатики

Звіт

до лабораторної роботи на тему

ФРАГМЕНТАРНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ТАБЛИЧНИМИ БАЗАМИ ДАНИХ

Виконав студент 4-го курсу
Бурлака Владислав Олегович

Київ – 2023

Постановка задачі. Варіант 1, 0

Створити власну версію СУБД.

Основні вимоги щодо структури бази:

- кількість таблиць принципово не обмежена (реляції між таблицями не враховувати);
- кількість полів та кількість записів у кожній таблиці також принципово не обмежені.

У кожній роботі для всіх варіантів для полів у таблицях треба забезпечити підтримку таких типів:

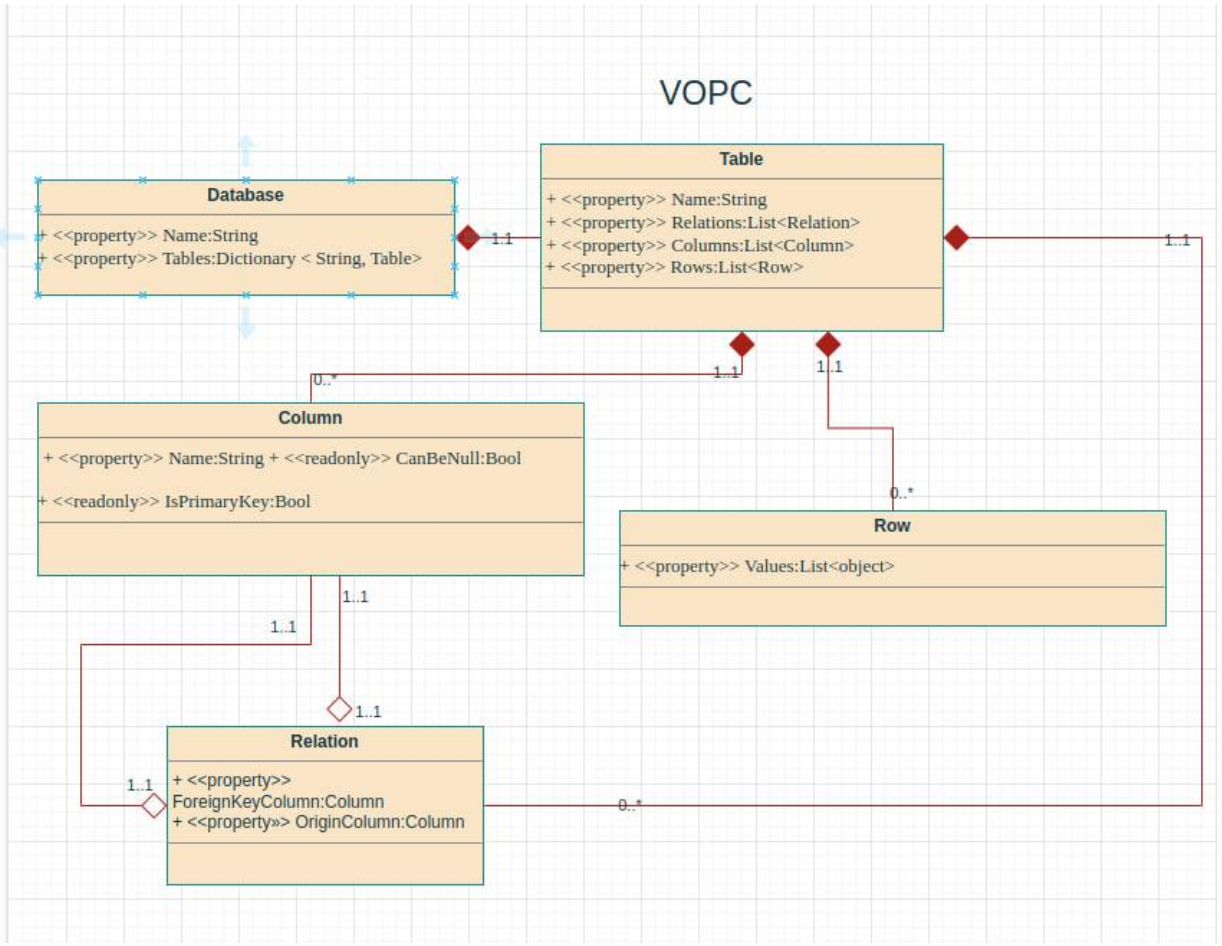
- integer;
- real;
- char;
- string.

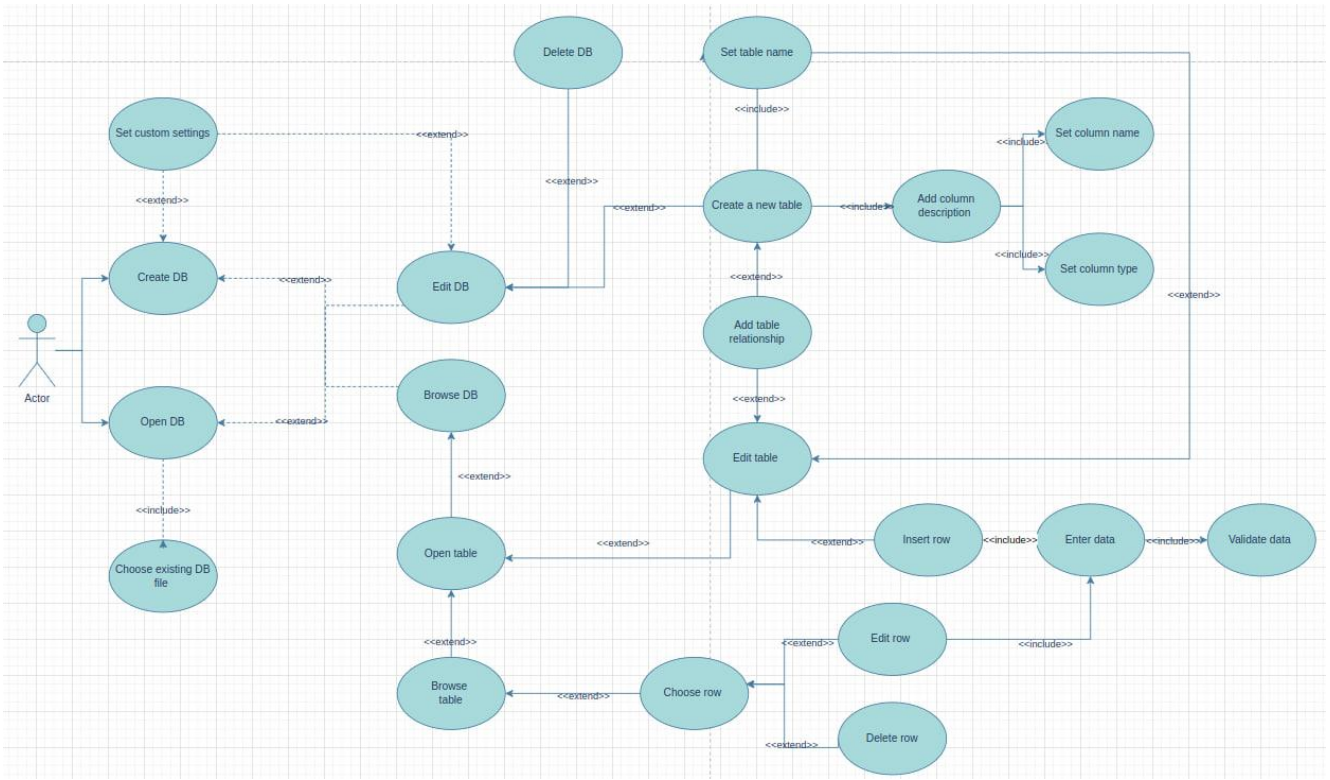
Також у кожній роботі треба реалізувати функціональну підтримку для:

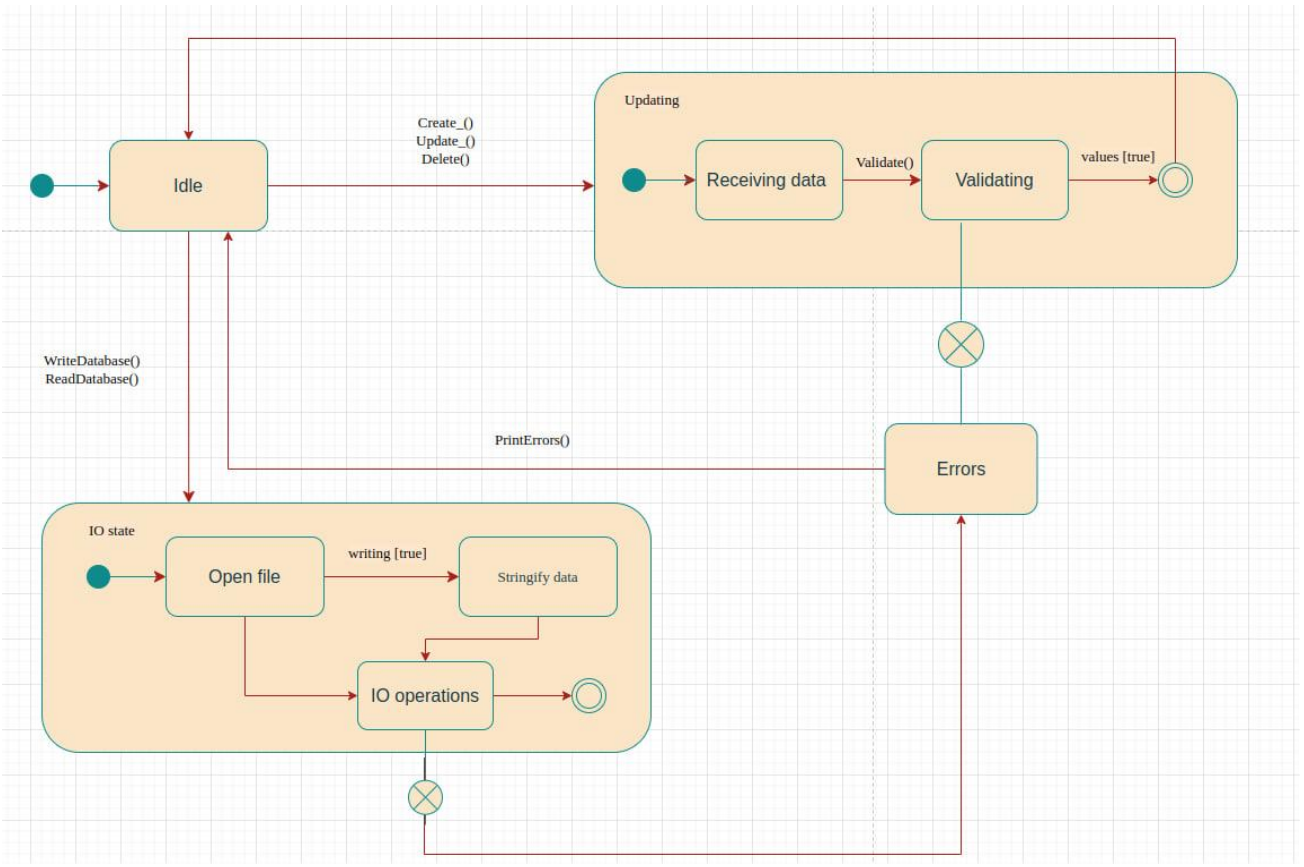
- створення бази;
- створення (із валідацією даних) та знищення таблиці з бази;
- перегляду та редагування рядків таблиці;
- збереження табличної бази на диску та зчитування її з диску.

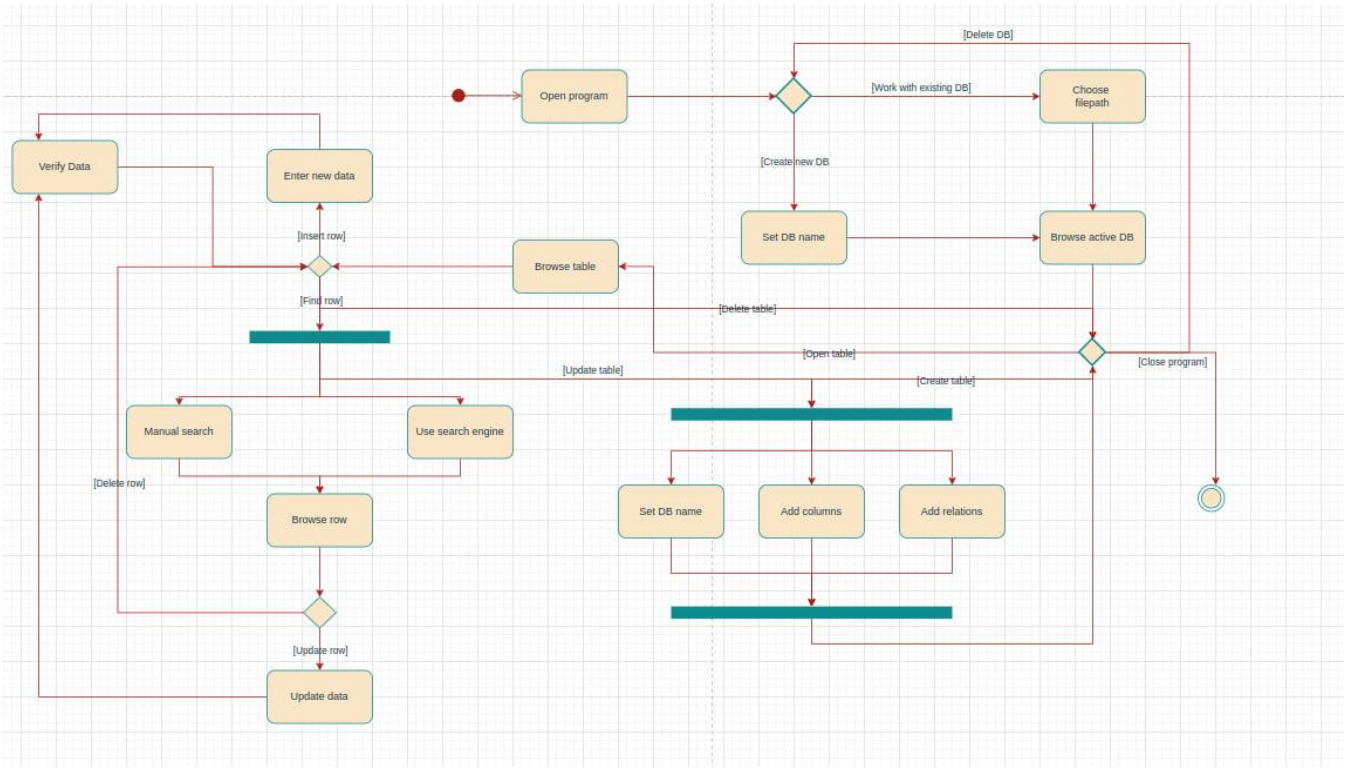
Потрібно забезпечити підтримку (для можливого використання у таблицях) двох додаткових типів: текстові файли та integerInvl; та додаткову функцію: сортування таблиці за одним полем;

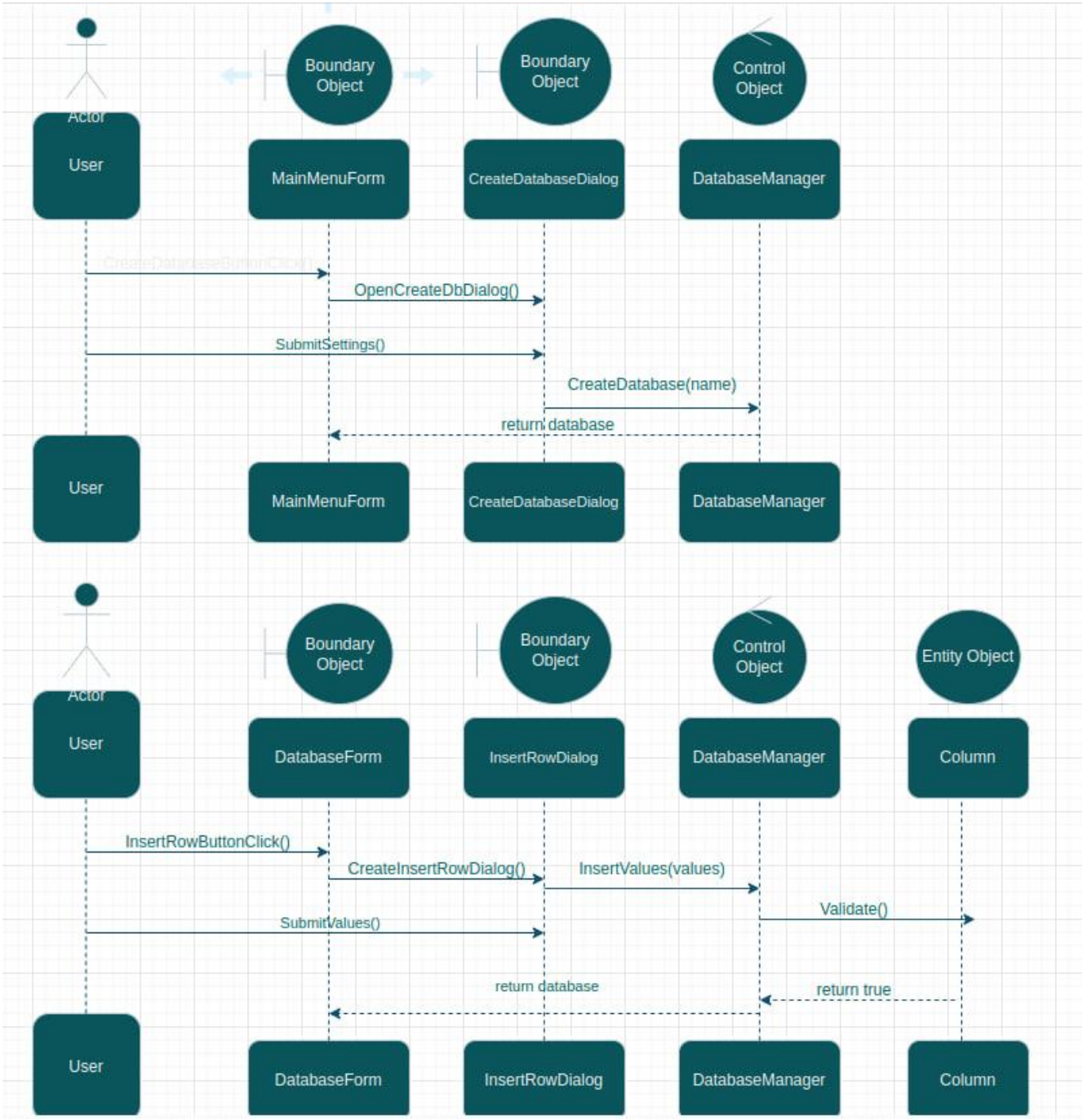
Діаграми до застосунку






Updating
Create_()
Update_()
Delete()
Idle
Receiving data
Validate()
Validating
values [true]
WriteDatabase()
ReadDatabase()
PrintErrors()
Errors
IO state
Open file
writing [true]
Stringify data
IO operations


[Delete DB]
Open program
[Work with existing DB]
Choose filepath
Verify Data
Enter new data
[Create new DB
Set DB name
Browse active DB
[Insert row]
Browse table
[Find row]
[Delete table]
[Open table]
[Close program]
[Update table]
[Create table]
Manual search
Use search engine
[Delete row]
Set DB name
Add columns
Add relations
Browse row
[Update row]
Update data


Actor
User
Boundary Object
MainMenuForm
Boundary Object
CreateDatabaseDialog
Control Object
DatabaseManager
OpenCreateDbDialog()
SubmitSettings()
CreateDatabase(name)
return database
User
MainMenuForm
CreateDatabaseDialog
DatabaseManager
Actor
User
Boundary Object
DatabaseForm
Boundary Object
InsertRowDialog
Control Object
DatabaseManager
Entity Object
Column
InsertRowButtonClick()
CreateInsertRowDialog()
InsertValues(values)
Validate()
SubmitValues()
return database
return true
User
DatabaseForm
InsertRowDialog
DatabaseManager
Column

Основні класи

```
public class Database
{
    3 references
    public string Name { get; set; }
    41 references
    public List<Table> Tables { get; set; } = new List<Table>();
    1 reference
    public Database(string name) => Name = name;

}
```

Клас **Database** – основа СУБД. Має назву та містить в собі список таблиць.

```
public class Table
{
    2 references
    public int NextRowId { get; set; } = 1;
    3 references
    public string Name { get; private set; }
    16 references
    public List<Row> Rows { get; set; } = new List<Row>();
    19 references
    public List<Column> Columns { get; set; } = new List<Column>();
    1 reference
    public Table(string name) => Name = name;
}
```

Клас **Table** – таблиця бази даних. Має назву, айді наступного рядка (тобто кількість вже створених коли-небудь рядків) та містить в собі список рядків і колонок.

```
public abstract class Column
{
    13 references
    public string Name { get; set; }
    7 references
    public abstract DataType Type { get; }
    3 references
    public bool IsNullable { get; set; }
    6 references
    public Column(string name, bool isNull)
    {
        Name = name;
        IsNullable = isNull;
    }
    8 references
    public abstract bool Validate(string value);
}
```

Абстрактний клас **Column** – має назву, тип даних (enum), визначає чи може поле бути null та абстракний метод Validate. Далі від нього наслідуються вже колонки певних типів даних, таких як Int, Real, Char, String, Color, ColorInterval.

```
public class Row
{
    7 references
    public List<string> Values { get; set; } = new List<string>();
    6 references
    public int Id { get; set; }

    1 reference
    public Row(int id) => Id = id;
}
```

Клас **Row**, що має Id, та список значень (індекс значення відповідає індексу колонки).

```
public static class DBController
{
    47 references
    public static Database db { get; private set; }
    1 reference
    public static void CreateDatabase(string name)...
    1 reference
    public static void DeleteDatabase()...
    1 reference
    public static void RenameDatabase(string name)...
    1 reference
    public static void ImportDatabase(string path)...
    1 reference
    public static void ExportDatabase(string path)...
    1 reference
    public static void AddTable(string name)...

    1 reference
    public static void DeleteTable(int table_id)...
    1 reference
    public static void AddColumn(int table_id, string name, string dataType, bool isNull)...

    1 reference
    public static void DeleteColumn(int table_id, string columnName)...
    1 reference
    public static void AddRow(int table_id, int rowId, Dictionary<int, string> values)...
    1 reference
    public static void DeleteRow(int table_id, int row_id)...
    1 reference
    public static void EditRow(int table_id, int row_id, Dictionary<int, string> values)...
}
```

Статичний клас **DBController**, що відповідає за роботу з БД та містить всі основні методи.

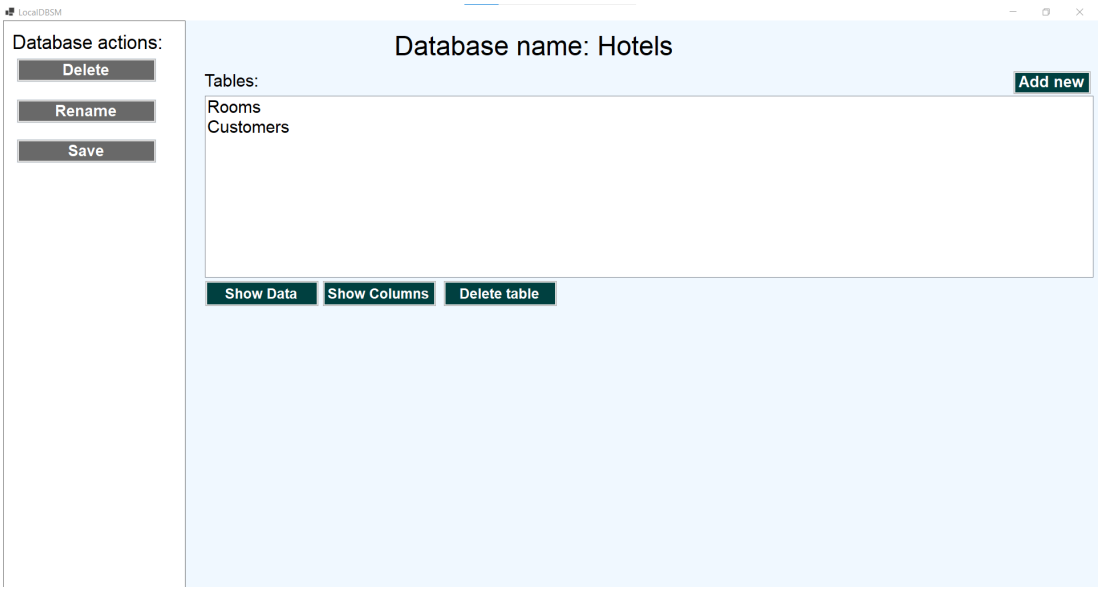
Інтерфейс користувача (GUI)

Для створення графічного інтерфейсу користувача було використано Windows Forms.

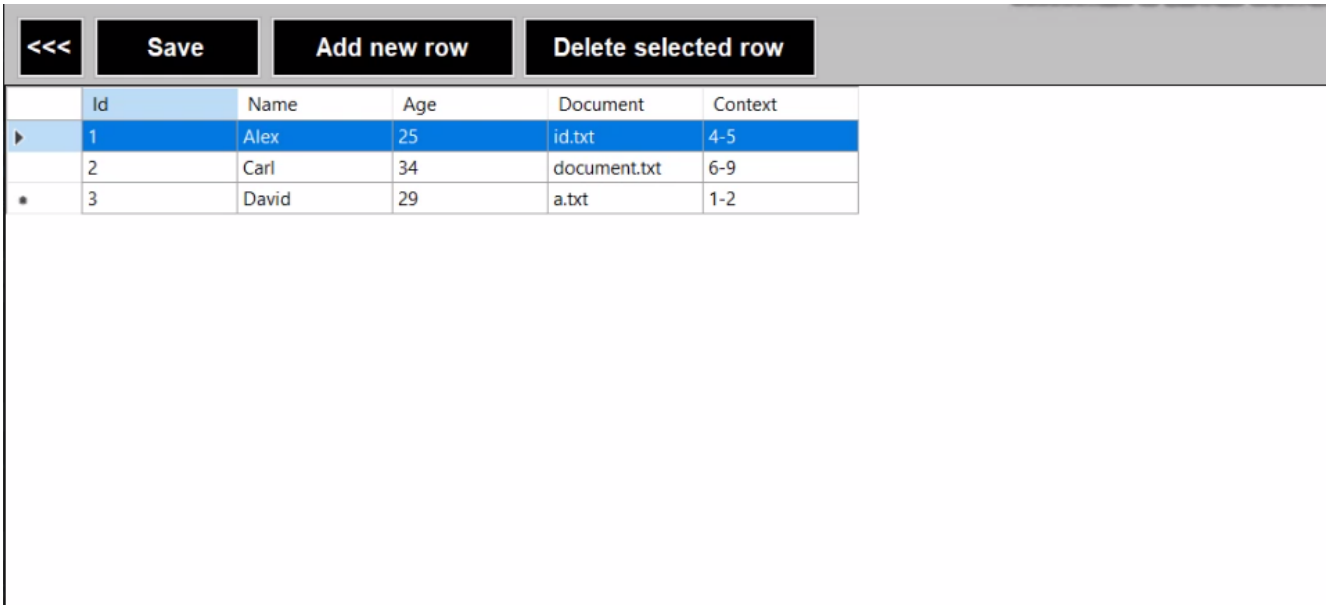
Розглянемо основні сторінки:

- Початкова сторінка. Тут користувач має на вибір дві опції: створити нову БД або завантажити вже існуючу у форматі файлу json.

- Сторінка інформації про БД. Тут користувач може зберегти БД, перейменувати її, видалити. Також переглянути таблиці та взаємодіяти з ними: створити нову, видалити, переглянути вміст та редагувати.



- Перегляд колонок таблиці: тут їх можна переглянути та відредагувати.



Тестування

Тести демонструють, чи правильно проводиться валідація вводу даних.


```cs
using LocalDBSM.Models;

namespace LocalTests
{
    public class ColumnsTests
    {
        private Column column;

        [Fact]
        public void TestColorColumnValidation()
        {
            //Arrange
            column = new TextFileColumn("Document", false);

            //Act
            bool result = column.Validate("id.txt");

            //Assert
            Assert.True(result);
        }

        [Fact]
        public void TestRealColumnValidation()
        {
            //Arrange
            column = new RealColumn("Price", false);

            //Act
            bool result = column.Validate("aboba");

            //Assert
            Assert.False(result);
        }

        [Fact]
        public void TestIntegerIntervalColumnValidation()
        {
            //Arrange
            column = new IntegerInvlColumn("Age", false);

            //Act
            bool result = column.Validate("10-12");

            //Assert
            Assert.True(result);
        }

        [Fact]
        public void TestIntegerColumnValidation()
        {
            //Arrange
            column = new IntColumn("Weight", true);

            //Act
            bool result = column.Validate(null);

            //Assert
            Assert.False(result);
        }
    }
}
```

Результати виконання тестів: