**Міністерство освіти і науки України**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

**«ДНІПРОВСЬКА ПОЛІТЕХНІКА»**



**Факультет інформаційних технологій**

**Кафедра програмного забезпечення комп'ютерних систем**

Мартиненко А.А.

Гуліна І.Г.

**ЗВІТ**

**до виконання навчальної практики бакалаврами**

**галузі знань 12 Інформаційні технології**

**спеціальності 122 Комп’ютерні науки**

Студентки 2 курсу групи 122-24-1

Спецівльності «Комп’ютерні науки»

Кравцової В.Є.

**Дніпро**

**НТУ «ДП»**

**2025**

**Пояснювальна записка:** 26 стор., 8 рис., 8 додатків, 5 джерел.

**Об’єкт дослідження:** процес розробки програмного забезпечення для керування обліком користувачів у настільному WPF-застосунку.

**Предмет дослідження:** методи проєктування, розробки та тестування програмного забезпечення для збереження, редагування та обробки даних користувачів із використанням мови С#, технологій WPF та реляційної бази даних(SQLite).

**Мета роботи:** створення зручного, інтуїтивно зрозумілого та функціонального настільного застосунку **UsersApp**, який забезпечує ефективне управління даними користувачів, включаючи їх додавання, видалення, редагування та збереження у базі даних.

**Методи дослідження.** Для розробки програмного забезпечення використано середовище **Microsoft Visual Studio** та мову програмування **C#** із застосуванням фреймворку **.NET** та технології **Windows Presentation Foundation (WPF)**. Для доступу до бази даних використано **Entity Framework** (ORM) з реляційною базою даних **Microsoft SQL Server**. Проєктування інтерфейсу виконано з урахуванням принципів MVVM-патерну та сучасних підходів до UX/UI-дизайну. Тестування проводилося за допомогою модульних тестів, інтеграційного тестування та перевірки юзабіліті.

**Наукова новизна** полягає у створенні WPF-застосунку, який поєднує сучасний підхід до побудови інтерфейсу (MVVM) з гнучким доступом до даних через ORM, забезпечуючи масштабованість та адаптивність системи для різних сфер використання.

**Практична цінність роботи** полягає у тому, що розроблений застосунок може бути використаний як готове рішення для ведення обліку користувачів у внутрішніх системах підприємств, навчальних закладів або організацій, а також легко модифікуватися під інші предметні області.

**Список ключових слів:** UsersApp, C#, WPF, Entity Framework, SQL Server, SQLite, MVVM, облік користувачів, база даних, програмне забезпечення.

**Зміст**

1. Титульний аркуш ......................................................... .1
2. Реферат .......................................................................... 2
3. Зміст ............................................................................... 3
4. Вступ .............................................................................. 4
5. Завдання ......................................................................... 5
6. Призначення розробки та галузь застосування .......... 6
7. Функціональне призначення програми .......................6
8. Вимоги до програми або програмного виробу.............6
9. Вимоги до функціональних характеристик .................7
10. Вимоги до складу та параметрів технічних засобів ....7
11. Загальна частина ............................................................ 8  
      11.1 Огляд та структура проєкту................................... 8  
      11.2. Використані технології та бібліотеки ................. 9  
      11.3 Алгоритм роботи ............................................ ....... 10  
      11.4 Анімація............................. .............. ...................... 12  
      11.5 Використання EF Core та ADO.NET......................13  
      11.6 Переваги обраної архітектури................................ 13  
      11.7 Обгрунтування та організація вхідних та вихідних даних програми.................................................................................. 14
12. Висновки ........................................................................ 14
13. Список використаних джерел ...................................... 15
14. Додатки .......................................................................... 15

## ****ВСТУП****

У сучасному цифровому суспільстві питання безпечної, зручної та ефективної реєстрації й авторизації користувачів набуває особливої актуальності. Більшість сучасних додатків, незалежно від своєї сфери застосування, потребують реалізації механізмів, які дозволяють чітко ідентифікувати користувача, забезпечувати збереження його особистих даних та надавати персоналізований доступ до ресурсів. Саме тому створення програмного забезпечення, яке реалізує цей процес, є важливим завданням у галузі розробки інформаційних систем.

Вибір технології **Windows Presentation Foundation (WPF)** обумовлений необхідністю створення сучасного, інтуїтивно зрозумілого інтерфейсу користувача. WPF надає широкі можливості для роботи з графічними елементами, стилями, шаблонами, анімаціями та підтримує розділення логіки програми (C#) від її візуальної частини (XAML), що підвищує гнучкість та масштабованість розробки.

Проєкт **UsersApp** орієнтований на демонстрацію повного циклу обробки користувацьких даних:

* створення нових записів у базі (реєстрація);
* перевірка введених облікових даних при вході;
* відображення інтерфейсу відповідно до статусу користувача;
* анімаційний супровід переходів між сторінками, що покращує візуальне сприйняття додатку.

При розробці особливу увагу було приділено організації коду та архітектурі програми. Логіка роботи з даними зосереджена у класах, що реалізують модель та доступ до бази (через DbContext EF Core), тоді як відображення та взаємодія з користувачем реалізовані у XAML-інтерфейсі та обробниках подій на C#. Такий підхід відповідає принципам патерну MVVM, що рекомендується для WPF-додатків.

Окремим аспектом стало впровадження візуальних ефектів та анімацій, які роблять взаємодію користувача з програмою більш комфортною. Для головної сторінки було реалізовано анімаційні ефекти на C#, які програмно змінюють властивості елементів інтерфейсу. У випадку сторінки авторизації, частина анімацій була створена безпосередньо у XAML, що дозволяє досягти більшої декларативності та зручності в налаштуванні інтерфейсу без додаткового кодування.

Тобто, мій проект є прикладом комплексної розробки, що поєднує:

* використання сучасних UI-технологій (**WPF**);
* застосування ORM для зручної роботи з базами даних (**Entity Framework Core**);
* створення реляційної бази на основі **SQLite**;
* поділ логіки та інтерфейсу відповідно до сучасних архітектурних принципів;
* реалізацію елементів анімації та ефектів для покращення UX.

Таким чином, **UsersApp** демонструє, як за допомогою сучасних технологій можна створити повноцінну настільну програму, яка забезпечує безпечну роботу з даними користувачів, привабливий інтерфейс і зручний користувацький досвід. Крім того, цей проєкт може слугувати основою для розширення функціоналу, наприклад, додавання ролей користувачів, інтеграції з віддаленими сервісами чи впровадження більш складних механізмів захисту даних.

**ЗАВДАННЯ**

Метою даного проєкту є розробка настільного WPF-додатку **UsersApp**, який забезпечує повний цикл роботи з обліковими записами користувачів: від їхньої реєстрації з внесенням у базу даних, до авторизації з перевіркою введених облікових даних. Для зберігання інформації використовується реляційна база даних **SQLite**, яка забезпечує компактність, автономність та простоту розгортання. Для роботи з базою даних застосовується сучасний ORM-фреймворк **Entity Framework Core**, що дозволяє взаємодіяти з даними у вигляді об’єктів і значно спрощує розробку та підтримку проекту.  
Основні завдання:

* створення інтуїтивного графічного інтерфейсу з використанням технології **WPF**;
* реалізація механізму аутентифікації користувачів;
* організація зберігання та обробки інформації за допомогою **SQLite**;
* забезпечення мінімальних апаратних вимог та стабільності роботи програми.

**ПРИЗНАЧЕННЯ РОЗРОБКИ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ**

Програма призначена для локального використання на персональних комп’ютерах з операційною системою Windows. Сфера застосування - це невеликі організації, навчальні заклади або індивідуальні користувачі, які потребують простої системи для управління обліковими записами.  
Продукт також може бути використаний як навчальний матеріал для студентів-програмістів, що вивчають мови програмування C#, технологію WPF та основи роботи з базами даних.

**ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ ПРОГРАМИ**

* реєстрація нових користувачів із внесенням основних даних;
* авторизація користувачів через форму входу;
* можливість перегляду профілю користувача після входу;
* збереження даних у базі **UsersAppDatabase.db**;
* розмежування вікон: вікно авторизації, головне вікно, сторінка користувача.

Продукт розрахований на роботу в локальному середовищі без підключення до мережі.

**ВИМОГИ ДО ПРОГРАМИ**

* Сумісність з операційними системами Windows 10 та вище.
* Наявність встановленого **.NET 6.0 або вище**.
* Можливість запуску як окремого виконуваного файлу без додаткових інструментів.
* Простота інсталяції: програма запускається без складного налаштування середовища.

**ВИМОГИ ДО ФУНКЦІОНАЛЬНИХ ХАРАКТЕРИСТИК**

* Час запуску програми - не більше 3 секунд на середньостатистичному ПК.
* Час відповіді на дії користувача - миттєвий (не більше 0,5 сек).
* Обробка даних здійснюється локально без затримок.
* Система підтримує одночасну роботу лише одного користувача, що відповідає концепції програми.

**ВИМОГИ ДО СКЛАДУ ТА ПАРАМЕТРІВ ТЕХНІЧНИХ ЗАСОБІВ**

* Мінімальні системні вимоги:
  + ОС: Windows 10/11;
  + Оперативна пам’ять: 2 ГБ і більше;
  + Процесор: 1,5 ГГц і вище;
  + Дисковий простір: 50 МБ;
  + Екран: мінімальна роздільна здатність 1280x720.
* Рекомендовані системні вимоги:
  + ОЗП: 4 ГБ;
  + Процесор: 2,5 ГГц і вище;
  + Дисковий простір: 100 МБ.

**ЗАГАЛЬНА ЧАСТИНА**

Проєкт UsersApp - це настільний додаток на C# з використанням Windows Presentation Foundation (WPF) та бази даних SQLite. Програма реалізує базову систему авторизації та реєстрації користувачів. Код побудований із застосуванням підходів об'єктно-орієнтованого програмування та принципів MVVM-патерну.

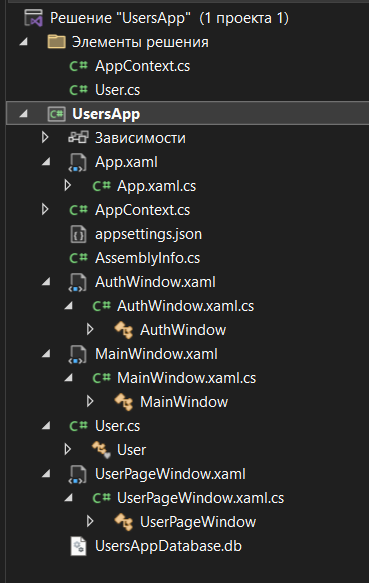


Рисунок 11.1 Структура проекту

1. **App.xaml / App.xaml.cs**
   * Файл налаштувань додатку та його точка входу.
   * Описує глобальні ресурси додатку та визначає логіку запуску в App.xaml.cs.
2. **AppContext.cs**
   * Клас для підключення до бази даних через **Entity Framework Core**.
   * Містить контекст бази даних та таблиці (DbSet), наприклад для користувачів.
3. **User.cs**
   * Модель користувача, яка визначає поля користувача (ID, логін, пароль).
   * Використовується для збереження та отримання даних із бази.
4. **AuthWindow.xaml / AuthWindow.xaml.cs**
   * Вікно авторизації користувача.
   * У XAML розміщено інтерфейс (поля введення логіна та пароля, кнопки входу/реєстрації).
   * У файлі .xaml.cs - логіка авторизації та перевірки даних.
5. **MainWindow.xaml / MainWindow.xaml.cs**
   * Головне вікно додатку після входу користувача.
6. **UserPageWindow.xaml / UserPageWindow.xaml.cs**
   * Окреме вікно для перегляду профілю користувача.
7. **UsersAppDatabase.db**
   * Файл бази даних SQLite, де зберігаються зареєстровані користувачі.
8. **appsettings.json**
   * Файл налаштувань додатку.
9. **AssemblyInfo.cs**
   * Інформація про збірку додатку (версія, авторські дані тощо).

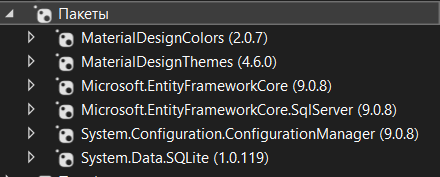


Рисунок 11.2 Використані технології та бібліотеки

* **MaterialDesignColors (2.0.7)**  - кольорові схеми.
* **MaterialDesignThemes (4.0.6)** - розробка графічного інтерфейсу.
* **Microsoft.EntityFrameworkCore (9.0.8)** - ORM для роботи з БД.
* **Microsoft.EntityFrameworkCore.SqlServer (9.0.8)** - провайдер EF Core.
* **System.Configuration.ConfigurationManager (**9.0.8) - читання конфігурації.
* **System.Data.SQLite (1.0.119) -** драйвер для SQLite.

**Алгоритм роботи:**

1. **Запуск програми**
   * Після запуску додатку користувач потрапляє у вікноавторизації **(AuthWindow)**

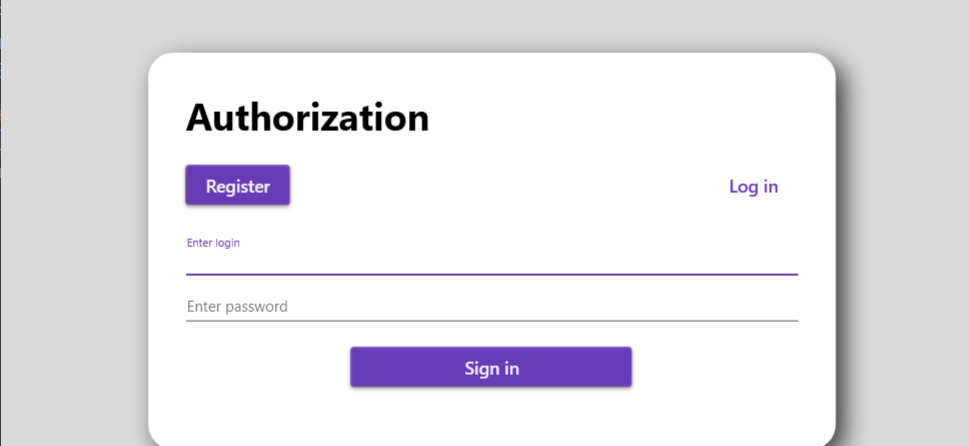


Рисунок 11.3 Вікноавторизації

* + Є два основних варіанти дій:
    - **Увійти в обліковий запис** (якщо вже зареєстрований).
    - **Створити новий обліковий запис**.

1. **Реєстрація нового користувача**
   * У вікні авторизації потрібно натиснути кнопку “Реєстрація.
   * Ввести логін та пароль.
   * Дані зберігаються в локальній базі даних SQLite.
   * Після успішної реєстрації користувач може виконати вхід у свій профіль.

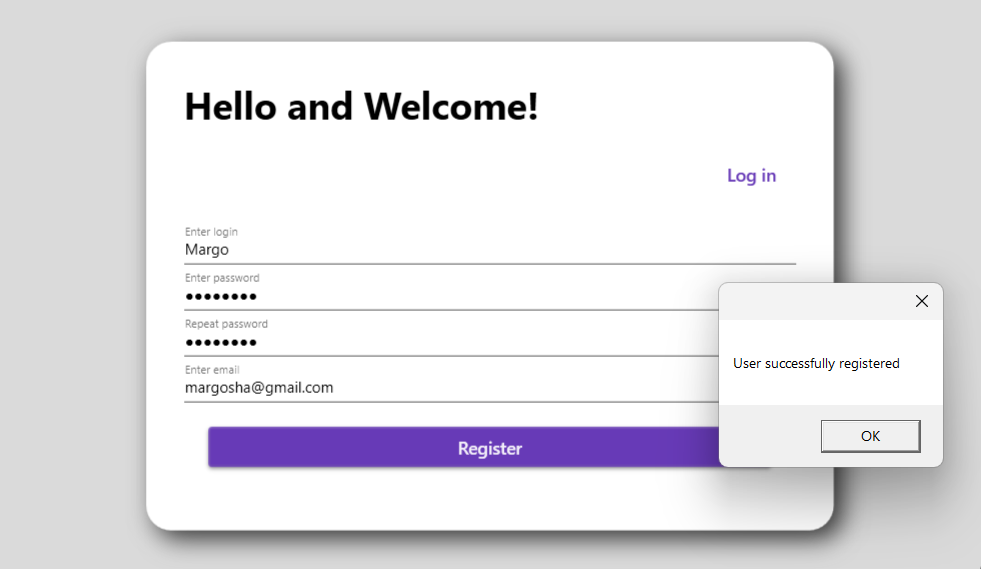


Рисунок 11.4: ВікноРеєстрації

1. **Авторизація користувача**
   * Користувач вводить свої дані (логін і пароль).
   * Програма перевіряє їх у базі даних.
   * Якщо дані коректні, відкривається **MainWindow** або **UserPageWindow**, де можна переглянути профіль користувача.

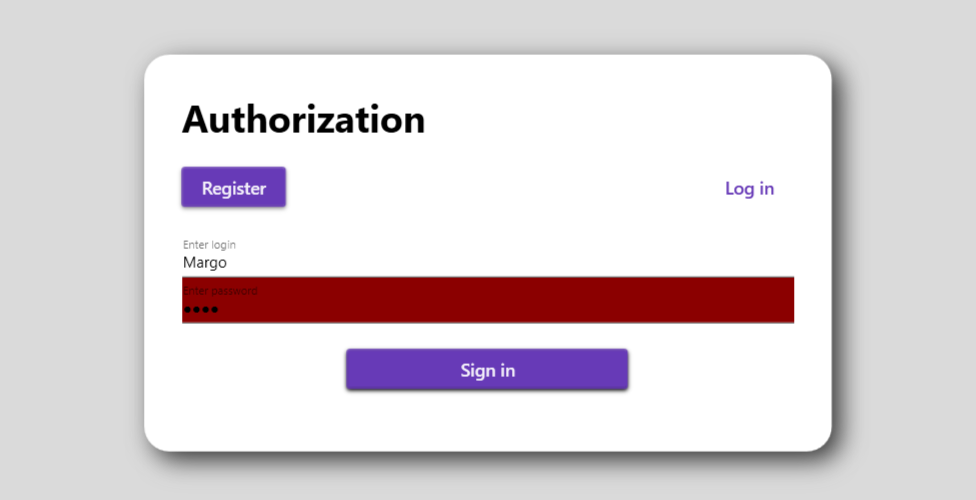


Рисунок 11.5 Помилка при авторизації через некоректно введенмй пароль

1. **Головне вікно (MainWindow)**
   * Відображає основну інформацію користувача та надає доступ до додаткових функцій.
   * Може містити анімацію та елементи інтерфейсу, що роблять програму зручною.
2. **Робота з профілем користувача (UserPageWindow)**
   * Користувач може переглянути свої дані.

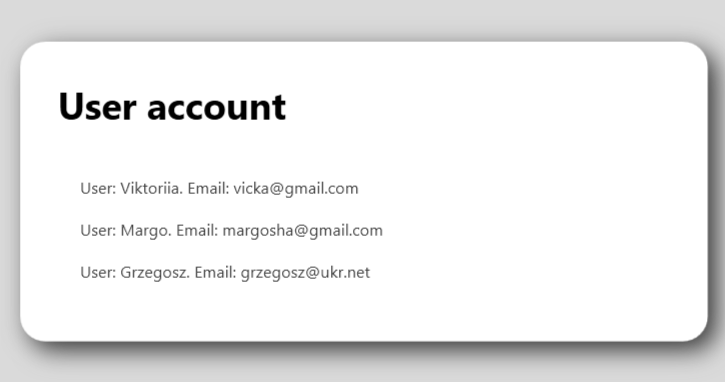


Рисунок 11.6 Аккаунт користувача

1. **База даних (UsersAppDatabase.db)**
   * Усі облікові записи зберігаються локально у файлі SQLite.
   * Це робить додаток автономним та простим у розгортанні.

**Анімація**

**У цьому проекті я використала два варіанти анімації кнопок.**

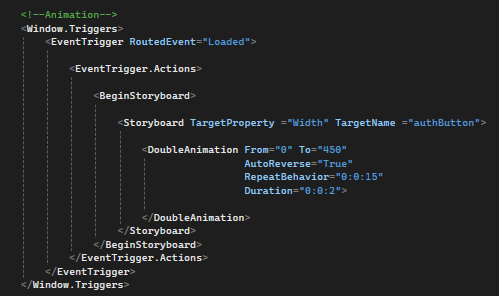
****

Рисунок 11.7 Анімація у вікніавторизації

**XAML-анімація** задається декларативно у розмітці, виконується автоматично після завантаження елемента.

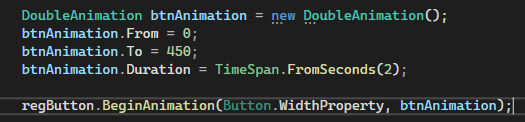


Рисунок 11.8 Анімація у головному вікні

**C#-анімація** (using System.Windows.Media.Animation;) створюється програмно, що дозволяє керувати її параметрами під час виконання (наприклад, змінювати швидкість або напрям).

**Використання EF Core та ADO.NET**

**EF Core** у цьому проєкті бере на себе:

* створення таблиць у SQLite;
* генерацію SQL-запитів;
* маппінг об’єктів C# на записи у БД.

**ADO.NET** використовується неявно — саме через нього EF Core взаємодіє з драйвером SQLite.  
Це дає змогу уникнути написання ручних SQL-запитів, але при потребі їх можна реалізувати напряму.

**Переваги обраної архітектури**

* **Мінімальна кількість коду для роботи з БД** завдяки EF Core.
* **Модульність** - окремі вікна виконують свої функції.
* **Масштабованість** - можна додати інші ролі користувачів або додаткові поля без значних змін у коді.
* **Зручний UI** - WPF дозволяє створювати сучасний інтерфейс із анімаціями.

**Обгрунтування та організація вхідних та вихідних даних програми**

* Вхідні дані: логін і пароль користувача при авторизації, персональні дані при реєстрації профілю.
* Вихідні дані: інформація про обліковий запис користувача, повідомлення про успіх або помилки операцій.
* Дані зберігаються у форматі SQLite і не потребують додаткових драйверів.

**ВИСНОВКИ**

У процесі роботи було створено настільний додаток **UsersApp**, який реалізує безпечну реєстрацію та авторизацію користувачів. Використання WPF дало змогу створити інтуїтивно зрозумілий інтерфейс, SQLite забезпечила зручне зберігання даних, а EF Core простоту роботи з базою.  
Програма може бути розширена додаванням функцій відновлення пароля, ролей користувачів та інтеграції з іншими сервісами.

Розроблений програмний продукт **UsersApp** є повноцінним прикладом настільного додатку з використанням технології **WPF** та роботи з базами даних за допомогою **ADO.NET**. У ході виконання цього проєкту були реалізовані всі ключові етапи створення програмного забезпечення - від аналізу вимог і проектування інтерфейсу до безпосередньої розробки, тестування та оптимізації роботи додатку.

Основною метою проєкту було створення зручного, надійного та інтуїтивно зрозумілого інструменту для управління даними користувачів, який би дозволяв виконувати операції **додавання та пошуку записів** у базі даних. У процесі розробки вдалося досягти поставленої мети, а реалізований функціонал повністю відповідає вимогам, сформованим на етапі постановки задачі.

Однією з ключових особливостей даного проєкту стало використання **WPF** (Windows Presentation Foundation) як платформи для побудови інтерфейсу. Це рішення забезпечило гнучкість у налаштуванні зовнішнього вигляду додатку та можливість реалізації складних графічних елементів, зокрема анімацій, стилів та шаблонів. Наприклад, анімації підвищують візуальну привабливість програми та роблять взаємодію користувача більш комфортною.

Для обробки та збереження даних була використана бібліотека **ADO.NET,** яка дозволила організувати надійний канал зв’язку між додатком та базою даних. За допомогою об’єктів **SqlConnection, SqlCommand, SqlDataAdapter** та **DataTable** було реалізовано виконання SQL-запитів, завантаження та оновлення даних у таблицях. Такий підхід забезпечив контрольовану роботу з базою даних, а також дозволив уникнути надмірного навантаження на систему завдяки вибірковому завантаженню даних.

Важливим аспектом розробки стала **архітектурна структура коду**. Логіка взаємодії з базою даних була винесена у окремі методи, що підвищує зручність підтримки та розширення програми в майбутньому. Інтерфейс користувача був побудований у файлах XAML, що дозволило відокремити візуальну частину від бізнес-логіки та дотримуватися принципів **MVVM-підходу.**

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Microsoft Docs - WPF Documentation: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/>
2. Microsoft Docs - Entity Framework Core: https://learn.microsoft.com/ru-ru/ef/core/
3. Офіційний сайт SQLite: https://www.sqlite.org/
4. <https://metanit.com/sql/sqlite/1.1.php>
5. <https://metanit.com/sharp/wpf/>

**ДОДАТКИ**

Файл **AppContext.cs** - клас для роботи з базою даних **SQLite,** встановлює підключення

using System;

using System.Data.Entity;

namespace UsersApp

{

class AppContext : DbContext

{

public AppContext() : base("DefaultConnection")

{

Database.SetInitializer(new DropCreateDatabaseIfModelChanges<AppContext>());

}

public DbSet<User> Users { get; set; }// подключение к БД

}

}

Файл **User.cs** - модель користувача, яка описує поля та властивості користувача

using System;

namespace UsersApp

{

class User

{

public int id { get; set; }

private string login, password, email;

public string Login

{

get { return login; }

set { login = value; }

}

public string Password

{

get { return password; }

set { password = value; }

}

public string Email

{

get { return email; }

set { email = value; }

}

public User() { }

public User(string login, string password, string email)

{

this.login = login;

this.email = email;

this.password = password;

}

//чтобы понять что за пользователи

public override string ToString() //переписываем метод ToString

{

// return base.ToString();

return "User: " + Login + ". Email: " + Email;

}

}

}

Файл **AuthWindow.xaml.cs** - вікно авторизації користувачів (С#)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace UsersApp

{

/// <summary>

/// Логика взаимодействия для AuthWindow.xaml

/// </summary>

public partial class AuthWindow : Window

{

public AuthWindow()

{

InitializeComponent();

}

private void Button\_Auth\_Click(object sender, RoutedEventArgs e)

{

string login = textBoxLogin.Text.Trim(); //удаляет пробелы

string pass = passBox.Password.Trim();

if (login.Length < 5)

{

textBoxLogin.ToolTip = "The login must contain more than 5 characters.";

textBoxLogin.Background = Brushes.DarkRed;

}

else if (pass.Length < 5)

{

passBox.ToolTip = "The password must contain more than 5 characters.";

passBox.Background = Brushes.DarkRed;

}

else

{

textBoxLogin.ToolTip = "";

textBoxLogin.Background = Brushes.Transparent;

passBox.ToolTip = "";

passBox.Background = Brushes.Transparent;

User authUser = null;

using (AppContext db = new AppContext())

{

//находим пользователя с подходязими данными

//authUser = db.Users.Where(b => b.Login == login && b.Password == pass).FirstOrDefault();

authUser = db.Users.FirstOrDefault(user => user.Login == login && user.Password == pass);

if (authUser != null)

{

// MessageBox.Show("User successfully registered");

//Переход на окно

UserPageWindow userPageWindow = new UserPageWindow();

userPageWindow.Show();

this.Hide();

}

else

MessageBox.Show("Unknown user. Sign up or link your account first.");

}

}

}

private void Button\_Click\_Reg(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Hide(); //скрыть текущее окно

}

private void Button\_Window\_Auth\_Click(object sender, RoutedEventArgs e)

{

AuthWindow authWindow = new AuthWindow();

authWindow.Show();

Hide(); //скрыть текущее окно

}

}

}

Файл **AuthWindow.xaml.cs** - вікно авторизації користувачів (XAML)

<Window x:Class="UsersApp.AuthWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:UsersApp"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

Title="Users App" Height="450" Width="800">

<!--Animation-->

<Window.Triggers>

<EventTrigger RoutedEvent="Loaded">

<EventTrigger.Actions>

<BeginStoryboard>

<Storyboard TargetProperty ="Width" TargetName ="authButton">

<DoubleAnimation From="0" To="450"

AutoReverse="True"

RepeatBehavior="0:0:15"

Duration="0:0:2">

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</EventTrigger.Actions>

</EventTrigger>

</Window.Triggers>

<Grid Background="#ffdadada">

<Border MinWidth="400" MaxWidth="550" Margin="15" Background="White" VerticalAlignment="Center" Padding="30" CornerRadius="20" >

<Border.Effect>

<DropShadowEffect BlurRadius="30" Color="Gray" ShadowDepth="10" />

</Border.Effect>

<StackPanel>

<TextBlock Text="Authorization" FontSize="30" FontWeight="Bold" Margin="0 0 0 20" />

<Grid Margin="0 0 0 20">

<Button HorizontalAlignment="Left" Content="Register" Click="Button\_Click\_Reg" />

<Button HorizontalAlignment="Right" Content="Log in" Click="Button\_Window\_Auth\_Click" Style="{StaticResource MaterialDesignFlatButton}" />

</Grid>

<TextBox x:Name="textBoxLogin" materialDesign:HintAssist.Hint="Enter login" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

<PasswordBox x:Name="passBox" materialDesign:HintAssist.Hint="Enter password" Style="{StaticResource MaterialDesignFloatingHintPasswordBox}"/>

<Button Content="Sign in" x:Name="authButton" Margin="0 20" Click="Button\_Auth\_Click"/>

</StackPanel>

</Border>

</Grid>

</Window>

Файл **MainWindow.xaml.cs** - головне вікно програми, яке відображається після входу (С#)

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Media.Animation;

namespace UsersApp

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

AppContext db;

public MainWindow()

{

InitializeComponent();

db = new AppContext();

DoubleAnimation btnAnimation = new DoubleAnimation(); // обьект для создания анимации using System.Windows.Media.Animation;

btnAnimation.From = 0; //первичное состояние

btnAnimation.To = 450; // куда движется

btnAnimation.Duration = TimeSpan.FromSeconds(2); // время на анимацию

regButton.BeginAnimation(Button.WidthProperty, btnAnimation); //(какое свойство меняем, анимация)

}

//Переход на окно

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

{

string login = textBoxLogin.Text.Trim(); //удаляет пробелы

string pass = passBox.Password.Trim();

string pass\_2 = passBox\_2.Password.Trim();

string email = textBoxEmail.Text.Trim().ToLower(); //в нижний регистр

if (login.Length < 5)

{

textBoxLogin.ToolTip = "The login must contain more than 5 characters.";

textBoxLogin.Background = Brushes.DarkRed;

}

else if (pass.Length < 5)

{

passBox.ToolTip = "The password must contain more than 5 characters.";

passBox.Background = Brushes.DarkRed;

}

else if (pass != pass\_2)

{

passBox\_2.ToolTip = "Incorrect input.";

passBox\_2.Background = Brushes.DarkRed;

}

else if (email.Length < 5 || !email.Contains("@") || !email.Contains("."))

{

textBoxEmail.ToolTip = "Incorrect input.";

textBoxEmail.Background = Brushes.DarkRed;

}

else

{

textBoxLogin.ToolTip = "";

textBoxLogin.Background = Brushes.Transparent;

passBox.ToolTip = "";

passBox.Background = Brushes.Transparent;

passBox\_2.ToolTip = "";

passBox\_2.Background = Brushes.Transparent;

textBoxEmail.ToolTip = "";

textBoxEmail.Background = Brushes.Transparent;

MessageBox.Show("User successfully registered");

User user = new User(login, pass, email);

db.Users.Add(user);

db.SaveChanges();

AuthWindow authWindow = new AuthWindow();

authWindow.Show();

this.Hide();

}

}

private void Button\_Wind\_Auth\_Click(object sender, RoutedEventArgs e)

{

AuthWindow authWindow = new AuthWindow();

authWindow.Show();

Hide();

}

}

}

Файл **MainWindow.xaml** - головне вікно програми, яке відображається після входу (XAML)

<Window x:Class="UsersApp.MainWindow"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:UsersApp"

mc:Ignorable="d"

Title="Users App" Height="500" Width="800">

<Grid Background="#ffdadada">

<!--отступы внешн 15 внутр 30 и округление CornerRadius-->

<Border MinWidth="400" MaxWidth="550" Margin="15" Background="White" VerticalAlignment="Center" Padding="30" CornerRadius="20" >

<Border.Effect>

<DropShadowEffect BlurRadius="30" Color="Gray" ShadowDepth="10" />

</Border.Effect>

<StackPanel>

<TextBlock Text="Hello and Welcome!" FontSize="30" FontWeight="Bold" Margin="0 0 0 20" />

<Grid Margin="0 0 0 20">

<Button HorizontalAlignment="Right" Content="Log in" Click="Button\_Wind\_Auth\_Click" Style="{StaticResource MaterialDesignFlatButton}" />

</Grid>

<TextBox x:Name="textBoxLogin" materialDesign:HintAssist.Hint="Enter login" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

<PasswordBox x:Name="passBox" materialDesign:HintAssist.Hint="Enter password" Style="{StaticResource MaterialDesignFloatingHintPasswordBox}"/>

<PasswordBox x:Name="passBox\_2" materialDesign:HintAssist.Hint="Repeat password" Style="{StaticResource MaterialDesignFloatingHintPasswordBox}"/>

<TextBox x:Name="textBoxEmail" materialDesign:HintAssist.Hint="Enter email" Style="{StaticResource MaterialDesignFloatingHintTextBox}"/>

<Button Content="Register" x:Name="regButton" Margin="0 20" Click="Button\_Click"/>

</StackPanel>

</Border>

</Grid>

</Window>

Файл **UserPageWindow.xaml.cs** - вікно перегляду та редагування інформації про користувача (С#)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace UsersApp

{

/// <summary>

/// Логика взаимодействия для UserPageWindow.xaml

/// </summary>

public partial class UserPageWindow : Window

{

public UserPageWindow()

{

InitializeComponent();

// подключение бд и выборка записей из нее

AppContext db = new AppContext();

List<User> users = db.Users.ToList(); //преобразовать данные к списку

listOfUsers.ItemsSource = users;

}

}

}

Файл **UserPageWindow.xaml -** вікно перегляду та редагування інформації про користувача (XAML)

<Window x:Class="UsersApp.UserPageWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:UsersApp"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

mc:Ignorable="d"

Title="Users App" Height="350" Width="600">

<Grid Background="#ffdadada">

<!--отступы внешн 15 внутр 30 и округление CornerRadius-->

<Border MinWidth="400" MaxWidth="550" Margin="15" Background="White" VerticalAlignment="Center" Padding="30" CornerRadius="20" >

<Border.Effect>

<DropShadowEffect BlurRadius="30" Color="Gray" ShadowDepth="10" />

</Border.Effect>

<StackPanel>

<TextBlock Text="User account" FontSize="30" FontWeight="Bold" Margin="0 0 0 20" />

<ListView x:Name="listOfUsers" Margin="10" />

</StackPanel>

</Border>

</Grid>

</Window>