МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 6

з дисципліни “Основи програмування”
тема “Подійно-орієнтований інтерфейс користувача”

Виконав
студент І курсу
групи КП-01
Пецеля Артем Володимирович
(*прізвище, ім’я, по батькові*)

Перевірив
“____” “_____” 20___ р.
викладач
Гадиняк Руслан Анатолійович
(*прізвище, ім’я, по батькові*)

варіант №12

Київ 2021

Мета роботи

Вивчити основні принципи подійно-орієнтованого підходу при побудові інтерфейсу користувача.

Реалізувати подійно-орієнтований інтерфейс користувача для виконання базових операцій над даними.

Постановка завдання

Реалізувати подійно-орієнтований інтерфейс користувача для керування сутностями (див. Додаток А) із бази даних, що дозволяє:

- Створити нову сутність.
- Переглянути пагінований список всіх сутностей.
- ~~Відфільтрувати список сутностей за пошуковим рядком символів~~
- Переглянути детальну інформацію про обрану сутність.
- Редагувати дані обраної сутності.
- Видалити обрану сутність.

Стани інтерфейсу користувача поділити на вікна:

- Головне вікно з пагінованим списком всіх сутностей
- Вікно створення нової сутності
- Вікно перегляду детальної інформації про обрану сутність
- Вікно редагування обраної сутності

Головне вікно програми має містити:

- кнопку для створення нової сутності

- відкриває вікно із формою вводу даних нової сутності. У форму можна ввести всі дані, крім тих, що генеруються автоматично (ідентифікатор, дата і час створення запису).
- після підтвердження створення переключатись на вікно перегляду створеної сутності

- пагінований список сутностей
  - виводити у списку коротку інформацію про кожну сутність, наприклад, назву і якесь інше поле (на вибір).
  - натискання на обрану сутність переключає програму на вікно перегляду сутності
- елементами інформації про сторінки та кнопками навігації по сторінках (наступна сторінка, попередня сторінка).
  - при перегляді списку виводити у інтерфейсу загальну кількість сторінок і номер поточної сторінки (нумерація з 1).
  - якщо у БД немає записів - виводити замість списку мітку з текстом про відсутність записів.
  - кнопки навігації по сторінках робити неактивними, якщо їх дію виконати неможливо (наприклад, на першій чи останній сторінці)
- ~~поле вводу рядка символів пошуку сутностей і кнопка для активації пошуку~~
- головне меню з пунктами:
  - _File
    - _New… - створення нового запису
    - _Quit - вихід з програм
  - _Help
    - _About - показати діалогове вікно з інформацією про програму і її автора.

Вікно перегляду сутності містить:

- дані обраної сутності:
  - виводити опис полів і їх значення
  - дату і час виводити не в ISO 8601, а залежно від локалізації
- кнопку повернення до головного вікна
- кнопку редагування відкриває вікно редагування обраної сутності.
  - Ідентифікатор і дату-час створення запису змінювати не можна.
  - Після підтвердження змін переключитись на вікно перегляду обраної сутності
- кнопку видалення сутності відкриває діалог підтвердження видалення (підтвердити видалення і видалити або відмінити спробу видалення).
  - Після підтвердження видалення переключитись на головне вікно

Вимоги до структури коду:

- модуль інтерфейсу користувача описати у проекті консольної програми
- модуль доступу до даних (клас репозиторія сутності за варіантом) та клас сутності винести у проект бібліотеки класів, яку підключити до проекту консольної програми.

Текст коду програми

Program.cs

```
using ClassLibrary;
using Terminal.Gui;

namespace lab6
{
    class Program
    {
        static void Main(string[] args)
        {
            string database = @"./../../../data.db";
            Application.Init();
            GoodRepository repo = new GoodRepository(database);
            Window win = new MainWindow(repo);
            Application.Top.Add(win);
            Application.Run();
        }
    }
}
```

MainWindow.cs

```
using ClassLibrary;
using System.Collections.Generic;
using Terminal.Gui;

namespace lab6
{
    class MainWindow : Window
    {
        GoodRepository repository;
        int totalPages;
        int currentPage;

        ListView list;
        Button prevPage;
        TextField bottomPageCounter;
        Button nextPage;
        public MainWindow(GoodRepository repository)
        {
            this.repository = repository;
            this.totalPages = repository.GetTotalPages();
            this.currentPage = 1;
            Y = Pos.Percent(0) + 1;
            Initialize();
        }
        private void Initialize()
        {
            MenuBar menu = new MenuBar(new MenuBarItem[]
            {
                new MenuBarItem ("_File", new MenuItem[]
                {
                    new MenuItem ("_New...", "Add new good", OnNewGoodClicked),
                    new MenuItem ("_Quit", "Help text", OnQuit)
                }),
                new MenuBarItem ("_Help", new MenuItem[]
                {
                    new MenuItem ("_About", "Information", OnAboutClicked)
                })
            });

            Application.Top.Add(menu);
```

```
            Button newGood = new Button("Add new good")
            {
                X = Pos.Center(),
                Y = Pos.Percent(10),
            };
            list = new ListView()
            {
                X = Pos.Percent(10),
                Y = Pos.Center(),
                Width = Dim.Percent(80),
                Height = 10
            };
            prevPage = new Button("<-")
            {
                X = Pos.Right(list) - 20,
                Y = Pos.Bottom(list) + 2,
            };
            bottomPageCounter = new TextField()
            {
                X = Pos.Right(prevPage) + 1,
                Y = Pos.Top(prevPage),
                Width = 3,
                Height = Dim.Height(prevPage)
            };
            Label bottomAllPage = new Label($"/{totalPages}")
            {
                X = Pos.Right(bottomPageCounter) + 1,
                Y = Pos.Top(bottomPageCounter),
                Height = Dim.Height(bottomPageCounter)
            };
            nextPage = new Button("->")
            {
                X = Pos.Right(bottomAllPage) + 1,
                Y = Pos.Top(bottomAllPage),
            };

            Label notFoundLabel = new Label("Goods not found")
            {
                X = Pos.Center(),
                Y = Pos.Center()
            };

            UpdateInfo();

            newGood.Clicked += OnNewGoodClicked;
            list.OpenSelectedItem += OnGoodClicked;

            prevPage.Clicked += OnPrevPageClicked;
            nextPage.Clicked += OnNextPageClicked;
            bottomPageCounter.KeyDown += OnCounterPressed;

            if (totalPages != 0)
            {
                this.Add(newGood, list, prevPage, bottomPageCounter, bottomAllPage, nextPage);
            }
            else
            {
                this.Add(newGood, notFoundLabel);
            }
        }

        private void OnGoodClicked(ListViewItemEventArgs obj)
        {
            Good good = (Good)obj.Value;
            Window viewWindow = new GoodViewWindow(good.id, repository);
            Application.Top.RemoveAll();
            Application.Top.Add(viewWindow);
            Application.RequestStop();
            Application.Run();
        }

        private void OnNewGoodClicked()
```

```
        {
            Window newGood = new GoodCreationWindow(repository);
            Application.Top.RemoveAll();
            Application.Top.Add(newGood);
            Application.RequestStop();
            Application.Run();
        }

        private void OnCounterPressed(KeyEventEventArgs obj)
        {
            if (obj.KeyEvent.Key == Key.Enter)
            {
                if (!int.TryParse(bottomPageCounter.Text.ToString(), out int number))
                {
                    bottomPageCounter.Text = string.Empty;
                    return;
                }
                if (number > totalPages || number < 1)
                {
                    bottomPageCounter.Text = string.Empty;
                    return;
                }
                currentPage = number;
                UpdateInfo();
            }
        }

        private void OnNextPageClicked()
        {
            currentPage++;
            UpdateInfo();
        }
        private void OnPrevPageClicked()
        {
            currentPage--;
            UpdateInfo();
        }

        private void UpdateInfo()
        {
            if (totalPages == 0)
            {
                bottomPageCounter.Text = "0";
                prevPage.Visible = false;
                nextPage.Visible = false;
                return;
            }

            prevPage.Visible = currentPage != 1;
            nextPage.Visible = currentPage != totalPages;
            bottomPageCounter.Text = $"{currentPage}";
            Application.Refresh();

            List<Good> source = repository.GetPage(currentPage);
            list.SetSource(source);
        }
        private void OnQuit()
        {
            Application.RequestStop();
        }
        private void OnAboutClicked()
        {
            MessageBox.Query("Info", "Lab6 by Artem Petselia", "Ok");
        }
    }
}
```

GoodViewWindow.cs

```
using ClassLibrary;
using Terminal.Gui;

namespace lab6
{
    class GoodViewWindow : Window
    {
        Good good;
        GoodRepository repository;

        TextField nameField;
        TextView descriptionField;
        TextField priceField;
        TextField creationDateField;
        Label availableLabel;
        public GoodViewWindow(long goodId, GoodRepository repository)
        {
            this.repository = repository;
            this.good = repository.GetById(goodId);
            Title = "Good";
            Initialize();
        }
        private void Initialize()
        {
            Label nameLabel = new Label("Name: ")
            {
                X = Pos.Percent(10),
                Y = Pos.Percent(10),
            };
            nameField = new TextField()
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(nameLabel) + 1,
                Width = Dim.Percent(80),
                ReadOnly = true
            };
            Label descriptionLabel = new Label("Description: ")
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(nameField) + 1,
            };
            Window inputWindow = new Window()
            {
                X = Pos.Left(descriptionLabel),
                Y = Pos.Bottom(descriptionLabel) + 1,
                Width = Dim.Percent(80),
                Height = Dim.Fill(17)
            };
            descriptionField = new TextView()
            {
                Width = Dim.Fill(),
                Height = Dim.Fill(),
                ReadOnly = true,
            };
            inputWindow.Add(descriptionField);
            Label priceLabel = new Label("Price: ")
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(inputWindow) + 1,
            };
            priceField = new TextField()
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(priceLabel) + 1,
                Width = Dim.Percent(80) - 4,
                ReadOnly = true
            };
            Label uahLabel = new Label("UAH")
            {
                X = Pos.Right(priceField) + 1,
```

```
            Y = Pos.Top(priceField)
        };
        Label creationDateLabel = new Label("Created at:")
        {
            X = Pos.Left(priceField),
            Y = Pos.Bottom(priceField) + 1,
        };
        creationDateField = new TextField()
        {
            X = Pos.Left(creationDateLabel),
            Y = Pos.Bottom(creationDateLabel) + 1,
            Width = 19,
            ReadOnly = true
        };
        availableLabel = new Label("1")
        {
            X = Pos.Left(creationDateField),
            Y = Pos.Bottom(creationDateField) + 1,
            Width = 30,
        };
        Button toMainWindow = new Button("To main window")
        {
            X = Pos.Center() - 20,
            Y = Pos.Bottom(availableLabel) + 2
        };
        Button edit = new Button("Edit")
        {
            X = Pos.Right(toMainWindow) + 3,
            Y = Pos.Top(toMainWindow)
        };
        Button delete = new Button("Delete")
        {
            X = Pos.Right(edit) + 3,
            Y = Pos.Top(edit)
        };
        UpdateInfo();

        toMainWindow.Clicked += OnToMainWindowClicked;
        edit.Clicked += OnEditClicked;
        delete.Clicked += OnDeleteClicked;

        this.Add(nameLabel, nameField,
            descriptionLabel, inputWindow,
            priceLabel, priceField, uahLabel,
            creationDateLabel, creationDateField,
            availableLabel,
            toMainWindow, edit, delete);
    }
    private void UpdateInfo()
    {
        nameField.Text = good.name;
        descriptionField.Text = good.description;
        priceField.Text = $"{good.price}";
        creationDateField.Text = good.createdAt.ToString();
        availableLabel.Text = $"Available: {good.isAvailable}";
    }
    private void OnDeleteClicked()
    {
        int result = MessageBox.Query("Info", "Are you sure, that you want to delete good?", "Yes", "No");
        if (result == 0)
        {
            repository.Delete(good.id);
            OnToMainWindowClicked();
        }
    }

    private void OnEditClicked()
    {
        Window editWindow = new GoodEditWindow(good.id, repository);
        Application.Run(editWindow);
        good = repository.GetById(good.id);
        UpdateInfo();
```

```
        }

        private void OnToMainWindowClicked()
        {
            Application.Top.RemoveAll();
            Window main = new MainWindow(repository);
            Application.Top.Add(main);
            Application.RequestStop();
            Application.Run();
        }
    }
}
```

GoodEditWindow.cs

```
using ClassLibrary;
using Terminal.Gui;

namespace lab6
{
    class GoodEditWindow : Window
    {
        GoodRepository repository;
        Good good;

        TextField nameField;
        TextView descriptionField;
        TextField priceField;
        CheckBox availableCheckBox;
        public GoodEditWindow(long goodId, GoodRepository repository)
        {
            this.repository = repository;
            this.good = repository.GetById(goodId);
            Title = "Good edit";
            Initialize();
        }
        private void Initialize()
        {
            Label nameLabel = new Label("Name: ")
            {
                X = Pos.Percent(10),
                Y = Pos.Percent(10),
            };
            nameField = new TextField(good.name)
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(nameLabel) + 1,
                Width = Dim.Percent(80)
            };
            Label descriptionLabel = new Label("Description: ")
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(nameField) + 1,
            };
            Window inputWindow = new Window()
            {
                X = Pos.Left(descriptionLabel),
                Y = Pos.Bottom(descriptionLabel) + 1,
                Width = Dim.Percent(80),
                Height = Dim.Fill(10)
            };
            descriptionField = new TextView()
            {
                Width = Dim.Fill(),
                Height = Dim.Fill(),
                Text = good.description
            };
            inputWindow.Add(descriptionField);
            Label priceLabel = new Label("Price: ")
```

```
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(inputWindow) + 1,
            };
            priceField = new TextField($"{good.price}")
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(priceLabel) + 1,
                Width = Dim.Percent(80) - 4
            };
            Label uahLabel = new Label("UAH")
            {
                X = Pos.Right(priceField) + 1,
                Y = Pos.Top(priceField)
            };
            availableCheckBox = new CheckBox("Available")
            {
                X = Pos.Left(priceField),
                Y = Pos.Bottom(priceField) + 1,
                Checked = good.isAvailable
            };
            Button confirm = new Button("Confirm")
            {
                X = Pos.Center() - 10,
                Y = Pos.Bottom(availableCheckBox) + 2
            };
            Button cancel = new Button("Cancel")
            {
                X = Pos.Right(confirm) + 3,
                Y = Pos.Top(confirm),
            };

            confirm.Clicked += OnConfirmClicked;
            cancel.Clicked += OnCancelClicked;

            this.Add(nameLabel, nameField,
                descriptionLabel, inputWindow,
                priceLabel, priceField, uahLabel,
                availableCheckBox,
                confirm, cancel);
        }
        private void OnConfirmClicked()
        {
            string name = nameField.Text.ToString();
            string description = descriptionField.Text.ToString();
            string priceString = priceField.Text.ToString();

            if (!double.TryParse(priceString, out double price))
            {
                MessageBox.ErrorQuery("Error", "Price should be number", "Ok");
                return;
            }
            if (name == "")
            {
                MessageBox.ErrorQuery("Error", "Name should be provided", "Ok");
                return;
            }
            if (description == "")
            {
                MessageBox.ErrorQuery("Error", "Description should be provided", "Ok");
                return;
            }

            good.name = name;
            good.description = description;
            good.price = price;
            good.isAvailable = availableCheckBox.Checked;

            repository.Edit(good);
            Application.RequestStop();
        }
        private void OnCancelClicked()
```

```
        {
            Application.RequestStop();
        }
    }
}
```

GoodCreationWindow.cs

```
using System;
using ClassLibrary;
using Terminal.Gui;

namespace lab6
{
    class GoodCreationWindow : Window
    {
        GoodRepository repository;

        TextField nameField;
        TextView descriptionField;
        TextField priceField;
        CheckBox availableCheckBox;
        public GoodCreationWindow(GoodRepository repository)
        {
            this.repository = repository;

            Title = "New good";

            Initialize();
        }
        private void Initialize()
        {
            Label nameLabel = new Label("Name: ")
            {
                X = Pos.Percent(10),
                Y = Pos.Percent(10),
            };
            nameField = new TextField()
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(nameLabel) + 1,
                Width = Dim.Percent(80)
            };
            Label descriptionLabel = new Label("Description: ")
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(nameField) + 1,
            };
            Window inputWindow = new Window()
            {
                X = Pos.Left(descriptionLabel),
                Y = Pos.Bottom(descriptionLabel) + 1,
                Width = Dim.Percent(80),
                Height = Dim.Fill(10)
            };
            descriptionField = new TextView()
            {
                Width = Dim.Fill(),
                Height = Dim.Fill()
            };
            inputWindow.Add(descriptionField);
            Label priceLabel = new Label("Price: ")
            {
                X = Pos.Left(nameLabel),
                Y = Pos.Bottom(inputWindow) + 1,
            };
            priceField = new TextField()
            {
                X = Pos.Left(nameLabel),
```

```
            Y = Pos.Bottom(priceLabel) + 1,
            Width = Dim.Percent(80) - 4
        };
        Label uahLabel = new Label("UAH")
        {
            X = Pos.Right(priceField) + 1,
            Y = Pos.Top(priceField)
        };
        availableCheckBox = new CheckBox("Available")
        {
            X = Pos.Left(priceField),
            Y = Pos.Bottom(priceField) + 1,
        };
        Button confirm = new Button("Confirm")
        {
            X = Pos.Center() - 10,
            Y = Pos.Bottom(availableCheckBox) + 2
        };
        Button cancel = new Button("Cancel")
        {
            X = Pos.Right(confirm) + 3,
            Y = Pos.Top(confirm),
        };

        confirm.Clicked += OnConfirmClicked;
        cancel.Clicked += OnCancelClicked;

        this.Add(nameLabel, nameField,
            descriptionLabel, inputWindow,
            priceLabel, priceField, uahLabel,
            availableCheckBox,
            confirm, cancel);
    }

    private void OnCancelClicked()
    {
        Application.Top.RemoveAll();
        Window main = new MainWindow(repository);
        Application.Top.Add(main);
        Application.RequestStop();
        Application.Run();
    }

    private void OnConfirmClicked()
    {
        string name = nameField.Text.ToString();
        string description = descriptionField.Text.ToString();
        string priceString = priceField.Text.ToString();

        if (!double.TryParse(priceString, out double price))
        {
            MessageBox.ErrorQuery("Error", "Price should be number", "Ok");
            return;
        }
        if (name == "")
        {
            MessageBox.ErrorQuery("Error", "Name should be provided", "Ok");
            return;
        }
        if (description == "")
        {
            MessageBox.ErrorQuery("Error", "Description should be provided", "Ok");
            return;
        }

        Good good = new Good()
        {
            name = name,
            description = description,
            price = price,
            isAvailable = availableCheckBox.Checked,
            createdAt = DateTime.Now
```

```
            };

            good.id = repository.Insert(good);

            Application.Top.RemoveAll();
            Window viewWindow = new GoodViewWindow(good.id, repository);
            Application.Top.Add(viewWindow);
            Application.RequestStop();
            Application.Run();
        }
    }
}
```

GoodRepository.cs

```
using System;
using Microsoft.Data.Sqlite;
using System.Collections.Generic;

namespace ClassLibrary
{
    public class GoodRepository
    {
        private SqliteConnection connection;
        public GoodRepository(string databaseFile)
        {
            this.connection = new SqliteConnection($"Data Source = {databaseFile}");
        }
        public long Insert(Good good)
        {
            connection.Open();
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"
                INSERT INTO goods (name, description, price, is_available, created_at)
                VALUES ($name, $description, $price, $is_available, $created_at);

                SELECT last_insert_rowid();
            ";
            command.Parameters.AddWithValue("$name", good.name);
            command.Parameters.AddWithValue("$description", good.description);
            command.Parameters.AddWithValue("$price", good.price);
            command.Parameters.AddWithValue("$is_available", good.isAvailable);
            command.Parameters.AddWithValue("$created_at", good.createdAt.ToString("o"));

            long newId = (long)command.ExecuteScalar();
            connection.Close();
            return newId;
        }
        public void Delete(long id)
        {
            connection.Open();
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"DELETE FROM goods WHERE id = $id";
            command.Parameters.AddWithValue("$id", id);
            command.ExecuteScalar();
            connection.Close();
        }
        public void Edit(Good good)
        {
            connection.Open();
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"
                UPDATE goods
                SET name = $name, description = $description, price = $price, is_available = $is_available, created_at = $create
d_at
                WHERE id = $id
            ";
            command.Parameters.AddWithValue("$id", good.id);
```

```
            command.Parameters.AddWithValue("$name", good.name);
            command.Parameters.AddWithValue("$description", good.description);
            command.Parameters.AddWithValue("$price", good.price);
            command.Parameters.AddWithValue("$is_available", good.isAvailable);
            command.Parameters.AddWithValue("$created_at", good.createdAt.ToString("o"));

            command.ExecuteNonQuery();
            connection.Close();
        }
        public Good GetById(long id)
        {
            connection.Open();
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT * FROM goods WHERE id = $id";
            command.Parameters.AddWithValue("$id", id);

            SqliteDataReader reader = command.ExecuteReader();

            if (reader.Read())
            {
                Good good = new Good()
                {
                    id = id,
                    name = reader.GetString(1),
                    description = reader.GetString(2),
                    price = reader.GetDouble(3),
                    isAvailable = reader.GetBoolean(4),
                    createdAt = reader.GetDateTime(5)
                };
                reader.Close();
                connection.Close();
                return good;
            }
            return null;
        }
        public List<Good> GetPage(int pageNumber)
        {
            const int pageSize = 10;
            int numberOfPages = GetTotalPages();
            if (pageNumber > numberOfPages || pageNumber <= 0)
            {
                throw new Exception("Page is not valid");
            }
            connection.Open();
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT * FROM goods LIMIT $limit OFFSET $offset";
            command.Parameters.AddWithValue("$limit", pageSize);
            command.Parameters.AddWithValue("$offset", (pageNumber - 1) * pageSize);

            SqliteDataReader reader = command.ExecuteReader();
            List<Good> page = new List<Good>();
            while (reader.Read())
            {
                Good good = new Good()
                {
                    id = reader.GetInt64(0),
                    name = reader.GetString(1),
                    description = reader.GetString(2),
                    price = reader.GetDouble(3),
                    isAvailable = reader.GetBoolean(4),
                    createdAt = reader.GetDateTime(5)
                };
                page.Add(good);
            }
            reader.Close();
            connection.Close();
            return page;
        }
        public int GetTotalPages()
        {
            const int pageSize = 10;
            return (int)Math.Ceiling(GetCount() / (double)pageSize);
```

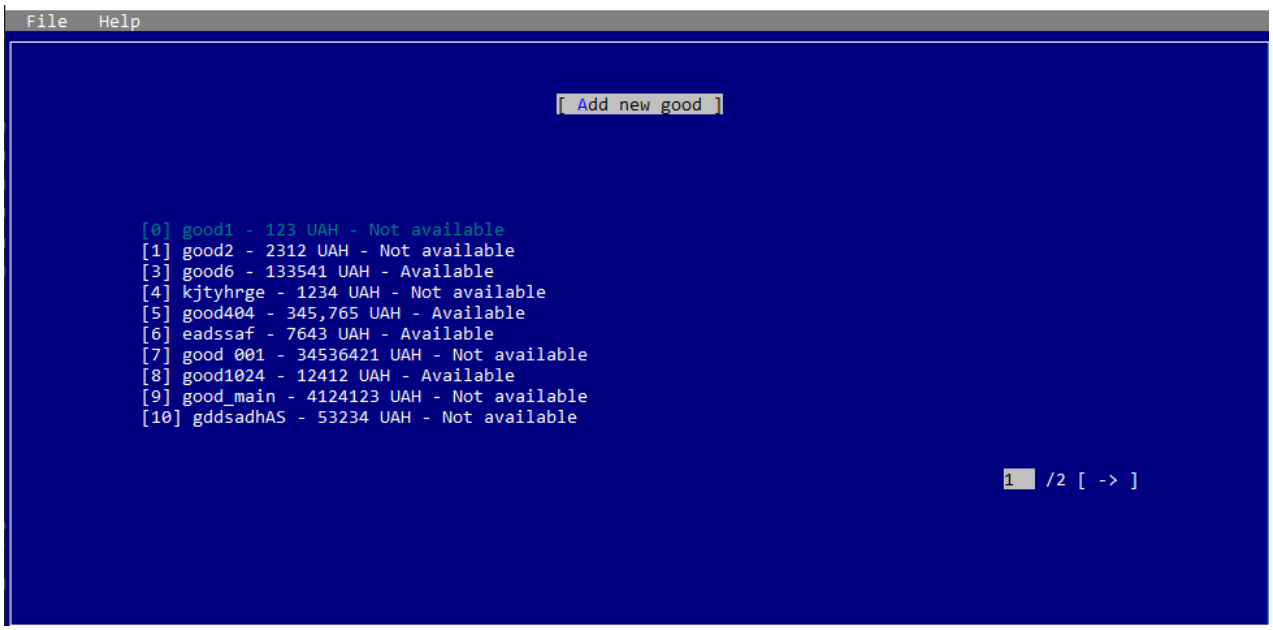
```
        }
        private long GetCount()
        {
            connection.Open();
            SqliteCommand command = connection.CreateCommand();
            command.CommandText = @"SELECT COUNT(*) FROM goods";
            long count = (long)command.ExecuteScalar();
            connection.Close();
            return count;
        }
    }
}
```

Good.cs

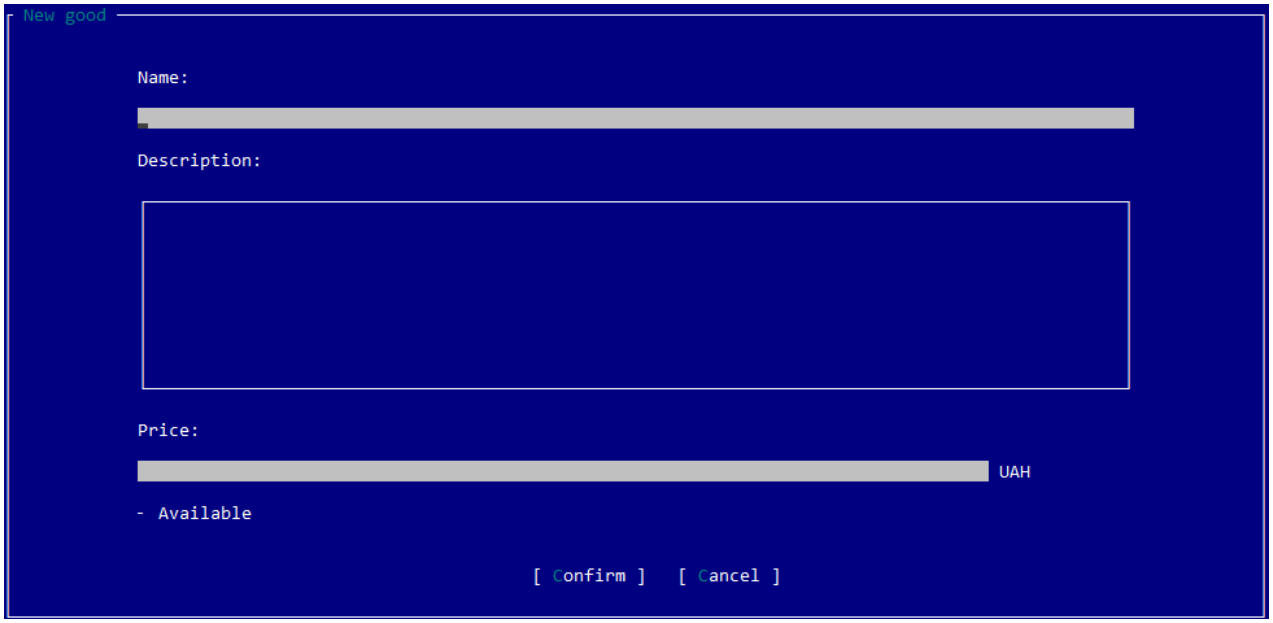
```
using System;

namespace ClassLibrary
{
    public class Good
    {
        public long id;
        public string name;
        public string description;
        public double price;
        public bool isAvailable;
        public DateTime createdAt;

        public override string ToString()
        {
            string available = isAvailable ? "Available" : "Not available";
            return $"[{id}] {name} - {price} UAH - {available}";
        }
    }
}
```

Приклади результатів

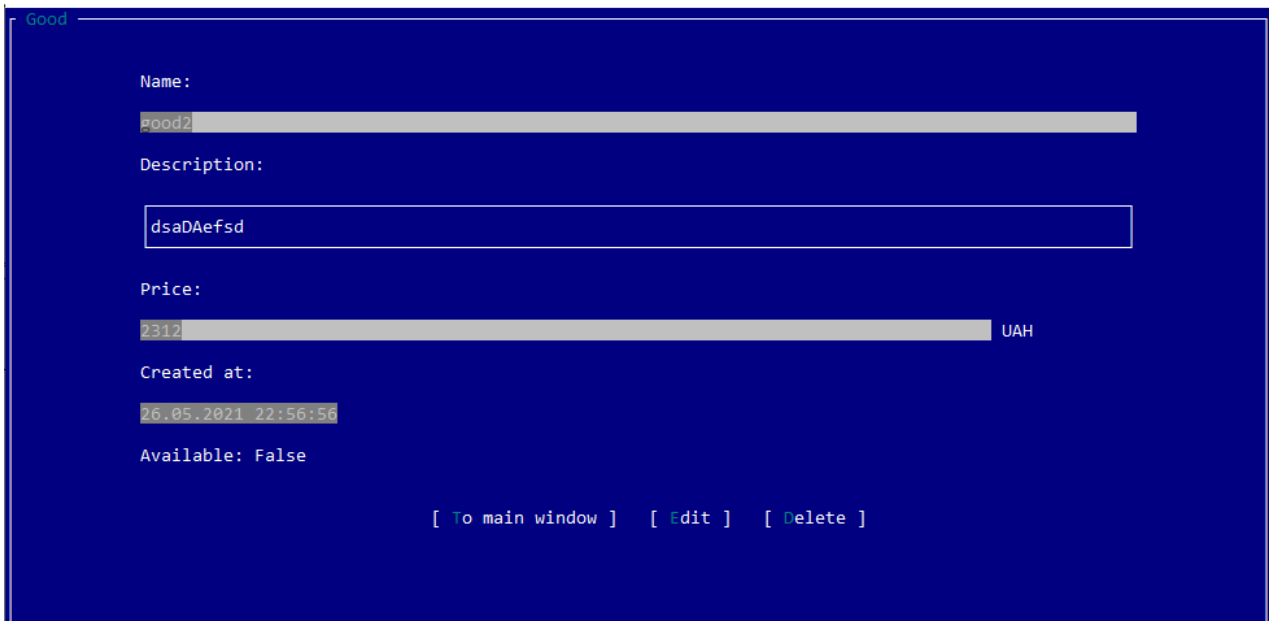
Приклад 1. Головне вікно



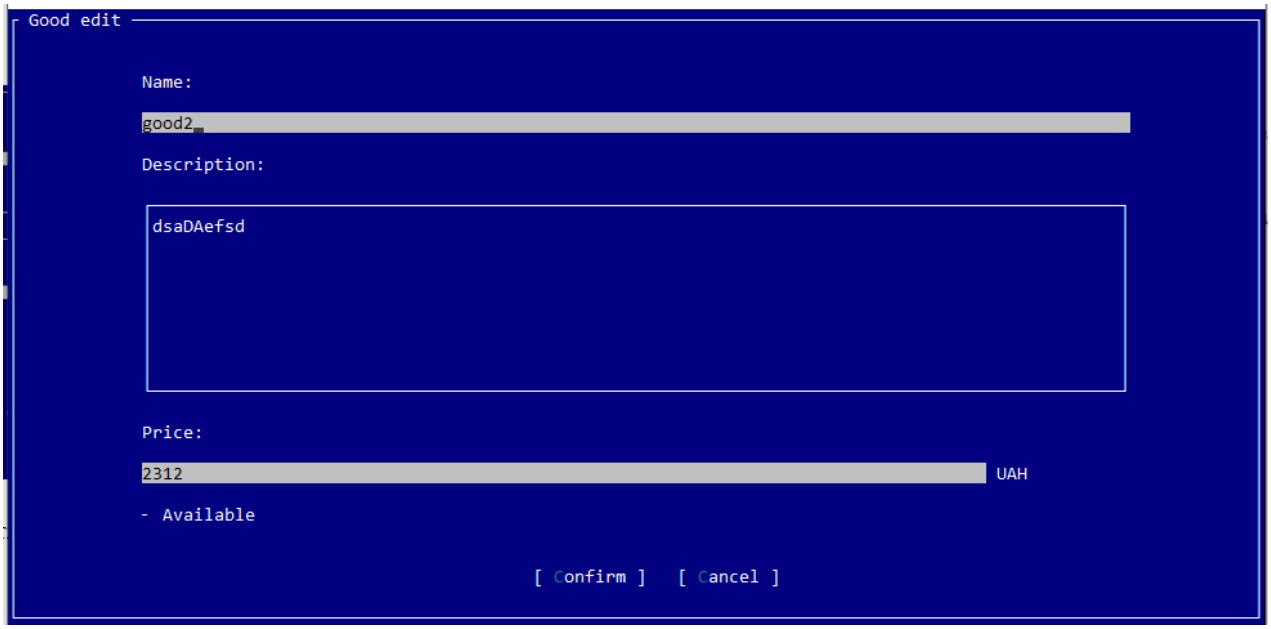
Приклад 2. Додавання нової сутності



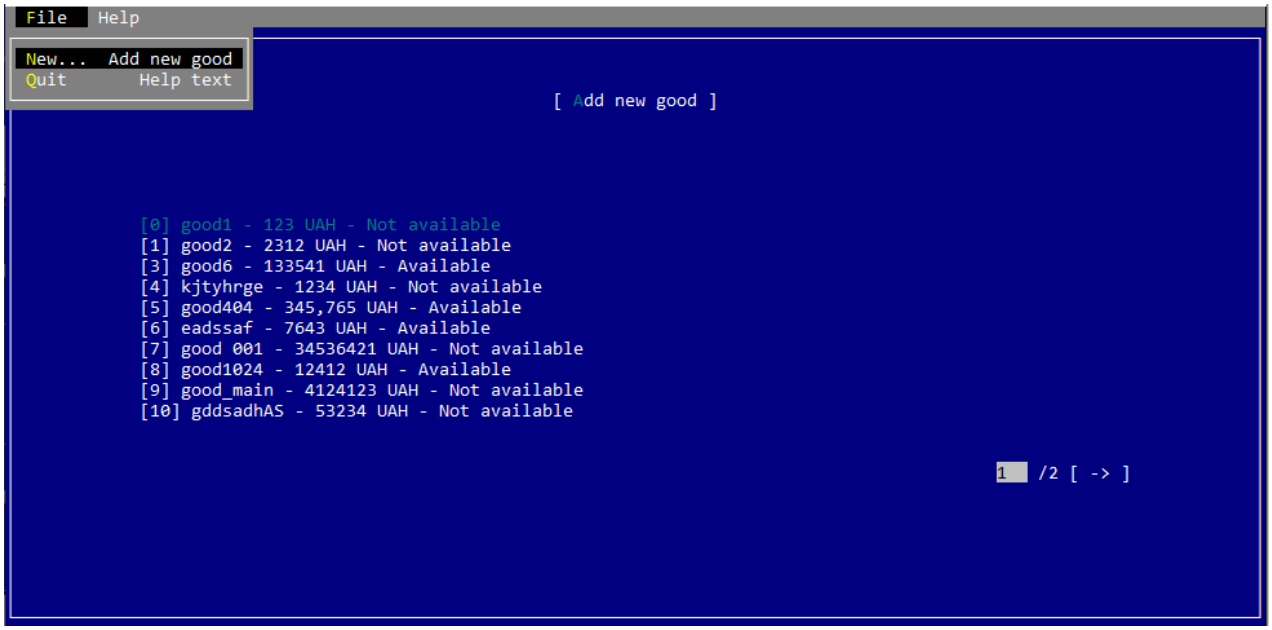
Приклад 3. Перегляд сутності



Приклад 4. Редагування сутності



Приклад 5. Меню головного вікна



Висновки

Виконавши дану лабораторну роботу було реалізовано подійно-орієнтований інтерфейс користувача (за допомогою бібліотеки Terminal GUI) для керування даними, що зберігаються у базі даних SQLite.

Програма компілювалася за допомогою утиліти dotnet.