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
Беліцький Олександр Сергійович
(*прізвище, ім’я, по батькові*)

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

варіант № 3

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.
- 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)
- 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

Сутність за варіантом:

3	**Активність**	ідентифікатор, тип (прогулянка|біг|велосипед|плавання|інше), назва, коментар, відстань (км.), дата і час створення запису про активність

Аналіз вимог і проектування

БД та модуль доступу до даних

Створити файл БД SQLite із однією таблицею для сутностей за варіантом. Налаштувати цілочисельний авто інкрементний ідентифікатор. Дату і час зберігати у текстовому полі як ISO 8601 рядок символів.

Створити клас сутності за варіантом. Використати для деяких полів цілочисельні типи даних та DateTime.

Створити клас репозиторія сутності за варіантом. Клас репозиторія має містити методи:

- додавання сутності в БД і отримання ідентифікатора доданої сутності
- видалення сутності з БД за ідентифікатором із результатом видалення (так-ні)
- оновлення даних сутності в БД із результатом оновлення (так-ні)
- отримання загальної кількості сторінок з сутностями (число)
- отримання колекції сутностей за номером сторінки (нумерація сторінок з 1)

Отримання кількості сторінок чи сторінки за її номером не повинні запитувати у БД всі записи з таблиці. Використовувати SQL оператори LIMIT, OFFSET і функцію COUNT().

Модуль інтерфейсу користувача

Містить набір класів вікон програми та підписки на події графічних елементів.

У інтерфейсі користувача використовувати графічні елементи за призначенням, відповідно до типів даних, які потрібно задавати:

- однорядкове поле вводу - для коротких рядків символів без переходів на новий рядок
- текстове поле - для багаторядково тексту
- спеціальні поля вводу для дати і часу
- чекбокс - для булевого значення
- радіокнопки або комбобокси - для вибору одного з фіксованих значень (перечислення)

Тексти коду програм

Activity.cs

```
using System;

public class Activity
{
    public long id;
    public string type;
    public string title;
    public string commentary;
    public double distance;
    public DateTime timeOfCreation;

    public override string ToString()
    {
        return $"[{id}] - {type}: {title}; {commentary}";
    }
}
```

ActivityRepository.cs

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

public class ActivityRepository
{
    private SqliteConnection connection;

    public ActivityRepository(SqliteConnection connection)
    {
        this.connection = connection;
    }

    public long Add(Activity activity)
    {
        connection.Open();

        SqliteCommand command = connection.CreateCommand();
        command.CommandText = @"INSERT INTO activities (type, title, commentary, distance,
timeOfCreation)
        VALUES ($type, $title, $commentary, $distance, $timeOfCreation);
        SELECT last_insert_rowid();";

        command.Parameters.AddWithValue("$type", activity.type);
        command.Parameters.AddWithValue("$title", activity.title);
        command.Parameters.AddWithValue("$commentary", activity.commentary);
        command.Parameters.AddWithValue("$distance", activity.distance);
        command.Parameters.AddWithValue("$timeOfCreation", DateTime.Now); //activity.timeOfCreation

        long newId = (long)command.ExecuteScalar();

        connection.Close();

        return newId;
    }

    public bool Delete(long id)
    {
        connection.Open();
```

```
        SqliteCommand command = connection.CreateCommand();
        command.CommandText = @"DELETE FROM activities WHERE id = $id";
        command.Parameters.AddWithValue("$id", id);

        int nChanged = command.ExecuteNonQuery();

        connection.Close();

        if (nChanged == 0)
        {
            return false;
        }

        return true;
    }

    public bool Update(long id, Activity activity)
    {
        connection.Open();
        SqliteCommand command = connection.CreateCommand();
        command.CommandText = $"UPDATE activities SET type = $type, title = $title, commentary =
$commentary, distance = $distance, timeOfCreation = $timeOfCreation WHERE id = $id";
        command.Parameters.AddWithValue("$id", id);
        command.Parameters.AddWithValue("$type", activity.type);
        command.Parameters.AddWithValue("$title", activity.title);
        command.Parameters.AddWithValue("$commentary", activity.commentary);
        command.Parameters.AddWithValue("$distance", activity.distance);
        command.Parameters.AddWithValue("$timeOfCreation", activity.timeOfCreation);
        int rowChange = command.ExecuteNonQuery();

        connection.Close();
        if (rowChange == 0)
        {
            return false;
        }

        return true;
    }

    public List<Activity> GetAll()
    {
        connection.Open();

        SqliteCommand command = connection.CreateCommand();
        command.CommandText = @"SELECT * FROM activities";

        List<Activity> activities = new List<Activity>();
        SqliteDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
            Activity activity = new Activity();

            activity.id = int.Parse(reader.GetString(0));
            activity.type = reader.GetString(1);
            activity.title = reader.GetString(2);
            activity.commentary = reader.GetString(3);
            activity.distance = double.Parse(reader.GetString(4), CultureInfo.InvariantCulture);
            activity.timeOfCreation = reader.GetDateTime(5);
            activities.Add(activity);
        }
        reader.Close();
        connection.Close();

        return activities;
    }

    public Activity GetById(int id)
    {
        connection.Open();
```

```
    SqliteCommand command = connection.CreateCommand();
    command.CommandText = @"SELECT * FROM activities  WHERE id = $id";
    command.Parameters.AddWithValue("$id", id);

    SqliteDataReader reader = command.ExecuteReader();

    Activity activity = new Activity();

    if (reader.Read())
    {
        activity.id = int.Parse(reader.GetString(0));
        activity.type = reader.GetString(1);
        activity.title = reader.GetString(2);
        activity.commentary = reader.GetString(3);
        activity.distance = double.Parse(reader.GetString(4), CultureInfo.InvariantCulture);
        activity.timeOfCreation = reader.GetDateTime(5);
        reader.Close();
    }

    reader.Close();
    connection.Close();

    return activity;
}

public int GetTotalPages()
{
    const int pageSize = 3;
    return (int)Math.Ceiling(this.GetCount() / (double)pageSize);
}
private long GetCount() //for GetTotalPages
{
    connection.Open();

    SqliteCommand command = connection.CreateCommand();
    command.CommandText = @"SELECT COUNT(*) FROM activities";

    long count = (long)command.ExecuteScalar();
    return count;
}

public List<Activity> GetPage(int pageNumber)
{
    if( pageNumber < 1)
    {
        throw new ArgumentOutOfRangeException(nameof(pageNumber));
    }
    connection.Open();

    SqliteCommand command = connection.CreateCommand();
    command.CommandText = @"SELECT * FROM activities LIMIT 3 OFFSET 3*($pageNumber-1)";
    command.Parameters.AddWithValue("$pageNumber", pageNumber);

    SqliteDataReader reader = command.ExecuteReader();
    List<Activity> activities = new List<Activity>();

    while (reader.Read())
    {

        Activity activity = new Activity();
        activity.id = int.Parse(reader.GetString(0));
        activity.type = reader.GetString(1);
        activity.title = reader.GetString(2);
        activity.commentary = reader.GetString(3);
        activity.distance = double.Parse(reader.GetString(4), CultureInfo.InvariantCulture);
        activity.timeOfCreation = reader.GetDateTime(5);
        activities.Add(activity);
    }

    reader.Close();
```

```
        connection.Close();

        return activities;
    }
}
```

EditActivityDialog.cs

```
using Terminal.Gui;

public class EditActivityDialog : newActivityDialog
{

    public EditActivityDialog()
    {
        this.Title = "Edit Activity";


    }

    public void SetActivity(Activity activity)
    {
        this.activityTitleTf.Text = activity.title;
        this.commentTf.Text = activity.commentary;
        this.typeTf.Text = activity.type;
        this.distanceTf.Text = activity.distance.ToString();
        this.timeTf.Text = activity.timeOfCreation.ToShortDateString();
        this.idTf.Text = activity.id.ToString();
    }

}
```

MainWindow.cs

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

public class MainWindow : Window
{
    private ListView allActivitiesView;
    private ActivityRepository repo;
    private int pageLength = 3;
    private int page = 1;

    private Label totalPagesLabel;
    private Label pageLabel;
    public MainWindow()
    {
        this.Title = "Main";

        Rect frame = new Rect(4, 8, 40, 20);
        allActivitiesView = new ListView(new List<Activity>())
        {
            Width = Dim.Fill(),
            Height = Dim.Fill(),
        };
        allActivitiesView.OpenSelectedItem += OpenActivity;


        Button prevPageBtn = new Button(2, 6, "prev");
        prevPageBtn.Clicked += ToPrevPage;
        pageLabel = new Label("?")
```

```
        {
            X = Pos.Right(prevPageBtn) + 2,
            Y = Pos.Top(prevPageBtn),
            Width = 5,
        };
        totalPagesLabel = new Label("?")
        {
            X = Pos.Right(pageLabel) + 2,
            Y = Pos.Top(prevPageBtn),
            Width = 5,
        };
        Button nextPageBtn = new Button("next")
        {
            X = Pos.Right(totalPagesLabel) + 2,
            Y = Pos.Top(prevPageBtn),
        };
        nextPageBtn.Clicked += ToNextPage;
        this.Add(prevPageBtn, pageLabel, totalPagesLabel, nextPageBtn);

        FrameView frameView = new FrameView("Activities")
        {
            X = 2,
            Y = 8,
            Width = Dim.Fill() - 4,
            Height = pageLength + 2,
        };
        frameView.Add(allActivitiesView);


        Button newActivityBtn = new Button(2, 4, "Create new activity");
        newActivityBtn.Clicked += ClickNew;


        this.Add(frameView);
        this.Add(newActivityBtn);

    }

    private void ToPrevPage()
    {
        if (page == 1)
        {
            return;
        }
        this.page -= 1;
        ShowCurrentPage();
    }
    private void ToNextPage()
    {
        if (page >= repo.GetTotalPages())
        {
            return;
        }
        this.page += 1;
        ShowCurrentPage();
    }
    private void ShowCurrentPage()
    {
        if(allActivitiesView == null)
        {
            List<string> noRecords = new List<string>();
            noRecords.Add("No record in database");
            allActivitiesView.SetSource(noRecords);
        }
        this.pageLabel.Text = page.ToString();
        this.totalPagesLabel.Text = repo.GetTotalPages().ToString();
        this.allActivitiesView.SetSource(repo.GetPage(page));
    }


    public void SetRepository(ActivityRepository repository)
```

```
{
    this.repo = repository;
    ShowCurrentPage();
}

public void ClickNew()
{
    newActivityDialog dialog = new newActivityDialog();
    Application.Run(dialog);

    if (!dialog.canceled)
    {
        Activity activity = dialog.GetActivity();
        if (!(activity == null))
        {
            long activityId = repo.Add(activity);
            activity.id = activityId;
            OpenActivityAfterAdding(activity);
        }
        else
            ClickNew();
    }
    allActivitiesView.SetSource(repo.GetPage(page));
}
private void OpenActivityAfterAdding(Activity activity)
{
    OpenActivityDialog dialog = new OpenActivityDialog();

    dialog.SetActivity(activity);

    Application.Run(dialog);
}

public void ClickQuit()
{
    Application.RequestStop();
}

public void ClickAbout()
{
    Button back = new Button(30, 16, "Back");
    back.Clicked += ClickQuit;

    Dialog dialog = new Dialog("About", back);
    TextView textView = new TextView()
    {
        X = 2,
        Y = 2,
        Width = 65,
        Height = 12,
        ReadOnly = true,
        Text = @"
    Цей додаток призначений для ведення списку подій:
        їхнього короткого опису, коментарів і тд.


        Знаходиться на ранній стадії розробки


            Автор: Беліцький Олександр

                ActivityApp Beta"

    };

    dialog.Add(textView);
    dialog.AddButton(back);

    Application.Run(dialog);
}
```

```
    private void OpenActivity(ListViewItemEventArgs args)
    {
        Activity activity = (Activity)args.Value;
        OpenActivityDialog dialog = new OpenActivityDialog();

        dialog.SetActivity(activity);

        Application.Run(dialog);

        if (dialog.deleted)
        {
            bool result = repo.Delete(activity.id);
            if (result)
            {
                int pages = repo.GetTotalPages();
                if (page > pages && page > 1)
                {
                    page -= 1;
                    this.ShowCurrentPage();
                }

                allActivitiesView.SetSource(repo.GetPage(page));
            }
            else
            {
                MessageBox.ErrorQuery("Delete activity", "Can not delete activity", "OK");
            }
        }

        if (dialog.updated)
        {
            bool result = repo.Update(activity.id, dialog.GetActivity());
            if (result)
            {
                allActivitiesView.SetSource(repo.GetPage(page));
            }
            else
            {
                MessageBox.ErrorQuery("Update activity", "Can not update activity", "OK");
            }
        }
    }
}
```

NewActivityDialog.cs

```
using System;
using Terminal.Gui;

public class newActivityDialog : Dialog
{
    public bool canceled;

    protected TextField activityTitleTf;
    protected TextField typeTf;
    protected TextField commentTf;
    protected TextField distanceTf;
    protected DateField timeTf;
    protected TextField idTf;

    public newActivityDialog()
    {
        this.Title = "Create new activity";

        Button okBtn = new Button("OK");
        okBtn.Clicked += DialogConfirm;
```

```
        Button cancelBtn = new Button("Cancel");
        cancelBtn.Clicked += DialogCanceled;

        this.AddButton(cancelBtn);
        this.AddButton(okBtn);

        int rightColumnX = 20;

        Label activityTitleLbl = new Label(2, 2, "Title: ");
        activityTitleTf = new TextField("")
        {
            X = rightColumnX,
            Y = Pos.Top(activityTitleLbl),
            Width = 40,
        };
        this.Add(activityTitleLbl, activityTitleTf);


        Label typeLbl = new Label(2, 4, "Type: ");
        typeTf = new TextField("")
        {
            X = rightColumnX,
            Y = Pos.Top(typeLbl),
            Width = 40,
        };
        this.Add(typeLbl, typeTf);


        Label commentLbl = new Label(2, 6, "Commentary: ");
        commentTf = new TextField("")
        {
            X = rightColumnX,
            Y = Pos.Top(commentLbl),
            Width = 40,
        };
        this.Add(commentLbl, commentTf);

        Label distanceLbl = new Label(2, 8, "Distance: ");
        distanceTf = new TextField("")
        {
            X = rightColumnX,
            Y = Pos.Top(distanceLbl),
            Width = 40,
        };
        this.Add(distanceLbl, distanceTf);

        Label timeLbl = new Label(2, 10, "Time of creation: ");
        timeTf = new DateField()
        {
            X = rightColumnX,
            Y = Pos.Top(timeLbl),
            Width = 40,
            ReadOnly = true,
        };
        this.Add(timeLbl, timeTf);

        Label idLbl = new Label(2, 12, "Id: ");
        idTf = new TextField("Id will be choose automatically")
        {
            X = rightColumnX,
            Y = Pos.Top(idLbl),
            Width = 40,
            ReadOnly = true,
        };
        this.Add(idLbl, idTf);


    }

    public Activity GetActivity()
    {
```

```
        if (!double.TryParse(distanceTf.Text.ToString(), out double result))
        {
            MessageBox.ErrorQuery("Error", $"Distance must be real number but have
{distanceTf.Text.ToString()}", "OK");
            return null;
        }
        return new Activity()
        {
            title = activityTitleTf.Text.ToString(),
            type = typeTf.Text.ToString(),
            commentary = commentTf.Text.ToString(),
            distance = result,
            timeOfCreation = DateTime.Now,
        };
    }

    public Activity GetEditActivity()
    {
        if (!double.TryParse(distanceTf.Text.ToString(), out double result))
        {
            MessageBox.ErrorQuery("Error", $"Distance must be real number but have
{distanceTf.Text.ToString()}", "OK");
            return null;
        }
        return new Activity()
        {
            title = activityTitleTf.Text.ToString(),
            type = typeTf.Text.ToString(),
            commentary = commentTf.Text.ToString(),
            distance = result,
            timeOfCreation = System.DateTime.Parse(timeTf.Text.ToString()),
            id = long.Parse(idTf.Text.ToString()),
        };
    }

    private void DialogCanceled()
    {
        this.canceled = true;
        Application.RequestStop();
    }

    private void DialogConfirm()
    {
        this.canceled = false;
        Application.RequestStop();
    }

}
```

OpenActivityDialog.cs

```
using System;
using Terminal.Gui;

public class OpenActivityDialog : Dialog
{
    public bool deleted;
    public bool updated;

    protected Activity activity;

    private TextField activityTitleTf;
    private TextField typeTf;
    private TextField commentTf;
    private TextField distanceTf;
    private DateField timeTf;
    private TextField idTf;
```

```
public OpenActivityDialog()
{
    this.Title = "Open activity";

    Button okBtn = new Button("Back");
    okBtn.Clicked += DialogConfirm;

    this.AddButton(okBtn);

    int rightColumnX = 20;

    Label activityTitleLbl = new Label(2, 2, "Title: ");
    activityTitleTf = new TextField("")
    {
        X = rightColumnX,
        Y = Pos.Top(activityTitleLbl),
        Width = 40,
        ReadOnly = true,
    };
    this.Add(activityTitleLbl, activityTitleTf);


    Label typeLbl = new Label(2, 4, "Type: ");
    typeTf = new TextField("")
    {
        X = rightColumnX,
        Y = Pos.Top(typeLbl),
        Width = 40,
        ReadOnly = true,
    };
    this.Add(typeLbl, typeTf);


    Label commentLbl = new Label(2, 6, "Commentary: ");
    commentTf = new TextField("")
    {
        X = rightColumnX,
        Y = Pos.Top(commentLbl),
        Width = 40,
        ReadOnly = true,
    };
    this.Add(commentLbl, commentTf);

    Label distanceLbl = new Label(2, 8, "Distance: ");
    distanceTf = new TextField("")
    {
        X = rightColumnX,
        Y = Pos.Top(distanceLbl),
        Width = 40,
        ReadOnly = true,
    };
    this.Add(distanceLbl, distanceTf);

    Label timeLbl = new Label(2, 10, "Time of creation: ");
    timeTf = new DateField()
    {
        X = rightColumnX,
        Y = Pos.Top(timeLbl),
        Width = 40,
        ReadOnly = true,
    };
    this.Add(timeLbl, timeTf);

    Label idLbl = new Label(2, 12, "Id: ");
    idTf = new TextField("")
    {
        X = rightColumnX,
        Y = Pos.Top(idLbl),
        Width = 40,
        ReadOnly = true,
```

```
        };
        this.Add(idLbl, idTf);


        Button editBtn = new Button(2, 14, "Edit");
        editBtn.Clicked += ActivityEdit;

        Button deleteBtn = new Button("Delete")
        {
            X = Pos.Right(editBtn) + 2,
            Y = Pos.Top(editBtn),
        };
        deleteBtn.Clicked += ActivityDelete;

        this.Add(editBtn);
        this.Add(deleteBtn);
    }

    private void ActivityEdit()
    {
        EditActivityDialog dialog = new EditActivityDialog();

        dialog.SetActivity(this.activity);

        Application.Run(dialog);

        if (!dialog.canceled)
        {
            Activity updatedActivity = dialog.GetEditActivity();
            this.updated = true;
            this.SetActivity(updatedActivity);
            activity = updatedActivity;
        }
    }



    private void ActivityDelete()
    {
        int index = MessageBox.Query("Delete activity", "Are you sure?", "No", "Yes");
        if (index == 1)
        {
            deleted = true;
            Application.RequestStop();
        }
    }

    public Activity GetActivity()
    {
        return this.activity;
    }

    public void SetActivity(Activity activity)
    {
        this.activity = activity;
        this.activityTitleTf.Text = activity.title;
        this.typeTf.Text = activity.type;
        this.commentTf.Text = activity.commentary;
        this.distanceTf.Text = activity.distance.ToString();
        this.timeTf.Text = activity.timeOfCreation.ToShortDateString();
        this.idTf.Text = activity.id.ToString();
    }


    private void DialogConfirm()
    {
        Application.RequestStop();
    }

}
```

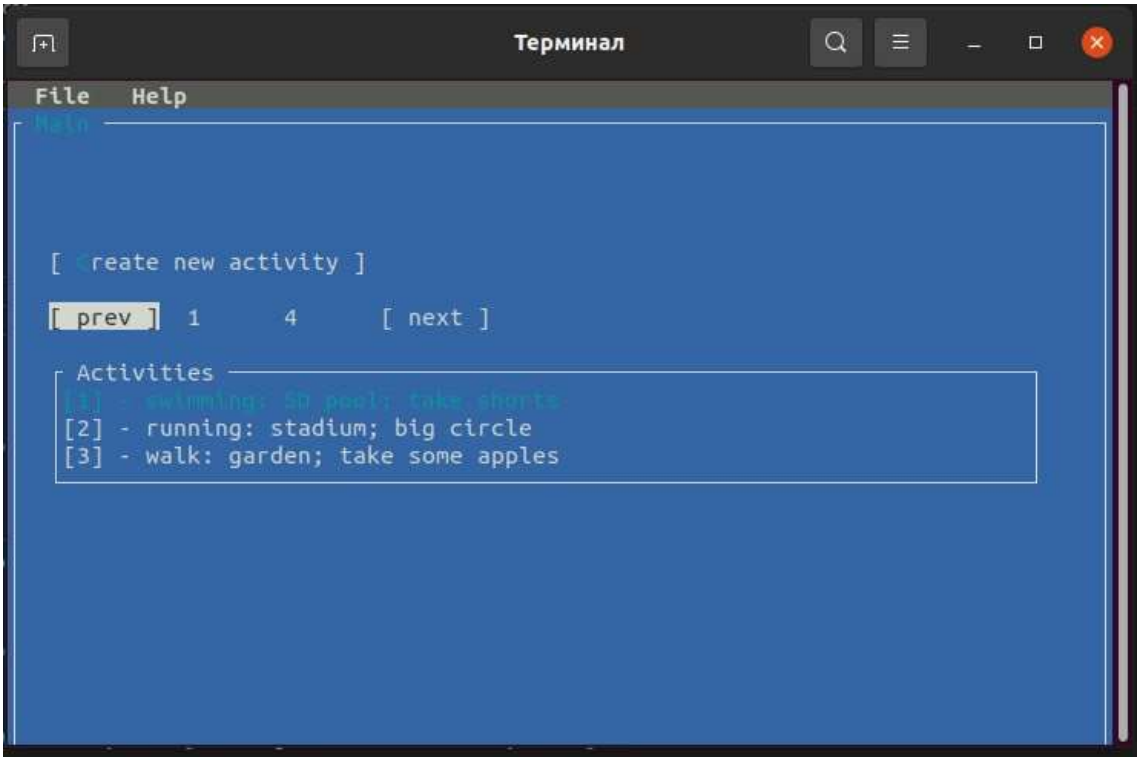
Program.cs

```
using System;
using Microsoft.Data.Sqlite;
using Terminal.Gui;

class Program
{

    static void Main(string[] args)
    {
        string filepath = "./activitydb";
        SqliteConnection connection = new SqliteConnection($"Data Source={filepath}");
        ActivityRepository repo = new ActivityRepository(connection);

        Application.Init();

        Toplevel top = Application.Top;

        MainWindow win = new MainWindow();
        win.SetRepository(repo);

        MenuBar menuBar = new MenuBar(new MenuBarItem[]
        {
            new MenuBarItem("_File", new MenuItem[]
            {
                new MenuItem("_New record", "",win.ClickNew),
                new MenuItem("_Quit", "", win.ClickQuit),
            }),
            new MenuBarItem("_Help", new MenuItem[]
            {
                new MenuItem("_About","", win.ClickAbout)
            })
        });


        top.Add(menuBar);

        top.Add(win);


        Application.Run();
    }


}
```

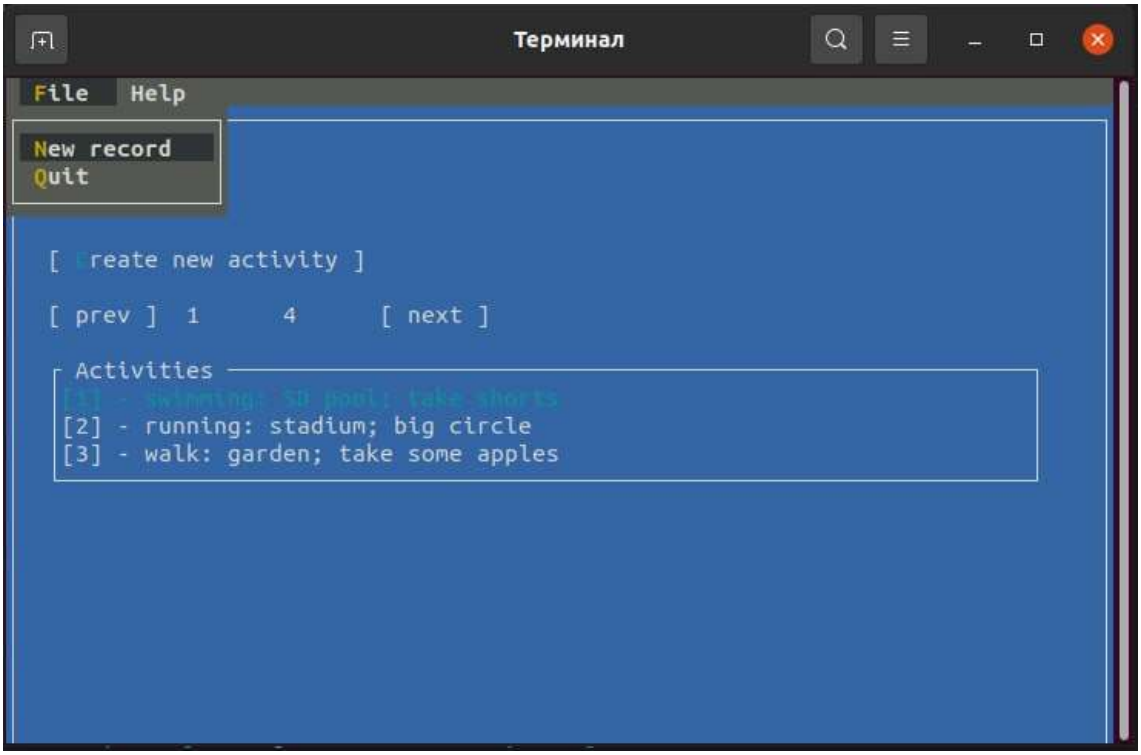
Приклади результатів

Тут буде перелік вікон, які можуть бути представлені для користувача:

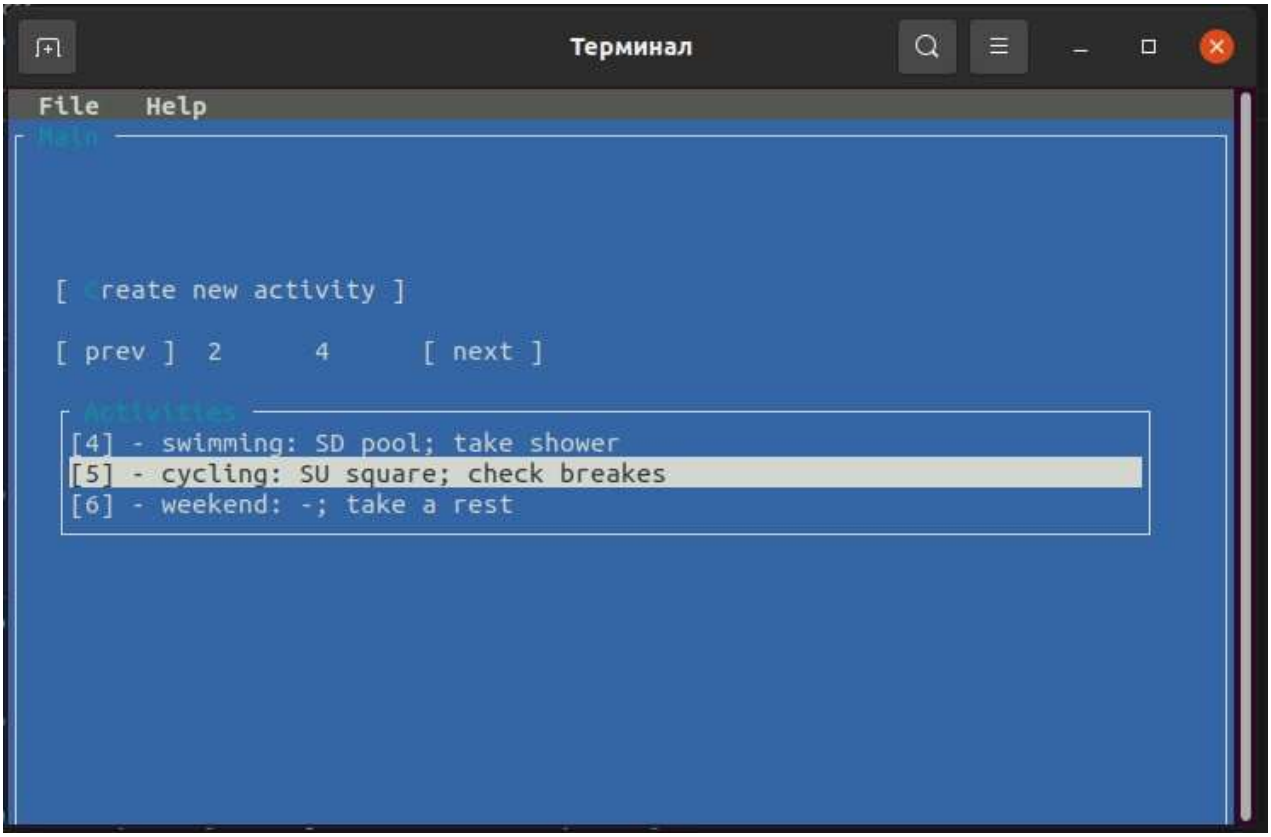
Головне (домашнє) вікно:



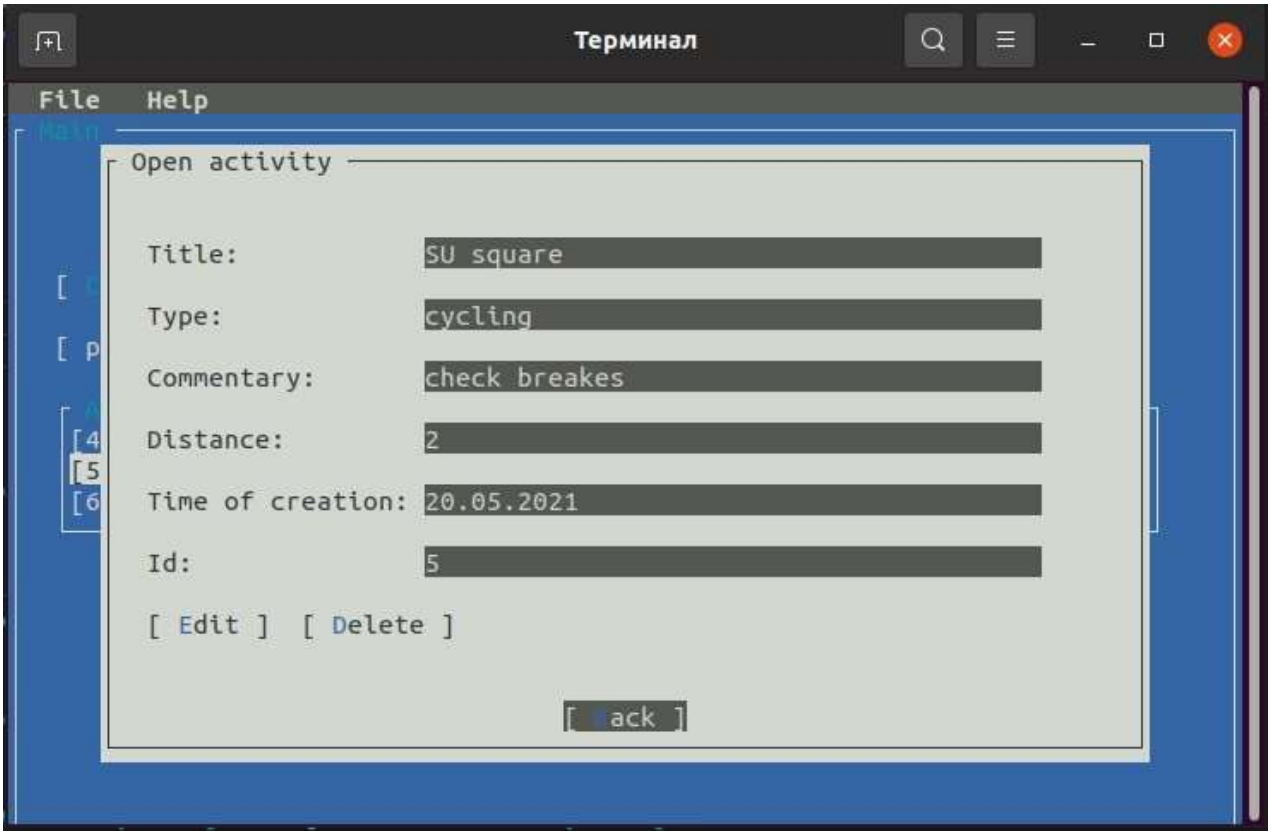
Вигляд панелі меню:



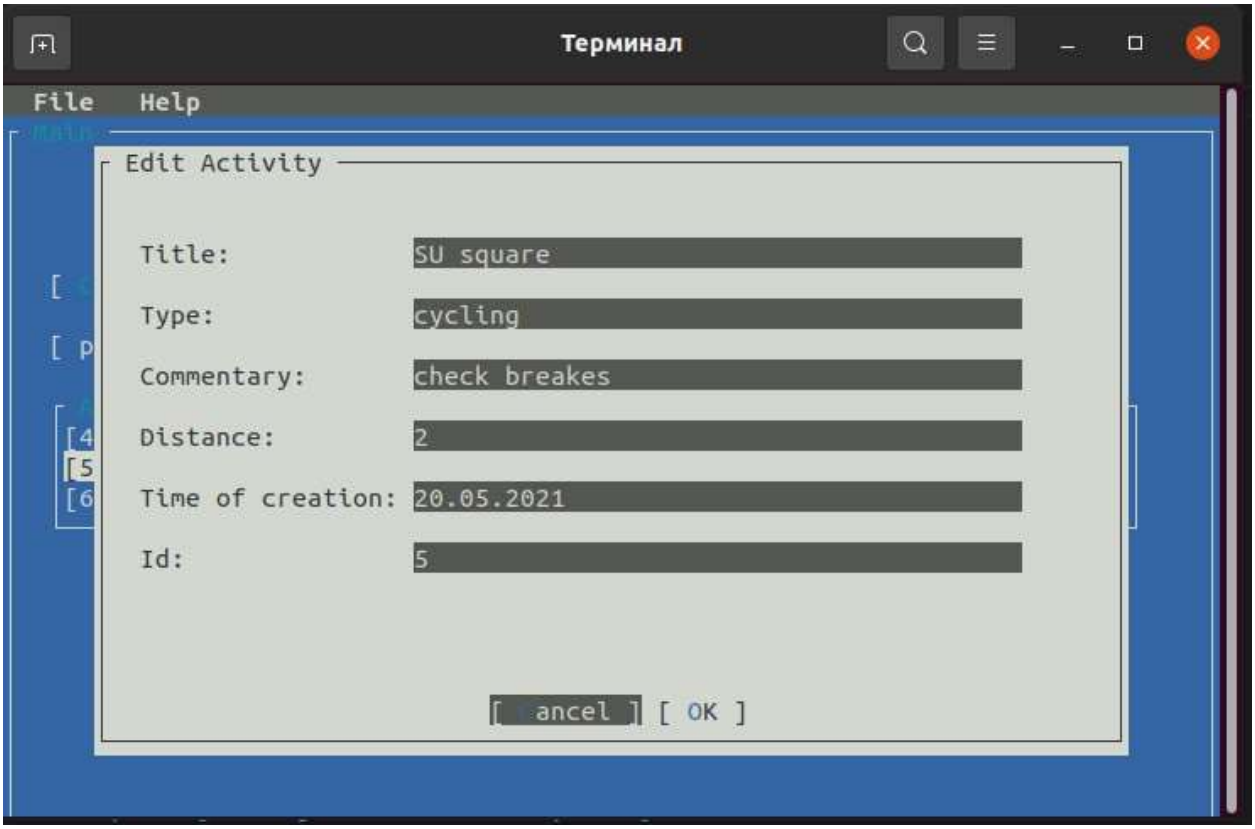
Приклад пересування по списку:



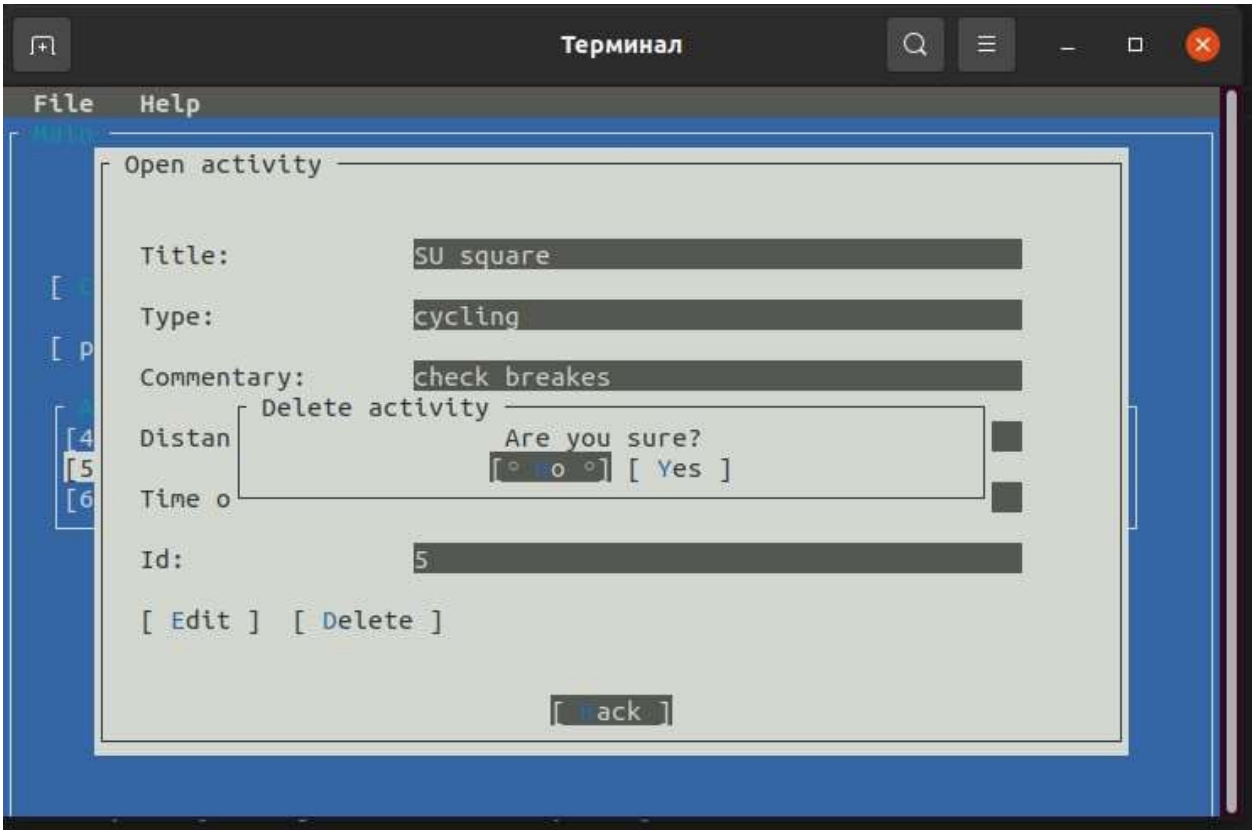
Вигляд вікна перегляду сутності:



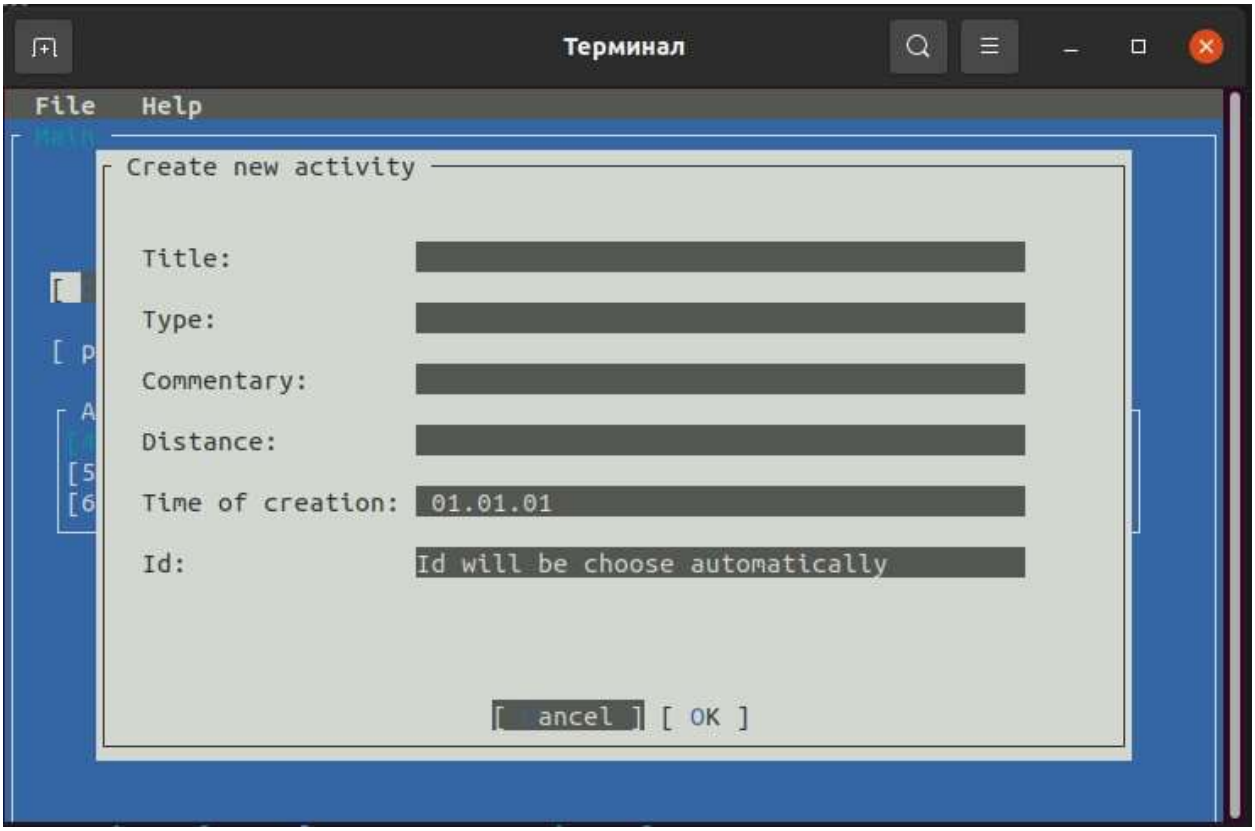
Вигляд вікна редагування сутності:



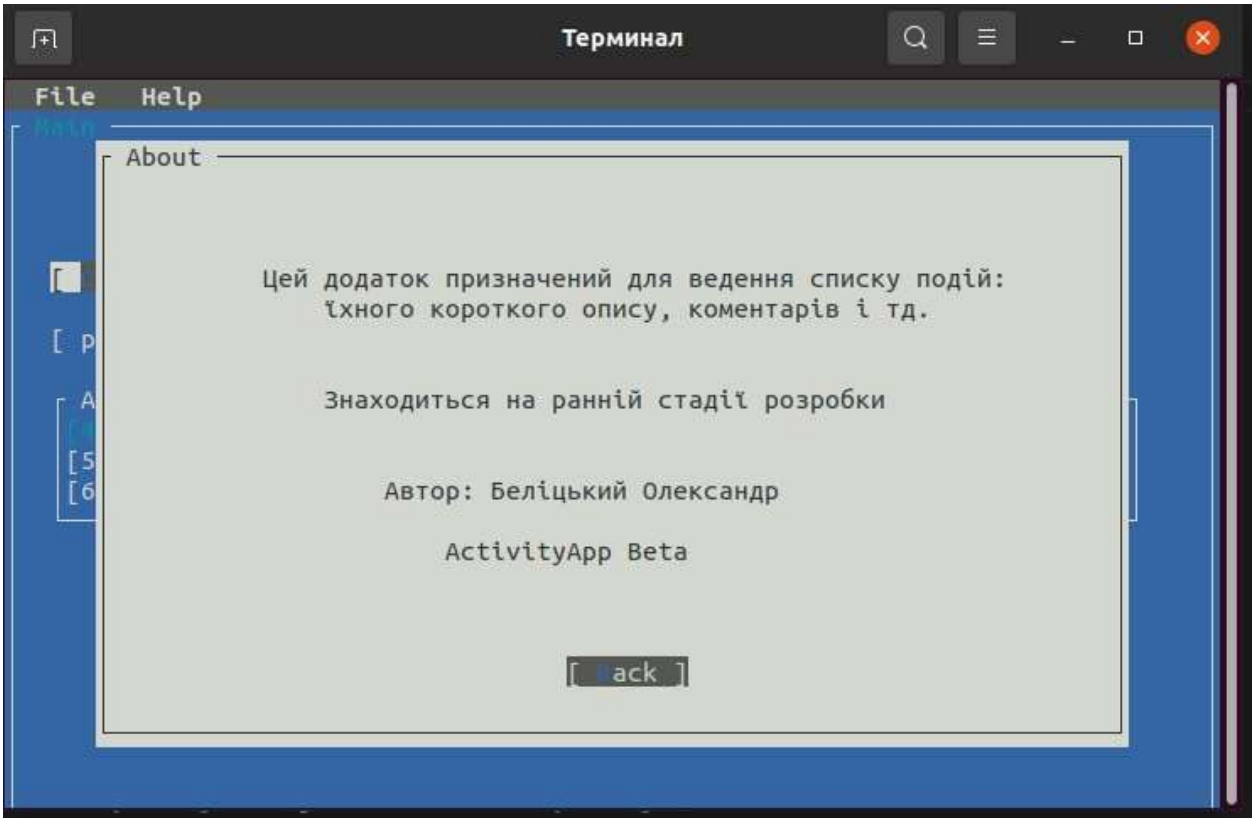
Інтерактив щодо впевненності видалення сутності:



Вигляд вікна створення сутності:



Вигляд вікна інформації про додаток:



Висновки

Виконавши дану лабораторну роботу було проведено вивчення основних принципів подійно-орієнтованого програмування. Було вивчено основні види графічних інтерфейсів користувача. Було встановлено зв'язок між користувацьким інтерфейсом та базою даних. Якщо трохи модернізувати даний інтерфейс і програму, то можна вважати, що це своєрідний тайм-менеджмент додаток.

Лабораторну роботу було виконано за допомогою бібліотеки Terminal.Gui. Terminal.Gui - це бібліотека, призначена для створення консольних програм на основі C #. Вона розроблена для спрощення написання програм, які працюватимуть на монохромних терміналах, а також сучасних кольорових терміналах з підтримкою миші.

Компіляція всього коду відбувалася за допомогою утиліти dotnet.