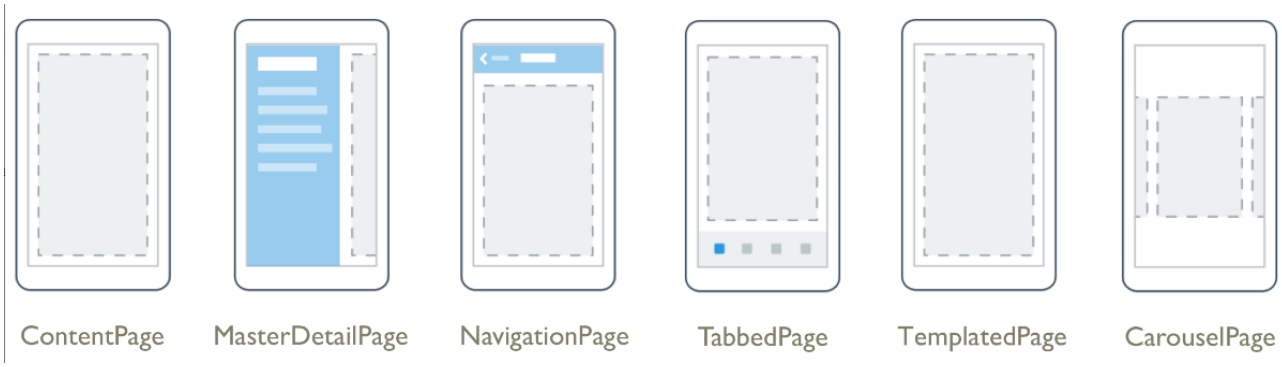
Крос-платформна розробка. Технологія Xamarin. Layout

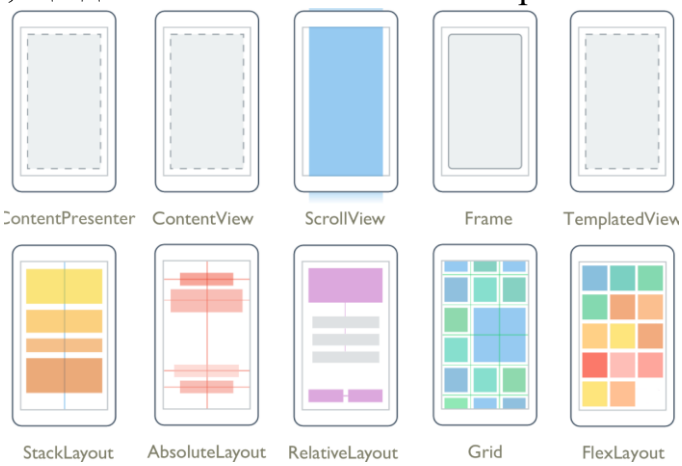
У Xamarin можемо використовувати низку елементів (кнопки, текстові поля, рисунки). Їх поєднує те, що вони успадковані від загального класу View і тому успадковують ряд загальних властивостей.

Крім звичайних елементів типу кнопок і текстових полів, у Xamarin Forms також є контейнери, які дозволяють скомпонувати вміст, розташувати його певним чином.

Елементи розташовуємо на сторінці ContentPage.



Для визначення вмісту сторінки клас сторінки ContentPage має властивість Content. За умовчанням цій властивості присвоюється один елемент Label. Але властивість Content має обмеження – для нього можна встановити лише один елемент. І щоб поміщати на сторінку відразу кілька елементів використовують один з елементів компонування. Елемент компонування є класом, який успадковується від базового класу Layout<T>: StackLayout, AbsoluteLayout, RelativeLayout, Grid, FlexLayout. Всі елементи компонування мають властивість **Children** , що дозволяє встановити або отримати вкладені елементи.



StackLayout самий простий. Елементи розташовуються один під одним по вертикалі (стопка тарілок) або по горизонталі. Елементам не задаємо зв’язку один з одним.

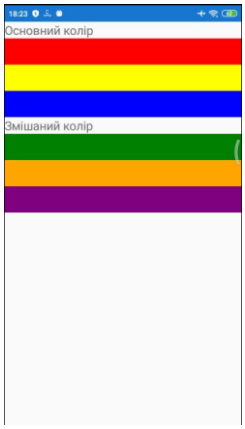
У StackLayout є 3 основних функції: Orientation, Spacing (вказує відстань між дочірніми елементами, по замовчуванню 6 одиниць), Children (колекція куда додаємо control, які хочемо відобразити на екрані)

Розглянемо на прикладі. Створюємо проект. У файлі ManePage.xaml видаляємо код в середині StackLayot та приводимо до вигляду:

```
<?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="StackLayoutApp.MainPage">

    <StackLayout>

    </StackLayout>

</ContentPage>
```

Пишемо код

```
<StackLayout Orientation="Vertical" Spacing="0">
        <Label Text="Основний колір"/>
        <BoxView Color="Red"/>
        <BoxView Color="Yellow"/>
        <BoxView Color="Blue"/>
        <Label Text="Змішаний колір"/>
        <BoxView Color="Green"/>
        <BoxView Color="Orange"/>
        <BoxView Color="Purple"/>
</StackLayout>
```



Повторюємо теж саме у коді на C#. Видаляємо усе з StackLayout та додаємо код в файл ManePage.xaml.cs. Створимо колекцію Children для швидкої ініціалізації елементів

```
protected override void OnAppearing()
{
   StackLayout stackLayout = new StackLayout()
   {
       Children =
       {
           new Label { Text="Основний колір" },
           new BoxView { Color =Color.Red },
           new BoxView { Color =Color.Yellow },
           new BoxView { Color =Color.Blue },
           new Label { Text="Змішаний колір" },
           new BoxView { Color =Color.Green },
           new BoxView { Color =Color.Orange },
           new BoxView { Color =Color.Purple },
         },
         Spacing = 15
      };
      Content = stackLayout;
  }
```

Розглянемо горизонтальну розмітку. Повторюємо код у файлі ManePage.xaml з атрибутом Orientation="Horizontal".

```
<StackLayout Margin="20" Orientation="Horizontal"/>
        <BoxView Color="Red"/>
        <BoxView Color="Yellow"/>
        <BoxView Color="Blue"/>
        <BoxView Color="Green"/>
        <BoxView Color="Orange"/>
        <BoxView Color="Purple"/>

    </StackLayout>
```

У Xamarin.Forms кожен View має дві властивості HorizontalOptions та VerticalOptions. Обидва мають тип LayoutOptions і можуть мати одне з наступних значень:

- LayoutOptions.Start
- LayoutOptions.Center
- LayoutOptions.End
- LayoutOptions.Fill
- LayoutOptions.StartAndExpand
- LayoutOptions.CenterAndExpand
- LayoutOptions.EndAndExpand
- LayoutOptions.FillAndExpand

Start, Center, End, Fill визначають View **вирівнювання в межах його простору.** Expand визначає доступність **займати більше місця.**

Структура LayoutOptions управляє двома різними типами поведінки:
Вирівнювання:

- Start: для вертикального вирівнювання подання переміщається у верхню частину. Для горизонтального вирівнювання це зазвичай ліва сторона.
- Center: по центру.
- End: Вирівнювання знизу або праворуч.
- Fill: розтягнутий на повний розмір батьківського елемента.

Якщо батько, не більше, ніж його діти, то не помітите жодної різниці між цими вирівнюваннями. Вирівнювання має значення лише для батьківських елементів із додатковим вільним простором.
Розширення: Чи буде займати більше місця, якщо він доступний?

- Суфікс Expand: якщо батьківський елемент більше, ніж об'єднаний розмір всіх його дочірніх елементів, тобто додатковий простір доступний, тоді простір пропорційний серед дочірніх елементів із цим суфіксом. Ці діти "займають" свій простір, але не обов'язково "заповнюють" його.
- Нема суфікса: дочірні елементи не отримають додатковий простір, навіть якщо є більше місця.

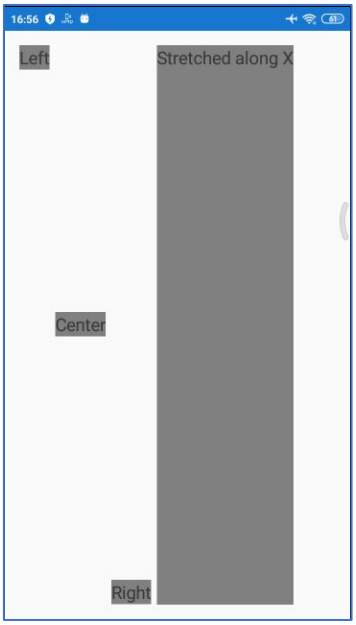
Код розмітки для файлу ManePage.xaml

```
<StackLayout Margin="15">
    <Label Text="Left"
           BackgroundColor="Gray"
           HorizontalOptions="Start"/>
    <Label Text="Center"
           BackgroundColor="Gray"
           HorizontalOptions="Center"/>
    <Label Text="Right"
           BackgroundColor="Gray"
           HorizontalOptions="End"/>
    <Label Text="Stretched along X"
           BackgroundColor="Gray"
           HorizontalOptions="Fill"/>
</StackLayout>
```

Код розмітки для файлу ManePage.xaml.cs

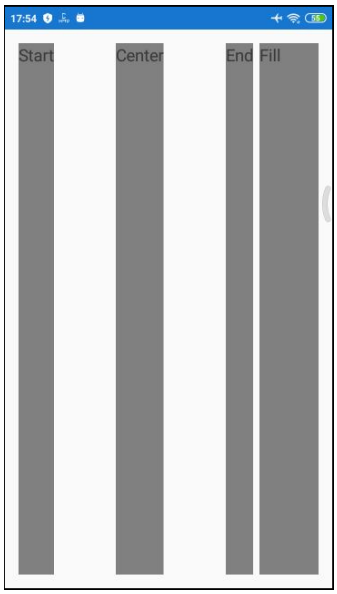
```
protected override void OnAppearing()
{
   StackLayout stackLayout1 = new StackLayout()
   {
      Margin = new Thickness(15),
      Children =
      {
        new Label { Text="Left", BackgroundColor=Color.Gray,
                    HorizontalOptions=LayoutOptions.Start},
        new Label { Text="Center", BackgroundColor=Color.Gray,
                    HorizontalOptions=LayoutOptions.Center},
        new Label { Text="Right", BackgroundColor=Color.Gray,
                    HorizontalOptions=LayoutOptions.End},
        new Label { Text="Stretched along X",
                    BackgroundColor=Color.Gray,
                    HorizontalOptions=LayoutOptions.Fill},
      }
   };
   Content = stackLayout1;
}
```

```
protected override void OnAppearing()
{
    StackLayout stackLayout1 = new StackLayout()
    {
        Margin = new Thickness(15),
        Orientation = StackOrientation.Horizontal,
        Children =
        {
          new Label { Text="Left",
              BackgroundColor=Color.Gray,
              VerticalOptions=LayoutOptions.Start},
          new Label { Text="Center",
              BackgroundColor=Color.Gray,
              VerticalOptions=LayoutOptions.Center},
          new Label { Text="Right",
            BackgroundColor=Color.Gray,
            VerticalOptions=LayoutOptions.End},
          new Label { Text="Stretched along X",
            BackgroundColor=Color.Gray,
            VerticalOptions=LayoutOptions.Fill},
        }
    };
    Content = stackLayout1;
}
```




```
<StackLayout Margin="15">
    <BoxView BackgroundColor="Red"
             HeightRequest="1"/>
    <Label Text="Start"
           BackgroundColor="Gray"

VerticalOptions="StartAndExpand"/>
    <BoxView BackgroundColor="Red"
             HeightRequest="5"/>
    <Label Text="Center"
           BackgroundColor="Gray"

VerticalOptions="CenterAndExpand"/>
    <BoxView BackgroundColor="Red"
             HeightRequest="10"/>
    <Label Text="End"
           BackgroundColor="Gray"
           VerticalOptions="EndAndExpand"/>
    <BoxView BackgroundColor="Red"
             HeightRequest="15"/>
    <Label Text="Fill"
           BackgroundColor="Gray"
           VerticalOptions="FillAndExpand"/>
    <BoxView BackgroundColor="Red"
             HeightRequest="10"/>
 </StackLayout>
```

```
<StackLayout Margin="15" Orientation="Horizontal">
    <Label Text="Start"
           BackgroundColor="Gray"
           HorizontalOptions="StartAndExpand"/>

    <Label Text="Center"
           BackgroundColor="Gray"
           HorizontalOptions="CenterAndExpand"/>

    <Label Text="End"
           BackgroundColor="Gray"
           HorizontalOptions="EndAndExpand"/>

     <Label Text="Fill"
            BackgroundColor="Gray"
            HorizontalOptions="FillAndExpand"/>
</StackLayout>
```



Контейнер Grid

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 та 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.

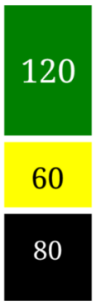
1) Absolute – це фіксований розмір, можна додати будь-яке число до параметра Wight/Height.

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) };
```



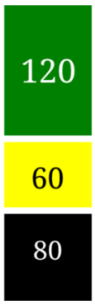
2) Auto – рядок/стовпець адаптуватиметься до дочірнього розміру.

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 можуть 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>
```

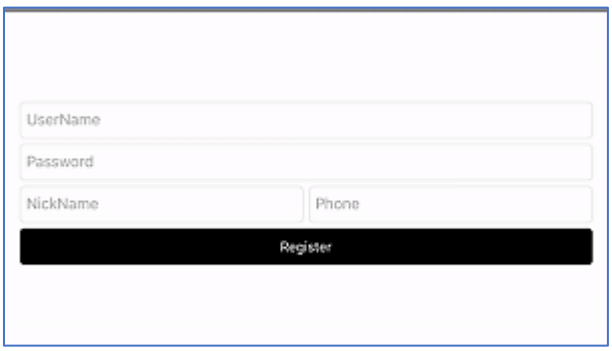


Рисунок 2 – Макет проекту з використанням Grid