**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 програми обліку відвідування занять студентами

Студента \_2\_ курсу КН–22–1 групи

Ступінь вищої освіти «Бакалавр»

(бакалавр, магістр)

Спеціальність 122 – «Комп’ютерні науки»

Освітньо-професійна програма

«Комп’ютерні науки»

Алєксєєвої А.А

(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)

І. Г. Оксанич

(підпис) (ініціали та прізвище)

В. В. Найда

(підпис) (ініціали та прізвище)

м. Кременчук 2023 рік

**Форма № Н-9.01у**

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Кафедра автоматизації та інформаційних систем \_\_\_

Дисципліна «Сучасні мови об’єктно-орієнтованого програмування» \_\_\_

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

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

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

Курс 2 група \_\_КН-22-1 семестр \_\_3\_\_

**ЗАВДАННЯ**

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алєксєєвій Анастасії Антонівні \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_

(прізвище, ім’я, по-батькові)

1. Тема роботи: Розробка програми обліку відвідування занять студентами
2. Термін здачі студентом роботи 1 грудня 2023 р
3. Вихідні дані до роботи:
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 інтерфейс програми.
5. Перелік графічного матеріалу:
6. Дата видачі завдання: 1 жовтня 2023 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 | 01.10.23–  25.10.23 |  |  | | |
| 2 | Етап 2  Створення моделі даних | 16.10.23–  28.10.23 |  |  | | |
| 3 | Етап 3  Розробка віконного інтерфейсу та створення основного програмного коду додатку | 29.10.23–  24.11.23 |  |  | | |
| 4 | Етап 4  Тестування програмного коду | 16.10.23  29.11.23 |  |  | | |
| 5 | Етап 5  Оформлення пояснювальної записки | 25.11.23–  31.11 .23 |  |  | | |
| 6 | Етап 9  Захист | 01.12.23 |  |  | | |
|  | Разом | 8 тижнів |  |  | | |

Студент

(підпис)

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

(підпис) (ініціали та прізвище)

«1» жовтня 2023 р.

РЕФЕРАТ

Курсова робота містить 43 сторінок, 2 розділи, 12 рисунків, 1 таблиця, 6 використаних джерел.

Об’єкт розробки – WPF додаток «Журнал відвідування».

Мета: створення багатовіконного Wpf додатку, що реалізує розробку програми обліку відвідувань занять студентами.

Під час виконання завдання, поставленого на курсову роботу, було виконано аналіз алгоритму «Облік відвідування занять студентами». Визначено основні функціональні та не функціональні вимоги до додатку, що створюється. Побудована модель даних, що дозволить вирішити наступні задачі: реалізація методів, розробка алгоритму реєстрації та авторизації користувача, розробка алгоритму редагування та додавання студентів/користувачів.

В якості мови програмування для створення Wpf додатку використовувалася мова С# та середовище програмування Visual Studio 22.

Результатом виконання всіх етапів є застосунок, створений за допомогю технологій WPF на мові програмування С#, який може реєструвати користувачів, додавати інформацію про студентів, вести журнал відвідувань та створювати звіти про відвідування.

ОБ’ЄКТ, ІНКАПСУЛЯЦІЯ, ІНТЕРФЕЙС, КЛАС

ЗМІСТ

[ВСТУП 2](#_Toc152849349)

[1 ОПИС ПРЕДМЕТНОЇ ЧАСТИНИ 4](#_Toc152849350)

[1.1 Основні принципи об’єктно-орієнтованого програмування 4](#_Toc152849351)

[1.2 Аналіз технічного завдання на роботу 6](#_Toc152849352)

[1.3 Структура програмного забезпечення 7](#_Toc152849353)

[2 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 9](#_Toc152849354)

[2.1 Опис ієрархії класів 9](#_Toc152849355)

[2.2 Опис роботи програми 11](#_Toc152849356)

[2.3 Опис інтерфейсу програми 12](#_Toc152849357)

[ВИСНОВКИ 17](#_Toc152849358)

[СПИСОК ЛІТЕРАТУРИ 18](#_Toc152849359)

[Додаток А 19](#_Toc152849360)

[Додаток Б 27](#_Toc152849361)

[Додаток В 35](#_Toc152849362)

# ВСТУП

Журнал відвідування студентів являється необхідним інструментом кожного викладача в університеті. Такий додаток завжди буде на разі допомогти викладачу відстежити продуктивність студентів.

Для створення програми, яка реалізує журнал відвідування, було вибрано найбільш актуальну платформу для розробки графічного інтерфейсу в коді C# — WPF.

Windows Presentation Foundation (WPF) широко використовується для розробки користувальницьких інтерфейсів. Він підтримує широкий спектр компонентів розробки додатків, таких як ресурси, елементи управління, графіка, макети, стилі та прив'язка даних.

Завдяки стилям, ми можемо легко створювати різноманітні інтерфейси вікон, які можуть бути взагалі не схожими на стандартні вікна Windows.

Binding — це потужний інструмент програмування в WPF, без якого не обходиться жоден серйозний додаток. Прив'язуючи дані до елементів інтерфейсу, ми можемо легко змінювати і відображати дані.

Завдяки безлічі вбудованих функцій, WPF практично не має конкурентів, коли мова йде про розробку складних додатків для Windows. Ми можемо легко змінювати і відображати дані, прив'язуючи дані до елементів інтерфейсу. WPF дозволяє спроектувати інтерфейс додатка, який є одночасно зручним і інтуїтивно зрозумілим.

# 1 ОПИС ПРЕДМЕТНОЇ ЧАСТИНИ

## Основні принципи об’єктно-орієнтованого програмування

Об'єктно-орієнтоване програмування - це програмування, сфокусоване на даних, причому дані та поведінка нерозривно пов'язані. Водночас дані та поведінка є класом, а об'єкти є екземплярами класу. Наприклад, багаточлен має область значень, і вона може змінюватися такими операціями, як додавання та множення багаточленів.

ООП розглядає обчислення як моделювання поведінки. Те, що моделюється, є об'єктами, представленими обчислювальною абстракцією. Допустимо, ми хочемо покращити наші навички гри в покер; для цього ми маємо навчитися обчислювати можливість випадання різних карткових комбінацій. Нам треба змоделювати перетасовування колоди, крім того слід знайти підходящий спосіб для оперування мастями та перевагами карт. Ми можемо відкрито використовувати назви мастей: піки, черв'яки, бубни та трефи, але технічно масті представлені цілими числами. Це внутрішнє представлення приховано і тому не може вплинути на наші розрахунки. Також як матеріальні колоди карт можуть мати різні фізичні властивості, але при цьому поводяться однаковим очікуваним від них чином, так і різні модельовані колоди повинні поводитися однаково.

Застосуємо термін абстрактний тип даних, АТД (Abstract data type, АDТ) для позначення визначуваного користувачем розширення вихідних типів мови. АТД складається з набору значень та операцій, які можуть впливати на ці значення. Наприклад, С немає типу даних для комплексних чисел, а С # дозволяє додати такий тип і інтегрувати його з існуючими.

Об'єкти (objects) є змінними класом. Об'єктно-орієнтоване програмування дозволяє легко створювати та використовувати АТД. Об'єктно-орієнтоване програмування використовує механізм спадкування (inheritance). На слідування вигідно тим, що дозволяє отримувати похідні типи вже визначених користувачем типів даних. Цей механізм схожий на біологічну таксономію. І гризуни, і кішки - ссавці. Якщо категорія «ссавці» несе в собі інформацію про властивості та поведінку, істинну для кожного з об'єктів відповідного біологічного класу, то створення категорій «котячих» та «гризунів» з категорії «ссавців» є надзвичайно економічним.

В ООП об'єкти відповідають за свою поведінку. Наприклад, багаточлени, комплексні числа, цілі числа, числа з плаваючою точкою — це об'єкти, «розуміють» додавання. Кожен з цих типів включає код для виконання складання. Компілятор надає належний код для цілих чисел з плаваючою точкою. АТД «багаточлен» містить функцію, що визначає складання, відповідно до особливостей своєї реалізації. Постачальник АТД повинен включати код для опису будь-якої поведінки, яке зазвичай можна очікувати від відповідних об'єктів. Те, що сам відповідає за свою поведінку, значно спрощує завдання програмування для користувача цього об'єкта.

Уявімо собі клас об'єктів під назвою «фігури». Якщо потрібно намалювати на екрані якусь фігуру, треба знати, де буде її центр і як її малювати. Деякі фігури, наприклад, багатокутники, відносно легко намалювати. У випадку, процедура малювання фігури може бути дуже трудомісткою, оскільки, можливо, потрібно зберігати велике число окремих граничних точок. Навпаки, варіант з багатокутником безперечно зручний. Якщо окрема фігура чудово розуміє, як намалювати себе, програміст при використанні такої фігури має лише передати об'єкту повідомлення «намалювати(ся)».

WPF, що означає Windows Presentation Foundation, є останнім підходом Microsoft до GUI framework, який використовується з .NET framework.

GUI розшифровується як графічний інтерфейс користувача, і ви, ймовірно, зараз дивитесь на нього. Windows має графічний інтерфейс для роботи з вашим комп’ютером, а браузер, у якому ви, ймовірно, читаєте цей документ, має графічний інтерфейс, який дозволяє вам переглядати веб-сторінки.

Структура графічного інтерфейсу дозволяє створювати програму з широким набором елементів графічного інтерфейсу, таких як мітки, текстові поля та інші добре відомі елементи. Без графічного інтерфейсу вам довелося б малювати ці елементи вручну та обробляти всі сценарії взаємодії користувача, як-от введення тексту та миші.

Існує багато фреймворків графічного інтерфейсу користувача, але для розробників .NET найцікавішими наразі є WinForms і WPF. WPF є найновішим, але Microsoft все ще обслуговує та підтримує WinForms.

## 1.2 Аналіз технічного завдання на роботу

Мета курсового проекту — Розробити WPF додаток «Облік відвідування занять студентами».

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

**Функціональні вимоги**

1. Можливість реєстрації та авторизації.
2. Можливість додавання інформації про студентів.
3. Можливість ведення журналу відвідування.
4. Можливість створення звіту про відвідування.

**Нефункціональні вимоги**

1. Зовнішній інтерфейс користувача має бути реалізованим за допомогою WPF.
2. Додаток має бути багатовіконним.
3. Усі поля введення повинні бути захищені від некоректного введення.

Використання патерну MVVM.

## 1.3 Структура програмного забезпечення

Структурою ПЗ являються усі розроблені модулі програми, які разом утворюють систему.

Кожен модуль реалізує свої конкретні функції, які вирішують основні задачі проекті.

На рисунку нижче можна ознайомитись з оглядачем рішень проекту (деревом проекту).

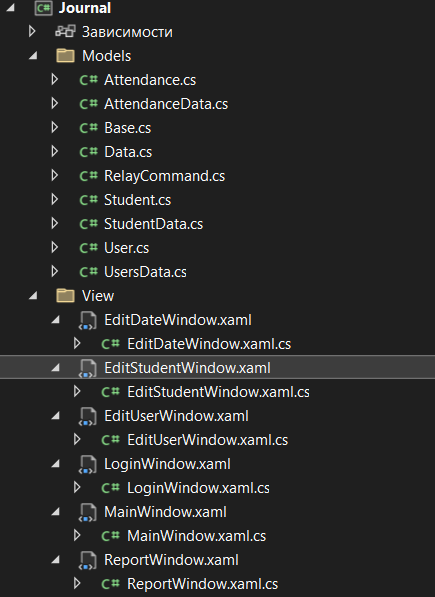


Рис. 1.1 – Оглядач рішень

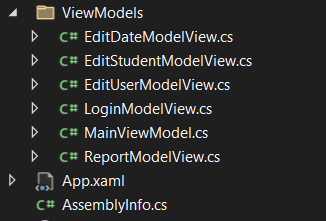


Рис. 1.2 – Оглядач рішень (продовження)

# 2 ОПИС РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Опис ієрархії класів

При розробці програмного забезпечення для зручності було створено універсальний клас Data<T> для зберігання та завантаження даних, необхідних для роботи програми.

Від цього класу були успадковані такі класи AttendanceData, StudentData та UsersData.

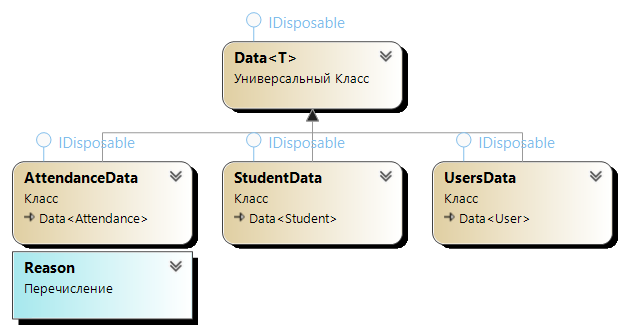


Рис 2.1 – Діаграма створених класів даних.

Для того щоб зручно працювати з даними та для автоматичного відстеження будь-яких змін, було створено абстрактний клас Base, який використовують майже більша частина класів додатку.

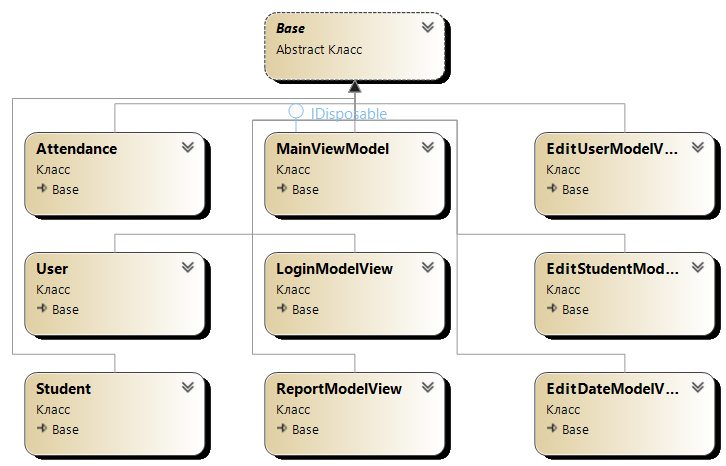


Рис. 2.2 – Діаграма класів з відстеженням змін

В подальшій розробці додатку крім основного вікна було створено вікно для редагування списку студентів, редагування дат відвідування, вікно для реєстрації нових користувачів та вікно для відображення звіту про відвідування студентів.

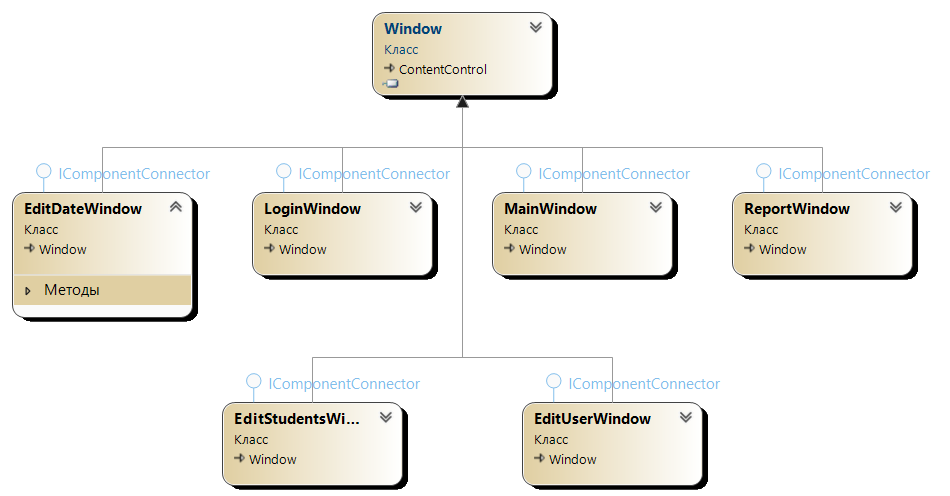


Рис. 2.3 – Діаграма класів вікон додатку

## 2.2 Опис роботи програми

Після запуску програми користувачу відкривається вікно входу, в якому користувач повинен ввести свій логін і пароль для отримання доступу до редагування журналу відвідування.

У вікні редагування списку студентів, користувач може додати в список студентів дані про нову особу або навпаки видалити дані про існуючу.

У вікні редагування користувачів програми, користувач може зареєструвати нового або навпаки видалити існуючого користувача журналу відвідування.

Також подібним чином у вікні редагування дат відвідування користувач може додавати або видаляти дату відвідування зі списку.

Відкривши вікно звіту про відвідування, користувач може ознайомитися з таблицею «якості» відвідування студентів, яка представлена в кількості пропусків і відсотком відвідування.

Детальніше робота програмного забезпечення описана нижче в функціональній схемі програми (рис. 2.4).

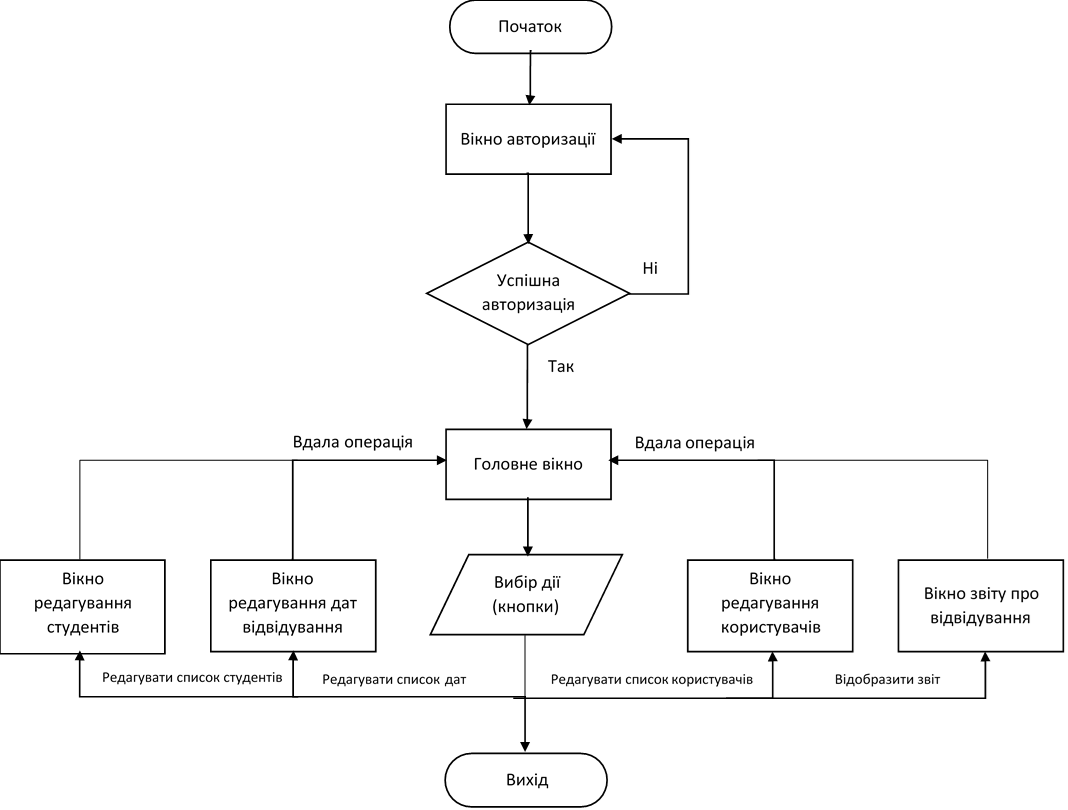


Рис. 2.4 – Функціональна схема роботи програми

## 2.3 Опис інтерфейсу програми

При запускі програми користувача зустрічає вікно авторизації в якому для входу треба ввести логін та пароль і натиснути кнопку «Вхід».

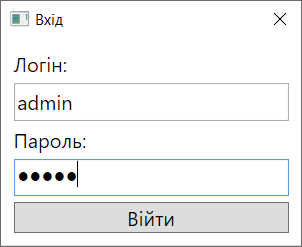


Рис. 2.5 – Вікно авторизації

Після входу для користувача відкривається головне вікно програми в якому відображена таблиця журналу. Для того щоб відмітити прогул студента необхідно в комірці навпроти імені студента та необхідної дати вписати букву «н» для відмічення невідомої причини прогулу або «хв» - якщо студент хворий.

Після редагування журналу необхідно натиснути Файл – Зберегти. Після редагування списку студентів або дат необхідно оновити таблицю шляхом Файл – Оновити.

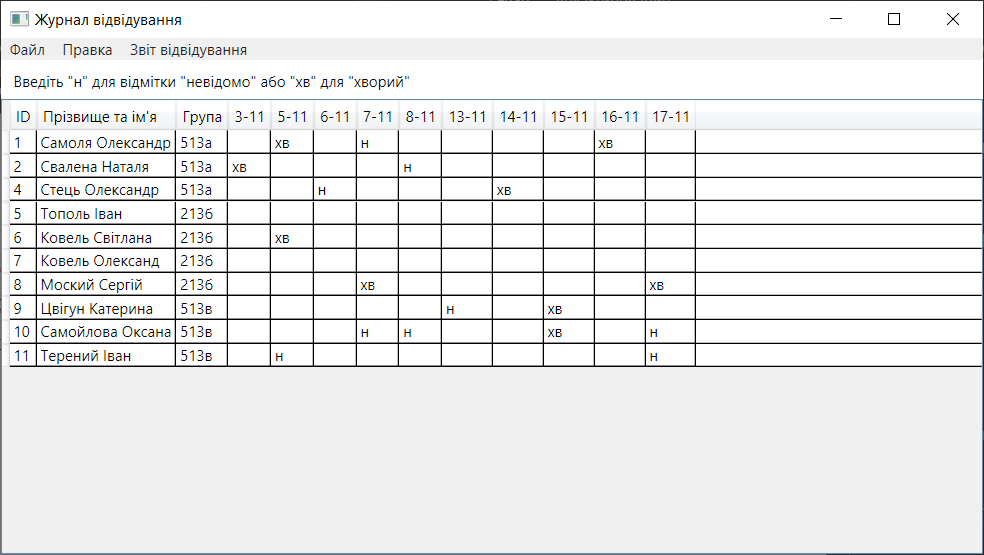


Рис. 2.6 – Головне вікно програми

Для того щоб відредагувати список студентів користувачу потрібно пройти в меню Правка – Редагувати список студентів. Після чого відкриється вікно редагування в якому для створення нового запису про студента необхідно ввести його дані і потім натиснути кнопку «Додати». Для видалення необхідно обрати інформацію з колонки нижче і потім натиснути кнопку «Видалити».

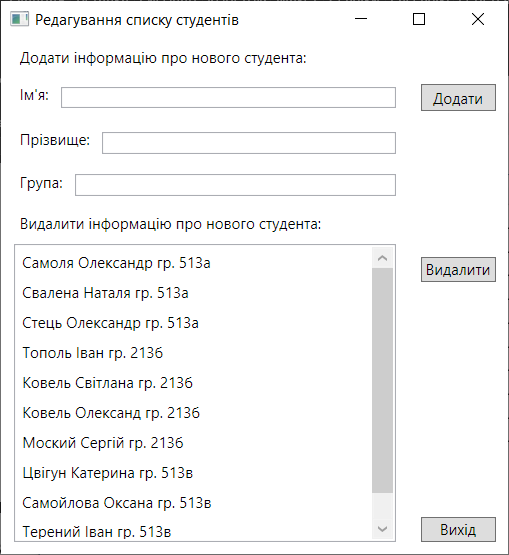


Рис. 2.7 – Вікно редагування інформації про студентів

Для того щоб зареєструвати нового користувача потрібно пройти в меню Правка – Редагувати користувачів. Після чого відкриється вікно редагування в якому для створення нового запису користувача необхідно ввести його дані і потім натиснути кнопку «Додати». Для видалення необхідно обрати інформацію з колонки нижче і потім натиснути кнопку «Видалити».

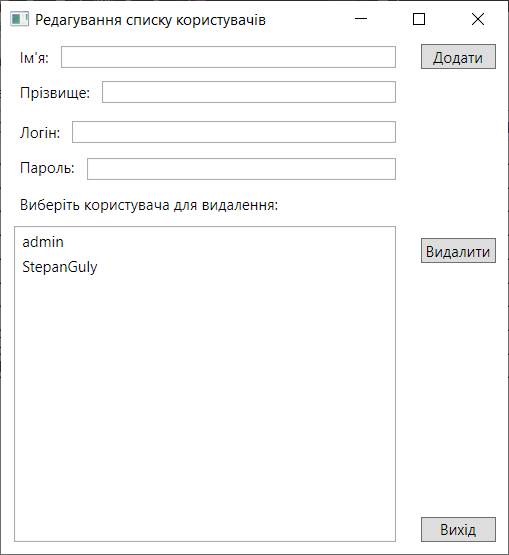


Рис. 2.7 – Вікно реєстрації користувачів

Для того щоб відредагувати список дат відвідування потрібно пройти в меню Правка – Редагувати список дат відвідування. Після чого відкриється вікно редагування в якому для створення нової дати необхідно ввести обрати її в календарі та натиснути кнопку «Додати». Для видалення необхідно обрати інформацію з колонки нижче і потім натиснути кнопку «Видалити».

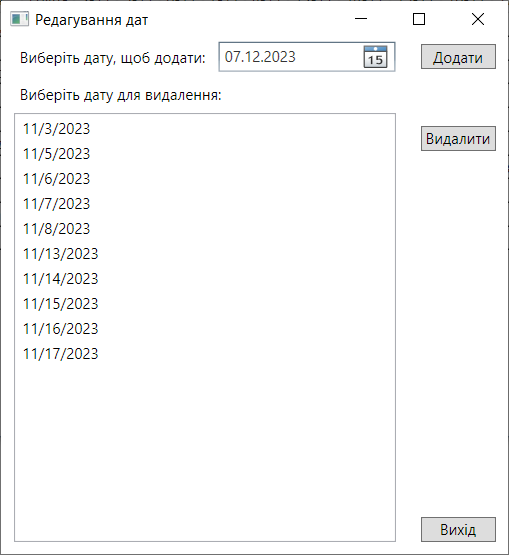


Рис. 2.8 – Вікно редагування дат

Для відображення звіту про відвідування студентів користувачу необхідно обрати пункт меню Звіт відвідування. Після цього користувач може ознайомитись зі списком студентів, в якому відображена кількість прогулів та відвідуваність студентів в відсотках.

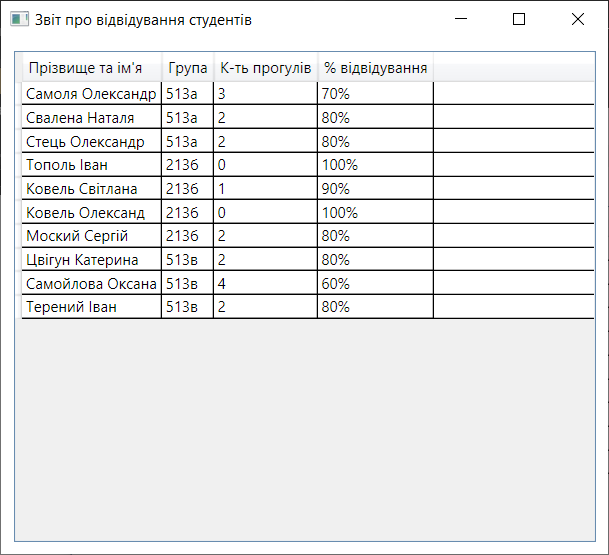


Рис. 2.9 – Вікно результатів відвідування

# ВИСНОВКИ

Під час створення курсового проекту були розглянуті базові принципи ООП, проаналізована предметна область, реалізовані: можливість реєстрації та авторизації, можливість редагування списку студентів та дат, генерація звіту про відвідування.

Для виконання даного проекту була використана мова програмування C#, яка добре підходить для розробки програм з використанням принципів ООП.

Важливим етапом розробки стали планування й реалізація користувальницького інтерфейсу.

Основним завданням створення інтерфейсу було задовольнити низку вимог, серед яких: зручність роботи з інтерфейсом, швидкість обміну даними, ефективність управління, реалізація усіх необхідних функцій, а також захист від усіх видів некоректного введення.

Задля відокремлення розробки графічного інтерфейсу та моделей даних, було використано шаблон проектування MVVM.

Курсовий проект зі створення WPF-додатку «Журнал відвідування» демонструє найважливіші принципи роботи з програмним середовищем WPF, його інструментами та функціями, патерном MVVM і загальними принципами об’єктно-орієнтованого програмування.

# СПИСОК ЛІТЕРАТУРИ

1. Microsoft Learn. Іннформаційний ресурс. https://learn.microsoft.com/uk-ua/dotnet.

2. Serializing .NET 6s new DateOnly to JSON. Інформаційний ресурс. https://storck.io/posts/serializing-net-6-dateonly-to-json/.

3. StackOverflow. Інформаційний ресурс. https://stackoverflow.com.

4. CLR за допомогою C#. Програмування на платформі Microsoft .NET Framework 4.5 мовою C#. Джеффри Рихтер. 2020р. 896 с.

5. itProger. Інтернет Ресурс. https://itproger.com/ua/course/csharp-wpf

6. 2. C Sharp. Інформаційний ресурс. https://uk.wikipedia.org/wiki/C\_Sharp

# Додаток А

**XAML-розмітка вікон додатку**

1) Розмітка вікна “MainWindow”

<Window

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:Journal"

xmlns:Models="clr-namespace:Journal.Models" xmlns:ViewModels ="clr-namespace:Journal.ViewModels" x:Class="Journal.MainWindow"

mc:Ignorable="d"

Title="Журнал відвідування" Height="450" Width="800" MinWidth="600" MinHeight="450">

<Window.DataContext>

<ViewModels:MainViewModel></ViewModels:MainViewModel>

</Window.DataContext>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" MinHeight="18.96"/>

<RowDefinition Height="30"/>

<RowDefinition Height="236\*"/>

</Grid.RowDefinitions>

<Menu Margin="0,0,0,1">

<MenuItem Header="Файл">

<MenuItem Header="Оновити" Command="{Binding FillFullDataCommand}" CommandParameter="{Binding ElementName=viewGrid}"/>

<MenuItem Header="Зберегти" Command="{Binding SaveFullDataCommand}"/>

<Separator/>

<MenuItem Header="Вихід" Click="MenuItem\_Click"/>

</MenuItem>

<MenuItem Header="Правка">

<MenuItem Header="Редагувати список студентів" Command="{Binding EditStudents}"/>

<MenuItem Header="Редагувати користувачів" Command="{Binding EditUsers}"/>

<Separator/>

<MenuItem Header="Редагувати список дат відвідування" Command="{Binding EditDate}"/>

</MenuItem>

<MenuItem Header="Звіт відвідування" Command="{Binding ReportCommand, Mode=OneWay}"/>

</Menu>

<DataGrid x:Name="viewGrid" Grid.Row="2" CanUserReorderColumns="False" CanUserResizeColumns="False" CanUserAddRows="False" ItemsSource="{Binding FullData}" CanUserResizeRows="False" CanUserDeleteRows="False" Grid.RowSpan="2" FrozenColumnCount="3">

</DataGrid>

<TextBlock HorizontalAlignment="Left" Margin="10,0,0,0" Grid.Row="1" TextWrapping="Wrap" Text="Введіть &quot;н&quot; для відмітки &quot;невідомо&quot; або &quot;хв&quot; для &quot;хворий&quot;" VerticalAlignment="Center"/>

</Grid>

</Window>

2) Розмітка вікна “LoginWindow”

<Window

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:Journal"

xmlns:ViewModels="clr-namespace:Journal.ViewModels" x:Class="Journal.LoginWindow"

mc:Ignorable="d"

Title="Вхід" WindowStartupLocation="CenterScreen" Width="241" Height="190"

SizeToContent="WidthAndHeight" ResizeMode="NoResize" Closed="Window\_Closed">

<Window.DataContext>

<ViewModels:LoginModelView/>

</Window.DataContext>

<Grid>

<TextBox x:Name="loginBox" Margin="10,36,10,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="30" FontSize="16" VerticalContentAlignment="Center" TabIndex="1" Text="{Binding UserLogin}"/>

<Button x:Name="checkButton" Content="Війти" HorizontalAlignment="Center" Margin="0,131,0,10" Width="220" FontSize="16" Focusable="False" TabIndex="3" Command="{Binding LogCommand}" Click="checkButton\_Click\_1" CommandParameter="{Binding ElementName=passwordBox}"/>

<PasswordBox x:Name="passwordBox" HorizontalAlignment="Center" VerticalAlignment="Top" Width="220" Height="29" Margin="10,97,10,0" TabIndex="2" FontSize="16"/>

<TextBlock HorizontalAlignment="Left" Margin="10,10,0,0" TextWrapping="Wrap" Text="Логін:" VerticalAlignment="Top" FontSize="16"/>

<TextBlock HorizontalAlignment="Left" Margin="10,71,0,0" TextWrapping="Wrap" Text="Пароль:" VerticalAlignment="Top" FontSize="16"/>

</Grid>

</Window>

3) Розмітка вікна “EditUserWindow”

<Window

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:Journal.View"

xmlns:ViewModels="clr-namespace:Journal.ViewModels" x:Name="Window" x:Class="Journal.View.EditUserWindow"

mc:Ignorable="d"

Title="Редагування списку користувачів" Height="448" Width="420" MinWidth="420" MinHeight="450">

<Window.DataContext>

<ViewModels:EditUserModelView/>

</Window.DataContext>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="80"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Grid.Column="1" Content="Додати" HorizontalAlignment="Center" VerticalAlignment="Center" Width="60" Command="{Binding AddUser, Mode=OneWay}"/>

<Button Grid.Column="1" Content="Видалити" HorizontalAlignment="Center" VerticalAlignment="Top" Width="60" Margin="0,10,0,0" Grid.Row="5" Command="{Binding DeleteUser, Mode=OneWay}" CommandParameter="{Binding SelectedItem, ElementName=listBox}"/>

<Button Grid.Column="1" Content="Вихід" HorizontalAlignment="Center" Width="60" Margin="0,0,0,10" Grid.Row="5" Height="20" VerticalAlignment="Bottom" Command="{Binding Exit, Mode=OneWay}" CommandParameter="{Binding ElementName=Window, Mode=OneWay}"/>

<ListBox x:Name="listBox" Grid.Row="5" Margin="10,0,10,10" ItemsSource="{Binding UserData.Collection}">

<ListBox.DataContext>

<ViewModels:MainViewModel/>

</ListBox.DataContext>

<ListBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Login}"/>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Label Content="Ім'я:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Center"/>

<Label Content="Прізвище:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Grid.Row="1"/>

<Label Content="Логін:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Center" Grid.Row="2"/>

<Label Content="Пароль:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Grid.Row="3"/>

<Label Content="Виберіть користувача для видалення:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Grid.Row="4"/>

<TextBox Margin="48,0,10,0" TextWrapping="Wrap" VerticalAlignment="Center" Text="{Binding NewName}"/>

<TextBox Margin="81,4,10,0" TextWrapping="Wrap" VerticalAlignment="Top" Grid.Row="1" Text="{Binding NewLastName}"/>

<TextBox Margin="57,0,10,0" TextWrapping="Wrap" VerticalAlignment="Center" Grid.Row="2" Text="{Binding NewLogin}"/>

<TextBox Margin="69,0,10,0" TextWrapping="Wrap" VerticalAlignment="Center" Grid.Row="3" Text="{Binding NewPassword}"/>

</Grid>

</Window>

4) Розмітка вікна “EditStudentWindow”

<Window

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:Journal.View"

xmlns:Models="clr-namespace:Journal.Models" xmlns:VisualBasic="clr-namespace:Microsoft.VisualBasic;assembly=Microsoft.VisualBasic.Core" x:Name="Window" x:Class="Journal.View.EditStudentsWindow"

xmlns:ViewModels ="clr-namespace:Journal.ViewModels"

mc:Ignorable="d"

Title="Редагування списку студентів" Height="450" Width="420" Closing="Window\_Closing" MinWidth="420" MinHeight="450">

<Window.DataContext>

<ViewModels:EditStudentModelView/>

</Window.DataContext>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="80"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="35"/>

<RowDefinition Height="30"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label Content="Додати інформацію про нового студента:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Center" Height="26" Width="239"/>

<Label Content="Ім'я:" HorizontalAlignment="Left" Margin="10,2,0,0" VerticalAlignment="Top" Grid.Row="1" Height="26" Width="33"/>

<TextBox Margin="48,0,10,0" TextWrapping="Wrap" VerticalAlignment="Center" Grid.Row="1" Height="17" Text="{Binding NewStudentName}"/>

<Label Content="Прізвище:" HorizontalAlignment="Left" Margin="10,3,0,0" VerticalAlignment="Top" Grid.Row="2" Height="26" Width="66"/>

<TextBox Margin="81,10,10,0" TextWrapping="Wrap" VerticalAlignment="Top" Grid.Row="2" Height="18" Text="{Binding NewStudentLastName}"/>

<Label Content="Група:" HorizontalAlignment="Left" Margin="10,2,0,0" VerticalAlignment="Top" Grid.Row="3" Height="26" Width="44"/>

<TextBox Margin="59,0,10,0" TextWrapping="Wrap" VerticalAlignment="Center" Grid.Row="3" Height="17" Text="{Binding NewStudentGroup}"/>

<Label Content="Видалити інформацію про нового студента:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Top" Height="26" Width="254" Grid.Row="4"/>

<ListBox x:Name="studentsList" Grid.Row="5" Margin="10,0,10,10" ItemsSource="{Binding StudentData.Collection}">

<ListBox.DataContext>

<ViewModels:MainViewModel/>

</ListBox.DataContext>

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Margin="0,2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="100" />

</Grid.ColumnDefinitions>

<TextBlock Text="{Binding LastName}" />

<TextBlock Grid.Column="1" Text=" "/>

<TextBlock Grid.Column="2" Text="{Binding Name}"/>

<TextBlock Grid.Column="3" Text=" гр. "/>

<TextBlock Grid.Column="4" Text="{Binding Group}"/>

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Button Grid.Column="1" Content="Вихід" Margin="10,0,10,10" Grid.Row="5" Height="20" VerticalAlignment="Bottom" Command="{Binding Exit}" CommandParameter="{Binding ElementName=Window}"/>

<Button Grid.Column="1" Content="Видалити" Margin="10,10,10,0" Grid.Row="5" Height="20" VerticalAlignment="Top" Command="{Binding DeleteStudent}" CommandParameter="{Binding Path=SelectedItem, ElementName=studentsList}"/>

<Button Grid.Column="1" Content="Додати" Margin="10,7,10,7" Grid.Row="1" Command="{Binding AddStudent}"/>

</Grid>

</Window>

5) Розмітка вікна “EditDateWindow”

<Window

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:Journal.View"

xmlns:ViewModels="clr-namespace:Journal.ViewModels" x:Name="Window" x:Class="Journal.View.EditDateWindow"

mc:Ignorable="d" Height="450" Width="420" MinWidth="420" MinHeight="450" Closing="Window\_Closing" Title="Редагування дат">

<Window.DataContext>

<ViewModels:EditDateModelView/>

</Window.DataContext>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition Height="30"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="80"/>

</Grid.ColumnDefinitions>

<DatePicker Margin="174,0,10,0" VerticalAlignment="Center" FirstDayOfWeek="Monday" SelectedDate="{Binding NewDate}"/>

<Label Content="Виберіть дату, щоб додати:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Center"/>

<Label Content="Виберіть дату для видалення:" HorizontalAlignment="Left" Margin="10,0,0,0" VerticalAlignment="Center" Grid.Row="1"/>

<Button Grid.Column="1" Content="Додати" HorizontalAlignment="Center" VerticalAlignment="Center" Width="60" Height="20" Command="{Binding AddDate}"/>

<Button Grid.Column="1" Content="Видалити" Margin="10,10,10,0" Grid.Row="2" Height="20" VerticalAlignment="Top" Command="{Binding DeleteDate}" CommandParameter="{Binding Path=SelectedItem, ElementName=dateList}"/>

<Button Grid.Column="1" Content="Вихід" Margin="10,0,10,10" Grid.Row="2" Height="20" VerticalAlignment="Bottom" Command="{Binding Exit}" CommandParameter="{Binding ElementName=Window}"/>

<ListBox x:Name="dateList" Grid.Row="2" Margin="10,0,10,10" ItemsSource="{Binding AttendanceData.Collection}">

<ListBox.DataContext>

<ViewModels:MainViewModel/>

</ListBox.DataContext>

<ListBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Date}"></TextBlock>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</Grid>

</Window>

6) Розмітка вікна “ReportWindow”

<Window

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:Journal.View"

xmlns:ViewModels="clr-namespace:Journal.ViewModels" x:Class="Journal.View.ReportWindow"

mc:Ignorable="d"

Title="Звіт про відвідування студентів" Height="450" Width="500" MinWidth="500" MinHeight="450">

<Window.DataContext>

<ViewModels:ReportModelView/>

</Window.DataContext>

<Grid>

<DataGrid Margin="10,10,10,10" ItemsSource="{Binding ReportData}" CanUserAddRows="False" CanUserDeleteRows="False" CanUserResizeRows="False" CanUserReorderColumns="False" IsReadOnly="True"/>

</Grid>

</Window>

# Додаток Б

**C# класи Model View**

1) Клас “MainViewModel”

using Journal.Models;

using Journal.View;

using System;

using System.Collections.Generic;

using System.Data;

using System.Windows.Controls;

namespace Journal.ViewModels

{

public class MainViewModel : Base, IDisposable

{

public static StudentData StudentData { get; set; } = new StudentData();

public static AttendanceData AttendanceData { get; set; } = new AttendanceData();

public static UsersData UserData { get; set; } = new UsersData();

public DataTable FullData { get; set; } = new DataTable();

public RelayCommand EditStudents { get; }

public RelayCommand EditUsers { get; }

public RelayCommand EditDate { get; }

public RelayCommand SaveFullDataCommand { get; }

public RelayCommand FillFullDataCommand { get; }

public RelayCommand ReportCommand { get; }

public MainViewModel()

{

EditStudents = new RelayCommand(EditStudentsClick);

EditUsers = new RelayCommand(EditUsersClick);

EditDate = new RelayCommand(EditDateClick);

SaveFullDataCommand = new RelayCommand(SaveFullDataClick);

FillFullDataCommand = new RelayCommand(FillFullDataClick);

ReportCommand = new RelayCommand(ReportClick);

FillFullData();

}

private void FillFullData()

{

FullData = new DataTable();

DataRow row;

FullData.Columns.Add("id", typeof(int));

FullData.Columns.Add("name", typeof(string));

FullData.Columns.Add("group", typeof(string));

foreach (Attendance a in AttendanceData.Collection)

{

DataColumn dataColumn = new DataColumn();

dataColumn.ColumnName = a.Date.Day.ToString() + "-" + a.Date.Month.ToString();

dataColumn.DataType = typeof(string);

FullData.Columns.Add(dataColumn);

}

foreach (Student s in StudentData.Collection)

{

row = FullData.NewRow();

row["id"] = s.Id;

row["name"] = s.LastName + " " + s.Name;

row["group"] = s.Group;

foreach (Attendance a in AttendanceData.Collection)

{

foreach (KeyValuePair<int, Reason> k in a.Students)

{

if (s.Id == k.Key)

{

if (k.Value == Reason.Unknown)

{

row[a.Date.Day.ToString() + "-" + a.Date.Month.ToString()] = "н";

}

else if (k.Value == Reason.Sick)

{

row[a.Date.Day.ToString() + "-" + a.Date.Month.ToString()] = "хв";

}

}

}

}

FullData.Rows.Add(row);

}

}

private void SaveFullData()

{

AttendanceData.Collection.Clear();

string[] str;

for (int j = 3; j < FullData.Columns.Count; j++)

{

str = FullData.Columns[j].ColumnName.Split("-");

DateOnly date = new DateOnly(DateTime.Now.Year, Int32.Parse(str[1]), Int32.Parse(str[0]));

Attendance att = new Attendance(date);

for (int i = 0; i < FullData.Rows.Count; i++)

{

if (FullData.Rows[i].Field<string>(j) == "н")

{

att.Students.Add(FullData.Rows[i].Field<int>("id"), Reason.Unknown);

}

else if (FullData.Rows[i].Field<string>(j) == "хв")

{

att.Students.Add(FullData.Rows[i].Field<int>("id"), Reason.Sick);

}

}

AttendanceData.Collection.Add(att);

}

}

private void EditStudentsClick(object o)

{

new EditStudentsWindow().ShowDialog();

}

private void EditUsersClick(object o)

{

new EditUserWindow().ShowDialog();

}

private void EditDateClick(object o)

{

new EditDateWindow().ShowDialog();

}

private void ReportClick(object o)

{

new ReportWindow().ShowDialog();

}

private void SaveFullDataClick(object sender)

{

SaveFullData();

AttendanceData.Save();

}

private void FillFullDataClick(object sender)

{

FillFullData();

var obj = (DataGrid)sender;

obj.ItemsSource = null;

obj.ItemsSource = FullData.DefaultView;

}

public void Dispose()

{

ReleaseUnmanagedResources();

GC.SuppressFinalize(this);

}

~MainViewModel()

{

ReleaseUnmanagedResources();

}

private void ReleaseUnmanagedResources()

{

StudentData.Dispose();

UserData.Dispose();

AttendanceData.Dispose();

}

}

}

2) Клас “LoginModelView”

using Journal.Models;

using System.Windows;

using System.Windows.Controls;

namespace Journal.ViewModels

{

public class LoginModelView : Base

{

public string UserLogin { get; set; }

public string Password { get; set; }

public RelayCommand LogCommand { get; }

public static bool IsLogged { get; set; }

public LoginModelView()

{

LogCommand = new RelayCommand(LogCommandClick);

}

private void LogCommandClick(object sender)

{

var pasBox = (PasswordBox)sender;

Password = pasBox.Password;

IsLogged = MainViewModel.UserData.LoginCheck(UserLogin, Password);

if (!IsLogged)

{

MessageBox.Show("Невірний логін або пароль.");

}

}

}

}

3) Клас “EditUserModelView”

using Journal.Models;

using Journal.View;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Journal.ViewModels

{

class EditUserModelView : Base

{

public string NewLogin { get; set; }

public string NewPassword { get; set; }

public string NewName { get; set; }

public string NewLastName { get; set; }

public RelayCommand AddUser { get; }

public RelayCommand DeleteUser { get; }

public RelayCommand Exit { get; }

public EditUserModelView()

{

AddUser = new RelayCommand(AddUserClick);

DeleteUser = new RelayCommand(DeleteUserClick);

Exit = new RelayCommand(ExitClick);

}

private void AddUserClick(object o)

{

MainViewModel.UserData.AddUser(NewLogin, NewName, NewLastName, NewPassword);

}

private void DeleteUserClick(object sender)

{

try

{

var st = (User)sender;

MainViewModel.UserData.RemoveUser(st.Login);

}

catch

{

//

}

}

private void ExitClick(object sender)

{

MainViewModel.UserData.Save();

var win = (EditUserWindow)sender;

win.Close();

}

}

}

4) Клас “EditStudentModelView”

using Journal.Models;

using Journal.View;

using System.Linq;

namespace Journal.ViewModels

{

internal class EditStudentModelView : Base

{

public string NewStudentName { get; set; }

public string NewStudentLastName { get; set; }

public string NewStudentGroup { get; set; }

public RelayCommand AddStudent { get; }

public RelayCommand DeleteStudent { get; }

public RelayCommand Exit { get; }

private int Id

{

get

{

return MainViewModel.StudentData.Collection.Max(a => a.Id) + 1;

}

}

public EditStudentModelView()

{

AddStudent = new RelayCommand(AddStudentClick);

DeleteStudent = new RelayCommand(DeleteStudentClick);

Exit = new RelayCommand(ExitClick);

}

private void AddStudentClick(object o)

{

MainViewModel.StudentData.AddStudent(Id, NewStudentName, NewStudentLastName, NewStudentGroup);

}

private void DeleteStudentClick(object sender)

{

try

{

var st = (Student)sender;

MainViewModel.StudentData.RemoveStudent(st.Id);

}

catch

{

//

}

}

private void ExitClick(object sender)

{

MainViewModel.StudentData.Save();

var win = (EditStudentsWindow)sender;

win.Close();

}

}

}

5) Клас “EditDateModelView”

using Journal.Models;

using Journal.View;

using System;

namespace Journal.ViewModels

{

class EditDateModelView

{

public DateTime NewDate { get; set; }

public RelayCommand AddDate { get; }

public RelayCommand DeleteDate { get; }

public RelayCommand Exit { get; }

public EditDateModelView()

{

AddDate = new RelayCommand(AddDateClick);

DeleteDate = new RelayCommand(DeleteDateClick);

Exit = new RelayCommand(ExitClick);

NewDate = DateTime.Now;

}

private void AddDateClick(object o)

{

MainViewModel.AttendanceData.AddAttandance(new DateOnly(NewDate.Year, NewDate.Month, NewDate.Day));

}

private void DeleteDateClick(object sender)

{

try

{

var st = (Attendance)sender;

MainViewModel.AttendanceData.RemoveAttendance(st.Date);

}

catch

{

//

}

}

private void ExitClick(object sender)

{

MainViewModel.AttendanceData.Save();

var win = (EditDateWindow)sender;

win.Close();

}

}

}

6) Клас “ReportModelView”

using Journal.Models;

using Journal.View;

using System;

namespace Journal.ViewModels

{

class EditDateModelView

{

public DateTime NewDate { get; set; }

public RelayCommand AddDate { get; }

public RelayCommand DeleteDate { get; }

public RelayCommand Exit { get; }

public EditDateModelView()

{

AddDate = new RelayCommand(AddDateClick);

DeleteDate = new RelayCommand(DeleteDateClick);

Exit = new RelayCommand(ExitClick);

NewDate = DateTime.Now;

}

private void AddDateClick(object o)

{

MainViewModel.AttendanceData.AddAttandance(new DateOnly(NewDate.Year, NewDate.Month, NewDate.Day));

}

private void DeleteDateClick(object sender)

{

try

{

var st = (Attendance)sender;

MainViewModel.AttendanceData.RemoveAttendance(st.Date);

}

catch

{

//

}

}

private void ExitClick(object sender)

{

MainViewModel.AttendanceData.Save();

var win = (EditDateWindow)sender;

win.Close();

}

}

}

# Додаток В

C# класи моделей

1) Клас Student

namespace Journal.Models

{

public class Student : Base

{

public int Id { get; set; }

public string Name { get; set; }

public string LastName { get; set; }

public string Group { get; set; }

public Student() { }

public Student(int id, string name, string lastName, string group)

{

Id = id;

Name = name;

LastName = lastName;

Group = group;

}

}

}

2) Клас User

using System.Linq;

namespace Journal.Models

{

public class User : Base

{

public string Login { get; set; }

public string Name { get; set; }

public string LastName { get; set; }

public string Password { get; set; }

public User() { }

public User(string login, string name, string lastName, string pass)

{

Login = login;

Name = name;

LastName = lastName;

Password = pass;

}

}

}

3) Клас Attendance

using System;

using System.Collections.Generic;

using System.Text.Json.Serialization;

namespace Journal.Models

{

public class Attendance

{

[JsonConverter(typeof(DateOnlyJsonConverter))]

public DateOnly Date { get; set; }

public Dictionary<int, Reason> Students { get; set; }

public Attendance() { }

public Attendance(DateOnly date)

{

Date = date;

Students = new Dictionary<int, Reason>();

}

}

}

4) Клас StudentData

using System.Linq;

namespace Journal.Models

{

public class StudentData : Data<Student>

{

public bool AddStudent(int id, string name, string lastName, string group)

{

if (Collection.Any(a => a.Id == id)) return false;

Collection.Add(new Student(id, name, lastName, group));

return true;

}

public bool RemoveStudent(int id)

{

if (!Collection.Any(a => a.Id == id)) return false;

Collection.Remove((Student)Collection.First(a => a.Id == id));

return true;

}

}

}

5) Кдас UserData

using System.Linq;

namespace Journal.Models

{

public class UsersData : Data<User>

{

public User? GetAccount(string login)

{

return Collection.Where(a => a.Login == login).ToList().FirstOrDefault();

}

public bool AddUser(string login, string name, string lastName, string pass)

{

if (Collection.Any(a => a.Login == login)) return false;

Collection.Add(new User(login, name, lastName, pass));

return true;

}

public bool RemoveUser(string login)

{

if (!Collection.Any(a => a.Login == login)) return false;

Collection.Remove((User)Collection.First(a => a.Login == login));

return true;

}

public bool LoginCheck(string login, string password)

{

if (GetAccount(login) is not User u) return false;

return u.Password == password;

}

}

}

6) Клас AttendaceData

using System;

using System.Linq;

namespace Journal.Models

{

public class AttendanceData : Data<Attendance>

{

public bool AddAttandance(DateOnly date)

{

if (Collection.Any(a => a.Date == date)) return false;

Collection.Add(new Attendance(date));

return true;

}

public bool RemoveAttendance(DateOnly date)

{

if (!Collection.Any(a => a.Date == date)) return false;

Collection.Remove((Attendance)Collection.First(a => a.Date == date));

return true;

}

}

public enum Reason

{

Unknown, Sick

}

}

7) Клас Base

using System.Collections.Generic;

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace Journal.Models

{

public abstract class Base

{

public event PropertyChangedEventHandler? PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string propertyName = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

protected bool Set<T>(ref T storage, T value, [CallerMemberName] string propertyName = "")

{

if (EqualityComparer<T>.Default.Equals(storage, value)) return false;

storage = value;

OnPropertyChanged(propertyName);

return true;

}

}

}

8) Клас RelayCommands

using System;

using System.Windows.Input;

namespace Journal.Models

{

public class RelayCommand : ICommand

{

private readonly Predicate<object>? canExecute;

private readonly Action<object> execute;

public RelayCommand(Action<object> execute, Predicate<object>? canExecute = null)

{

this.execute = execute;

this.canExecute = canExecute;

}

public bool CanExecute(object? parameter = null)

{

return canExecute == null || canExecute(parameter);

}

public void Execute(object? parameter = null)

{

execute(parameter);

}

public event EventHandler CanExecuteChanged

{

add => CommandManager.RequerySuggested += value;

remove => CommandManager.RequerySuggested -= value;

}

}

}

9) Клас Data

using System;

using System.Collections.ObjectModel;

using System.Globalization;

using System.IO;

using System.Text.Encodings.Web;

using System.Text.Json;

using System.Text.Json.Serialization;

using System.Text.Unicode;

namespace Journal.Models

{

public class Data<T> : IDisposable

{

private const string path = "dat/";

private readonly JsonSerializerOptions options = new()

{

WriteIndented = true,

DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull,

Encoder = JavaScriptEncoder.Create(UnicodeRanges.BasicLatin, UnicodeRanges.Cyrillic),

IncludeFields = false

};

public Data()

{

Collection = Load();

}

public ObservableCollection<T> Collection { get; set; }

public void Dispose()

{

ReleaseUnmanagedResources();

GC.SuppressFinalize(this);

}

~Data()

{

ReleaseUnmanagedResources();

}

public void Save()

{

try

{

var name = GetType().Name;

Directory.CreateDirectory(path);

using var fs = new FileStream($"{path}{name}.json", FileMode.Create);

JsonSerializer.Serialize(fs, Collection, options);

}

catch (Exception)

{

//

}

}

protected ObservableCollection<T> Load()

{

try

{

var name = GetType().Name;

using var fs = new FileStream($"{path}{name}.json", FileMode.OpenOrCreate);

return JsonSerializer.Deserialize<ObservableCollection<T>>(fs, options)!;

}

catch

{

return new ObservableCollection<T>();

}

}

protected void ReleaseUnmanagedResources()

{

Save();

}

}

public class DateOnlyJsonConverter : JsonConverter<DateOnly>

{

private const string Format = "yyyy-MM-dd";

public override DateOnly Read(ref Utf8JsonReader reader, Type typeToConvert, JsonSerializerOptions options)

{

return DateOnly.ParseExact(reader.GetString()!, Format, CultureInfo.InvariantCulture);

}

public override void Write(Utf8JsonWriter writer, DateOnly value, JsonSerializerOptions options)

{

writer.WriteStringValue(value.ToString(Format, CultureInfo.InvariantCulture));

}

}

}