**MainPage.xaml.cs**

1. Importy i deklaracja przestrzeni nazw

Na początku kodu znajdują się importy bibliotek:

System, System.Collections.Generic, System.IO, System.Threading.Tasks: Dostarczają podstawowe funkcje języka C#, w tym operacje na plikach i kolekcjach.

Microsoft.Maui.Controls: Biblioteka .NET MAUI do tworzenia interfejsu użytkownika.

Deklarowana jest przestrzeń nazw OliwiaSmalcugaCounter, w której znajduje się klasa MainPage.

2. Klasa MainPage i pola prywatne

Klasa MainPage dziedziczy po ContentPage i zawiera logikę dla głównego ekranu aplikacji. Zawiera pola prywatne:

\_database: Instancja bazy danych liczników.

\_counters: Lista obiektów typu Counter, reprezentująca wszystkie zapisane liczniki.

3. Konstruktor MainPage()

W konstruktorze wywoływane są metody inicjujące:

InitializeComponent(): Inicjalizuje elementy interfejsu, zdefiniowane w pliku XAML.

InitializeDatabase(): Ustawia połączenie z bazą danych.

LoadCounters(): Ładuje liczniki z bazy danych do listy \_counters.

LoadPickerItems(): Wypełnia element wyboru kolorów (ColorPicker).

4. LoadPickerItems()

Metoda przypisuje do ColorPicker.ItemsSource listę kolorów dostępnych dla liczników i ustawia początkowo wybrany kolor na "Pink".

5. OnAppearing()

Metoda OnAppearing jest wywoływana po każdym pojawieniu się strony i ponownie ładuje listę liczników, aby odświeżyć widok, jeśli coś się zmieniło.

6. InitializeDatabase()

Tworzy połączenie z bazą danych (plik counters.db w lokalnej pamięci urządzenia). Ta metoda inicjalizuje instancję CounterDatabase, aby można było zapisywać i odczytywać dane liczników.

7. LoadCounters()

Ta metoda asynchronicznie pobiera wszystkie liczniki z bazy danych (GetCountersAsync) i przypisuje je do źródła danych dla CountersCollectionView, aby wyświetlić liczniki na stronie.

8. OnAddCounterClicked

Metoda jest wywoływana po kliknięciu przycisku dodawania nowego licznika. Sprawdza, czy:

Nazwa licznika (CounterNameEntry.Text) jest niepusta.

Wartość początkowa (InitialValueEntry.Text) jest liczbą.

Wybrano kolor (ColorPicker.SelectedItem).

Jeśli te warunki są spełnione, tworzy nowy licznik (Counter) i zapisuje go w bazie danych. Następnie ładuje listę liczników, aby odświeżyć widok.

9. OnIncrementClicked, OnDecrementClicked, OnResetClicked, OnDeleteClicked

Każda z tych metod zarządza operacją na liczniku po kliknięciu odpowiedniego przycisku:

OnIncrementClicked: Zwiększa wartość licznika o 1.

OnDecrementClicked: Zmniejsza wartość licznika o 1.

OnResetClicked: Resetuje wartość licznika do wartości początkowej (InitialValue).

OnDeleteClicked: Usuwa licznik z bazy danych.

Po wykonaniu każdej operacji baza danych jest aktualizowana, a LoadCounters() jest wywoływane, aby odświeżyć wyświetlaną listę.

**CouterDatabase**

1. Importy i Deklaracja Klasy

csharp

Skopiuj kod

using SQLite;

using System.Collections.Generic;

using System.Threading.Tasks;

public class CounterDatabase

{

private readonly SQLiteAsyncConnection \_database;

SQLite i SQLiteAsyncConnection: Używane są tutaj klasy z biblioteki SQLite do zarządzania asynchronicznymi operacjami na bazie danych SQLite.

CounterDatabase: Klasa zarządza operacjami na bazie danych dla obiektów typu Counter.

\_database: Pole prywatne, które przechowuje połączenie z bazą danych.

2. Konstruktor CounterDatabase

csharp

Skopiuj kod

public CounterDatabase(string dbPath)

{

\_database = new SQLiteAsyncConnection(dbPath);

\_database.CreateTableAsync<Counter>().Wait();

}

Konstruktor inicjalizuje bazę danych:

dbPath: Ścieżka do pliku bazy danych (przekazana jako argument podczas tworzenia obiektu CounterDatabase).

SQLiteAsyncConnection(dbPath): Tworzy asynchroniczne połączenie z bazą danych.

CreateTableAsync<Counter>().Wait(): Tworzy tabelę dla obiektów Counter w bazie danych, jeśli taka tabela nie istnieje. Metoda Wait() zapewnia, że operacja tworzenia tabeli zakończy się przed przejściem do dalszego kodu.

3. Metoda GetCountersAsync

csharp

Skopiuj kod

public Task<List<Counter>> GetCountersAsync()

{

return \_database.Table<Counter>().ToListAsync();

}

GetCountersAsync: Metoda asynchronicznie pobiera wszystkie rekordy z tabeli Counter i zwraca je jako listę (List<Counter>).

Table<Counter>().ToListAsync(): Tworzy zapytanie do tabeli Counter i konwertuje wyniki do listy.

4. Metoda GetCounterAsync

csharp

Skopiuj kod

public Task<Counter> GetCounterAsync(int id)

{

return \_database.Table<Counter>().Where(i => i.Id == id).FirstOrDefaultAsync();

}

GetCounterAsync(int id): Pobiera licznik na podstawie jego identyfikatora id.

Table<Counter>().Where(i => i.Id == id).FirstOrDefaultAsync(): Tworzy zapytanie, które wyszukuje licznik o określonym id. Jeśli nie znajdzie takiego rekordu, zwraca null.

5. Metoda SaveCounterAsync

csharp

Skopiuj kod

public Task<int> SaveCounterAsync(Counter counter)

{

if (counter.Id != 0)

{

return \_database.UpdateAsync(counter);

}

else

{

return \_database.InsertAsync(counter);

}

}

SaveCounterAsync(Counter counter): Zapisuje licznik do bazy danych. Sprawdza, czy licznik ma ustawione Id różne od 0, co oznacza, że jest to istniejący rekord.

Jeśli Id jest różne od 0 (czyli licznik już istnieje w bazie), wywoływana jest metoda UpdateAsync, aby zaktualizować rekord.

Jeśli Id jest równe 0 (nowy licznik), wywoływana jest metoda InsertAsync, aby dodać licznik do bazy.

Zwraca Task<int>, gdzie int jest liczbą wskazującą na liczbę zmienionych rekordów.

6. Metoda DeleteCounterAsync

csharp

Skopiuj kod

public Task<int> DeleteCounterAsync(Counter counter)

{

return \_database.DeleteAsync(counter);

}

DeleteCounterAsync(Counter counter): Usuwa przekazany licznik z bazy danych.

DeleteAsync(counter): Usuwa rekord na podstawie przekazanego obiektu Counter.

Zwraca Task<int>, gdzie int oznacza liczbę usuniętych rekordów.

Podsumowanie

Klasa CounterDatabase zarządza operacjami na bazie danych dla obiektów typu Counter. Zapewnia ona:

Tworzenie i inicjalizację tabeli.

Pobieranie wszystkich liczników lub pojedynczego licznika na podstawie id.

Dodawanie nowego licznika lub aktualizację istniejącego.

Usuwanie licznika z bazy danych.