Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

\_\_\_\_\_\_\_Програмного забезпечення комп’ютерних систем\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЄКТ**

з Принципи конструювання програмного забезпечення

(назва дисципліни)

на тему:\_ **Система управління базою даних**

**довідникової книги засобами мови С#**

Виконав: студент (ка) \_\_3\_\_\_ курсу, групи\_\_\_343\_ск\_\_\_

спеціальності\_\_\_\_\_\_\_\_\_\_\_\_\_\_121\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр спеціальності)

\_\_\_\_Інженерія програмного забезпечення\_\_\_\_\_

(назва спеціальності)

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Чорнописький А. В. \_\_\_\_

(підпис) (прізвище, ініціали)

Керівник \_\_\_\_\_\_\_Комісарчук В.В.\_\_\_\_\_\_

(прізвище, ініціали)

| **До захисту допущено:** |  |  | **Дата захисту** «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |
| --- | --- | --- | --- |
| від «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |  |  | **Оцінка:** |
| Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | за національною шкалою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (словами) |
| Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (цифра) |
|  |  |  | за шкалою ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  | (літера) |

Чернівці, 2023

Чернівецький національний університет імені Юрія Федьковича

Кафедра Програмного забезпечення комп’ютерних систем\_\_\_

Спеціальність Інженерія програмного забезпечення

# **ЗАВДАННЯ**

**на курсовий проєкт студента**

\_ Чорнописького Андрія Володимировича

(прізвище, ім’я, по-батькові)

1. Тема проєкту Система управління

базою даних Довідникова Книга»

2. Вихідні дані до проєкту:

- розробити можливість переглядати записи в базі даних

- розробити можливість змінювати записи в базі даних

- розробити можливість видаляти записи з бази даних

- розробити можливість фільтрувати записи в базі даних за заданою умовою

3. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити):

- розробити загальні вимоги до програми

- описати методи програми

- описати користувацький інтерфейс

- зробити висновки по розробленій програмі

4. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):

- скріншоти роботи з програмою

- блок-схеми роботи з програмою

Завдання прийняв (ла) до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Керівник проєкту \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис керівника)

# **РЕФЕРАТ**

В курсовому проєкті розроблено систему управління базою даних Довідникової Книги для комп’ютерів, які працюють під керуванням операційної системи Windows.

Область застосування – програмне забезпечення для комп’ютерів, що працюють на операційній системі Windows.

Програма реалізована засобами розробки середовища JetBrains Rider мовою C# із застосуванням графічної підсистеми WPF.

Ця система може бути розширена в майбутньому із додаванням нового функціоналу та зміною користувацького інтерфейсу.

Курсовий проєкт містить 40 с., 14 рис., 3 табл., 1 додатки, 6 джерел

*ОПЕРАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗОЮ ДАНИХ,* Rider*, WINDOWS, C#,* WPF*.*

# **SUMMARY**

The course project involves the development of a database management system for a Reference Book, designed to operate on computers running the Windows operating system. The application's scope is software for computers operating on the Windows operating system. The program is implemented using the development tools of the JetBrains Rider environment, using the C# programming language and the WPF graphical subsystem. This system can be expanded in the future by adding new functionality and modifying the user interface.

The course project contains 59 pages, 15 figures, 5 tables, 1 appendix, 6 sources

OPERATING SYSTEM, DATABASE, DATABASE MANAGEMENT SYSTEM, Rider, WINDOWS, C#, WPF.

# **ЗМІСТ**

Зміст

[ЗАВДАННЯ](about:blank) [2](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.gjdgxs)

[РЕФЕРАТ](about:blank) [3](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.30j0zll)

[SUMMARY](about:blank) [4](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.1fob9te)

[ЗМІСТ](about:blank) 5

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ](about:blank) [5](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.2et92p0)

[1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ](about:blank) [7](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.tyjcwt)

[1.1 Загальні вимоги до програми](about:blank) [7](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.3dy6vkm)

[1.1.1 Вимоги до графічного інтерфейсу користувача](about:blank) [7](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.1t3h5sf)

[1.1.2 Вимоги до архітектури програми](about:blank) [7](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.4d34og8)

[1.1.3 Вимоги до функціональності додатка](about:blank) [7](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.2s8eyo1)

[1.2 Призначення та область застосування](about:blank) [8](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.17dp8vu)

[1.3 Функціональні вимоги](about:blank) [8](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.3rdcrjn)

[2. ОПИС ПРОГРАМИ](about:blank) [9](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.26in1rg)

[2.1 Структура програми](about:blank) [9](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.lnxbz9)

[2.1.1 Модулі програми](about:blank) [9](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.35nkun2)

[2.1.2 Алгоритми роботи програми](about:blank) [10](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.1ksv4uv)

[2.2 Опис методів програми](about:blank) [11](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.44sinio)

[2.3 Програмні засоби](about:blank) [16](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.z337ya)

[2.4 Опис користувацького інтерфейсу](about:blank) [17](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.3j2qqm3)

[ВИСНОВКИ](about:blank) [25](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.1y810tw)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ](about:blank) [26](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.4i7ojhp)

[ДОДАТКИ](about:blank) [27](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.2xcytpi)

[ДОДАТОК А. Скролінг (текст) програми](about:blank) [27](https://docs.google.com/document/d/1iPz7F1fTuIg7NWQy1wGmMrek5fgmQ5gX/edit#heading=h.1ci93xb)

# **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ**

СУБД – система управління базою даних.

ОС – операційна система.

БД – база даних.

WF – Windows Form

GUI – graphical user interface (графічний інтерфейс користувача).

IDE – integrated development environment (інтегроване середовище розробки).

Rider - IDE

# **АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ**

## **1.1 Загальні вимоги до програми**

### **1.1.1 Вимоги до графічного інтерфейсу користувача**

1. Робоча мова інтерфейсу – українська
2. Використовувати терміни зрозумілі для користувача
3. Навігаційна панель, яка повинна забезпечувати перегляд, редагування, додавання, видалення даних
4. Панель для фільтрації, сортування та пошуку по ключу
5. Посібник користувача з усією потрібною інформацією для користування системою
6. Кольорова гама повинна наслідувати загальноприйняті рекомендації.

### **1.1.2 Вимоги до архітектури програми**

1. Використання стандартних класів, класів нащадків, абстрактних класів та класів-інтерфейсів:

− подання вхідних даних користувачем на GUI;

− відображення результатів у віконному режимі;

− збереження даних у файл та зчитування даних з файлу.

2) Стійкість програми:

– програма повинна не втрачати працездатності при будь-яких діях користувача;

– інформація, що вводиться, скрізь, де це можливо, піддається логічному забезпеченню цілісності даних;

– при будь-яких діях користувача не повинні втрачатись дані або їх цілісність.

### **1.1.3 Вимоги до функціональності додатка**

1) Запис даних в базу даних.

2) Зчитування файлів з бази даних.

3) Засіб для обробки даних.

* Фільтрування записів
* Сортування записів
* Видалення записів
* Створення записів
* Редагування записів

4) Вікно з інформацією про програму.

## **1.2 Призначення та область застосування**

**Мета роботи:** керування базою даних довідникової книги для користувачів ОС Windows.

Реалізована версія системи контролю довідникової книги призначена для ділового використання працівниками довідникової книги для керування інформацією про людей.

Область застосування – програмне забезпечення для комп’ютерів під керуванням ОС Windows.

## **1.3 Функціональні вимоги**

До програмного забезпечення висуваються такі функціональні вимоги:

1. Сторінка на якій відбуваються всі операції з керуванням інформацією про людей.
2. Наявність меню для навігації між сторінками керування інформацією.
3. Наявність фільтрації записів.
4. Наявність сортування записів в порядку спадання та зростання по різним ключам.
5. Функціональні вимоги до пошуку записів:

* Пошук записів за вказаним текстом по приблизним символам.

1. Функціональні вимоги до додавання записів:

* Оновлення бази даних при додаванні нових записів.
* Оновлення записів у вкладці перегляду записів в базі даних.

# 

# **2. ОПИС ПРОГРАМИ**

## **2.1 Структура програми**

### **2.1.1 Модулі програми**

Робота розробленого програмного забезпечення реалізується наступними модулями:

* UserAddWindow - клас, що дозволяє добавити нового користувача в базу даних
* UserGuideWindow - клас, що дозволяє відобразити “Посібник Користувача”
* OneStepGuide-FourStepGuide - класи, що демонструють як користуватись кожним розділом в додатку та взаємодіє з класом “UserGuideWindow”
* ViewUsers - клас, який надає функціонал для перегляду існуючих користувачів
* User - клас, який є моделлю, що представляє користувача в системі з властивостями
* FileStaticService - це клас, що представляє статичний сервіс для роботи з файлом, використовуючи об’єкт користувача з класу “User”
* ValidateUser - статичний клас, який використовується для перевірки валідності даних користувача

### **2.1.2 Алгоритми роботи програми**

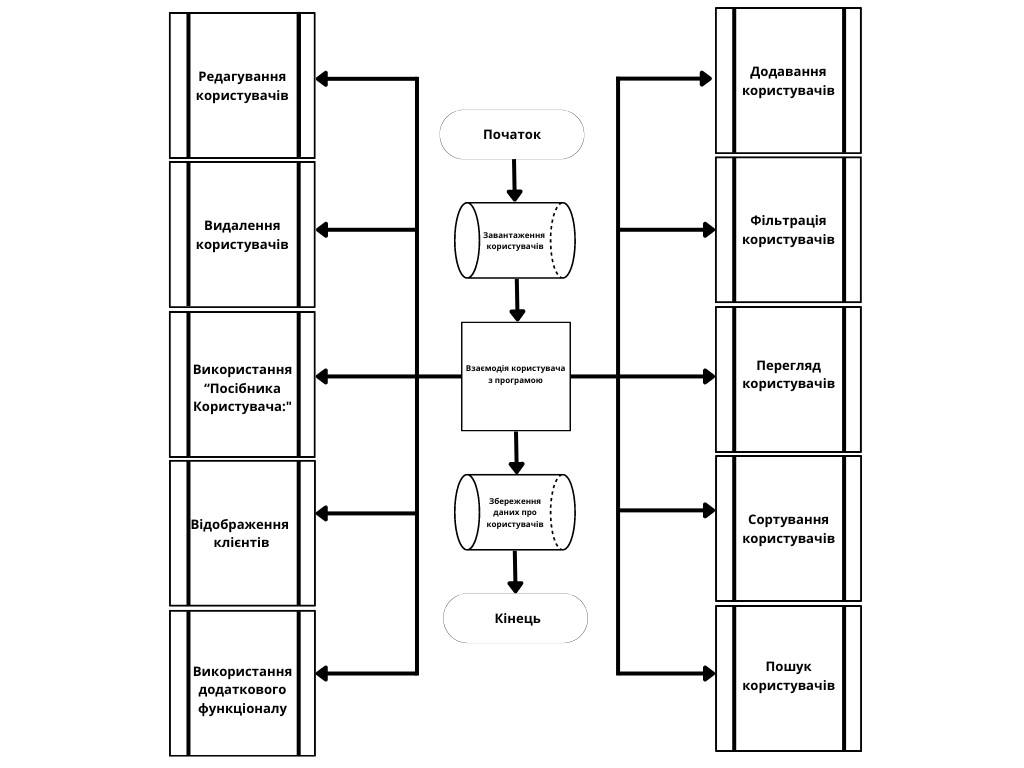


Рисунок 1 – Узагальнена блок-схема взаємодії користувача із програмою

## **2.2 Опис методів програми**

Список методів класу UserAddWindow та їх опис наведено в табл.1

Таблиця 1 – Основні методи класу

| № | Назва методу | Короткий опис |
| --- | --- | --- |
| 1. | AddUser(); | Цей метод викликається при спробі додати нового користувача. Він отримує дані користувача з поля введення, валідує їх, а потім зберігає користувача, якщо вони є дійсними. Якщо дані недійсні, виводиться повідомлення про помилку. |
| 2. | ClearFields() | Цей метод очищає всі поля введення у вікні додавання користувача. |
| 3. | GetUser() | Цей метод отримує дані з поля введення, створює об'єкт користувача на основі цих даних і повертає його. Цей об'єкт потім використовується для подальшої обробки, валідації та збереження. |

| № | Назва методу | Короткий опис |
| --- | --- | --- |
| 1. | ReturnToMainMenu\_Click(object sender, RoutedEventArgs e) | Це подія, яка відбувається при натисканні кнопки "Повернутися до головного меню". Вона створює новий екземпляр головного вікна MainWindow, показує його та закриває поточне вікно UserGuideWindow. |
| 2. | OneStepGuide() до FiveStepGuide(): | Ці приватні методи створюють новий екземпляр вікна кроку посібника (наприклад, OneStepGuide, TwoStepGuide тощо), показують його та закривають поточне вікно UserGuideWindow. |

Список методів класу UserGuideWindow та їх опис наведено в табл. 2

Таблиця 2 – Основні методи класу

Список методів класу ViewUsers та їх опис наведено в табл.3

Таблиця 3 – Основні методи класу

| № | Назва методу | Короткий опис |
| --- | --- | --- |
| 1. | ReadAllLines() | Цей метод зчитує всі рядки з файлу "users.txt" і зберігає їх у масиві lines. |
| 2. | PopulateListBox(string[] lines) | Цей метод очищує список користувачів (userListBox) і заповнює його рядками, які зчитані з файлу. |
| 3. | SortUsers(string[] filteredLines, bool ascending) | Цей метод сортує відфільтровані рядки користувачів. Він використовує метод Array.Sort() для сортування за зростанням або за спаданням, в залежності від значення параметру ascending. Після сортування список користувачів оновлюється за допомогою методу PopulateListBox(). |
| 4. | FilterUsers() | Цей метод фільтрує користувачів згідно з вибраними параметрами (характер знайомства, ділові якості, стать). Після фільтрації список користувачів оновлюється за допомогою методу PopulateListBox(). |

Список методів класу FileStaticService та їх опис наведено в табл.4

Таблиця 4 – Основні методи класу

| № | Назва методу | Короткий опис |
| --- | --- | --- |
| 1. | SaveUser(User user) | Цей метод приймає об'єкт користувача User і зберігає його дані у файлі. Він використовує StreamWriter для запису даних користувача у текстовий файл. Інформація про користувача записується у вигляді рядка, де кожен параметр розділений комою. Метод також повертає рядок із підтвердженням успішного додавання користувача або повідомлення про помилку, якщо виникла. |
| 2. | DeleteUser(int index) | Цей метод приймає індекс користувача, якого потрібно видалити з файлу. Він спочатку зчитує всі рядки з файлу, потім видаляє рядок, що відповідає вказаному індексу. Після цього оновлений список рядків записується назад у файл. Метод також повертає рядок із підтвердженням успішного видалення користувача або повідомлення про помилку, якщо виникла. |

Список методів класу FileStaticService та їх опис наведено в табл.5

Таблиця 5 – Основні методи класу

| № | Назва методу | Короткий опис |
| --- | --- | --- |
| 1. | ValidateUser(User user) | Цей метод приймає об'єкт користувача User і перевіряє, чи всі поля коректно заповнені. Якщо яке-небудь поле порожнє, метод повертає відповідне повідомлення про те, що всі поля повинні бути заповнені. Потім метод перевіряє правильність формату телефонного номера та дати народження, викликаючи відповідні приватні методи. Якщо дані відповідають всім критеріям, метод повертає порожній рядок, що означає, що дані користувача є коректними. |
| 2. | ValidatePhoneNumber(string phoneNumber) | Цей приватний статичний метод приймає рядок з телефонним номером і перевіряє його формат. Телефонний номер вважається правильним, якщо він починається з "+", за яким слідує 12 цифр. |
| 3. | ValidateBirthDate(string birthDate) | Цей приватний статичний метод приймає рядок з датою народження і перевіряє його формат. Дата народження вважається правильною, якщо вона відповідає формату "dd.MM.yyyy" і може бути успішно розпізнана за допомогою методу DateTime.TryParseExact. |

## 2.3 Програмні засоби

Розробка реалізована засобами середовища JetBrains Rider на мові програмування C# з використанням технологій WPF

JetBrains Rider - це інтегроване середовище розробки (IDE) від компанії JetBrains, яке використовується для створення програмного забезпечення для різних платформ.

Windows – це операційна система, орієнтована на керування за допомогою графічних інструментів. Також широкого поширення набула через її повноту взаємодії з компонентами комп’ютера. Оскільки операційна система націлена на графічний метод керування, широкого поширення набули настільні додатки, та інше додаткове ПЗ.

## 

## 2.4 Опис користувацького інтерфейсу

В головному меню додатку відображається список користувачів, методи впорядкування, поле пошуку по довільному ключу.

Також є навігаційна панель з вікнами, що містять інший функціонал, пов’язаний з списком користувачів, а також посібник.

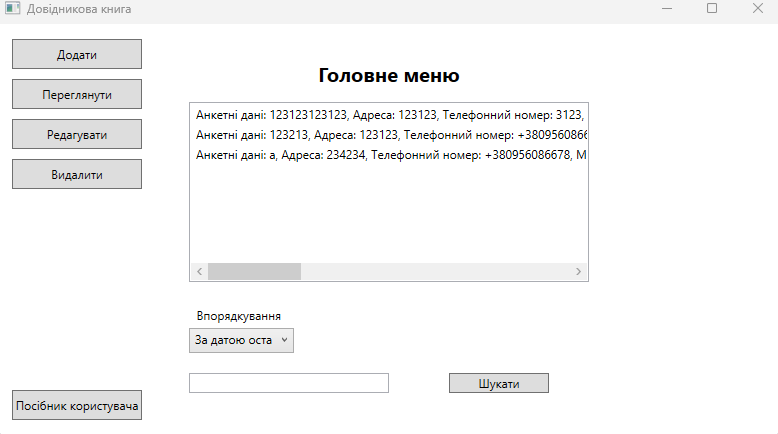


Рисунок 2 – Головне меню

Щоб добавити користувача - потрібно дотримуватись правил коректності введення значень в кожному полі під кожен тип поля, що потребує власний формат вводу. Після заповнення всіх полів, щоб добавити користувача - потрібно натиснути кнопку “Додати”, або “Очистити”, щоб очистити всі поля для вводу, а також є кнопка “Назад”, щоб повернутись в головне меню.

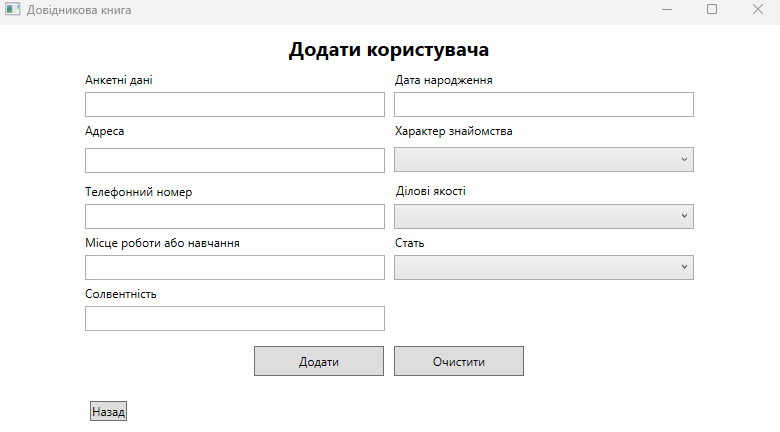


Рисунок 3 – Додати користувача

Користувачу в цьому вікні доступний перегляд користувачів без сортування(в початковому стані), або з сортуванням по вибраним параметрам.

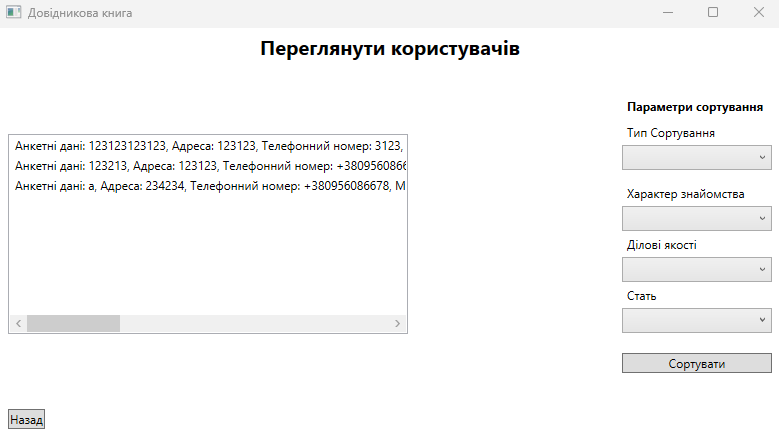


Рисунок 4 – Переглянути користувачів

В даному вікні користувачу доступний функціонал для того, щоб знайти користувача по номеру після введення номеру в полі та нажавши “Знайти” - після цієї дії заповняться даними користувача пусті поля, які можна редагувати та нажати кнопку “Зберегти” для того, щоб зберегти введені дані.

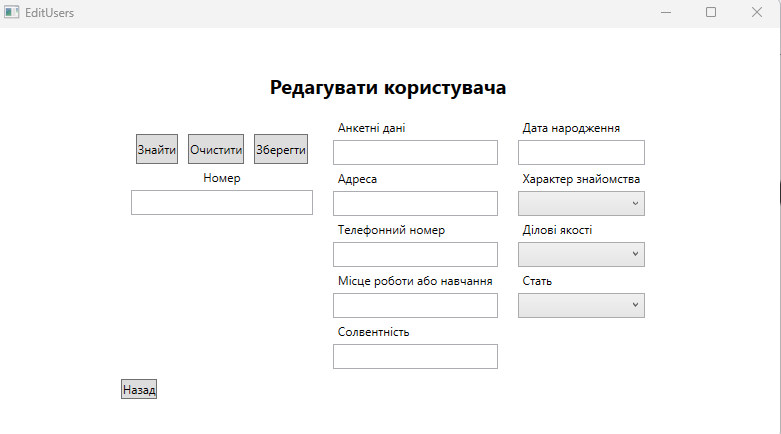


Рисунок 5 – Редагувати користувачів

В цьому вікні можна видалити користувача по його номеру, заповнивши поле та нажати кнопку “Видалити”

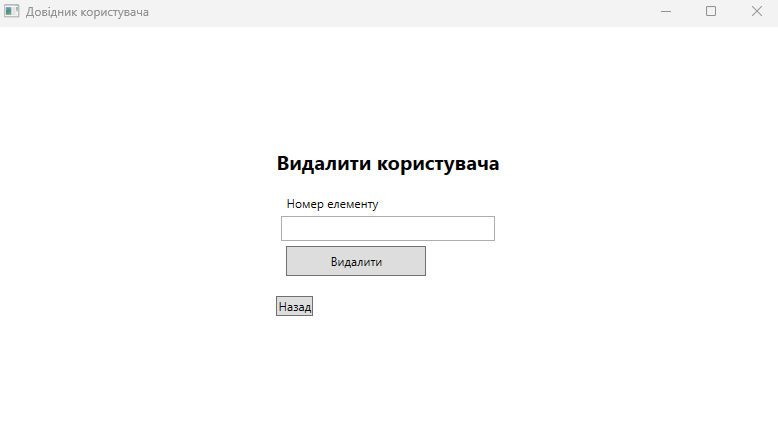


Рисунок 6 – Видалити користувача

Посібник користувача представляє навігаційну панель з інструкцією, яка пояснює як користуватись кожною сторінкою, що має функціонал з маніпуляцією списку користувачів

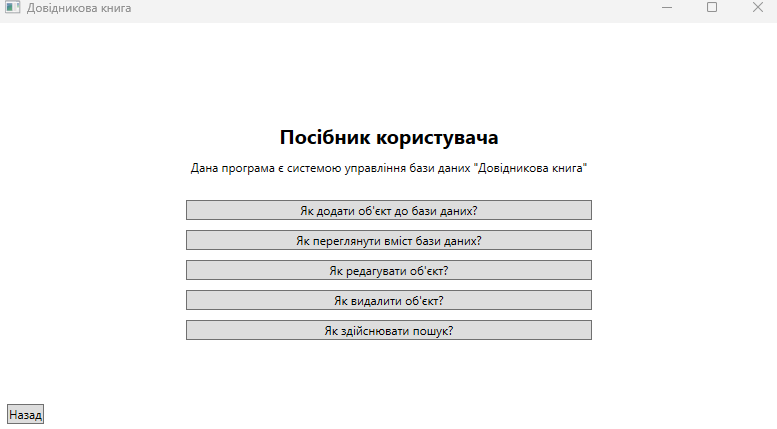


Рисунок 7 – Посібник користувача

# ВИСНОВКИ

В результаті виконання курсового проекту розроблена система управління

базою даних «Довідникова Книга» для ОС Windows.

Розроблене програмне забезпечення має наступні можливості:

1. Можливість додавання, видалення, редагування користувачів.
2. Фільтрація, сортування та пошук серед користувачів в базі даних.
3. Перегляд користувачів за допомогою різних типів впорядкування.
4. Перегляд посібника користувача.

В даній роботі було застосовано технології WPF для створення графічних інтерфейсів користувача (GUI) у програмах для операційної системи Windows. Наведена розробка у майбутньому може бути розширена із додаванням нового функціоналу.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. <https://learn.microsoft.com/en-us/dotnet/csharp/linq/>
2. <https://learn.microsoft.com/en-us/dotnet/csharp/>
3. [https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic?view=net-8.0](https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic?view=net-7.0)

# ДОДАТКИ

## ДОДАТОК А. Скролінг (текст) програми

using System.Windows;

using YourAppName;

namespace kursova.Base;

public class BaseView : Window

{

public void ReturnToMainMenu\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

namespace kursova;

/// <summary>

/// Interaction logic for App.xaml

/// </summary>

public partial class App

{

}

using System.Windows;

using kursova.Base;

using kursova.services.StaticServices;

namespace kursova

{

public partial class DeleteUser

{

public DeleteUser()

{

InitializeComponent();

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

if (int.TryParse(ProfileDataTextBox.Text.Trim(), out int index))

{

var result = FileStaticService.DeleteUser(index);

if (!string.IsNullOrEmpty(result))

{

MessageBox.Show(result);

}

}

else

{

MessageBox.Show("Введіть коректний номер користувача для видалення.");

}

}

}

}

using System.IO;

using System.Windows;

using System.Windows.Controls;

namespace kursova

{

public partial class EditUsers

{

private string filePath = "users.txt";

private int currentIndex = -1;

public EditUsers()

{

InitializeComponent();

}

// Метод для очищення полів введення

private void ClearFields()

{

ProfileDataTextBox.Clear();

AddressTextBox.Clear();

PhoneNumberTextBox.Clear();

WorkplaceOrEducationTextBox.Clear();

RelationshipTextBox.SelectedIndex = -1;

BusinessQualitiesTextBox.SelectedIndex = -1;

GenderComboBox.SelectedIndex = -1;

Birth.Clear();

Solvency.Clear();

}

// Метод для збереження оновлених даних у файлі

private void SaveDataToFile()

{

if (currentIndex >= 0)

{

try

{

string[] lines = File.ReadAllLines(filePath);

if (currentIndex < lines.Length)

{

lines[currentIndex] = $"{ProfileDataTextBox.Text.Trim()}, Адреса: {AddressTextBox.Text.Trim()}, Телефонний номер: {PhoneNumberTextBox.Text.Trim()}, Місце роботи або навчання: {WorkplaceOrEducationTextBox.Text.Trim()}, Солвентність: {Solvency.Text.Trim()}, Дата народження: {Birth.Text.Trim()}, Характер знайомства: {((ComboBoxItem)RelationshipTextBox.SelectedItem).Content}, Ділові якості: {((ComboBoxItem)BusinessQualitiesTextBox.SelectedItem).Content}, Стать: {((ComboBoxItem)GenderComboBox.SelectedItem).Content}";

File.WriteAllLines(filePath, lines);

MessageBox.Show("Дані успішно збережено у файлі.");

}

}

catch (Exception ex)

{

MessageBox.Show("Помилка при збереженні даних у файлі: " + ex.Message);

}

}

}

// Метод для пошуку користувача за індексом

private void FindUser(int index)

{

if (File.Exists(filePath))

{

try

{

string[] lines = File.ReadAllLines(filePath);

if (index >= 0 && index < lines.Length)

{

string userData = lines[index];

string[] userDataArray = userData.Split(',');

ProfileDataTextBox.Text = userDataArray[0].Split(':')[1].Trim();

AddressTextBox.Text = userDataArray[1].Split(':')[1].Trim();

PhoneNumberTextBox.Text = userDataArray[2].Split(':')[1].Trim();

WorkplaceOrEducationTextBox.Text = userDataArray[3].Split(':')[1].Trim();

Solvency.Text = userDataArray[4].Split(':')[1].Trim();

Birth.Text = userDataArray[5].Split(':')[1].Trim();

RelationshipTextBox.SelectedItem = RelationshipTextBox.Items.Cast<ComboBoxItem>()

.FirstOrDefault(item => item.Content.ToString() == userDataArray[6].Substring(userDataArray[6].IndexOf(':') + 1).Trim());

BusinessQualitiesTextBox.SelectedItem = BusinessQualitiesTextBox.Items.Cast<ComboBoxItem>()

.FirstOrDefault(item => item.Content.ToString() == userDataArray[7].Substring(userDataArray[7].IndexOf(':') + 1).Trim());

GenderComboBox.SelectedItem = GenderComboBox.Items.Cast<ComboBoxItem>()

.FirstOrDefault(item => item.Content.ToString() == userDataArray[8].Substring(userDataArray[8].IndexOf(':') + 1).Trim());

currentIndex = index;

}

else

{

MessageBox.Show("Немає користувача з таким індексом.");

}

}

catch (Exception ex)

{

MessageBox.Show("Помилка: " + ex.Message);

}

}

else

{

MessageBox.Show("Файл користувачів не знайдено.");

}

}

// Обробник події натискання кнопки "Пошук"

private void SearchButton\_Click(object sender, RoutedEventArgs e)

{

if (int.TryParse(NumberTextBox.Text.Trim(), out int index))

{

FindUser(index);

}

else

{

MessageBox.Show("Введіть коректний номер користувача.");

}

}

// Обробник події натискання кнопки "Очистити"

private void ClearFields\_Click(object sender, RoutedEventArgs e)

{

ClearFields();

}

// Обробник події натискання кнопки "Зберегти"

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

SaveDataToFile();

}

}

}

using System.IO;

using System.Windows;

using System.Windows.Controls;

using kursova;

using kursova.models.Models;

namespace YourAppName

{

public partial class MainWindow

{

private List<User> users = new List<User>();

public MainWindow() {

InitializeComponent();

LoadDataFromFile();

}

private void LoadDataFromFile() {

string filePath = "users.txt";

try

{

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

// Припустимо, що User - це ваш клас користувача з методом ToString(), який повертає інформацію про користувача у вигляді рядка

UserListBox.Items.Add(line);

}

}

catch (IOException ex)

{

MessageBox.Show($"Помилка зчитування файлу: {ex.Message}");

}

}

private void arrangement\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

string selectedSortingOption = ((ComboBoxItem)arrangement.SelectedItem)?.Content?.ToString();

if (selectedSortingOption == "За алфавітом")

{

users = users.OrderBy(user => user.ProfileData).ToList();

}

else if (selectedSortingOption == "За датою останнього коригування")

{

// Впорядковуємо за допомогою додаткового поля, наприклад, DataPublication

users = users.OrderBy(user => user.DataPublication).ToList();

}

// Очищаємо ListBox перед оновленням даних

UserListBox.Items.Clear();

// Додаємо відсортовані дані до ListBox

foreach (var user in users)

{

// Додати рядок, який відображатиме користувача у форматі, який вам потрібен

UserListBox.Items.Add($"{user.ProfileData}, {user.Address}, {user.PhoneNumber}, {user.WorkplaceOrEducation}, {user.Solvency}, {user.Birth}, {user.Relationship}, {user.BusinessQualities}, {user.Gender}");

}

}

private void ViewUsers() {

// Створюємо новий об'єкт вікна UserGuideWindow

ViewUsers ViewUsers = new ViewUsers();

// Показуємо вікно UserGuideWindow

ViewUsers.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void EditUsers() {

// Створюємо новий об'єкт вікна UserGuideWindow

EditUsers EditUsers = new EditUsers();

// Показуємо вікно UserGuideWindow

EditUsers.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void DeleteUsers() {

// Створюємо новий об'єкт вікна UserGuideWindow

DeleteUser DeleteUser = new DeleteUser();

// Показуємо вікно UserGuideWindow

DeleteUser.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void UserGuideButton\_Click(object sender, RoutedEventArgs e) {

// Створюємо новий об'єкт вікна UserGuideWindow

UserGuideWindow userGuideWindow = new UserGuideWindow();

// Показуємо вікно UserGuideWindow

userGuideWindow.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void UserAddButton\_Click(object sender, RoutedEventArgs e) {

// Створюємо новий об'єкт вікна UserGuideWindow

UserAddWindow UserAddWindow = new UserAddWindow();

// Показуємо вікно UserGuideWindow

UserAddWindow.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void SearchButton\_Click(object sender, RoutedEventArgs e) {

string searchInput = SearchTextBox.Text;

arrangement\_SelectionChanged(null, null);

// Очищаємо ListBox перед пошуком нових результатів

UserListBox.Items.Clear();

try

{

// Зчитуємо дані з файлу users.txt та створюємо об'єкти користувачів

string filePath = "users.txt";

string[] lines = File.ReadAllLines(filePath);

users = new List<User>();

foreach (string line in lines)

{

string[] userData = line.Split(','); // Зміна роздільника на кому

User user = new User

{

ProfileData = userData[0],

Address = userData[1],

PhoneNumber = userData[2],

WorkplaceOrEducation = userData[3],

Solvency = userData[4],

Birth = userData[5],

Relationship = userData[6],

BusinessQualities = userData[7],

Gender = userData[8]

};

users.Add(user);

}

// Виконуємо пошук серед користувачів за введеними критеріями

var searchResults = users.Where(user =>

user.ProfileData.Contains(searchInput) ||

user.Address.Contains(searchInput) ||

user.PhoneNumber.Contains(searchInput) ||

user.WorkplaceOrEducation.Contains(searchInput) ||

user.Solvency.Contains(searchInput) ||

user.Birth.Contains(searchInput) ||

user.Relationship.Contains(searchInput) ||

user.BusinessQualities.Contains(searchInput) ||

user.Gender.Contains(searchInput)).ToList();

// Отримуємо вибраний елемент з ComboBox

string selectedSortingOption = ((ComboBoxItem)arrangement.SelectedItem)?.Content?.ToString();

// Впорядковуємо дані відповідно до вибору користувача

if (selectedSortingOption == "За алфавітом")

{

searchResults = searchResults.OrderBy(user => user.ProfileData).ToList();

}

else if (selectedSortingOption == "За датою останнього коригування")

{

// Впорядковуємо за допомогою додаткового поля, наприклад, DataPublication

searchResults = searchResults.OrderBy(user => user.DataPublication).ToList();

}

// Додаємо результати пошуку до ListBox

foreach (var user in searchResults)

{

UserListBox.Items.Add(user.ProfileData + user.Address + user.PhoneNumber + user.WorkplaceOrEducation + user.Solvency + user.Birth + user.Relationship + user.BusinessQualities + user.Gender);

}

}

catch (IOException ex)

{

MessageBox.Show($"Помилка зчитування файлу: {ex.Message}");

}

}

private void ViewUsers\_Click(object sender, RoutedEventArgs e) {

ViewUsers(); // Викликаємо функцію перегляду користувачів

}

private void EditUsers\_Click(object sender, RoutedEventArgs e) {

EditUsers(); // Викликаємо функцію перегляду користувачів

}

private void DeleteUsers\_Click(object sender, RoutedEventArgs e) {

DeleteUsers(); // Викликаємо функцію перегляду користувачів

}

}

}

using System.Windows;

using System.Windows.Controls;

using kursova.Base;

using kursova.models.Models;

using kursova.services.StaticServices;

using kursova.services.StaticServices.Validators;

namespace kursova

{

public partial class UserAddWindow

{

public UserAddWindow()

{

InitializeComponent();

}

private void AddUser()

{

var user = GetUser();

var validation = ValidatorStaticService.ValidateUser(user);

if (!string.IsNullOrEmpty(validation))

{

MessageBox.Show(validation);

}

else

{

// Записуємо дані користувача у текстовий файл без індексу

var result = FileStaticService.SaveUser(user);

MessageBox.Show(result);

// Очищаємо поля введення після додавання користувача

ClearFields();

}

}

private void AddUser\_Click(object sender, RoutedEventArgs e)

{

AddUser(); // Викликаємо функцію додавання користувача

}

private void ClearFields()

{

// Очищаємо всі поля введення

ProfileDataTextBox.Clear();

AddressTextBox.Clear();

PhoneNumberTextBox.Clear();

WorkplaceOrEducationTextBox.Clear();

RelationshipTextBox.SelectedIndex = -1;

BusinessQualitiesTextBox.SelectedIndex = -1;

GenderComboBox.SelectedIndex = -1;

Birth.Clear();

Solvency.Clear();

}

private void ClearFields\_Click(object sender, RoutedEventArgs e)

{

ClearFields(); // Викликаємо функцію очищення полів введення

}

private User GetUser()

{

// Отримуємо дані з полів введення

string profileData = ProfileDataTextBox.Text;

string address = AddressTextBox.Text;

string phoneNumber = PhoneNumberTextBox.Text;

string workplaceOrEducation = WorkplaceOrEducationTextBox.Text;

string relationship = RelationshipTextBox.Text;

string businessQualities = BusinessQualitiesTextBox.Text;

string gender = ((ComboBoxItem)GenderComboBox.SelectedItem)?.Content?.ToString();

string birth = Birth.Text;

string solvency = Solvency.Text;

// Створюємо нового користувача з отриманими даними

User newUser = new User

{

ProfileData = profileData,

Address = address,

PhoneNumber = phoneNumber,

WorkplaceOrEducation = workplaceOrEducation,

Relationship = relationship,

BusinessQualities = businessQualities,

Gender = gender,

Birth = birth,

Solvency = solvency,

DataPublication = DateTime.Now.ToString("dd.MM.yyyy") // Додаємо поточну дату публікації

};

return newUser;

}

}

}

using System.Windows;

namespace kursova;

public partial class FiveStepGuide : Window

{

public FiveStepGuide()

{

InitializeComponent();

}

public void ReturnToUserGuide\_Click(object sender, RoutedEventArgs e)

{

// Створюємо новий екземпляр MainWindow

UserGuideWindow UserGuideWindow = new UserGuideWindow();

// Показуємо нове головне вікно

UserGuideWindow.Show();

// Закриваємо поточне вікно (UserAddWindow)

this.Close();

}

}

using System.Windows;

namespace kursova;

public partial class FourStepGuide : Window

{

public FourStepGuide()

{

InitializeComponent();

}

public void ReturnToUserGuide\_Click(object sender, RoutedEventArgs e)

{

// Створюємо новий екземпляр MainWindow

UserGuideWindow UserGuideWindow = new UserGuideWindow();

// Показуємо нове головне вікно

UserGuideWindow.Show();

// Закриваємо поточне вікно (UserAddWindow)

this.Close();

}

}

using System.Windows;

namespace kursova;

public partial class OneStepGuide : Window

{

public OneStepGuide()

{

InitializeComponent();

}

public void ReturnToUserGuide\_Click(object sender, RoutedEventArgs e)

{

// Створюємо новий екземпляр MainWindow

UserGuideWindow UserGuideWindow = new UserGuideWindow();

// Показуємо нове головне вікно

UserGuideWindow.Show();

// Закриваємо поточне вікно (UserAddWindow)

this.Close();

}

}

using System.Windows;

namespace kursova;

овне вікно

UserGuideWindow.Show();

// Закриваємо поточне вікно (UserAddWindow)

this.Close();

}

}

using System.Windows;

namespace kursova;

public partial class TwoStepGuide : Window

{

public TwoStepGuide()

{

InitializeComponent();

}

public void ReturnToUserGuide\_Click(object sender, RoutedEventArgs e)

{

// Створюємо новий екземпляр MainWindow

UserGuideWindow UserGuideWindow = new UserGuideWindow();

// Показуємо нове головне вікно

UserGuideWindow.Show();

// Закриваємо поточне вікно (UserAddWindow)

this.Close();

}

}

using System.Windows;

using System.Windows.Controls;

using YourAppName;

namespace kursova

{

public partial class UserGuideWindow

{

public UserGuideWindow()

{

InitializeComponent();

}

public void ReturnToMainMenu\_Click(object sender, RoutedEventArgs e)

{

// Створюємо новий екземпляр MainWindow

MainWindow mainWindow = new MainWindow();

// Показуємо нове головне вікно

mainWindow.Show();

// Закриваємо поточне вікно (UserGuideWindow)

this.Close();

}

private void OneStepGuide\_Click(object sender, RoutedEventArgs e) {

OneStepGuide(); // Викликаємо функцію перегляду користувачів

}

private void TwoStepGuide\_Click(object sender, RoutedEventArgs e) {

TwoStepGuide(); // Викликаємо функцію перегляду користувачів

}

private void ThreeStepGuide\_Click(object sender, RoutedEventArgs e) {

ThreeStepGuide(); // Викликаємо функцію перегляду користувачів

}

private void FourStepGuide\_Click(object sender, RoutedEventArgs e) {

FourStepGuide(); // Викликаємо функцію перегляду користувачів

}

private void FiveStepGuide\_Click(object sender, RoutedEventArgs e) {

FiveStepGuide(); // Викликаємо функцію перегляду користувачів

}

private void OneStepGuide() {

// Створюємо новий об'єкт вікна UserGuideWindow

OneStepGuide OneStepGuide = new OneStepGuide();

// Показуємо вікно UserGuideWindow

OneStepGuide.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void TwoStepGuide() {

// Створюємо новий об'єкт вікна UserGuideWindow

TwoStepGuide TwoStepGuide = new TwoStepGuide();

// Показуємо вікно UserGuideWindow

TwoStepGuide.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void ThreeStepGuide() {

// Створюємо новий об'єкт вікна UserGuideWindow

ThreeStepGuide ThreeStepGuide = new ThreeStepGuide();

// Показуємо вікно UserGuideWindow

ThreeStepGuide.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void FourStepGuide() {

// Створюємо новий об'єкт вікна UserGuideWindow

FourStepGuide FourStepGuide = new FourStepGuide();

// Показуємо вікно UserGuideWindow

FourStepGuide.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

private void FiveStepGuide() {

// Створюємо новий об'єкт вікна UserGuideWindow

FiveStepGuide FiveStepGuide = new FiveStepGuide();

// Показуємо вікно UserGuideWindow

FiveStepGuide.Show();

// Закриваємо поточне вікно (MainWindow)

this.Close();

}

}

}

using System.IO;

using System.Windows;

using System.Windows.Controls;

using kursova.Base;

using YourAppName;

namespace kursova

{

public partial class ViewUsers

{

private string[] lines;

public ViewUsers()

{

InitializeComponent();

ReadAllLines();

PopulateListBox(lines);

}

private void ReadAllLines()

{

// Зчитуємо всі рядки з файлу

string filePath = "users.txt";

lines = File.ReadAllLines(filePath);

}

private void PopulateListBox(string[] lines)

{

// Очищаємо попередні елементи списку

userListBox.Items.Clear();

// Додаємо кожний рядок (користувача) до ListBox

foreach (string line in lines)

{

userListBox.Items.Add(line);

}

}

private void SearchButton\_Click(object sender, RoutedEventArgs e)

{

// Перевіряємо, чи вибрано значення у комбо-боксі "Тип Сортування"

if (SortComboBox.SelectedIndex == -1)

{

// Фільтруємо користувачів

FilterUsers();

return; // Припиняємо виконання коду сортування

}

// Створюємо копію оригінального масиву lines

string[] filteredLines = new string[lines.Length];

Array.Copy(lines, filteredLines, lines.Length);

// Фільтр за характером знайомства

if (RelationshipTextBox.SelectedItem != null)

{

string selectedRelationship = ((ComboBoxItem)RelationshipTextBox.SelectedItem).Content.ToString();

filteredLines = filteredLines.Where(line => line.Contains(selectedRelationship)).ToArray();

}

// Фільтр за діловими якостями

if (BusinessQualitiesTextBox.SelectedItem != null)

{

string selectedBusinessQuality = ((ComboBoxItem)BusinessQualitiesTextBox.SelectedItem).Content.ToString();

filteredLines = filteredLines.Where(line => line.Contains(selectedBusinessQuality)).ToArray();

}

// Фільтр за статтю

if (GenderComboBox.SelectedItem != null)

{

string selectedGender = ((ComboBoxItem)GenderComboBox.SelectedItem).Content.ToString();

filteredLines = filteredLines.Where(line => line.Contains(selectedGender)).ToArray();

}

// Отримуємо тип сортування (за зростанням або за спаданням)

bool ascending = SortComboBox.SelectedIndex == 0; // 0 - За зростанням, 1 - За спаданням

// Сортуємо відфільтрованих користувачів

SortUsers(filteredLines, ascending);

}

private void SortUsers(string[] filteredLines, bool ascending)

{

Array.Sort(filteredLines, StringComparer.CurrentCulture);

if (!ascending)

{

Array.Reverse(filteredLines);

}

PopulateListBox(filteredLines);

}

private void FilterUsers()

{

// Створюємо копію оригінального масиву lines

string[] filteredLines = new string[lines.Length];

Array.Copy(lines, filteredLines, lines.Length);

// Фільтр за характером знайомства

if (RelationshipTextBox.SelectedItem != null)

{

string selectedRelationship = ((ComboBoxItem)RelationshipTextBox.SelectedItem).Content.ToString();

filteredLines = filteredLines.Where(line => line.Contains(selectedRelationship)).ToArray();

}

// Фільтр за діловими якостями

if (BusinessQualitiesTextBox.SelectedItem != null)

{

string selectedBusinessQuality = ((ComboBoxItem)BusinessQualitiesTextBox.SelectedItem).Content.ToString();

filteredLines = filteredLines.Where(line => line.Contains(selectedBusinessQuality)).ToArray();

}

// Фільтр за статтю

if (GenderComboBox.SelectedItem != null)

{

string selectedGender = ((ComboBoxItem)GenderComboBox.SelectedItem).Content.ToString();

filteredLines = filteredLines.Where(line => line.Contains(selectedGender)).ToArray();

}

// Оновлюємо дані для відображення в ListBox

PopulateListBox(filteredLines);

}

}

}

namespace kursova.models.Models;

public class User

{

public string ProfileData { get; set; }

public string Address { get; set; }

public string PhoneNumber { get; set; }

public string WorkplaceOrEducation { get; set; }

public string Solvency { get; set; }

public string Birth { get; set; }

public string Relationship { get; set; }

public string BusinessQualities { get; set; }

public string Gender { get; set; }

public string DataPublication { get; set; }

}

using kursova.models.Models;

namespace kursova.services.StaticServices;

public static class FileStaticService

{

private const string \_path = "users.txt";

public static string SaveUser(User user)

{

try

{

using StreamWriter writer = new StreamWriter(\_path, true);

writer.WriteLine(

$"Анкетні дані: {user.ProfileData}, " +

$"Адреса: {user.Address}, " +

$"Телефонний номер: {user.PhoneNumber}, " +

$"Місце роботи або навчання: {user.WorkplaceOrEducation}, " +

$"Солвентність: {user.Solvency}, " +

$"Дата народження: {user.Birth}, " +

$"Характер знайомства: {user.Relationship}, " +

$"Ділові якості: {user.BusinessQualities}, " +

$"Стать: {user.Gender}, " +

$"Дата останнього коригування: {DateTime.Now}");

return "Новий користувач успішно доданий та записаний у файл.";

}

catch (Exception e)

{

return "Помилка під час запису.";

}

}

public static string DeleteUser(int index)

{

try

{

// Зчитуємо всі рядки з файлу

string[] lines = File.ReadAllLines(\_path);

// Видаляємо користувача зі списку за вказаним індексом

if (index >= 0 && index < lines.Length)

{

lines = lines.Where((source, i) => i != index).ToArray();

// Записуємо оновлений список користувачів назад у файл

File.WriteAllLines(\_path, lines);

return "Користувача видалено успішно.";

}

else

{

return "Немає користувача з таким індексом.";

}

}

catch (Exception ex)

{

return "Помилка під час запису.";

}

}

}

using System.Globalization;

using System.Text.RegularExpressions;

using kursova.models.Models;

namespace kursova.services.StaticServices.Validators;

public static class ValidatorStaticService

{

public static string ValidateUser(User user)

{

// Перевіряємо коректність введених даних

if (string.IsNullOrEmpty(user.ProfileData) ||

string.IsNullOrEmpty(user.Address) ||

string.IsNullOrEmpty(user.PhoneNumber) ||

string.IsNullOrEmpty(user.WorkplaceOrEducation) ||

string.IsNullOrEmpty(user.Relationship) ||

string.IsNullOrEmpty(user.BusinessQualities) ||

string.IsNullOrEmpty(user.Gender) ||

string.IsNullOrEmpty(user.Birth) ||

string.IsNullOrEmpty(user.Solvency)

)

{

return "Будь ласка, заповніть всі поля.";

}

// Перевіряємо телефонний номер

if (!ValidatePhoneNumber(user.PhoneNumber))

{

return "Неправильний формат телефонного номера. Введіть номер у форматі +380123456789.";

}

// Перевіряємо дату народження

if (!ValidateBirthDate(user.Birth))

{

return "Неправильний формат дати народження. Введіть дату у форматі dd.MM.yyyy (наприклад, 01.01.2000).";

}

return string.Empty;

}

private static bool ValidatePhoneNumber(string phoneNumber)

{

// Перевірка формату телефонного номера (наприклад, формат +380123456789)

if (!Regex.IsMatch(phoneNumber, @"^\+\d{12}$"))

{

return false;

}

return true;

}

private static bool ValidateBirthDate(string birthDate)

{

// Перевірка формату дати народження (наприклад, формат 01.01.2000)

if (!DateTime.TryParseExact(birthDate, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None,

out \_))

{

return false;

}

return true;

}

}