**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Природничо-гуманітарний фаховий коледж**

**ДВНЗ «Ужгородський національний університет»**

***Реєстраційний номер***

***Дата реєстрації***

Валько Максим Сергійович

студент ІІ курсу

групи КН-24

денної форми навчання

особова карта №253001ФМБД

**ЗВІТНА РОБОТА**

**з навчальної практики**

**на тему:**

**Розробка консольного додатку «CatalogGenius Studio»**

|  |  |
| --- | --- |
| **Допущено до захисту** | **Науковий керівник:** |
| Дата « » 2023 р.  Підпис | **Викл. Верещагін О. О.** |

**Робота захищена «**  **»**    **2024 р.**

**з оцінкою «**  **/**  **/**  **»**

**Підписи членів комісії**

Ужгород 2024

ЗМІСТ

ВСТУП 3

[1 ТЕОРЕТИЧНА ЧАСТИНА 5](#_Toc164966789)

[1.1 Теоретичні відомості про предметну область 5](#_Toc164966790)

[1.1.1 Каталог фізичних товарів 5](#_Toc164966791)

[1.2 Опис предметної області та постановка задачі 6](#_Toc164966792)

[1.3 Огляд аналогів 7](#_Toc164966793)

[1.4 Обґрунтування вибору методу розв’язку задачі 8](#_Toc164966794)

[1.5 Загальні теоретичні відомості про використані засоби розробки 9](#_Toc164966795)

[2 ПРАКТИЧНА ЧАСТИНА 13](#_Toc164966800)

[2.1 Проектна частина 13](#_Toc164966801)

[2.1.1 Інформаційна структура моделі 13](#_Toc164966804)

[2.1.2 Опис алгоритму 13](#_Toc164966805)

[2.1.3 Функціональні частини програми 15](#_Toc164966806)

[Таблиця підсумку 17](#_Toc164966807)

[2.1.4 Фізична структура проекту 17](#_Toc164966808)

[2.1.5 Опис структури даних та використаних змінних 20](#_Toc164966809)

[2.1.6 Тестування проекту 21](#_Toc164966810)

[2.2 Експлуатаційна частина 29](#_Toc164966811)

[2.2.1 Вимоги до технічних засобів 29](#_Toc164966813)

[2.2.2 Склад інсталяційного пакету та інструкція 29](#_Toc164966814)

[2.2.3 Опис інтерфейсу 30](#_Toc164966815)

[2.2.4 Ієрархічна структура інтерфейсу 30](#_Toc164966816)

[2.2.5 Керівництво користувача та приклади використання програмного продукту 30](#_Toc164966817)

[ВИСНОВКИ 31](#_Toc164966818)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 32](#_Toc164966819)

[Додаток 1. Блок-схема алгоритму 33](#_Toc164966820)

[Додаток 2. Сценарій роботи програми 35](#_Toc164966821)

[3 Лістинг програми: 36](#_Toc164966822)

Темою навчальної практики є розробка консольного додатка "CatalogGenius Studio" для створення фізичних каталогів товарів.

Метою даної навчальної практики є навчитись розробляти консольні додатки з використанням мови програмування C# та середовища розробки Visual Studio 2019. Оволодіти навичками роботи з зовнішніми файлами для забезпечення ефективного створення та управління фізичними каталогами товарів.

Завдання − Створення консольного додатка "CatalogGenius Studio", який надасть можливість зручного створення, оновлення та управління фізичними каталогами товарів. Додаток повинен забезпечувати введення, виведення та редагування інформації про товари, а також ефективну роботу з зовнішніми файлами для зберігання даних.

Фізичні каталоги товарів в епоху цифрових технологій, фізичні каталоги товарів залишаються актуальними завдяки тактильному взаємодії, стимулюванню креативності, можливості перегляду без інтернет-з'єднання, ефективності у стимулюванні покупок, а також здатності представляти концепції бренду та підвищувати свідомість про них, надаючи бізнесу широкий інструментарій для взаємодії з аудиторією та створення запам'ятовувального враження.

Об’єктом дослідження є мова програмування високого рівня, та її застосування для створення додатків.

Предметами дослідження є мова програмування C#, середовище розробки Visual Studio 2022 => Windows Presentation Foundation (WPF), робота із максимально зручним та зрозумілим дизайном.

# **ТЕОРЕТИЧНА ЧАСТИНА**

## **Теоретичні відомості про предметну область**

### Каталог фізичних товарів

Фізичні каталоги товарів – зручний і доступний інструмент для отримання інформації про різноманітні товари. На відміну від електронних платформ, вони зберігаються у паперовій або іншій матеріальній формі, надаючи користувачам можливість огляду та порівняння продукції без використання електронних пристроїв. Це забезпечує зручний доступ до широкого асортименту товарів, а також використання каталогів у ділових відносинах для представлення асортименту партнерам і клієнтам. Можливість експорту інформації та порівняльні переваги використання фізичних каталогів роблять їх універсальним інструментом в сучасному бізнесі та споживацькому середовищі. [1]

## **Опис предметної області та постановка задачі**

* Даний каталог працює як звичайна база данних зберігаюча в собі велику кількість товарів.
* В даному додатку має бути реалізовано авторізація, навігація.
* Постановка задачі для цієї навчальої практики це створення дизайну для додатку, авторізацію та базової навігації. По можливості зробити адаптивність.

Основні функції застосунку «CatalogGenius Studio» можуть включати в себе:

* Виведення інформації з текстових файлів;
* Навігація між вікнами;
* Видалення;
* Додавання;
* Інший функціонал такий як редагування данних буде реалізовано в курсовій роботі);

Для зручності користувачів важливо розробити зрозумілий та простий інтерфейс користувача з підтримкою красивого дизайну та ергономічного розміщення елементів. В даному випадку розробка інтерфейсу може бути здійснена за допомогою технології WPF (Windows Presentation Foundation) в середовищі розробки Visual Studio [2].

## Огляд аналогів

За аналог було взято сайт «Каталог товарів TRX» та сайт «Rost» [9,10].

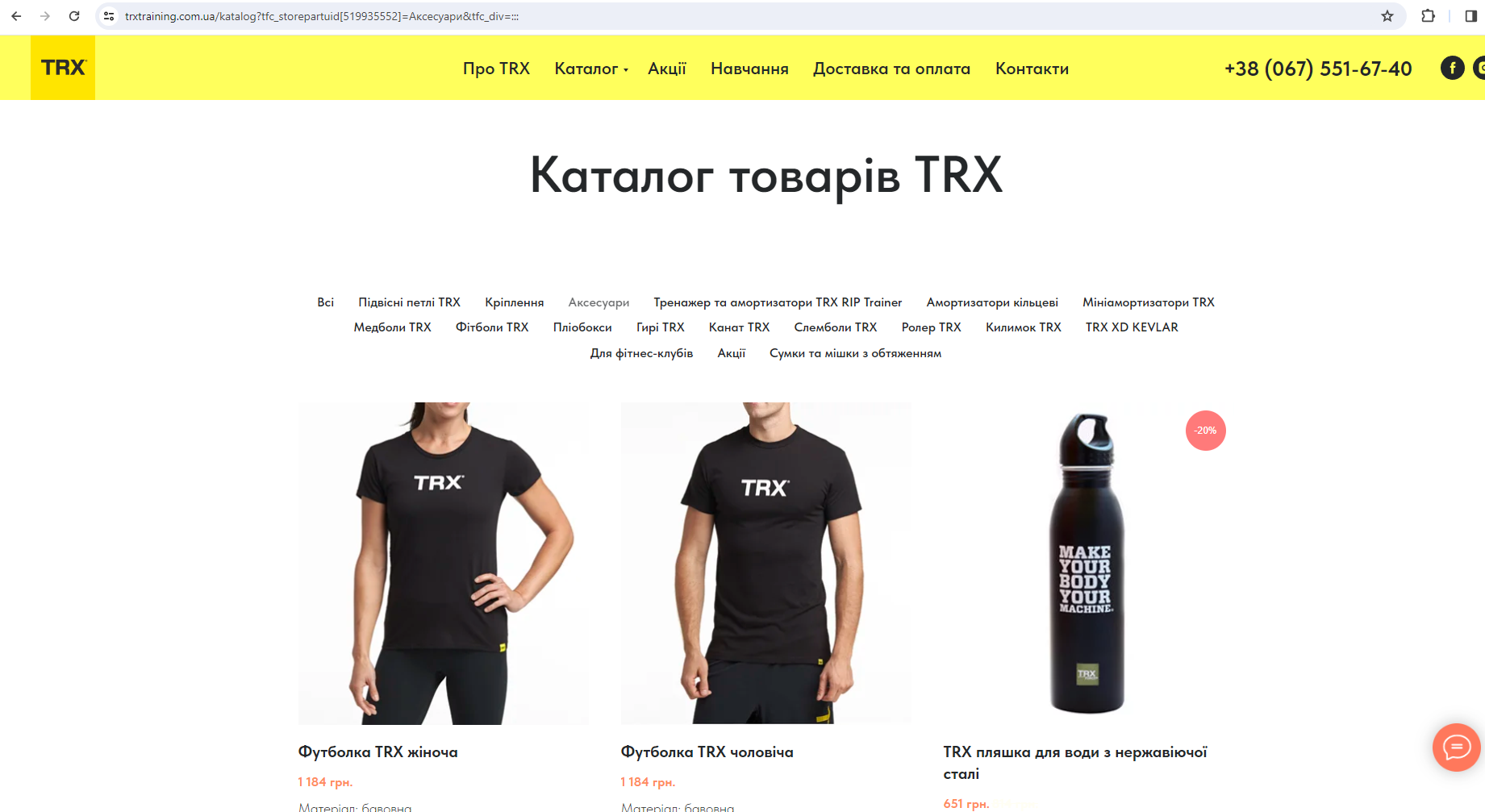


Рисунок 1.2 Сайт «Каталог товарів TRX»

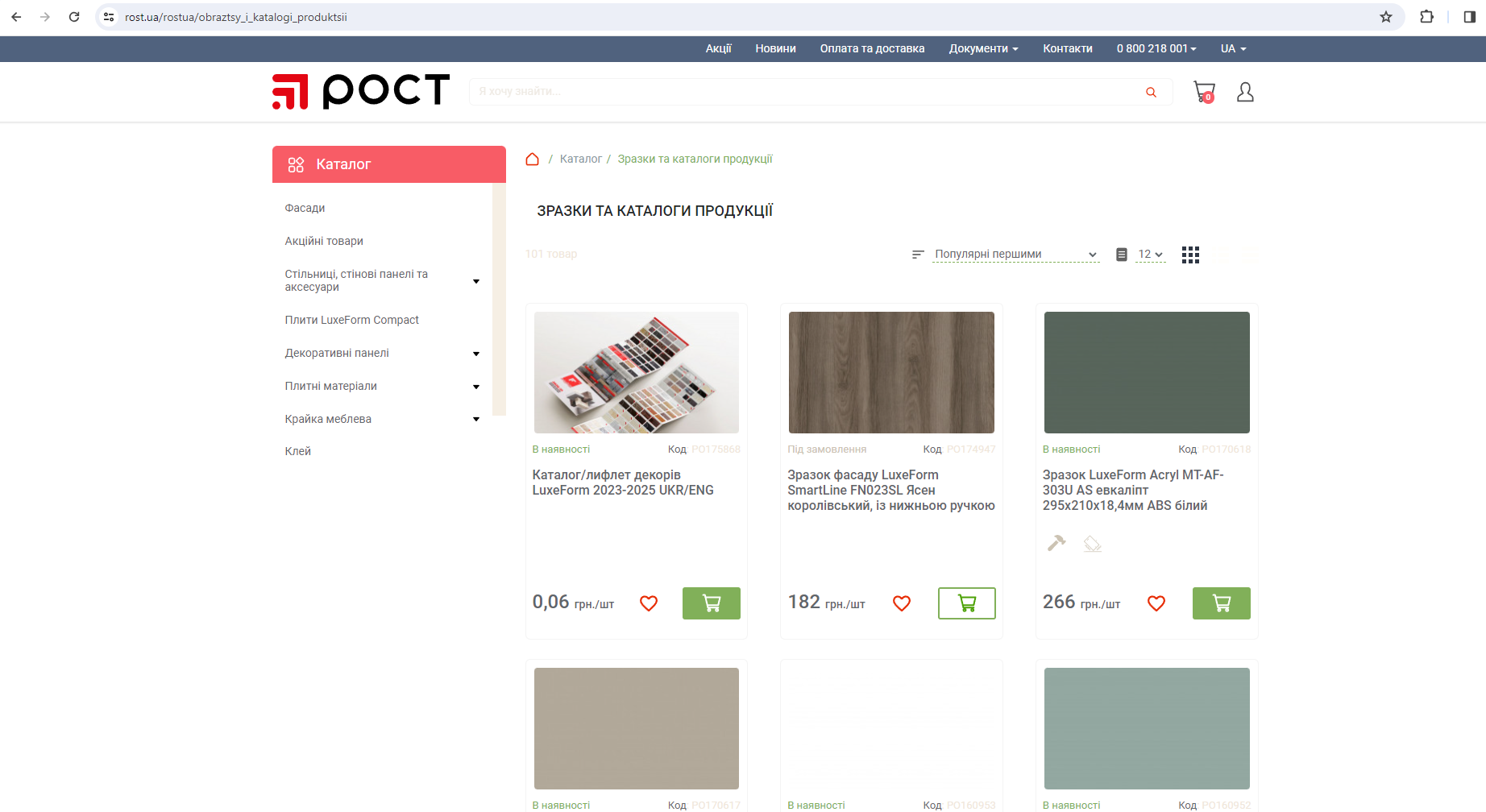


Рисунок 1.3 Сайт «Rost»

Плюси: Сайт працює з візуальними данними, а не тільки текстовими. Легкий дизайн, швидка робота сайту, зручна та легка навігація.

Мінуси: Сайт неможе працювати без інтернету та містить в собі багато реклами. На сайті неможна редагувати дані.

## Обґрунтування вибору методу розв’язку задачі

Обсяг даних, які необхідно зберігати, не є дуже великим. Якщо використовувати базу даних, може виникнути зайва складність в управлінні зв'язками між таблицями та запитами до бази даних.

Формат файлів, що використовується (XML), є досить зручним для зберігання даних, що мають ієрархічну структуру, таку як адресна книга.

Наявність резервних копій даних, які можна зберігати на зовнішньому носії, такому як флешка або хмарне сховище.

Як зберігаються файли (наприклад, картинки)? Чому саме так?

У випадку з XML-файлами, вони зберігаються у текстовому форматі, що дозволяє зручно переглядати та редагувати дані, а також зберігати їх у резервних копіях. В картинок та інших бінарних файлів, які можуть зберігатися в адресній книзі, зазвичай використовують формати зображення, такі як JPEG або PNG. Такі формати забезпечують зменшення обсягу файлів при зберіганні, що зменшує час доступу до файлів та займає менше місця на диску[7].

* Binding (Прив'язка): WPF пропонує потужний механізм прив'язки даних, що дозволяє зв'язувати властивості елементів інтерфейсу з даними з джерела даних, такими як об'єкти C# або файли XML. Це дозволяє автоматично оновлювати інтерфейс при зміні даних.
* Styles (Стилі): WPF дозволяє описувати стилі для елементів інтерфейсу, що дозволяє змінювати зовнішній вигляд декількох елементів інтерфейсу за одним разом.
* Templates (Шаблони): WPF пропонує шаблони для елементів інтерфейсу, що дозволяє змінювати структуру та зовнішній вигляд елементів інтерфейсу.
* Control Customization (Кастомізація елементів управління): WPF дозволяє створювати власні елементи управління, що дозволяє реалізовувати унікальну функціональність та зовнішній вигляд.

## **Загальні теоретичні відомості про використані** засоби розробки



**WPF c#**

C# (вимовляється як «Сі-шарп») - це об'єктно-орієнтована мова програмування, яку створила компанія Microsoft. Вона була розроблена на основі мов C++ та Java, та має ряд переваг перед ними.

Однією з головних переваг C# є простота синтаксису та легкість вивчення. Синтаксис мови схожий на синтаксис мови Java, та вона має багато спільного з іншими мовами програмування, що робить перехід між ними легким.

C# також є статично-типізованою мовою програмування, що означає, що всі змінні повинні бути оголошені заздалегідь з визначенням їх типу, що дозволяє покращити безпеку програми та попереджує баги, пов'язані з неправильним типом даних.

Іншою важливою особливістю C# є його підтримка для платформи .NET, що надає розробникам доступ до багатої бібліотеки класів .NET Framework, яка дозволяє розробляти широкий спектр програм з різними функціональними можливостями.

У загальному, C# є потужною та популярною мовою програмування, яка дозволяє розробникам створювати високоякісні та ефективні програми для різних платформ.

Windows Presentation Foundation (WPF) - це технологія розробки Windows-додатків, яка дозволяє розробникам створювати красиві, динамічні та ефективні інтерфейси користувача. WPF була розроблена компанією Microsoft і вперше вийшла у 2006 році.

Основна перевага WPF полягає в тому, що вона дозволяє розробникам відокремити логіку додатка від його візуального представлення. Це дозволяє створювати більш складні та динамічні інтерфейси, які можуть реагувати на дії користувача та змінювати свій вигляд в залежності від умов.

WPF включає в себе такі елементи, як вікна, кнопки, тексти, таблиці, списки та інші. Кожен елемент може бути налаштований окремо, що дозволяє створювати унікальний та персоналізований вигляд інтерфейсу.

Узагальнюючи, WPF є потужним інструментом для створення красивих та динамічних інтерфейсів користувача, що дозволяє розробникам додатків зосередитися на функціональності програми, не витрачаючи багато часу на розробку візуальної частини.

WPF надає розробникам можливість працювати з дизайном через код XAML, який є декларативною мовою опису візуального представлення інтерфейсу. Це дозволяє розробникам відокремлювати логіку програми від візуального представлення, що робить розробку більш ефективною та простішою.

**Робота з фалами xaml WPF c#**

Файли XAML в WPF є основним інструментом для візуального опису інтерфейсу користувача. Вони містять опис елементів інтерфейсу користувача та їхні властивості, що використовуються для створення вікон, кнопок, полів вводу, списків тощо.

У C# робота з файлами XAML зазвичай включає в себе дві операції: зчитування XAML-файлу та створення візуальних елементів на основі його вмісту, а також оновлення властивостей елементів при необхідності.

Зчитування XAML-файлу в C# можна здійснити за допомогою класу System.Windows.Markup.XamlReader, який здійснює розбір XAML-файлу та повертає об'єкт, який можна використовувати для побудови вікна.

У загальному, робота з файлами XAML в WPF дозволяє розробникам створювати візуально привабливі та функціональні інтерфейси користувача, які можна легко управляти за допомогою C#.

**Grid та робота з xaml WPF c#**

Grid - це один з основних елементів макетування в WPF, який дозволяє організовувати вміст у вигляді таблиці з рядками та стовпцями. Grid зазвичай використовується для розміщення елементів інтерфейсу користувача на вікні та налаштування їх позиції та розміру.

У XAML Grid визначається за допомогою тега <Grid>, а рядки та стовпці визначаються за допомогою тегів <RowDefinition> та <ColumnDefinition> відповідно. Кожна RowDefinition визначає один рядок

Для розміщення елементів інтерфейсу користувача в Grid можна використовувати атрибути Grid.Row та Grid.Column, які вказують, в якому рядку та стовпці має розміститись елемент.

У C# робота з Grid виконується за допомогою класу Grid, який дозволяє динамічно додавати та видаляти рядки та стовпці, а також налаштовувати розміщення елементів в Grid.

Крім того, Grid має додаткові властивості та методи для налаштування розмірів та позицій рядків та стовпців, налаштування відступів між елементами та інші. Це дозволяє створювати більш складні макети в WPF з великою кількістю елементів.

**Blend Visual Studio**

Blend для Visual Studio (раніше відомий як Microsoft Expression Blend) - це інструмент для розробки графічних інтерфейсів користувача (GUI) для веб- та настільних додатків, розроблений компанією Microsoft. Він є частиною родини продуктів Visual Studio та інтегрується з Visual Studio як розширення.

Blend для Visual Studio підтримує різні технології, включаючи XAML, HTML, CSS, JavaScript та C#. Це дозволяє розробникам та дизайнерам працювати разом безперешкодно, щоб створювати візуально привабливі та реагуючі додатки.

Однією з ключових функцій Blend для Visual Studio є можливість створювати динамічні та інтерактивні інтерфейси користувача. За допомогою підтримки анімацій, сценаріїв та часових ліній, дизайнери можуть створювати залучаючі та захоплюючі враження для користувачів.

Ще одна корисна функція Blend для Visual Studio - це можливість створювати адаптивні макети. Це дозволяє розробникам проектувати інтерфейси, які можуть адаптуватися до різних розмірів екрану та роздільної здатності, зробивши їх додатки більш реагуючими та мобільно-дружніми.

Blend для Visual Studio також включає низку вбудованих елементів керування та шаблонів, які розробники можуть використовувати для швидкого створення типових елементів інтерфейсу, таких як кнопки, тексти тощо.

# **ПРАКТИЧНА ЧАСТИНА**

## **Проектна частина**



### Інформаційна структура моделі

* В програмі використовуються лише txt файли. Вони використовуються при кожному вводі даних. Наприклад при виводі контактів. Також при редагуванні через програму файли виводяться й там можна вводити зміни.

### Опис алгоритму

Сценарій роботи програми подано в додатку 2.

Перше що бачить користувач коли відкриває програму це вікно авторизації, якщо в користувача немає аккаунту то він має можливість його зареєструвати натиснувши на RegisterButton «Зареєструвати аккаунт».

* В першому вікні AuthorisationWindow розміщені 1 TextBox, 1 PasswordBox, 2 Label, 2 Button. При вводі інформації в TextBox та PasswordBox

Треба натиснути на LoginButton який перевіряє:

* Чи логін або пароль введені вірно.
* Якщо пароль та логін не правильні то програма виводить messegebox та недає доступ до поки небуде введено правильний логін або пароль MainWindow.
* Якщо пароль та логін правильні то програма виводить messegebox та відкривається MainWindow.
* При натисканні на RegisterButton відкривається RegisterWindow в якому розміщені 1 TextBox, 2 PasswordBox, 3 Label, 3 Button. При вводі інформації в TextBox та PasswordBox

Після входу в систему користувача перекидує на головне вікно в якому знаходяться наступні компоненти:

* Кнопка «Menu» відкриває список товарів в католозі.
* Кнопка «Add Catalog» відкриває сторінку на якій можна додати або видалити категорію.
* Кнопка «Relogin» повертає на вікно авторизації та виходить з акаунта.
* Кнопка «Exit» закриває додаток.

### Функціональні частини програми

Описуєте структуру проекту, починаючи із самого верхнього рівня (App) і поступово розкриваючи підпорядковані. (всі файли з розширенням xaml та cs). Сюди також треба додати скрін проекту:

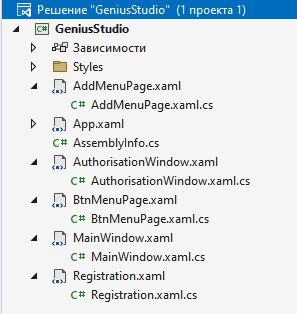


Рисунок 2.1 Структура проекту

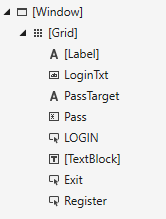


Рисунок 2.2 Об’єкти розміщені на AuthorisationWindow

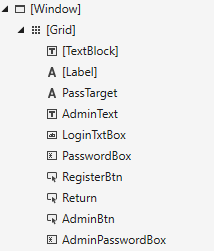


Рисунок 2.3 Об’єкти розміщені на Registration Window

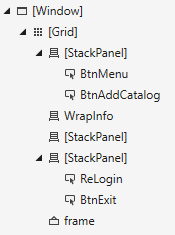


Рисунок 2.4 Об’єкти розміщені на MainWindow

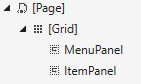


Рисунок 2.5 Об’єкти які розміщені на BtnMenuPage

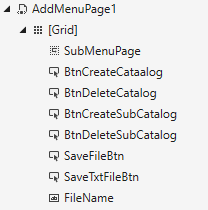


Рисунок 2.5 Об’єкти які розміщені на AddMenuPage

Таблиця підсумку

Таблиця 2.1

|  |  |
| --- | --- |
| **Назва елементу управління** | **Кількість** |
| Button | 16 |
| TextBox | 2 |
| PasswordBox | 3 |
| TextBlock | 2 |
| Label | 4 |
| StackPanel | 2 |
| Frame | 1 |

### Фізична структура проекту

Головна папка, в якій знаходяться усі файли програми називається «CatalogGenius Studio». В ній знаходиться файл CatalogGenius Studio.sln та іще одна папка «CatalogGenius Studio»

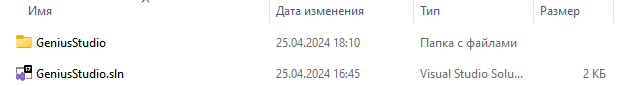


Рисунок 2.6 Bміст папки «CatalogGenius Studio»

В папці CatalogGenius Studio знаходяться 2 папки та 2 файла.

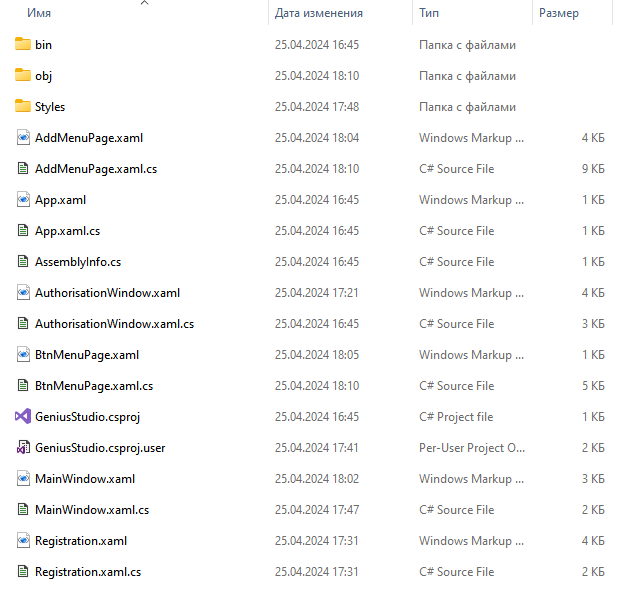


Рисунок 2.7 Вміст папки «CatalogGenius Studio»



Рисунок 2.8 Вміст папки «bin»



Рисунок 2.9 Вміст папки «Debug»

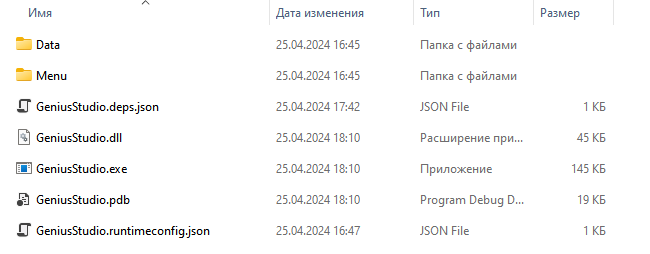


Рисунок 2.10 Вміст папки «net6.0-windows»

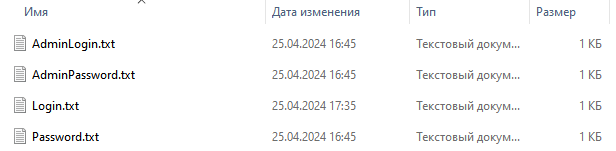


Рисунок 2.11 Вміст папки «Data»

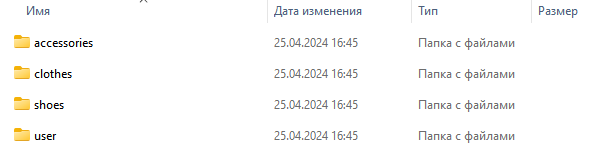


Рисунок 2.13 Вміст папки «Menu»



Рисунок 2.14 Вміст папки «Resources»

В папках «Debug» знаходяться паки «net6.0». В папках «net6.0» знаходяться усі файли потрібні для правильного функціонування програми.

### Опис структури даних та використаних змінних

* дати табличку всіх змінних, або структур;
* вказати область дії, належніть, тип, призначення;
* можна згрупувати по класах.

Таблиця 2.2 Використані елементи

|  |  |
| --- | --- |
| **Назва елементу властивості** | **Властивості, Тип та дії** |
| Grid.Row та Grid.Column | є властивостями, які вказують рядок і стовпець, відповідно, в якому елемент міститься у Grid (сітці). |
| Height та Width | властивості розміру елемента в пікселях, відповідно висоти та ширини елемента. |
| Margin та Padding | властивості, які вказують відстань між межами елементів та внутрішньої відстані між вмістом елемента та його межами відповідно. |
| ToolTip | властивість, яка встановлює текст, що відображається, коли користувач наводить курсор миші на елемент. |
| FontSize та FontFamily | властивості, що встановлюють розмір та тип шрифту для текстового вмісту. |
| Foreground та Background | властивості, які встановлюють колір тексту та тла елемента відповідно. |
| Content | властивість, яка вказує текстовий вміст елемента. |
| BorderThickness | властивість, яка вказує товщину межі елемента. |
| VerticalAlignment HorizontalAlignment | властивості, які встановлюють вертикальне та горизонтальне вирівнювання елемента відносно його контейнера. |
| Click | це подія, яка виникає, коли користувач клацає на елементі мишею або натискає на нього за допомогою клавіатури. Вона використовується для відслідковування натискання користувача на елемент, щоб виконати певну дію. |

### Тестування проекту

Тестування валідації.

Тестування нормальної роботи інтерфейсу.

Тестування тригерів (наведення на елементи управління).

**Вікно авторизації:**

* Тригери

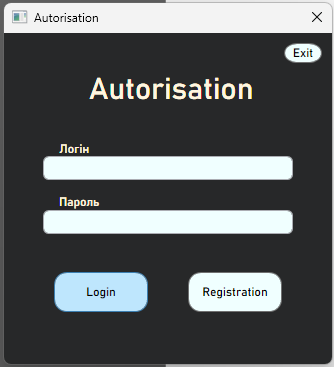


Рисунок 2.15 Тестування тригерів (Наведення на Login)

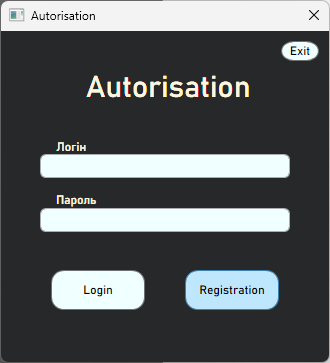


Рисунок 2.16 Тестування тригерів (Наведення на Register)

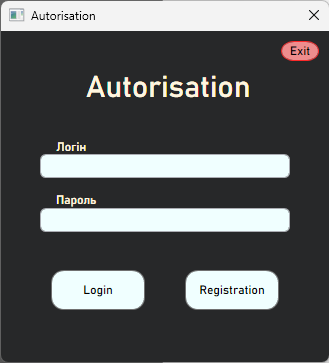


Рисунок 2.17 Тестування тригерів (Наведення на Exit)

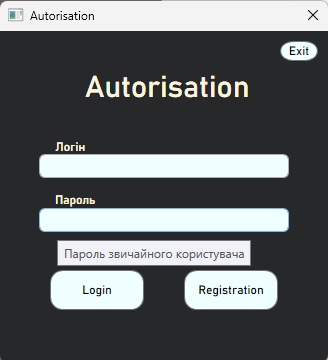


Рисунок 2.18 Тестування тригерів (Наведення на Password)

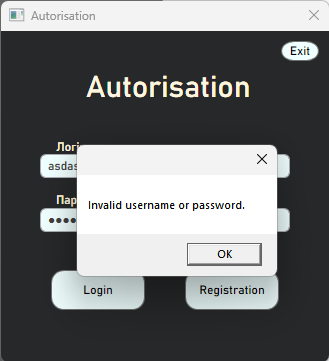


Рисунок 2.19 Невірний логін або пароль

**Вікно реєстрації:**

* Тригери

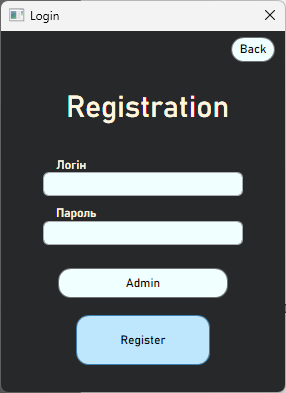


Рисунок 2.20 Тестування тригeрів (Наведення на Register)

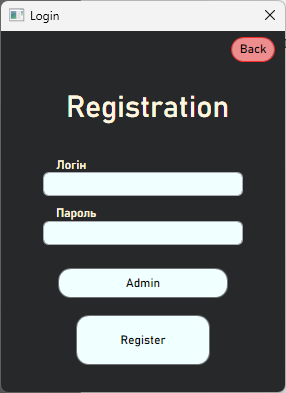


Рисунок 2.21 Тестування тригерів (Наведення на Back)

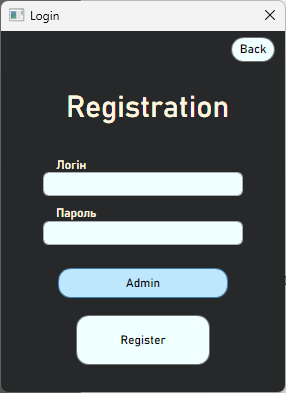


Рисунок 2.22 Тестування тригерів (Наведення на Admin)

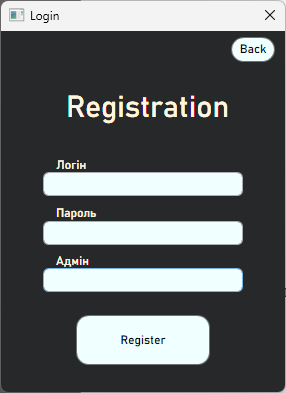


Рисунок 2.23 Тестування тригерів (Натискання на Admin)

**Основне вікно:**

* Тригери

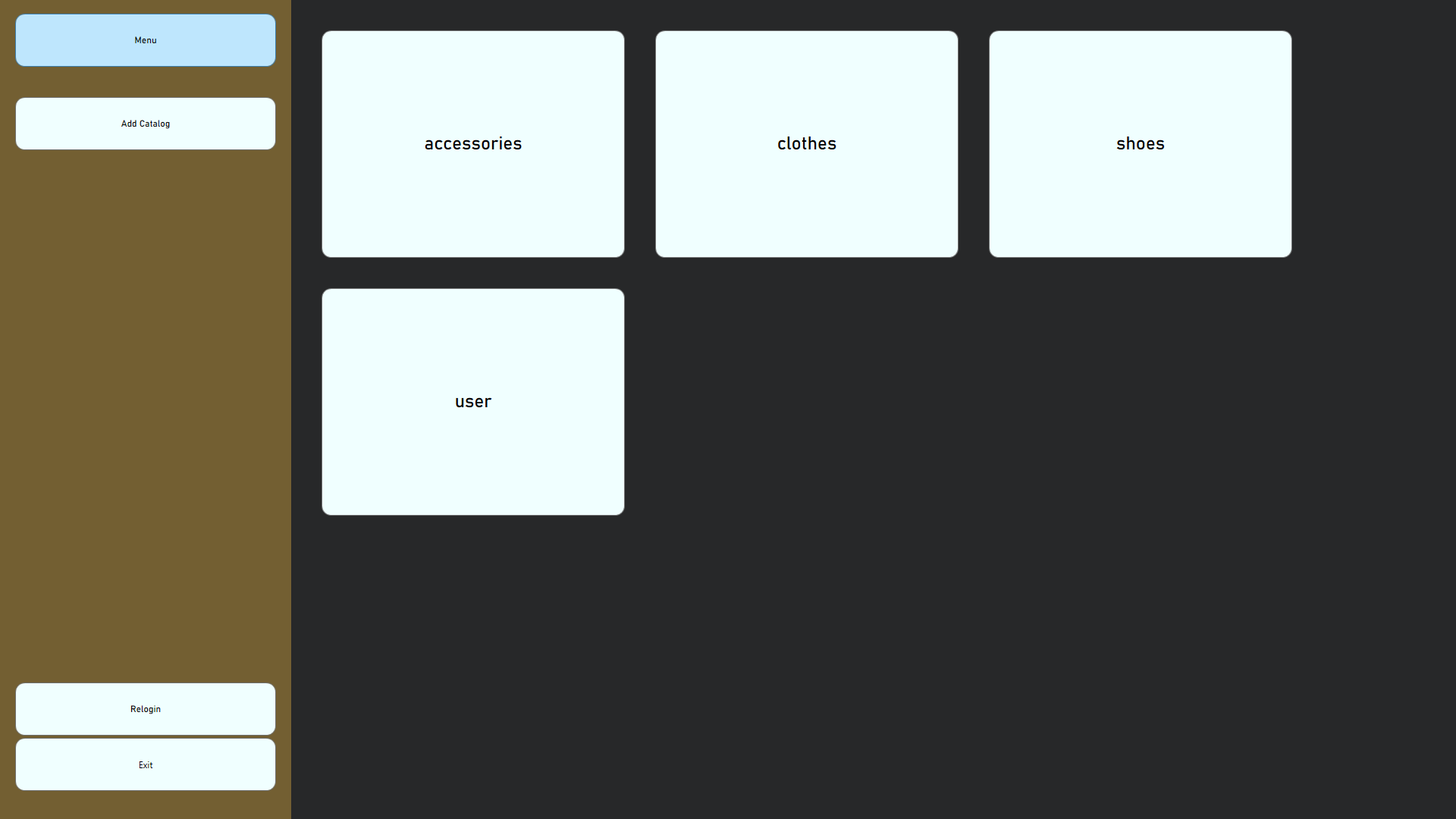


Рисунок 2.26 Тестування тригерів (Наведення на Menu)

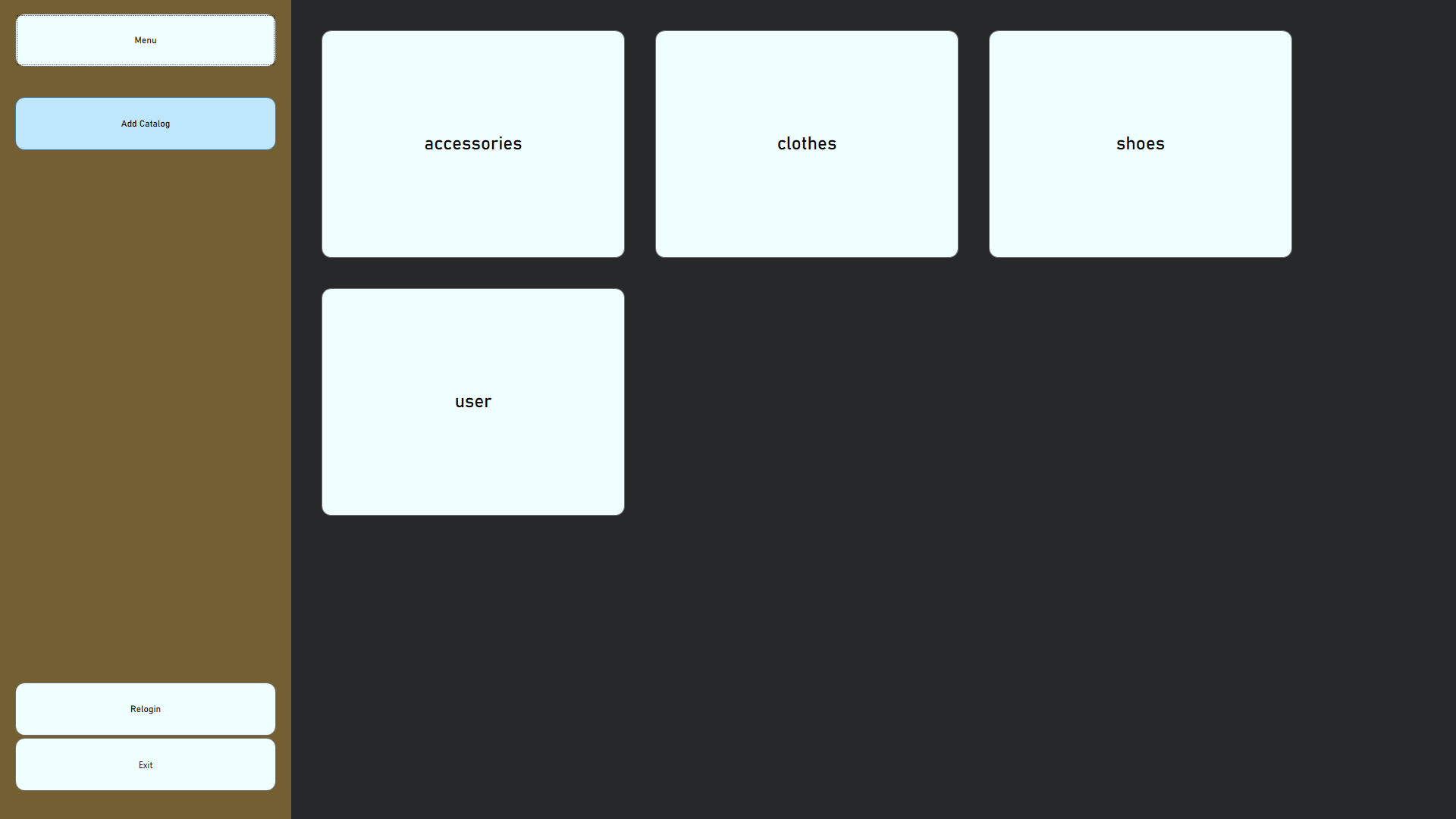


Рисунок 2.27 Тестування тригерів (Наведення на Add Catalog)

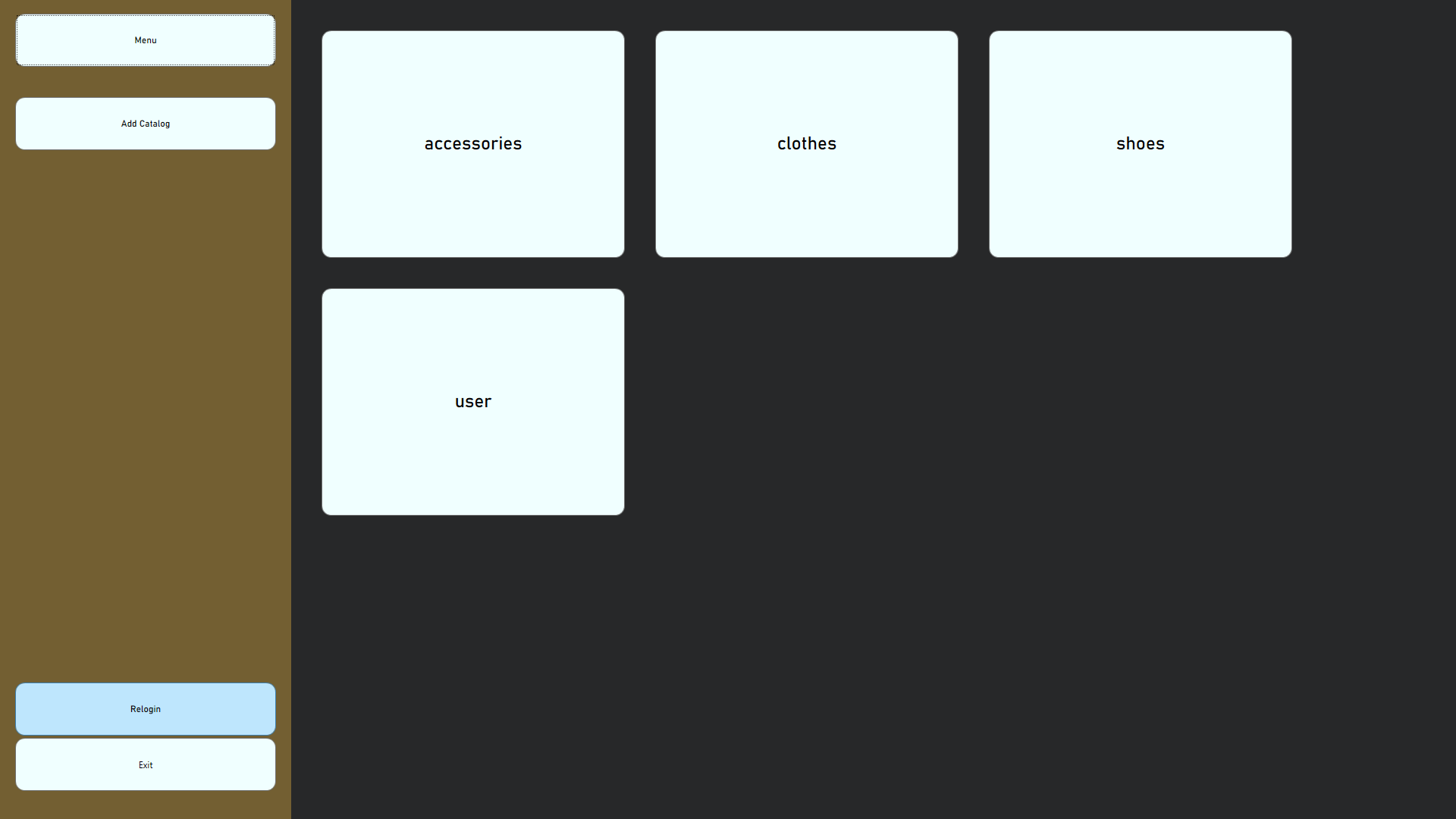


Рисунок 2.28 Тестування тригерів (Наведення на Relogin)

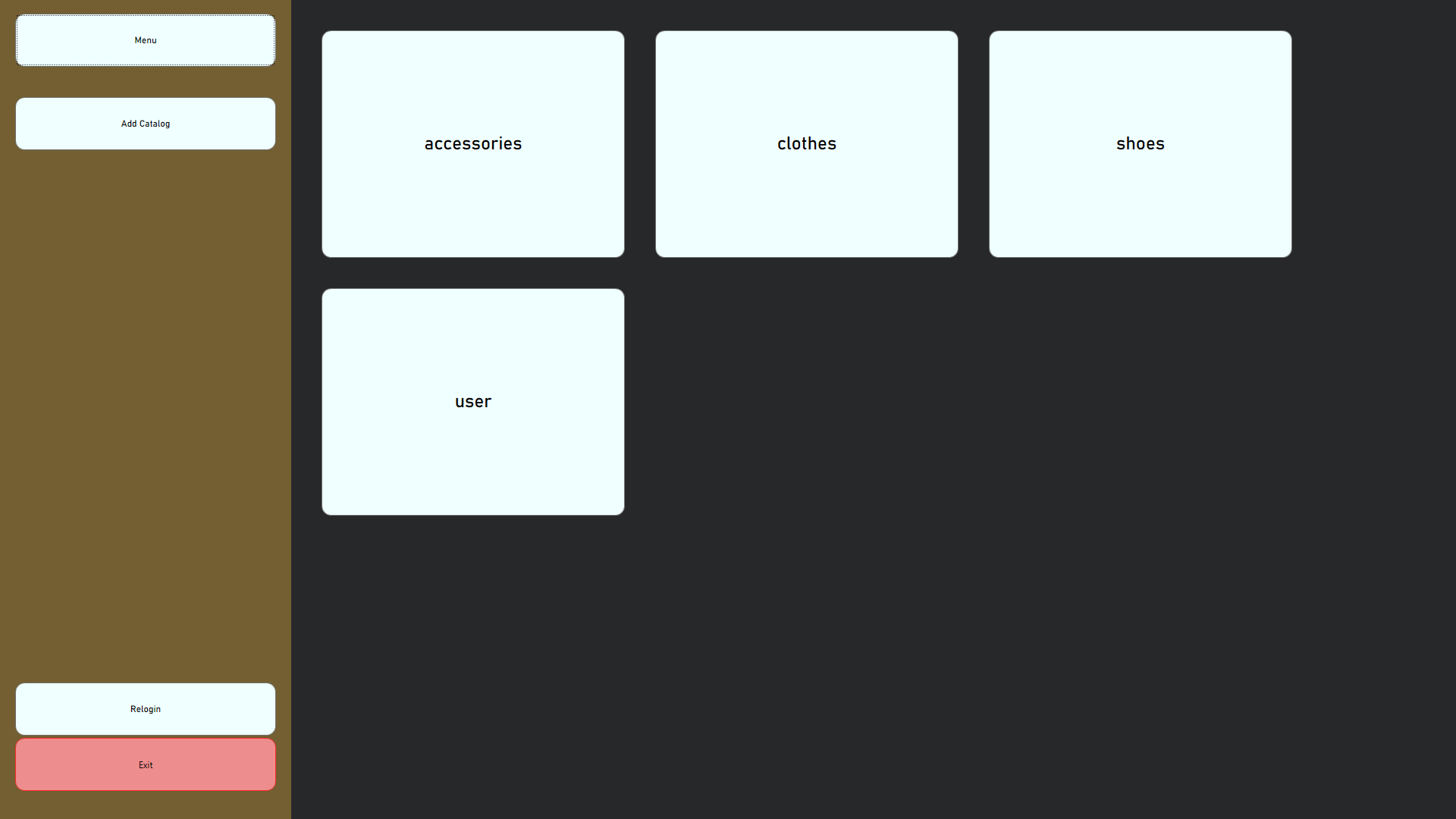


Рисунок 2.29 Тестування тригерів (Наведення на Exit)

**Сторінки:**

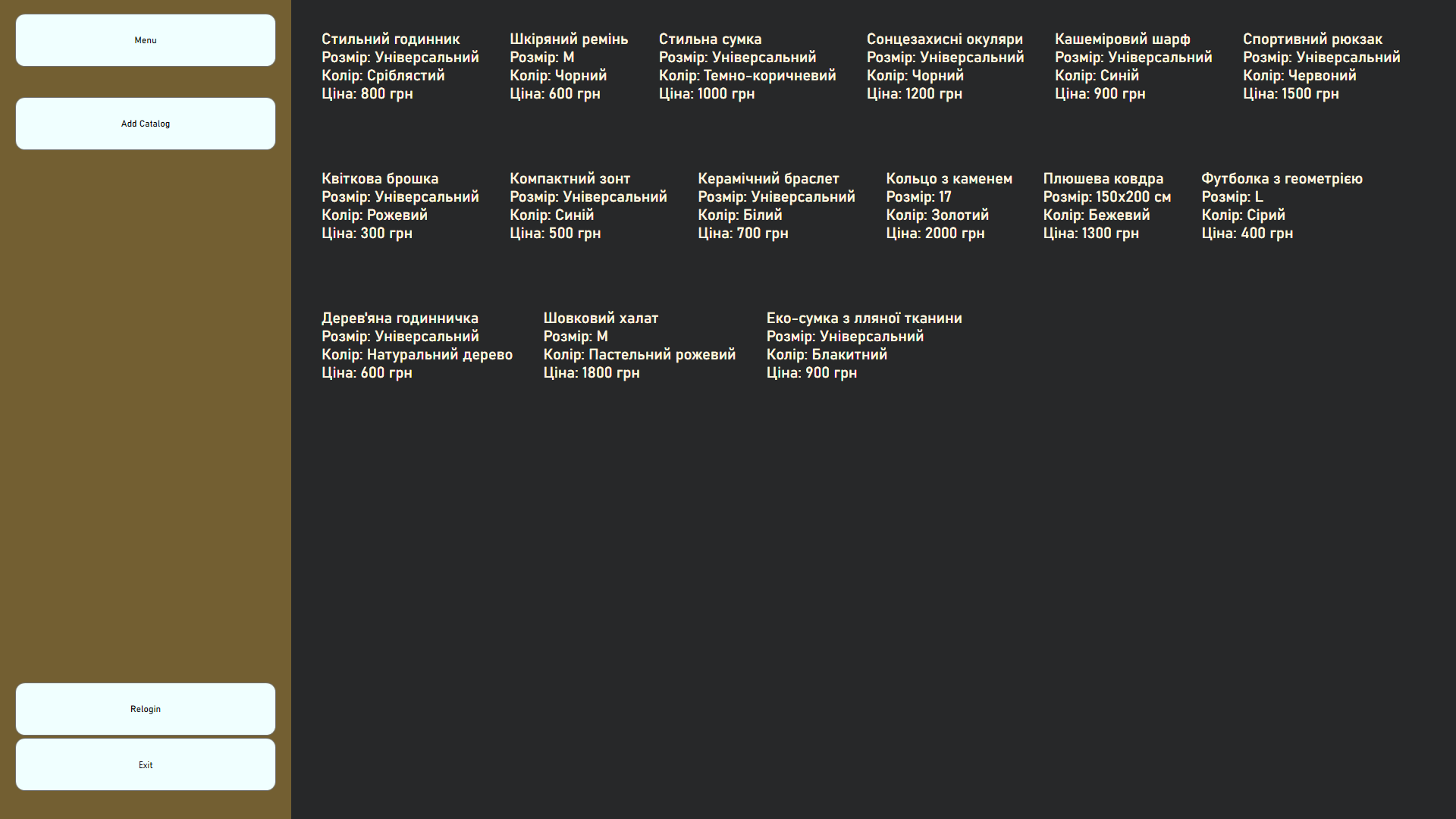


Рисунок 2.30 Сторінка Menu



Рисунок 2.31 Сторінка Add Catalog

## Експлуатаційна частина



### Вимоги до технічних засобів

Для коректної роботи програми має бути установлена платформа .NET

### Склад інсталяційного пакету та інструкція

Для роботи програми потрібні наступні файли(Див. Рисунок 2.32). А також щоб в додатку не виникало помилок має бути встановлені усі файли папки «CatalogGenius Studio»().

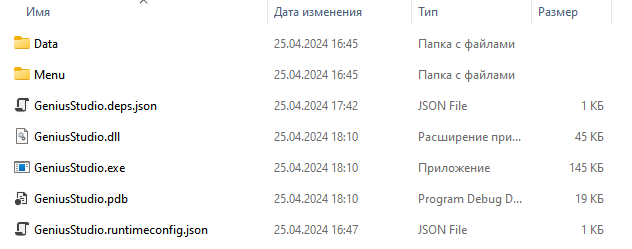


Рисунок 2.33 Склад інсталяційного пакету

Для початку роботи програми користувачу потрібно:

* Завантажити всі необхідні файли.
* Запусти файл CatalogGenius Studio.exe.
* Почати користуватись додатком

### Опис інтерфейсу

Запустивши додаток перед нами постає авторизація (Рисунок 2.15).

Тут ви маєте ввійти в свій аккаунт або створити його натиснувши на RegisterButton.

Після цього відкривається вікно реєстрації (Рисунок 2.20).

Де потрібно ввести данні правильно щоб не було помилок і після цього в вікні авторизації ввійти в аккаунт по цим самим данним.

Після успішного входу відкриється головне вікно з якого можна відкривати ще деякі вікна. (Рисунок 2.26).

Сторінка Menu (Рисунок 2.30). Сторінка Add Catalog (Рисунок 2.31).

Якщо ви натиснете на кнопку виходу, то ви виходите з програми.

### Ієрархічна структура інтерфейсу

Авторизація==Реєстрація:  
Основне вікно:

* Menu
* Add Catalog
* Relogin
* Exit

### Керівництво користувача та приклади використання програмного продукту

Для того щоб відкрити додаток клацніть мишкою два рази по файлу CatalogGenius Studio.exe. Перед вами відкриється авторизація.

Після входу в систему у користувача вибір:

* Menu – кнопка, яка при натисканні відкриває сторінку на якій можна переглянути товари.
* Add catalog – кнопка, яка при натисканні відкриває сторінку на якій можна додати або видалити товари.
* Relogin – кнопка, яка при натисканні повертає вас на авторизацію.
* Exit – кнопка, яка при натисканні виходить з додатку.

ВИСНОВКИ

Під час розробки додатку "CatalogGenius Studio" для створення фізичних каталогів товарів була вибрана стратегія, спрямована на глобальний ринок. Програма дозволяє легко виводити та обробляти інформацію з різних джерел, спрощуючи процес створення та управління каталогами.

В ході проекту було реалізовано ефективний вивід даних з різноманітних форматів, а також імплементовані різноманітні функції для зручного маніпулювання цією інформацією. Виникла необхідність в роботі з різними розділовими символами, але було розроблено єдиний механізм, який дозволяє гнучко впоратися з різними форматами даних.

Важливим аспектом стало уникнення створення різних варіантів виведення за допомогою різних OutputData, що сприяє більшій ефективності та структурованості розробленого програмного рішення.

Дизайн програми розроблявся з фокусом на простоті та чіткості, щоб забезпечити легке сприйняття та зручне користування. Використання Visual Studio виявилося ключовим для зручного програмування та відладки, що позитивно позначилося на продуктивності розробки.

Робота над "CatalogGenius Studio" викликала позитивні враження, і подальший розвиток навичок у створенні консольних додатків для обробки та візуалізації інформації виглядає дуже зацікавлюючим.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

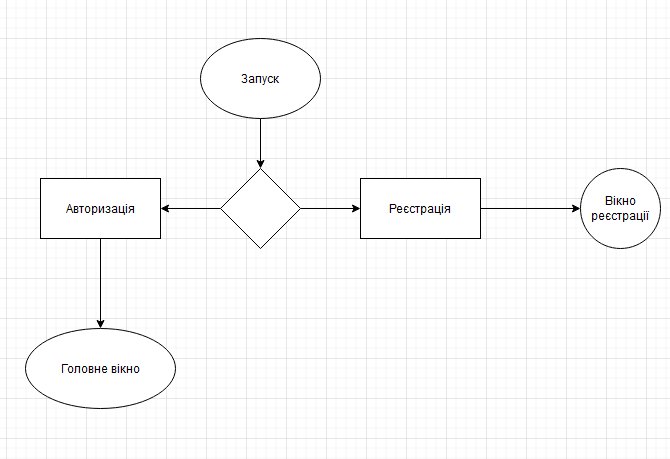
1. Двадцять сім правил джентльмена XIX сторіччя – урок для драматургів ХХІ-го. URL: https://dramaturg.org.ua/ двадцять-сім-правил-джентльмена-xix-стор/ (дата звернення: 20.04.2024).
2. Java | Класс File. Работа с файлами и каталогами. METANIT.COM - Сайт о программировании. URL: https://metanit.com/java/tutorial/6.11.php (дата звернення: 21.04.2024).
3. FileInfo (System.IO). Microsoft Learn: Build skills that open doors in your career.URL:https://learn.microsoft.com/ru-ru/dotnet/api/system.io.fileinfo?view=net-7.0 (дата звернення: 21.04.2024).
4. Учасники проектів Вікімедіа. C Sharp – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/C\_Sharp (дата звернення: 21.04.2024).
5. Каталог товарів TRX. URL: https://trxtraining.com.ua/katalog?tfc\_storepartuid[519935552] (дата звернення: 14. 01.2024).
6. Rost. URL: https://rost.ua/rostua/obraztsy\_i\_katalogi\_produktsii (дата звернення: 14. 01.2024).
7. Учасники проектів Вікімедіа. Microsoft Visual Studio – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Microsoft\_Visual\_Studio (дата звернення: 22.04.2024).
8. 23 правила поведінки справжнього джентльмена, які необхідно знати кожному чоловікові. URL: https://www.sneznoe.com/articles/muzhchiny/23-pravila-dzhentlmena.html (дата звернення: 22.04.2024).
9. Tutorial: Create a simple WPF application with C#. URL:

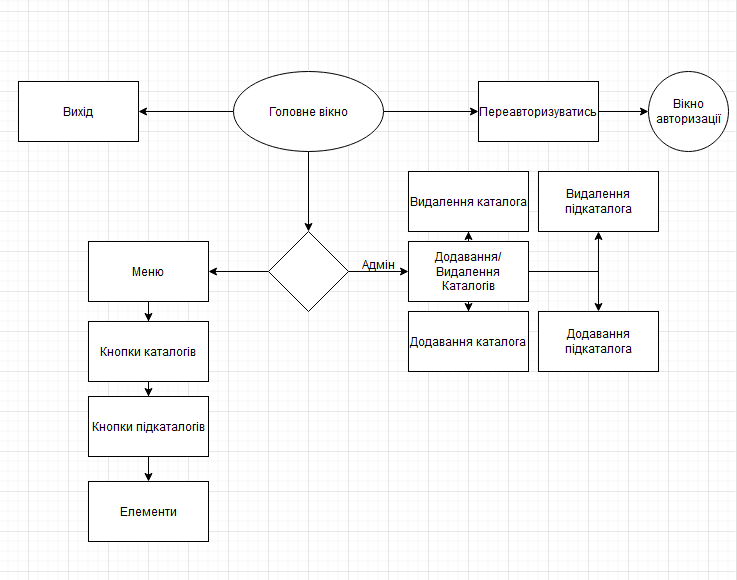
https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2022 (дата звернення: 23.04.2024).

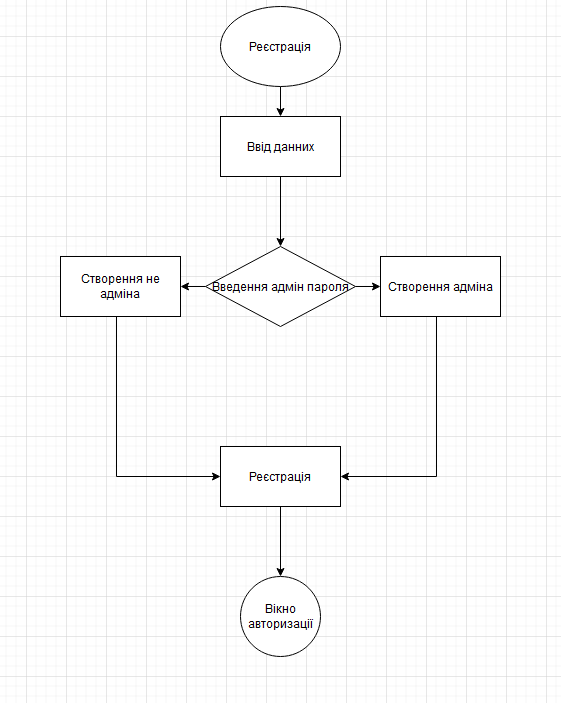
1. Grid inside Grid in XAML. URL:

https://stackoverflow.com/questions/3185485/grid-inside-grid-in-xaml (дата звернення: 24.04.2024).

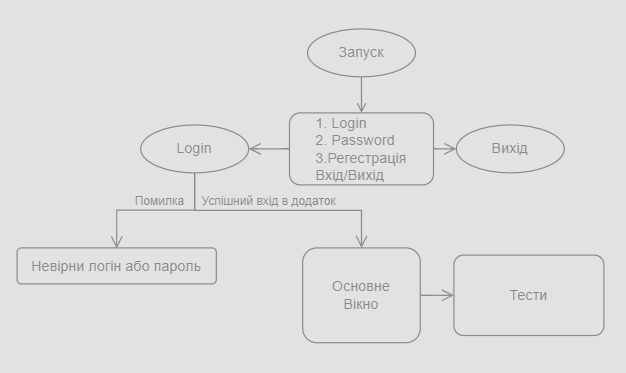
Додаток 1. Блок-схема алгоритму







Додаток 2. Сценарій роботи програми



# Лістинг програми:

AuthorisationWindow.xaml

<Window

x:Class="GeniusStudio.WinLog"

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:local="clr-namespace:GeniusStudio"

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

Title="Autorisation"

Width="344"

Height="370"

ResizeMode="NoResize" WindowStartupLocation="CenterScreen"

Background="#272829"

mc:Ignorable="d">

<Grid>

<Label

Foreground="#FFF6E0"

Margin="50,104,0,0"

HorizontalAlignment="Left"

VerticalAlignment="Top"

FontFamily="Bahnschrift"

Target="{Binding ElementName=LoginTxt}">

\_Логін

</Label>

<TextBox

x:Name="LoginTxt"

MinWidth="250"

Margin="0,123,0,0"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Cursor="Hand"

FontFamily="Bahnschrift"

Style="{DynamicResource ResourceKey=TextBoxStyle}"

Height="24"

Foreground="#404040"

Padding="5 4"/>

<Label

Foreground="#FFF6E0"

x:Name="PassTarget"

Margin="50,157,0,0"

HorizontalAlignment="Left"

VerticalAlignment="Top"

FontFamily="Bahnschrift"

Target="{Binding ElementName=Pass}">

\_Пароль

</Label>

<PasswordBox

x:Name="Pass"

MinWidth="250"

HorizontalAlignment="Center"

VerticalAlignment="Top"

Cursor="Hand"

IsEnabled="True"

Style="{DynamicResource ResourceKey=PassBoxStyle}"

ToolTip="Пароль звичайного користувача"

Margin="0,177,0,0"

FontFamily="Bahnschrift"

Foreground="#404040"

Padding="5 4"

Height="24" />

<Button

x:Name="LOGIN"

Height="40"

Margin="50,239,184,0"

VerticalAlignment="Top"

Click="Login\_Click"

Content="Login"

Cursor="Hand"

FontFamily="Bahnschrift"

Style="{DynamicResource ResourceKey=ButtonStyle}"

RenderTransformOrigin="0.5,1.376" />

<TextBlock

Height="40"

Margin="85,39,66,0"

VerticalAlignment="Top"

FontFamily="Bahnschrift"

FontSize="30"

Foreground="#FFF6E0">

Autorisation

</TextBlock>

<Button

x:Name="Exit"

Height="20"

Margin="280,10,10,0"

VerticalAlignment="Top"

Click="Exit\_Click"

Content="Exit"

Cursor="Hand"

FontFamily="Bahnschrift"

IsCancel="True"

Style="{DynamicResource ResourceKey=ExitButtonStyle}" />

<Button

x:Name="Register"

Height="40"

Margin="184,239,50,0"

VerticalAlignment="Top"

Click="Register\_Click"

Content="Registration"

Cursor="Hand"

FontFamily="Bahnschrift"

Style="{DynamicResource ResourceKey=ButtonStyle}" />

</Grid>

</Window>

AuthorisationWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Security.Cryptography;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using GeniusStudio;

using GeniusStudio.Windows;

namespace GeniusStudio

{

/// <summary>

/// Логика взаимодействия для WinLog.xaml

/// </summary>

public partial class WinLog : Window

{

static String data = @"Data\";

static String login = "Login.txt";

public WinLog()

{

InitializeComponent();

}

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

{

String loginTxtBox = LoginTxt.Text;

String passwordPassBox = Pass.Password;

string filePath = Path.Combine(data, login);

try

{

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

bool loginSuccessful = false;

foreach (string line in lines)

{

string[] elements = line.Split(',');

if (elements.Length >= 3 && elements[0] == loginTxtBox && elements[1] == passwordPassBox)

{

string adminLevel = elements[2];

if (adminLevel == "1")

{

MainWindow newWindow = new MainWindow(adminLevel);

newWindow.Show();

this.Close();

}

else if (adminLevel == "0")

{

MainWindow newWindow = new MainWindow(adminLevel);

newWindow.Show();

this.Close();

}

loginSuccessful = true;

break;

}

}

if (!loginSuccessful)

{

MessageBox.Show("Invalid username or password.");

}

}

catch (Exception ex) { MessageBox.Show("Error reading the file: " + ex.Message); }

}

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

{

Environment.Exit(0);

}

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

{

Registration registration = new Registration();

registration.Show();

this.Close();

}

}

}

RegistrationWindow.xaml

<Window x:Class="GeniusStudio.Windows.Registration"

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:GeniusStudio.Windows"

mc:Ignorable="d"

Title="Login" Height="400" Width="300"

ResizeMode="NoResize" WindowStartupLocation="CenterScreen"

Background="#272829">

<Grid>

<TextBlock

Height="40"

Margin="65,59,46,0"

VerticalAlignment="Top"

FontFamily="Bahnschrift"

FontSize="30"

Foreground="#FFF6E0">

Registration

</TextBlock>

<Label

Foreground="#FFF6E0"

Margin="50,122,0,0"

HorizontalAlignment="Left"

VerticalAlignment="Top"

FontFamily="Bahnschrift"

Target="{Binding ElementName=LoginTxt}">

\_Логін

</Label>

<Label

Foreground="#FFF6E0"

x:Name="PassTarget"

Margin="50,170,0,0"

HorizontalAlignment="Left"

VerticalAlignment="Top"

FontFamily="Bahnschrift"

Target="{Binding ElementName=Pass}">

\_Пароль

</Label>

<TextBlock

Foreground="#FFF6E0"

x:Name="AdminText"

Margin="55,223,0,0"

HorizontalAlignment="Left"

VerticalAlignment="Top"

Visibility="Collapsed"

FontFamily="Bahnschrift">

Адмін

</TextBlock>

<TextBox

x:Name="LoginTxtBox"

Height="24"

Width="200"

VerticalAlignment="Top"

HorizontalAlignment="Center"

Margin="0,141,0,0"

Style="{DynamicResource ResourceKey=TextBoxStyle}"

Foreground="#404040"

Padding="5 4"

FontFamily="Bahnschrift"

/>

<PasswordBox

x:Name="PasswordBox"

Height="24"

Width="200"

VerticalAlignment="Top"

HorizontalAlignment="Center"

Margin="0,190,0,0"

Style="{DynamicResource ResourceKey=PassBoxStyle}"

Foreground="#404040"

Padding="5 4"

FontFamily="Bahnschrift"

/>

<Button

x:Name="RegisterBtn"

Height="50"

VerticalAlignment="Top"

Margin="75,284,75,0"

Content="Register"

FontFamily="Bahnschrift"

Style="{StaticResource ButtonStyle}"

Click="RegisterBtn\_Click"/>

<Button

x:Name="Return"

Height="25"

VerticalAlignment="Bottom"

Margin="230,0,10,330"

Content="Back"

FontFamily="Bahnschrift"

Style="{DynamicResource ExitButtonStyle}"

Click="Back\_Click"/>

<Button

x:Name="AdminBtn"

Height="30"

Width="170"

VerticalAlignment="Top"

HorizontalAlignment="Center"

Margin="0,237,0,0"

Content="Admin"

FontFamily="Bahnschrift"

Style="{StaticResource ButtonStyle}"

Click="Admin\_Click"/>

<PasswordBox

x:Name="AdminPasswordBox"

Height="24"

Width="200"

Style="{DynamicResource ResourceKey=PassBoxStyle}"

Foreground="#404040"

Padding="5 4"

Visibility="Collapsed"

FontFamily="Bahnschrift"

VerticalAlignment="Top"

HorizontalAlignment="Center"

Margin="0,237,0,0"

/>

</Grid>

</Window>

RegistrationWindow.xaml.cs

using GeniusStudio;

using System;

using System.Collections.Generic;

using System.IO;

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 GeniusStudio.Windows

{

/// <summary>

/// Логика взаимодействия для Registration.xaml

/// </summary>

public partial class Registration : Window

{

static String Data = @"Data/";

static String loginFile = "Login.txt";

public Registration()

{

InitializeComponent();

}

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

{

string login = LoginTxtBox.Text;

string password = PasswordBox.Password;

string admin;

if (AdminPasswordBox.Password == "860790")

{

admin = "1";

}

else

{

admin = "0";

}

string data = $"{login},{password},{admin}";

using (StreamWriter writer = new StreamWriter(Data + loginFile, true))

{

writer.WriteLine(data);

writer.Close();

}

MessageBox.Show ("Зареестровано");

}

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

{

AdminPasswordBox.Visibility = Visibility.Visible;

AdminText.Visibility = Visibility.Visible;

AdminBtn.Visibility = Visibility.Collapsed;

}

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

{

WinLog login = new WinLog();

login.Show();

this.Close();

}

}

}

MainWindow.xaml

<Window x:Class="GeniusStudio.MainWindow"

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:GeniusStudio"

mc:Ignorable="d"

Title="MainWindow"

Height="1080"

Width="1920"

WindowStyle="None"

WindowState="Maximized">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="5\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="4\*" />

</Grid.ColumnDefinitions>

<StackPanel

Grid.Row="0"

Grid.Column="0"

Background="#735F32">

<Button

x:Name="BtnMenu"

MinWidth="110"

MinHeight="70"

Margin="20,18,20,20"

Click="BtnMenu\_Click"

Content="Menu"

Cursor="Hand"

Style="{DynamicResource ResourceKey=ButtonStyle}"

FontFamily="Bahnschrift" />

<Button

x:Name="BtnAddCatalog"

FontFamily="Bahnschrift"

MinWidth="110"

MinHeight="70"

Margin="20"

Click="BtnAddCatalog\_Click"

Content="Add Catalog"

Cursor="Hand"

Style="{DynamicResource ResourceKey=ButtonStyle}" />

</StackPanel>

<StackPanel

x:Name="WrapInfo"

Grid.Row="0"

Grid.Column="1"

Background="#272829" Grid.RowSpan="2" />

<StackPanel

Grid.Row="1"

Grid.Column="2"

Grid.ColumnSpan="2">

</StackPanel>

<StackPanel

Grid.Row="1"

Grid.Column="0"

Background="#735F32">

<Button

x:Name="ReLogin"

MinWidth="110"

MinHeight="70"

Margin="20,0"

Click="ReLogin\_Click"

Content="Relogin"

FontFamily="Bahnschrift"

Style="{DynamicResource ResourceKey=ButtonStyle}" />

<Button

x:Name="BtnExit"

MinWidth="110"

MinHeight="70"

Margin="20,3,20,20"

Click="BtnExit\_Click"

Content="Exit"

IsCancel="True"

Style="{DynamicResource ResourceKey=ExitButtonStyle}" />

</StackPanel>

<Frame

x:Name="frame"

Grid.Row="0"

Grid.Column="1"

Margin="20,20,20,20"

Loaded="frame\_Loaded"

NavigationUIVisibility="Hidden" Grid.RowSpan="2" />

</Grid>

</Window>

MainWindow.xaml.cs

using System;

using System.Windows;

namespace GeniusStudio

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow(string adminLevel)

{

InitializeComponent();

if (adminLevel == "1")

{

BtnAddCatalog.Visibility = Visibility.Visible;

}

else if (adminLevel == "0")

{

BtnAddCatalog.Visibility = Visibility.Collapsed;

}

}

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

{

frame.Content = new BtnMenuPage();

}

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

{

Environment.Exit(0);

}

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

{

WinLog authorization = new WinLog();

authorization.Show();

this.Close();

}

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

{

frame.Content = new AddMenuPage();

}

private void frame\_Loaded(object sender, RoutedEventArgs e)

{

frame.Content = new BtnMenuPage();

}

}

}

BtnMenuPage.xaml

<Page

x:Class="GeniusStudio.BtnMenuPage"

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:local="clr-namespace:GeniusStudio"

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

Title="BtnMenuPage"

d:DesignHeight="450"

d:DesignWidth="800"

mc:Ignorable="d">

<Grid>

<WrapPanel

x:Name="MenuPanel"

Grid.Row="0"

Grid.Column="1"

Background="Transparent"

Visibility="Visible"

Orientation="Horizontal" />

<WrapPanel Name="ItemPanel" />

</Grid>

</Page>

BtnMenuPage.xaml.cs

using System;

using System.Xml;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

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.Navigation;

using System.Windows.Shapes;

using System.Xml.Linq;

namespace GeniusStudio

{

/// <summary>

/// Логика взаимодействия для BtnMenuPage.xaml

/// </summary>

public partial class BtnMenuPage : Page

{

public BtnMenuPage()

{

InitializeComponent();

CreateMenuButtons();

}

private void CreateMenuButtons()

{

string MenuFolderPath = "Menu";

string[] menuFolders = Directory.GetDirectories(MenuFolderPath);

foreach (string menuFolder in menuFolders)

{

string folderName = new DirectoryInfo(menuFolder).Name;

Button folderButton = new Button

{

Content = folderName,

Width = 400,

Height = 300,

FontSize = 24,

FontFamily = new FontFamily("Bahnschrift"),

Margin = new Thickness(20),

Style = (Style)FindResource("ButtonStyle")

};

folderButton.Click += (sender, e) => LoadItemsFromFolder(folderName);

MenuPanel.Children.Add(folderButton);

}

}

private void LoadItemsFromFolder(string folderName)

{

string folderPath = $"Menu/{folderName}";

string[] itemFiles = Directory.GetFiles(folderPath);

MenuPanel.Children.Clear();

foreach (string itemFile in itemFiles)

{

string itemName = System.IO.Path.GetFileNameWithoutExtension(itemFile);

string[] itemInfo = File.ReadAllLines(itemFile);

Button itemButton = new Button

{

Content = itemName,

Width = 400,

Height = 300,

FontSize = 24,

FontFamily = new FontFamily("Bahnschrift"),

Margin = new Thickness(20),

Style = (Style)FindResource("ButtonStyle")

};

itemButton.Click += (sender, e) => DisplayItemInfo(itemFile);

MenuPanel.Children.Add(itemButton);

}

}

private void DisplayItemInfo(string itemFilePath)

{

MenuPanel.Children.Clear();

string[] itemInfo = File.ReadAllLines(itemFilePath);

try

{

foreach (string line in itemInfo)

{

string[] itemData = line.Split('|');

if (itemData.Length >= 2)

{

TextBlock itemTextBlock = new TextBlock

{

FontFamily = new FontFamily("Bahnschrift"),

FontSize = 20,

Foreground = new SolidColorBrush((Color)ColorConverter.ConvertFromString("#FFF6E0")),

TextWrapping = TextWrapping.Wrap,

TextAlignment = TextAlignment.Left,

Margin = new Thickness(20) // Add margin between TextBlocks

};

for (int i = 1; i < itemData.Length; i++)

{

string itemName = itemData[i].Trim(); // Extracting the item name

itemTextBlock.Text += itemName + Environment.NewLine;

}

MenuPanel.Children.Add(itemTextBlock);

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

AddMenuPage.xaml

<Page

x:Class="GeniusStudio.AddMenuPage"

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:local="clr-namespace:GeniusStudio"

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

x:Name="AddMenuPage1"

Title="AddMenuPage"

d:DesignHeight="1040"

d:DesignWidth="1496"

mc:Ignorable="d">

<Grid>

<WrapPanel

Name="SubMenuPage"

Grid.RowSpan="2"

Grid.ColumnSpan="3"

Background="Transparent" />

<Button

x:Name="BtnCreateCataalog"

Width="708"

Height="427"

Margin="20,80,0,0"

VerticalAlignment="Top"

HorizontalAlignment="Left"

Content="Create Catalog"

Cursor=""

Style="{DynamicResource ResourceKey=ButtonStyle}"

Click="BtnCreateCatalog\_Click"

FontSize="24"

FontFamily="Bahnschrift" />

<Button

x:Name="BtnDeleteCatalog"

Width="707"

Height="425"

Margin="0,82,20,0"

VerticalAlignment="Top"

HorizontalAlignment="Right"

Content="Delete Catalog"

Cursor="Hand"

Style="{DynamicResource ResourceKey=ButtonStyle}"

Click="BtnDeleteCatalog\_Click"

FontSize="24"

FontFamily="Bahnschrift"/>

<Button

x:Name="BtnCreateSubCatalog"

Width="708"

Height="424"

Margin="20,0,0,80"

VerticalAlignment="Bottom"

HorizontalAlignment="Left"

Content="Create SubCatalog"

Cursor="Hand"

Style="{DynamicResource ResourceKey=ButtonStyle}"

Click="BtnCreateSubCatalog\_Click"

FontSize="24"

FontFamily="Bahnschrift"/>

<Button

x:Name="BtnDeleteSubCatalog"

Width="707"

Height="424"

Margin="0,0,20,80"

VerticalAlignment="Bottom"

HorizontalAlignment="Right"

Content="Delete SubCatalog"

Cursor="Hand"

Style="{DynamicResource ResourceKey=ButtonStyle}"

Click="BtnDeleteSubCatalog\_Click"

FontSize="24"

FontFamily="Bahnschrift"/>

<Button

x:Name="SaveFileBtn"

Grid.Column="1"

Width="150"

Height="50"

HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Visibility="Collapsed"

Click="SaveFileBtn\_Click"

Content="Save"

Style="{DynamicResource ButtonStyle}" />

<Button

x:Name="SaveTxtFileBtn"

Grid.Column="1"

Width="150"

Height="50"

HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Visibility="Collapsed"

Click="SaveTxtFileBtn\_Click"

Content="Save"

Style="{DynamicResource ButtonStyle}" />

<TextBox

x:Name="FileName"

Grid.ColumnSpan="3"

Width="700"

Height="50"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Visibility="Collapsed"

FontSize="24"

Style="{DynamicResource TextBoxStyle}" />

</Grid>

</Page>

AddMenuPage.xaml.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Reflection;

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.Navigation;

namespace GeniusStudio

{

/// <summary>

/// Логика взаимодействия для AddMenuPage.xaml

/// </summary>

public partial class AddMenuPage : Page

{

public AddMenuPage()

{

InitializeComponent();

}

string folders;

string toDelete;

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

{

SaveFileBtn.Visibility = Visibility.Visible;

FileName.Visibility = Visibility.Visible;

SaveTxtFileBtn.Visibility = Visibility.Collapsed;

BtnCreateCataalog.Visibility = Visibility.Collapsed;

BtnCreateSubCatalog.Visibility = Visibility.Collapsed;

BtnDeleteCatalog.Visibility = Visibility.Collapsed;

BtnDeleteSubCatalog.Visibility = Visibility.Collapsed;

}

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

{

BtnCreateCataalog.Visibility = Visibility.Collapsed;

BtnCreateSubCatalog.Visibility = Visibility.Collapsed;

BtnDeleteCatalog.Visibility = Visibility.Collapsed;

BtnDeleteSubCatalog.Visibility = Visibility.Collapsed;

string MenuFolderPath = "Menu";

string[] menuFolders = Directory.GetDirectories(MenuFolderPath);

foreach (string menuFolder in menuFolders)

{

string folderName = new DirectoryInfo(menuFolder).Name;

folders = folderName;

Button folderButton = new Button

{

Content = folderName,

Width = 110,

Height = 100,

Margin = new Thickness(20),

Style = (Style)FindResource("ButtonStyle")

};

folderButton.Click += (s, args) =>

{

SaveFileBtn.Visibility = Visibility.Collapsed;

FileName.Visibility = Visibility.Visible;

SaveTxtFileBtn.Visibility = Visibility.Visible;

};

SubMenuPage.Children.Add(folderButton);

}

}

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

{

BtnCreateCataalog.Visibility = Visibility.Collapsed;

BtnCreateSubCatalog.Visibility = Visibility.Collapsed;

BtnDeleteCatalog.Visibility = Visibility.Collapsed;

BtnDeleteSubCatalog.Visibility = Visibility.Collapsed;

string MenuFolderPath = "Menu";

string[] menuFolders = Directory.GetDirectories(MenuFolderPath);

foreach (string menuFolder in menuFolders)

{

string folderName = new DirectoryInfo(menuFolder).Name;

folders = folderName;

Button folderButton = new Button

{

Content = folderName,

Width = 400,

Height = 300,

FontSize = 24,

FontFamily = new FontFamily("Bahnschrift"),

Margin = new Thickness(20),

Style = (Style)FindResource("ButtonStyle")

};

folderButton.Click += (s, args) =>

{

string toDelete = $"Menu/{folderName}";

try

{

Directory.Delete(toDelete, true);

}

catch (Exception ex)

{

MessageBox.Show($"Не вдалося видалити папку '{folderName}': {ex.Message}");

}

NavigationService?.Navigate(new AddMenuPage());

};

SubMenuPage.Children.Add(folderButton);

}

}

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

{

BtnCreateCataalog.Visibility = Visibility.Collapsed;

BtnCreateSubCatalog.Visibility = Visibility.Collapsed;

BtnDeleteCatalog.Visibility = Visibility.Collapsed;

BtnDeleteSubCatalog.Visibility = Visibility.Collapsed;

string MenuFolderPath = "Menu";

string[] menuFolders = Directory.GetDirectories(MenuFolderPath);

foreach (string menuFolder in menuFolders)

{

string folderName = new DirectoryInfo(menuFolder).Name;

Button folderButton = new Button

{

Content = folderName,

Width = 400,

Height = 300,

FontSize = 24,

FontFamily = new FontFamily("Bahnschrift"),

Margin = new Thickness(20),

Style = (Style)FindResource("ButtonStyle")

};

folderButton.Click += (sender, e) => LoadItemsFromFolder(folderName);

SubMenuPage.Children.Add(folderButton);

}

}

private void LoadItemsFromFolder(string folderName)

{

string folderPath = $"Menu/{folderName}";

string[] itemFiles = Directory.GetFiles(folderPath);

SubMenuPage.Children.Clear();

foreach (string itemFile in itemFiles)

{

string itemName = System.IO.Path.GetFileNameWithoutExtension(itemFile);

string[] itemInfo = File.ReadAllLines(itemFile);

Button itemButton = new Button

{

Content = itemName,

Width = 400,

Height = 300,

FontSize = 24,

FontFamily = new FontFamily("Bahnschrift"),

Margin = new Thickness(20),

Style = (Style)FindResource("ButtonStyle")

};

itemButton.Click += (sender, e) =>

{

try

{

// Видаляємо файл, представлений поточним itemFile

File.Delete(itemFile);

MessageBox.Show($"Файл '{itemName}' був успішно видалений!");

}

catch (Exception ex)

{

MessageBox.Show($"Не вдалося видалити файл '{itemName}': {ex.Message}");

}

// Після видалення перенаправляємо на поточну сторінку знову

NavigationService?.Navigate(new AddMenuPage());

};

SubMenuPage.Children.Add(itemButton);

}

}

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

{

string folderName = FileName.Text;

// Перевірка, чи введена назва не порожня

if (!string.IsNullOrWhiteSpace(folderName))

{

try

{

// Створення папки

Directory.CreateDirectory($"Menu/{folderName}");

}

catch (Exception ex)

{

MessageBox.Show($"Не вдалося створити папку '{folderName}': {ex.Message}");

}

}

else

{

MessageBox.Show("Введіть назву папки!");

}

}

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

{

string fileName = FileName.Text;

// Перевірка, чи введена назва не порожня

if (!string.IsNullOrWhiteSpace(fileName))

{

try

{

// Створення папки

File.Create($"Menu/{folders}/{fileName}.txt");

MessageBox.Show($"Папка '{fileName}' створена!");

}

catch (Exception ex)

{

MessageBox.Show($"Не вдалося створити папку '{fileName}': {ex.Message}");

}

}

else

{

MessageBox.Show("Введіть назву папки!");

}

}

}

}