**Форма № Н-6.01у**

Кременчуцький національний університет імені Михайла Остроградського

(повне найменування вищого навчального закладу)

Кафедра автоматизації та інформаційних систем

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЄКТ**

**(РОБОТА)**

з дисципліни «Сучасні мови об’єктно-орієнтованого програмування»

(назва дисципліни)

на тему Розробка WPF додатку «Бюро знайомств»

Студента \_2\_ курсу КН–23–1 групи

Ступінь вищої освіти «Бакалавр»

(бакалавр, магістр)

Спеціальність 122 – «Комп’ютерні науки»

Освітньо-професійна програма

«Комп’ютерні науки»

Гур’єва Д.П.

(прізвище та ініціали)

Керівник старший викладач кафедри АІС

\_\_\_\_\_\_\_ Бельська В. Ю.\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_. Оцінка: ЄКTС \_\_\_\_

Члени комісії  . В. Ю. Бельська

(підпис) (ініціали та прізвище)

Н.М. Істоміна

(підпис) (ініціали та прізвище)

О.В.Кліменко

(підпис) (ініціали та прізвище)

м. Кременчук 2024 рік

**Форма № Н-9.01у**

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Кафедра автоматизації та інформаційних систем

Дисципліна «Сучасні мови об’єктно-орієнтованого програмувння»

Освітній ступінь \_\_ «Бакалавр»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність 122 – «Комп’ютерні науки»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Освітня програма «Комп’ютерні науки»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курс 2 група \_\_КН-23-1 семестр \_\_3\_\_

**ЗАВДАННЯ**

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Гур’єва Дмитра Петровича \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_

(прізвище, ім’я, по-батькові)

1. Тема роботи: Розробка WPF додатку «Бюро знайомств»
2. Термін здачі студентом роботи 07.12.2024 р
3. Вихідні дані до роботи: users.json (зберігає інформацію про користвувачів) cities.json(Зберігає інформацію про міста та посилання на чати)
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

постановка завдання, аналіз технічного завдання, розробка алгоритму роботи програми, опис моделі даних, структура програмного забезпечення, функціональна схема та опис моделей та моделей представлення програми.

1. Перелік графічного матеріалу:
2. Дата видачі завдання: 9.09. 2024 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  пор. | Назва етапів курсового проекту | Терміни виконання етапів проекту | Вказівки та  зауваження викладача (з зазначенням дати консультації) | Оцінювання етапів проекту | | |
| за націо- нальною шкалою | за шкалою ЄКТС | кількість балів |
| 1 | Етап 1  Аналіз предметної області | 09.11.2024-15.11.2024 |  |  | | |
| 2 | Етап 2  Створення моделі даних | 15.11.2024-21.11.2024 |  |  | | |
| 3 | Етап 3  Розробка віконного інтерфейсу та створення основного програмного коду додатку | 22.11.2024-04.12.2024 |  |  | | |
| 4 | Етап 4  Тестування програмного коду | 04.12.2024 |  |  | | |
| 5 | Етап 5  Оформлення пояснювальної записки | 07.12.2024 |  |  | | |
| 6 | Етап 9  Захист | 08.12.2024 |  |  | | |
|  | Разом | 8 тижнів |  |  | | |

Студент

(підпис)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ В. Ю. Бельська \_\_\_\_

(підпис) (ініціали та прізвище)

«6» грудня 2024 р.

РЕФЕРАТ

Курсова робота містить 29 сторінок, 2 розділи, 17 рисунків, 5 використаних джерел.

Об’єкт розробки – WPF додаток «Тема».

Мета: створення багатовіконного Wpf додатку, що реалізує знаходження чатів для спілкування по обраному місту.

У проекті розглянуті базові принципи об'єктно-орієнтованого програмування (ООП), проведений аналіз предметної області та сформовано технічне завдання. До функціональних вимог курсової роботи віднесені: можливість реєстрації та авторизації користувачів, вибору міста та ортримання чату, перевірка валідності файлів, збереження та віжтворення інформації з файлів, створення зручного та функціонального профілю. Усі функціональні вимоги були реалізовані у коді проекту.

Проект розроблено на мові програмування C#. Основним інструментом для створення графічного інтерфейсу використовувалась бібліотека WPF. Створення проекту відбувалось за шаблоном MVVM.

Основна увага у розробці проекту була зосереджена на моделі даних та моделях представлення. Але в проекті також реалізовано зручний інтерфейс для забезпечення комфортної роботи користувача. Прив’язка даних, що реалізована в програмному коді, є забезпеченням швидкодії обміну даними та зручним механізмом корегування та створення об’єктів моделі даних.

Як результат виконання курсової роботи є створений багатовіконний WPF додаток «Бюро знайомств».

ОБ’ЄКТ, ІНКАПСУЛЯЦІЯ, ІНТЕРФЕЙС, КЛАС

**Зміст**

[Вступ 2](#_Toc151681430)

[1 Опис предметної областІ 3](#_Toc151681431)

[1.1 У кожного студента буде індивідуальна тема для висвітлення 3](#_Toc151681432)

[1.2 Аналіз технічного завдання на роботу 3](#_Toc151681433)

[1.2.1 Функціональні вимоги 3](#_Toc151681434)

[1.2.2 Нефункціональні вимоги 3](#_Toc151681435)

[1.3 Опис алгоритму основних задач/підзадач у роботі 4](#_Toc151681436)

[1.3.1 Структура програмного забезпечення 5](#_Toc151681437)

[Висновки до розділу 5](#_Toc151681438)

[2 Опис розробки програмного забезпечення 6](#_Toc151681439)

[2.1 Структура програмного забезпечення 6](#_Toc151681440)

[2.2 Опис роботи програми 10](#_Toc151681441)

[2.3 Функціональна схема програми 12](#_Toc151681442)

[2.4 Опис інтерфейсу програми 13](#_Toc151681443)

[Висновки до розділу 23](#_Toc151681444)

[ВИСНОВКИ 24](#_Toc151681445)

[СПИСОК ЛІТЕРАТУРИ 25](#_Toc151681446)

[Додаток А XAML розмітки вікон 26](#_Toc151681447)

[Додаток Б C# коди класів вікон 56](#_Toc151681448)

[Додаток B C# основні класи 74](#_Toc151681449)

[Додаток Г Тестування додатку 80](#_Toc151681450)

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

*1*

*122 – КР.2024.01.000 ПЗ*

Розроб.

Гур’євД.П.

Перевір.

Бельська В. Ю О.

*ва*

Н. контр.

*Найда В. В.*

*Розробка WPF додатку “Бюро знайомств”*

*Пояснювальна записка*

Літ.

Аркушів

*29*

КрНУ Кафедра АІС

ВСТУП

С# є однією з найпопулярніших мов програмування в сучасному програмуванні. Вона володіє широким спектром застосувань, але однією з найцікавіших областей її використання є створення графічних інтерфейсів за допомогою платформи WPF (Windows Presentation Foundation). WPF - це технологія, яка надає потужні засоби для розробки візуально привабливих та функціональних програм для операційної системи Windows.

Однією з ключових переваг WPF є його розширені можливості для створення різноманітних програм. Він надає великий набір компонентів, включаючи не лише графічні елементи, а й засоби для роботи з даними, макетами, ресурсами та безпекою. У цій платформі прив'язка даних (data binding) виступає як потужний інструмент, що дозволяє зв'язувати дані програми з її інтерфейсом, спрощуючи процес роботи з інформацією.

Розробка програми для покупки авіабілетів через WPF є чудовим прикладом використання цієї технології. Вона дозволяє створити інтуїтивно зрозумілий та зручний інтерфейс для користувача, де можна легко відобразити різноманітну інформацію про рейси, доступні місця, вартість квитків та інші важливі дані. Завдяки можливостям WPF з прив'язкою даних, редагування та оновлення інформації про рейси стають зручними та ефективними для користувача.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Що таке дженерики? Які проблеми вони вирішують? Array, List, HashSet, Dictionary - складність операцій пошуку, вставки, видалення елементів

Різноманітні структури даних та алгоритми в програмуванні є невід'ємною складовою для розв'язання різноманітних завдань. Зокрема, коли йдеться про операції зі збереженням, пошуком, вставкою та видаленням елементів, важливо розглянути джерела, які пропонують оптимальні та ефективні методи вирішення таких завдань. У цьому контексті виникають дженерики, що надають універсальні рішення для різних типів даних та операцій.

Дженерики - це загальний механізм в багатьох мовах програмування, який дозволяє створювати загальні класи або методи, які працюють з будь-яким типом даних. Їх використання дозволяє писати більш універсальний код, який може працювати з різними типами об'єктів, зберігаючи при цьому високий рівень абстракції та перевикористовування.

Серед таких дженеричних структур даних можна виділити Array, List, HashSet та Dictionary. Array - це найпростіша структура даних, яка забезпечує доступ до елементів за індексом, але операції вставки та видалення можуть бути витратними через потребу у зсуві елементів. List надає більшу гнучкість, але відомий своєю лінійною складністю для операцій вставки та видалення в середині списку. HashSet і Dictionary, які базуються на хеш-таблицях, забезпечують оптимальну складність для операцій пошуку, вставки та видалення, оскільки вони використовують хеш-функції для швидкого доступу до даних.

Проте, важливо враховувати, що кожна з цих структур має свої сильні та слабкі сторони. Вибір між ними повинен базуватися на конкретних потребах задачі, яку необхідно вирішити. Іноді оптимальним рішенням буде використання комбінації різних структур даних для досягнення найкращої продуктивності в конкретному контексті програмування.

1.2 Аналіз технічного завдання на роботу

Мета курсового проекту – Розробити багатовіконний WPF додаток «Продаж квитків на поточні авіарейси».

Над цим проектом працювала група з двох студентів, тому необхідно використовувати шаблон MVVM, задля розмежування обов’язків при виконанні проекту. Ми розподілили таким чином: я працюю над моделями та моделями представлень, а колега працює над зовнішнім інтерфейсом.

Під час розробки моделей даних та графічного інтерфейсу були висунуті наступні вимоги

**1.2.1 Функціональні вимоги**

1. Можливість реєстрації та авторизації(користувач або адміністратор).
2. Видаляння або редагування профілю користувача.
3. Можливість зберігання даних користувачів.
4. Розмежування прав доступу: користувач має доступ до вікна свого профілю з можливістю зміни міста та отрмання нового чату, а також редагування та видалення його. Для адміністратора доступно додавати, редагувати, видаляти міста та їх посилання на чати, а також переглядати зареєстрованих користувачів.

**1.2.2 Нефункціональні вимоги**

1. Використання патерну MVVM.
2. Усі дані повинні зберігатися у json форматі.
3. Реалізація логіки моделей та моделей вигляду.
4. Забезпечити коректність даних
5. WPF середовище.

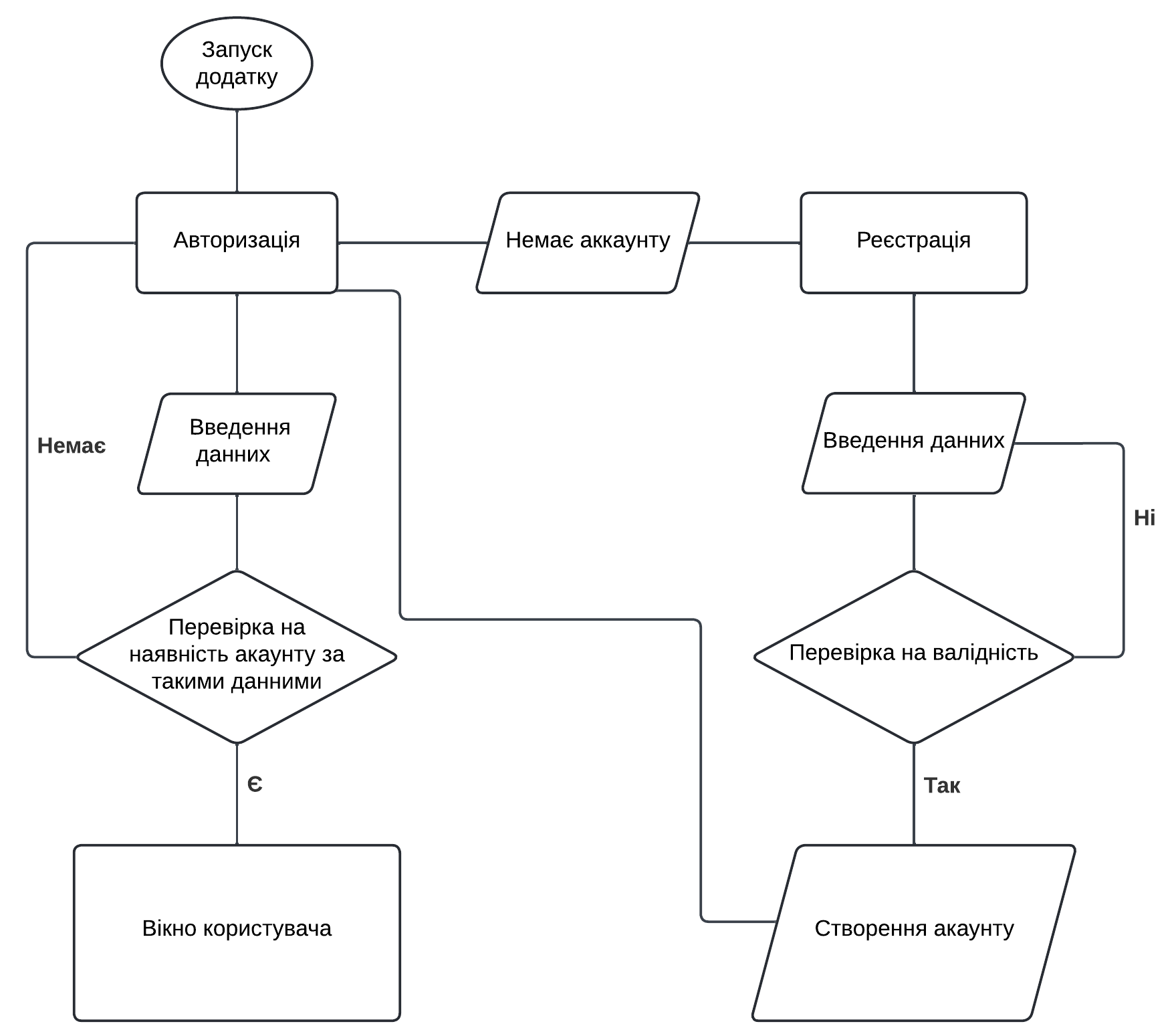
1.3 Опис алгоритму основних задач/підзадач у роботі

У рамках даної курсової роботи основною задачею є створення додатку, використовуючи патерн MVVM (рис. 1.1), так як цей додаток створюється групою, тому необхідне розмежування обов’язків. Моїми обов’язками в рамках даної роботи є створення моделей даних та моделей вигляду.



Мал. 1.1 Схема патерну MVVM

Мають бути реалізовані як мінімум 2 моделі: модель користувач, для того щоб працював механізм авторизації, реєстрації (рис 1.2) – якщо користувач має обліковий запис, то він вводить вхідні дані, з файлу зчитуються дані про усіх користувачів, якщо вхідні дані підібрано правильно, то користувач отримує доступ до основної частини, у випадку реєстрації вхідні дані користувача перевіряються на валідність, якщо всі данні введено коректно, створюється його обліковий запис та з’являється доступ до основної частини.



Мал. 1.2 Блок-схема реєстрації та авторизації

Повинно бути збереження даних у коректному вигляді, для чого потріно створити такі моделі: модель білету, без якої не можна буде реалізувати те, для чого цей додаток й створений – отримання чатів, та модель користувача, для реалізації механізмів з роботою з обліковим записом.

Висновки до розділу

У першому розділі розглянуто питання, що являють собою дженерики та їх використання, створена розширена постановка задачі, проаналізована предметна область, сформовано вимоги до функціональних можливостей програми та розглянуто технічне завдання. Складений план розробки програми та функції, що повинні бути реалізовані у програмі.

2 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Структура програмного забезпечення

Структура програмного забезпечення є поєднанням усіх застосованих модулів програми, що утворюють працюючу систему. Кожний модуль повинен реалізовувати функції програми, що вирішують основні завдання проєкту. Так як це груповий проект, буде висвітле тільки частина, виконана мною, а саме класи моделей вигляду та власне моделей.

Застосунок створений за допомою модулю Windows Presentation Foundation (WPF) на базі мови с#.

На рис. 2.1 зображено оглядач рішень програмного застосунку. В табл.2.1 представлено опис модулів(класів) проекту курсової роботи

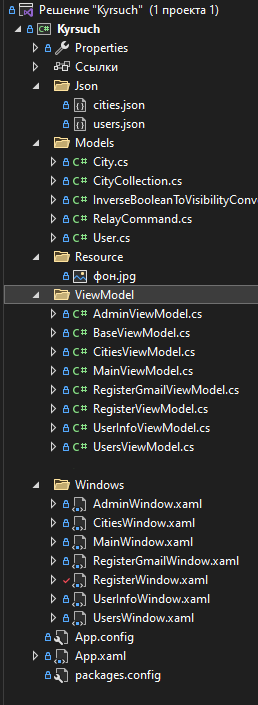
****

Рисунок 2.1 – Оглядач рішень

Таблиця 2.1 – Призначення модулів програми

|  |  |
| --- | --- |
| **Назва модуля** | **Призначення модуля** |
| User.cs | Клас, що зберігає статистику та дані користувача |
| City.cs | Клас, що зберігає дані про міста та посилання на чати. |
| CityCollection.cs | Клас, що відповідає за завантаженння данних з файлу та зберігання в файл. |
| RelayCommand.cs | Клас, який використовується для реалізації патерна команди (Command pattern) в програмній розробці. |
| InverseBooleanToVisibilityConverter.cs | Клас InverseBooleanToVisibilityConverter реалізує інтерфейс IValueConverter. |
| BaseViewModel.cs | Базовий клас, для класів моделей вигляду. |
| CitiesViewModel.cs | Клас який відповідає за логіку додавання, зберігання та редагування місто з їхї посиланнями. |
| AdminViewModel.cs | Клас який відповідає за логіку вибору дії адміністратора. |
| MainViewModel.cs | Клас який відповідає за логіку авторизації. |
| RegisterGmailViewModel.cs | Клас який відповідає за логіку реєстрації за поштою gmail. |
| RegisterViewModel.cs | Клас який відповідає за логіку реєстрації. |
| UserInfoViewModel.cs | Клас який відповідає за логіку виведення інформації користувачу про нього, отримання чату та можливість її редагування інформації. |

Продовження таблиці 2.1

|  |  |
| --- | --- |
| **Назва модуля** | **Призначення модуля** |
| UsersViewModel.cs | Клас який відповіає за роботу з користувачами перегляд хто заеєстрований та можливість видалення користувача |

Під час створення програми було створено 7 вікон, і відповідно до них було створено 7 моделей вигляду її діаграми зображені на рис. 2.2. Всі класи моделей вигляду наслідують клас ViewModel, який реалізує інтерфейс INotifyPropertyChanged

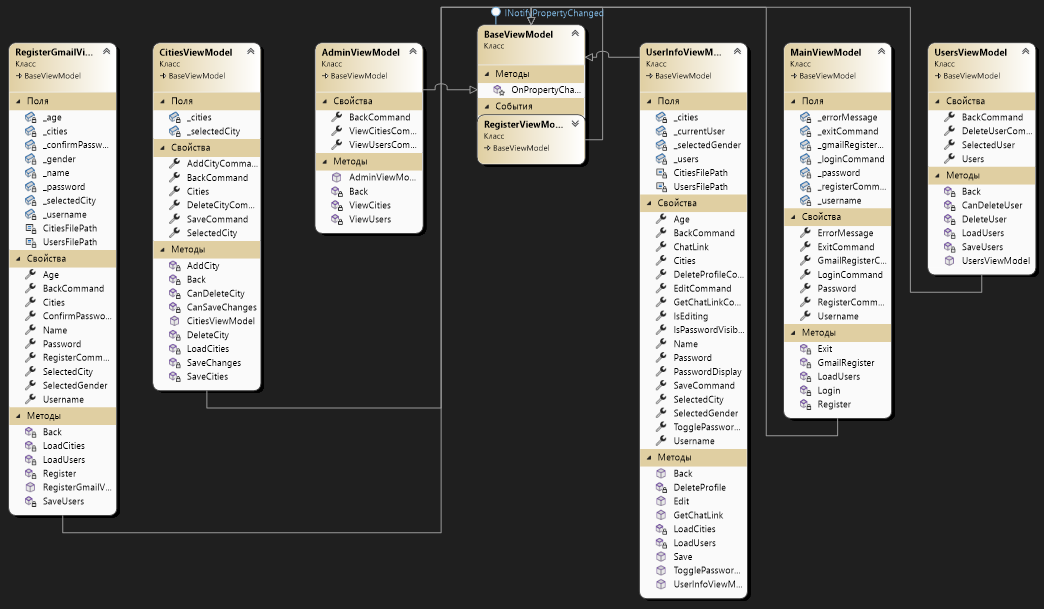


Рисунок 2.2 – Діаграма модулей вигляду

Також у всіх классах використовуются класи RelayCommand, який реалізує ICommand. На рис. 2.4 можна побачити схему моделей.

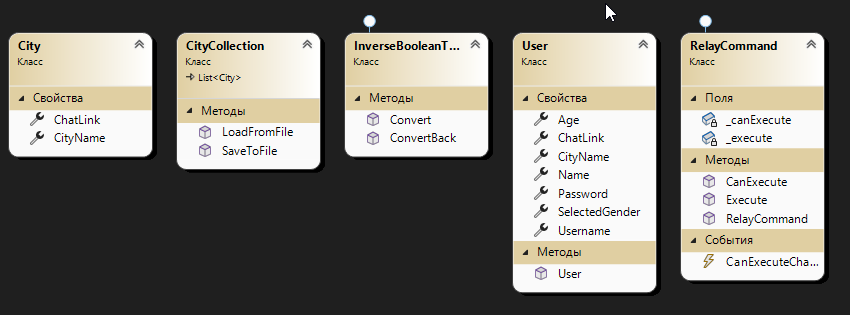


Рисунок 2.3 – Схема класів Models

2.2 Опис роботи програми

Після запуску програми відкривається вікно з авторизацією, з якого можна перейти у інші вікна такі як вікно з інформацією про користувача(Якщо є аккаунт) відкриваєтся вікно користувача де він бачить свій профіль та може його редагувати або видалити на цьому вікні він і отримує посилання на чат для спілкування за обраним містом.

-Вікно інформації показує користувачу всі данні що користувач вводив при реєстрації.

Якщо аккаунту немає користувач настискає кнопки зареєструватися(або зареєструватися через Gmail) тоді відкриваєтся вікно реєстрації де треба запомниити всі піля які також перевіряють введені данні.

-Вікно реєстрації має поля для вводу логіну, паролю , підтвердження паролю, вибору статі, введення віку та вибору міста

Якщо користувач заходить з данними адміністратора він отримує можливість перейти у два вікна.

-Вікно перегляду користувачів дає можливість переглянути зареєстрованих користувачів та видаляти їх.

-Вікно перегляду міст де адміністратор бачить міста та посилання на чати для знайомства в цих містах з можливістью редагувати, додавати та видаляти їх.

**2.3 Функціональна схема програми**

Детальний опис й пояснення процесів окремих функцій розробляємої системи утворюють функціональну схему програми (рис. 2.4).

На рис. 2.4 видно, що програма починається з авторизації. В цьому вікні користувач може ввести вхідні данні, це логін та пароль. Програма шукає, чи є користовач з таким логіном, якщо є, то перевіряється пароль. Якщо все вірно, далі іде перевірка, чи є користувач адміністратором, якщо так, то відкривається вікно адміністратора, якщо ні, то відкривається вікно користувача

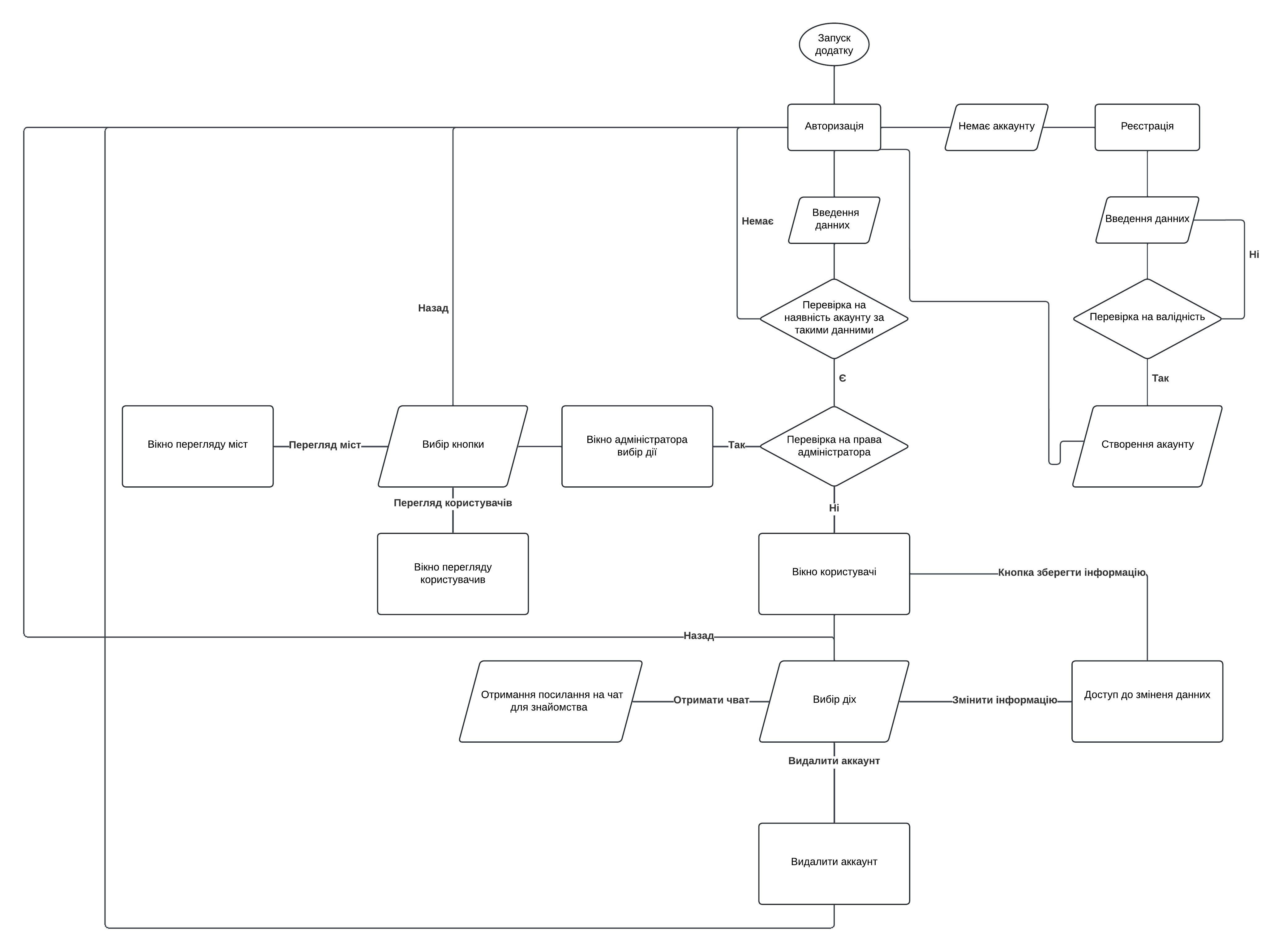


Рисунок 2.4 – Функціональна схема програмного коду

Також з вікна авторизації можна потрапити у вікно реєстрації. В ньому користувач вводить дані, і якщо дані пройшли перевірку на валідність, то на основі цих даних створюється новий акаунт. Після цього відкривається вікно з авторизацією.

2.4 Опис розробки модулів ViewModel та Model проекту

**2.4.1 MVVM та Model**

Почнемо з розгляду класу User.cs, як є одніє з основних моделей даних в рамках цього проекту.Цей клас містить в собі 7 полів та конструктор з параметрами, як зображено на рис. 2.5

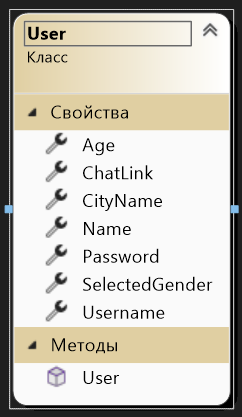


Рисунок 2.5 – Схема класу User

Не менш важливим є клас City. Цей клас має 2 поля, як зображено на рис.2.6

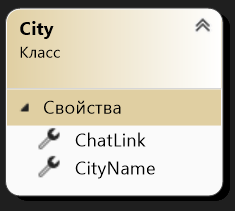


Рисунок 2.6 – Схема класу City

Класи RelayCommand.cs, ViewModelBase.cs утворюють основні складові шаблону проекту для використання архітектури MVVM (Model-View-ViewModel) в додатках, розроблених на платформі .NET. КласRelayCommand.cs визначає можливість створення команд для взаємодії між елементами користувацького інтерфейсу, а BaseViewModel.cs надає базовий функціонал для властивостей та подій, які використовуються в ViewModel.

**2.4.2 Огляд View Model**

Першим розглянемо клас MainViewModel. Він є частиною паттерну проектування MVVM (Model-View-ViewModel) у програмі, яка відповідає за авторизацію користувачів або перенаправляє його на вікна з реєстрацією. Його основна функція полягає в управлінні введенням логіну та пароля, перевірці їх правильності та наданні доступу до програми на основі цих даних. Клас містить властивості для зберігання поточного користувача та введених даних, а також методи, які дозволяють авторизувати користувача та виконувати дії в залежності від його статусу (адміністратор або звичайний користувач).

Клас MainViewModel відповідає за авторизацію користувачів у додатку, використовуючи паттерн проектування MVVM (Model-View-ViewModel). Він містить поля для зберігання списку користувачів, введеного логіну та пароля, а також об'єктів для навігації у програмі.

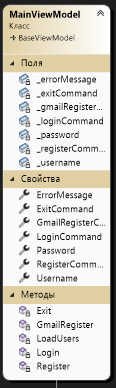


Рисунок 2.7 – Схема класу LoginViewModel

Метод Login(), який надає даний клас, відповідає за перевірку введених користувачем данх, визначеням його ролі, та в залежності від цього наданям доступу.  
Метод Register()(або GmailRegister())(), який надає даний клас, відповідає за відкриття нових вікона для реїстрації.

Метод Exit(), який надає даний клас, відповідає за вихід з програми

Метод LoadUser(), який надає даний клас, відповідає за завантаження данних про користувачів з фай

Клас, який реалізує головний функціонал нашого застосунку InfoUserViewModel.cs, його основна функція відображення данних користувача для його перегляду та отримання посилання на чат для знайомства по обраному місту

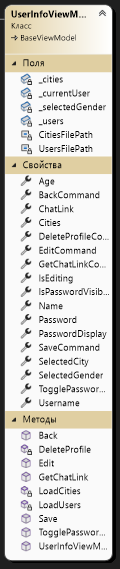


Рисунок 2.8 – Схема класу InfoUserViewModel

Висновки до розділу

У другому розділі розглянуті: ієрархія класів, ініціалізація даних, функціональна схема програми та опис основних функціональних моделей та представлень моделей.

Крім того, розроблено моделі, функціонал, і моделі представлення, які пов’язували представлення, яке баче користувач, з функціоналом

ВИСНОВКИ

Метою курсової роботи є створення багатовіконного Wpf додатку, що реалізує додаток для пошуку знайомств «Бюро знайомств».

Над проектом працювала команда з двох студентів, тому обв’язки було розподілені на такі частини View– Ляпко Н.І. та ViewModel, Model– Гур’єв Д.П.

У проекті розглянуті базові принципи об'єктно-орієнтованого програмування (ООП), проведений аналіз предметної області та сформовано технічне завдання. Усі функціональні вимоги були реалізовані у коді проекту.

Проект розроблено на мові програмування C#. Основним інструментом

для створення графічного інтерфейсу використовувалась бібліотека WPF.

Створення проекту відбувалось за шаблоном MVVM. Основна увага у розробці проекту була зосереджена на моделі даних та моделях представлення. Але в проекті також реалізовано зручний інтерфейс для забезпечення комфортної роботи користувача. Прив’язка даних, що реалізована в програмному коді, є забезпеченням швидкодії обміну даними та зручним механізмом корегування та створення об’єктів моделі даних.

СПИСОК ЛІТЕРАТУРИ

1. Повне керівництво з мови програмування С# та платформі .NET 8. Інформаційний ресурс. <https://metanit.com/sharp/tutorial/>
2. Блог про патерн MVVM: [MVVM: Model-View-ViewModel Explained](https://www.mvvmcross.com/) – Все про MVVM для .NET розробників.
3. [MVVM Examples on GitHub](MVVM%20Examples%20on%20GitHub%20–%20Відкриті%20репозиторії%20з%20прикладами%20WPF%20додатків.) – Відкриті репозиторії з прикладами WPF додатків.
4. Офіційна документація Microsoft: [ICommand Interface](https://learn.microsoft.com/en-us/dotnet/api/system.windows.input.icommand?view=windowsdesktop-7.0) – Як створювати та використовувати команди в WPF.

Тестування

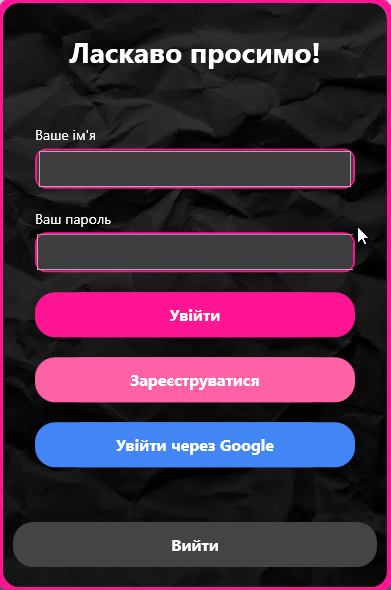


Рис. Т.1 - Вікно авторизації

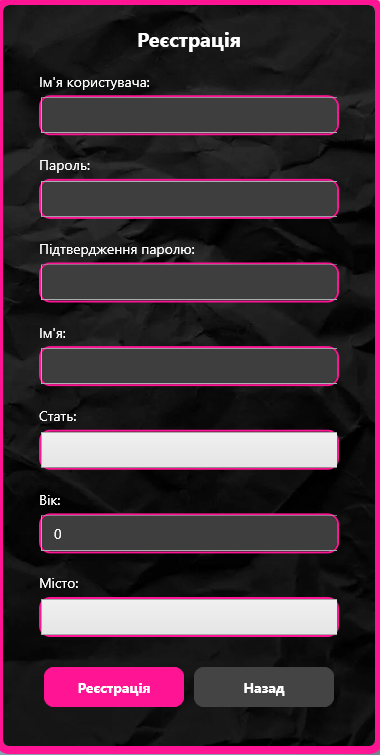


Рис. Т.2 - Вікно реєстрації

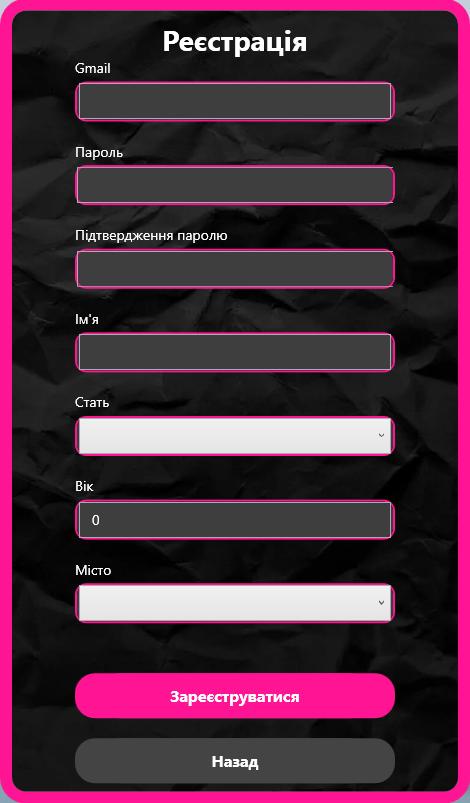


Рис. Т.3 - Вікно реєстрації Gmail

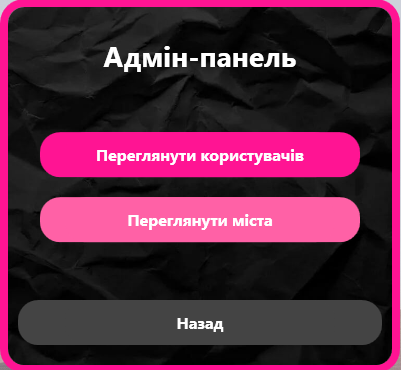


Рис. Т.4 - Вікно адміністратора

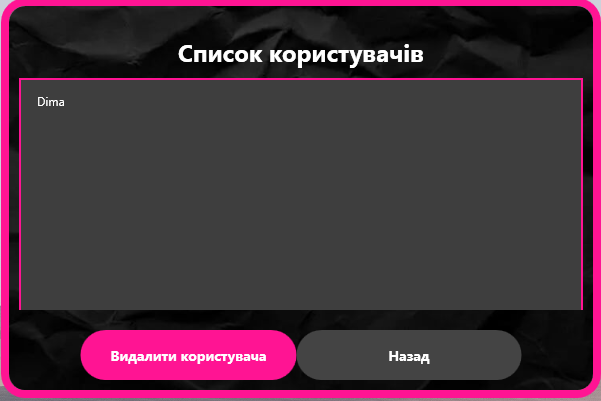


Рис. Т.5 - Вікно керування користувачами

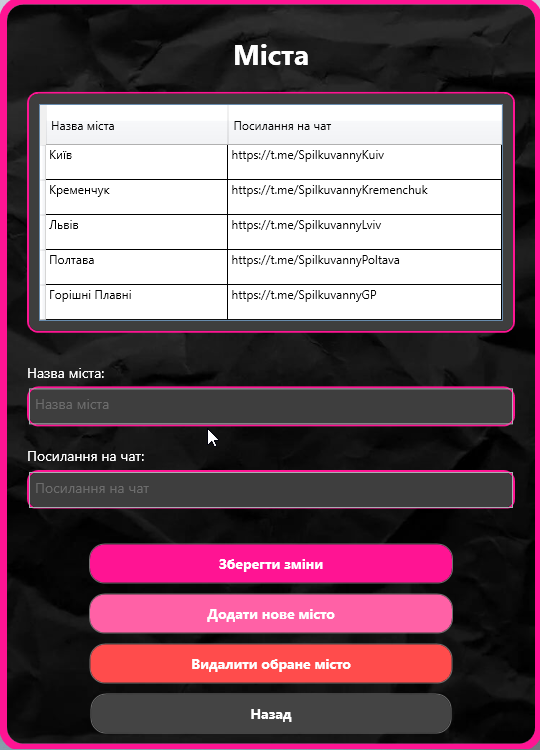


Рис. Т.6 - Вікно керування містами

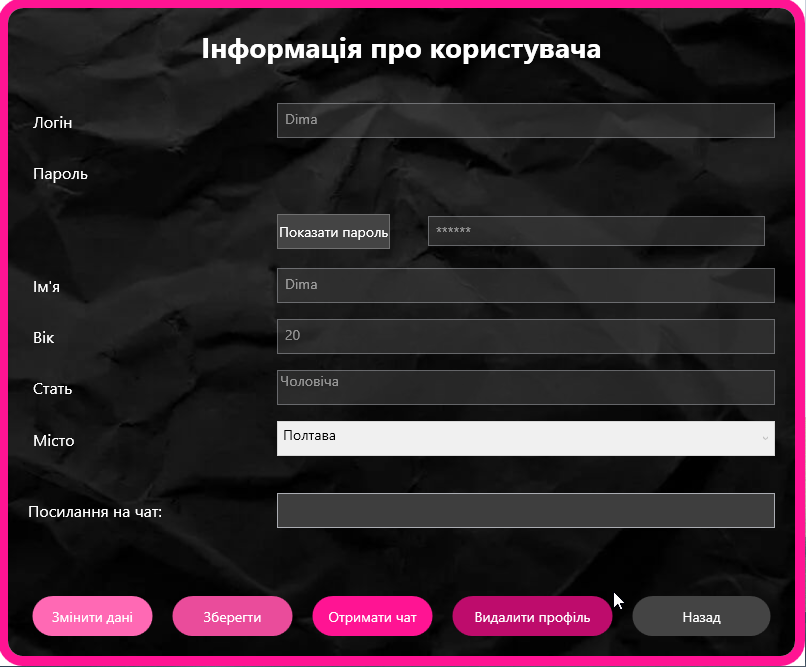


Рис. Т.7 - Вікно користувача