ПРАКТИЧНА РОБОТА № 12. СТВОРЕННЯ COLLECTION VIEW ЗАСОБАМИ XAMARIN.FORMS

Мета: набуття здобувачами освіти навичок створення контейнеру розмітки Grid; створення списку CollectionView об'єктів класу Language; розміщення елементів списку в Grid; оброблення події вибору елементу списку та виведення даних про об'єкт методом DisplayAlert.

Короткі теоретичні відомості

Xamarin Forms Grid – це тип макета для розміщення компонентів (Label, Entry, CheckBox) за допомогою рядків і стовпців. В уявній таблиці розташовуємо компоненти так, щоб вони складалися принаймні з одного рядка та одного стовпця. Сітку можна використовувати в проектах, де компоненти виглядають згрупованими. Або, найпростішим способом, його можна використовувати в дизайні з виглядом таблиці.

За замовчуванням Grid містить один рядок nf один стовпець. Створити Grid дуже просто, потрібно додати тег Grid:

XAML:

```
< Grid > </ Grid >
```

Код C#:

```
var grid = new Grid ( ) ;
```

Є 3 способи визначити розмір комірки.

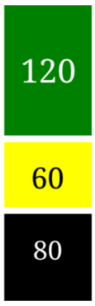
1) Absolute – це фіксований розмір, можна додати будь-яке число до параметра Wight/Height.

XAML:

```
<RowDefinition Height="120" />
<RowDefinition Height="60" />
<RowDefinition Height="80" />
```

Код C#:

```
var row1 = new RowDefinition() { Height = new GridLength(120)};
var row2 = new RowDefinition() { Height = new GridLength(60)};
var row3 = new RowDefinition() { Height = new GridLength(80) };
```



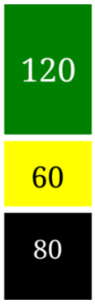
2) Auto – рядок/стовпець адаптуватиметься до дочірнього розміру.

XAML:

```
<RowDefinition Height="120" />
<RowDefinition Height="60" />
<RowDefinition Height="Auto" />
```

Код C#:

```
var row1 = new RowDefinition() { Height = new GridLength(120)};
var row2 = new RowDefinition() { Height = new GridLength(60)};
var row3 = new RowDefinition() { Height = new GridLength(1,
GridUnitType.Auto) };
```



3) Star–розширюється пропорційно розміру простору. У XAML замість 1* можна використовувати лише *

XAML:

```
<RowDefinition Height="5*" />
<RowDefinition Height="2*" />
<RowDefinition Height="3*" />
```

Код C#:

```
var row1 = new RowDefinition() {Height = new GridLength(5, GridUnitType.Star)};
var row2 = new RowDefinition() {Height = new GridLength(2, GridUnitType.Star)};
var row3 = new RowDefinition() {Height = new GridLength(3, GridUnitType.Star)};
```

Щоб додати кожен рядок/стовпець до сітки, є дві властивості: Grid.RowDefinitions (для додавання рядків) і Grid.ColumnDefinitions (для додавання стовпців), обидві є колекціями, які отримують список кожного визначення рядка та стовпця. Крім того можуf комбінувати всі властивості з розміром.

```
<Grid RowDefinitions="1*, Auto, 25, 14, 20"
      ColumnDefinitions="*, 2*, Auto, 300">
    ...
</Grid>
```

У цьому прикладі Grid має п'ять рядків та чотири стовпці. Третій, четвертий та п'ятий рядки мають абсолютні висоти, а другий рядок автоматично визначає його вміст. Висота, що залишилася, потім виділяється першому рядку.

Для стовпця чотири встановлено абсолютну ширину з автоматичною зміною розміру третього стовпця для його вмісту. Ширина, що залишилася, виділяється пропорційно між першими та другим стовпцями на основі числа перед зіркою. У цьому прикладі ширина другого стовпця вдвічі перевищує ширину 1*першого стовпця.

Щоб додати елемент до Grid, потрібно вказати, у якому стовпці та рядку потрібно його розмістити, наприклад:

XAML:

```
<BoxView BackgroundColor = "Pink" Grid.Colum ="0" HorizontalOptions =
"FillAndExpand" VerticalOptions = "FillAndExpand"/>
```

Код C#:

```
var boxview = new BoxView() {BackgroundColor = Color.Pink, HorizontalOptions =
LayoutOptions.FillAndExpand, VertitalOptions = LayoutOptions.FillAndExpand};
grid.Children.Add(boxview, 0, 0);
```

У коді наведений приклад додавання рожевого прямокутника до стовпця 0/ рядку 0.

Дочірні елементи в об'єкті Grid можуть розміщуватися у своїх комірках за

властивостями та VerticalOptions, HorizontalOptions. Ці властивості можна задати в таких полях структури LayoutOptions: Start, Center, End, Fill.

Якщо необхідно щоб комірка займала більше одного рядка/стовпця, то можна скористатися властивістю Row.Span або Column.Span та вказати, які клітинки хочете об'єднати. Наприклад, код, коли рожеве поле об'єднує 2 стовпця (рис. 1), має вигляд:

XAML:

```
<BoxView BackgroundColor = "Pink" Grid.Colum ="0" Grid.Row ="0"
Grid.ColumnSpan ="2"/>
```

Код C#:

```
var boxview = new BoxView() {BackgroundColor = Color.Pink};
grid.Children.Add(boxview, 0, 2, 0, 1);
//0 - Column, 1 - ColumnSpan, 0 - Row, 1 – RowSpan
```

Рисунок 1 – До пояснення властивості Row.Span та Column.Span

Значення інтервалу між рядками сітки, якщо не вказано, то приймається як 6. Для зміни його значення застосовують властивості RowSpacing, ColumnSpacing.

```
<Grid RowSpacing ="20" ColumnSpacing ="20">
```

Для відображення табличних даних рекомендується використовувати ListView, CollectionView або TableView.

CollectionView застосовують представлення списків даних із використанням різних специфікацій макета. CollectionView заповнюється даними шляхом встановлення його ItemsSource властивості будь-яку колекцію, що реалізує IEnumerable. Зовнішній вигляд кожного елемента у списку можна визначити, задавши ItemTemplate для властивості DataTemplate значення. За замовчуванням CollectionView елементи відображаються у вертикальному списку. Однак можна вказати вертикальні та горизонтальні списки та сітки.

Клас Grid є похідним від Layout<T>класу, що визначає Children властивість типу IList<T>. Властивість Children є ContentProperty класом Layout<T>.

Приклад коду з використанням Grid, макет розмітки наведений на рис.2.

```
<?xml version="1.0" encoding="utf-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:GridSample"
             x:Class="GridSample.GridSamplePage">

    <Grid Padding="20" VerticalOptions="CenterAndExpand">
      <Grid.RowDefinitions>
         <RowDefinition Height="40" />
         <RowDefinition Height="40" />
         <RowDefinition Height="40" />
         <RowDefinition Height="40" />
       </Grid.RowDefinitions>

      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="5*" />
        <ColumnDefinition Width="5*" />
      </Grid.ColumnDefinitions>

     <Entry Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2" Placeholder="UserName"
HorizontalOptions="FillAndExpand"/>
     <Entry Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="2" Placeholder="Password"
HorizontalOptions="FillAndExpand"/>
     <Entry Grid.Row="2" Grid.Column="0" Placeholder="NickName"
HorizontalOptions="FillAndExpand"/>
     <Entry Grid.Row="2" Grid.Column="1" Placeholder="Phone"
HorizontalOptions="FillAndExpand"/>
     <Button Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2"
HorizontalOptions="FillAndExpand" Text="Register" TextColor="White" BackgroundColor="Black"/>
    </Grid>
</ContentPage>
```

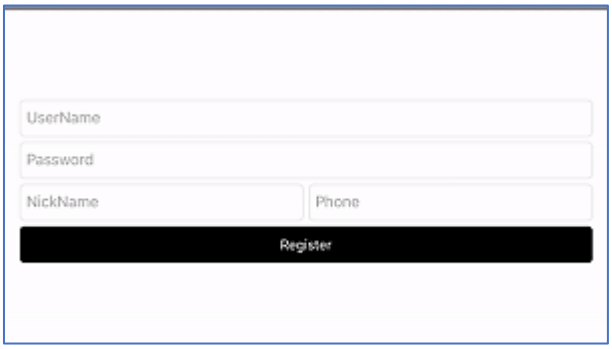


Рисунок 2 – Макет проекту з використанням Grid

Хід роботи

1. Створюємо проект CollectionViewSample в Visual Studio 19. Вибираємо шаблон Blank. При створенні проекту в опції Platform вибираємо платформу Android під яку буде створюватись проект (рис. 3).

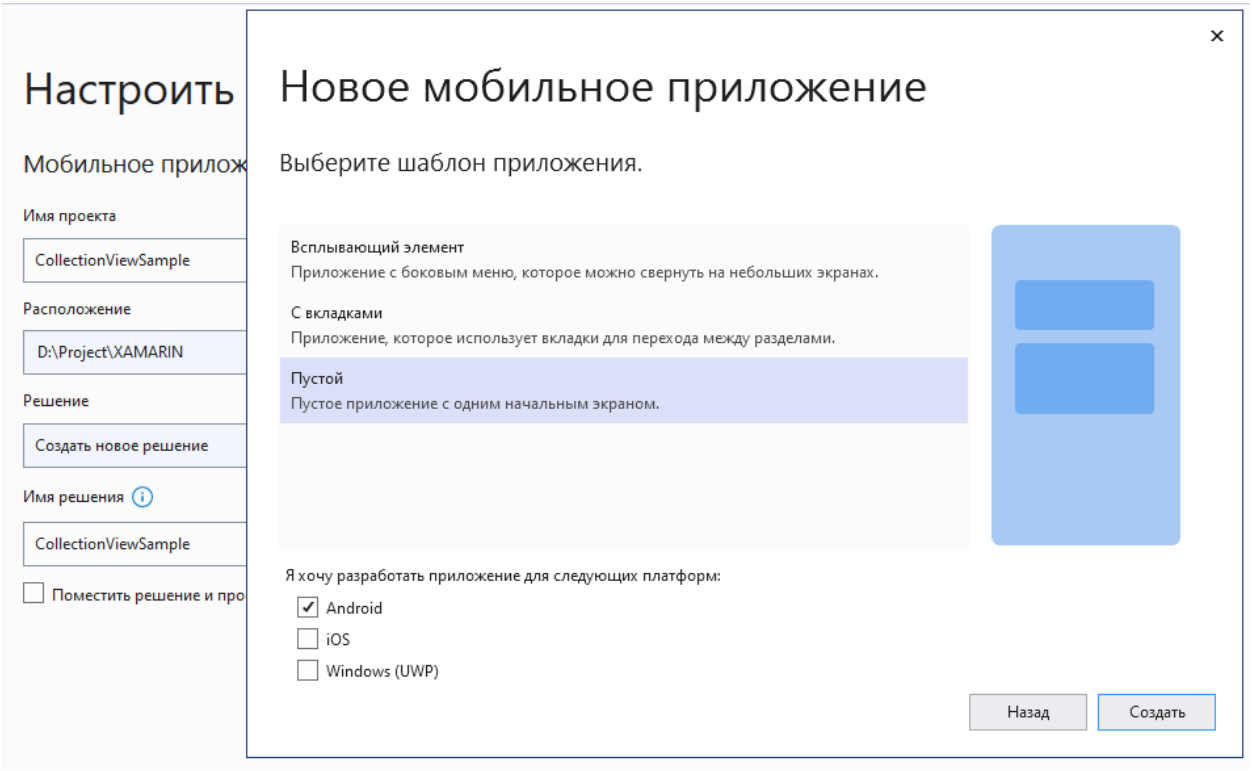


Рисунок 3 – Створення проекту CollectionViewSample

2. Як джерело даних вибираємо список, у якому зберігається інформація про різні мови програмування. Відкриваємо MainPage.xaml. Приводимо код файлу до виду:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="CollectionViewSample.MainPage">

    <StackLayout>

    </StackLayout>

</ContentPage>
```

3. Додаємо до проекту клас Language. Для цього викликаємо контекстне меню провідника рішень (Solution Explorer). Вибираємо Add -> Create element -> Class та вказуємо його назву (рис. 4). Відкриваємо файл Language.cs. Додаємо поля Name, Description, Image та властивість для встановлення та отримання даних з поля даних класу.

```
namespace CollectionViewSample
{
    public class Language
    {
        public string Name { get; set; }
        public string Description { get; set; }
        public string Image { get; set;}
  }
}
```

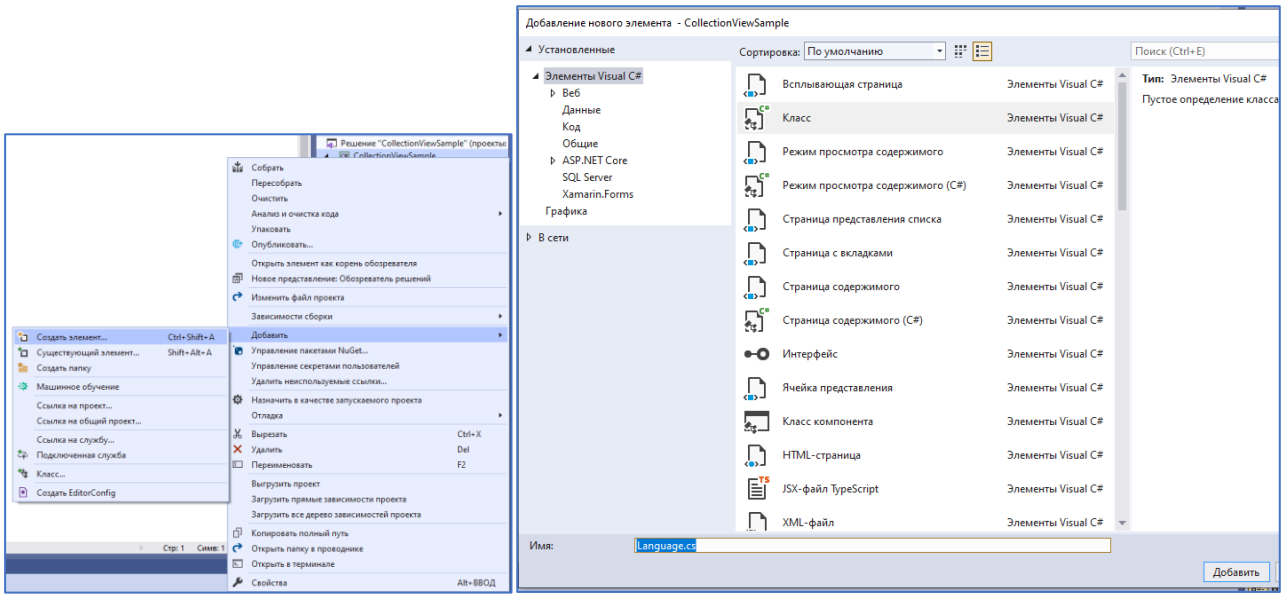


Рисунок 4 – Додавання до проекту класу Language

4. Переходимо до файлу MainPage.xaml.cs та створюємо колекцію елементів Language

```
namespace CollectionViewSample
{
        public partial class MainPage : ContentPage
    {
        public IList<Language> Languages { get; set; }

        public MainPage()
        {
            InitializeComponent();
        }

        protected override void OnAppearing()
        {
            Languages = new List<Language>();
            base.OnAppearing();
        }
    }
}
```

5. Додаємо елементи в колекцію. Вносимо зміни у метод OnAppearing файлу MainPage.xaml.cs.

```
protected override void OnAppearing()
{
   Languages = new List<Language>();
   Languages.Add(new Language()
   {
      Name = "C#",
      Description = "Об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET",
      Image = "https://upload.wikimedia.org/wikipedia/commons/thumb/4/4f/Csharp_Logo.png/200px-Csharp_Logo.png"
    });
```

```
    Languages.Add(new Language()
     {
        Name = "JAVA",
        Description = "Java є однією з найпопулярніших мов програмування, що використовуються розробниками програмного забезпечення на сьогоднішній день. Ядро мови використовується при розробці Android-додатків, а також широко використовується в веб-розробці, а саме в серверній частині.",
         Image = "https://upload.wikimedia.org/wikipedia/en/thumb/3/30/Java_programming_language_logo.svg/121px-Java_programming_language_logo.svg.png"
      });

      Languages.Add(new Language()
      {
         Name = "Python",
         Description = "Python широко використовується в інтернет-додатках, розробці програмного забезпечення, науці даних і машинному навчанні (ML). Він ефективний, простий у вивченні та працює на різних платформах",
         Image = "https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Python-logo-notext.svg/121px-Python-logo-notext.svg.png"
       });

      Languages.Add(new Language()
      {
         Name = "C++",
         Description = "мова програмування загального призначення з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої, процедурної",
         Image = "https://upload.wikimedia.org/wikipedia/commons/thumb/1/18/ISO_C%2B%2B_Logo.svg/130px-ISO_C%2B%2B_Logo.svg.png"
       });

       Languages.Add(new Language()
       {
          Name = "PHP",
          Description = "скриптова мова програмування, була створена для генерації HTML-сторінок на стороні вебсервера. " +
                "PHP є однією з найпоширеніших мов, що використовуються у сфері веброзробок",
          Image = "https://quaded.com/data/php-logo.png"
        });

        Languages.Add(new Language()
        {
           Name = "GO",
           Description = "Компільована мова програмування із вбудованими засобами для паралельних обчислень і засобами віддаленого керування пакунками",
           Image = "https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/Go_Logo_Blue.svg/215px-Go_Logo_Blue.svg.png"
         });
           base.OnAppearing();
         }
```

6. До методу OnAppearing необхідно ще додати рядок «BindingContext = this;». Ключеве слово this дозволяє отримати посилання на поточний елемент класу з оперативної пам’яті. Код має вигляд:

```
Languages.Add(new Language()
{
    Name = "GO",
    Description = "Компільована мова програмування із вбудованими засобами для паралельних обчислень і засобами віддаленого керування пакунками",
    Image = "https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/Go_Logo_Blue.svg/215px-Go_Logo_Blue.svg.png"
});

BindingContext = this;
base.OnAppearing();
```

7. Переходимо до файлу MainPage.xaml. Додаємо код до контейнера компоновки StackLayout:

```
<StackLayout>
   <CollectionView ItemsSource="{Binding Languages}"
        SelectionMode="Single" SelectionChanged="OnSelectionChanged">

   </CollectionView>

</StackLayout>
```

CollectionView заповнюється даними шляхом встановлення його ItemsSource властивості колекцію Languages з ключовим словом Binding у фігурних дужках. SelectionMode="Single" указує скільки користувач може виділити елементів у списку, SelectionChanged = "OnSelectionChanged" дозволяє обробляти Single Tap (натискання на елементі списку)

8. Створюємо шаблон для елементів

```
<StackLayout>
        <CollectionView ItemsSource="{Binding Languages}"
        SelectionMode="Single" SelectionChanged="OnSelectionChanged">
            <CollectionView.ItemTemplate>
                <DataTemplate>

                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>

</StackLayout>
```

Основним елементом списку є мова програмування, яка представляється назвою, описом та логотипом. Елементи списку будуть розміщатися у контейнері Grid з атоматичним визначенням ширини та висоти комірки. Логотип буде займати 2 рядки

```
<StackLayout>
        <CollectionView ItemsSource="{Binding Languages}"
        SelectionMode="Single" SelectionChanged="OnSelectionChanged">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Grid Padding="10"
                          RowDefinitions="Auto, *"
                          ColumnDefinitions="Auto, *">
                        <Image Grid.RowSpan="2"
                            Source="{Binding Image}"
                            Aspect="AspectFill"
                            HeightRequest="60"
                            WidthRequest="60"/>
                        <Label Grid.Column="1"
                               Text="{Binding Name}"
                               FontAttributes="Bold"/>
```

```
                        <Label Grid.Row="1"
                               Grid.Column="1"
                               Text="{Binding Description}"
                               VerticalOptions="End"/>
                    </Grid>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>

    </StackLayout>
```

9. Додаємо обробку Single Tap по елементу списку. Для цього у файлі MainPage.xaml.cs створюється метод OnSelectionChanged, який отримуємо дані з sender (об'єкт, який викликав подію) та SelectionChangedEventArgs (дозволяє отримати різні дані об’єкту даної події).

```
async void OnSelectionChanged(object sender, SelectionChangedEventArgs e)
{
     Language selectedLang = e.CurrentSelection[0] as Language;
     await DisplayAlert(selectedLang.Name, selectedLang.Description,
"Ok");
 }
```

Об'єкт CurrentSelection[0] отримує елемент списку за яким був Single Tap. Отримуємо не просто елемент списку, а екземпляр класу який прив'язаний з колекції даних до даного елементу списку. Відповідно отримуємо екземпляр класу Language та всі властивості й дані, які є в цьому екземплярі. Елемент списку нульовий, так як був встановлений режим SelectionMode="Single" для CollectionView. Приводимо його до типу Language за допомогою ключового слова as та в DisplayAlert показуємо назву та опис.

10. Запускаємо додаток та дивимось на результат (рис.5). Тестуємо його функціональність.

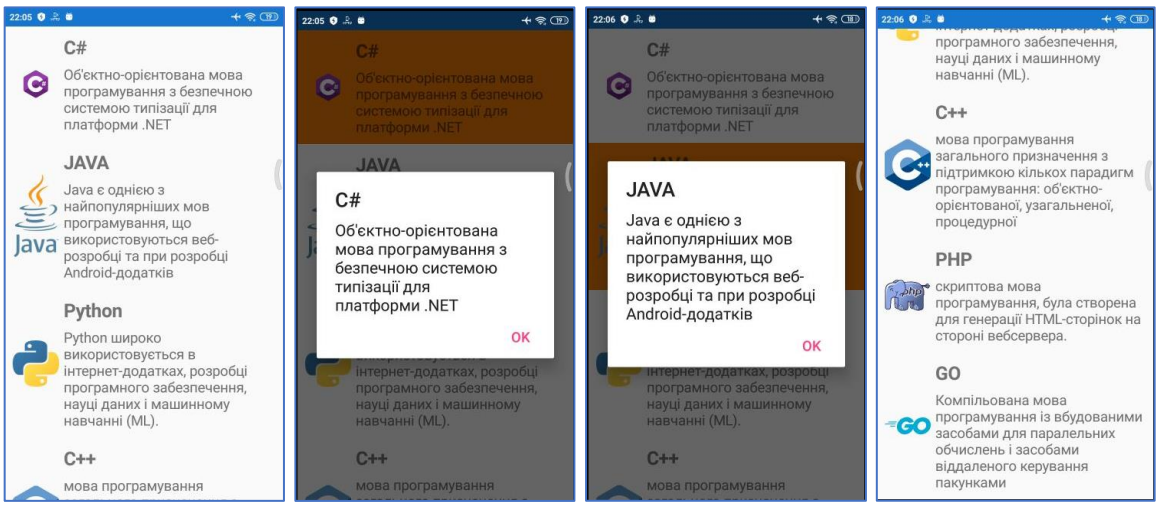


Рисунок 5 – Результати роботи додатку CollectionViewSample

Контрольні питання

1. Як виконується прив'язка даних (data binding) на платформі Xamarin Forms?
2. Як задати джерело прив'язки до будь-якого елемента списку?
3. Як налаштувати контейнер Grid 4×3 в XAML?
4. Як налаштувати контейнер Grid 4×3 в коді C#?
5. Як помістити Label у стовпчик 2 рядка 3?
6. Як помістити Label у об’єднані стовпчики 1,2 рядка 0?
7. Як помістити Label у стовпчик 1 об’єднаних рядків 2,3?
8. Які є способи задати розмір комірки Grid?
9. Як створити список CollectionView?
10. Як зчитати дані з елементу списку CollectionView?

Список використаних джерел

1. Xamarin.Forms Grid. URL: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/layouts/grid
2. Xamarin Forms Grid. URL: https://moonsunblog.com/xamarin-forms-grid/
3. Grids in Xamarin Forms Made Simple. URL: https://xamgirl.com/grids-xamarin-forms-made-simple
4. Grid In Xamarin.Forms. URL: https://www.c-sharpcorner.com/article/grid-in-xamarin-forms
5. Xamarin Forms GridView. URL: https://github.com/kishandonga/Xamarin-GridView
6. Xamarin.Forms CollectionView. URL: https://docs.microsoft.com/ en-us /xamarin/xamarin-forms/user-interface/collectionview