Charlin Agramonte
char0394@gmail.com
Twitter: @chard003
FB: chard003
https://xamgirl.com/

Rendy Del Rosario
rendy.delrosario@gmail.com
Twitter: @rdelrosario
FB: RendyDelRosario

CLASS PROGRAM

Week 1

- *Introduction to Xamarin (SHARE PROJECTS/PCL)*
- *Introduction to Xamarin Forms (Let's start our first project)*
- *Navigation, Pages, Controls, Layouts*
- *UI in Xamarin forms (XAML/Code Behind)*

Week 2

- *Resources and styles*
- *ListView*
- *Model- View -View model*
- *Data binding*

Week 3

- *Behaviours*
- *Triggers*
- *Dependency Services*
- *Effects*
- *Renderers*

Week 4

- Introduction to MVVM Frameworks in Xamarin Forms
- *Dependency Injection (Plugins/Services)*
- *Prism (Navigation, Service Container, Messaging)*

Week 5

- *API integrations*
- *Introduction to Refit*
- *Azure App Services*
- *Realm Local Database*

Week 6

- iOS & Android Specifics (Configuration files, Icons, Images)
- *Mobile Center (Crash, Analytics, Building, Distribution)*
- *Distribution (Testing, Store)*
- *Publishing (AppStore, PlayStore)*

Week 1

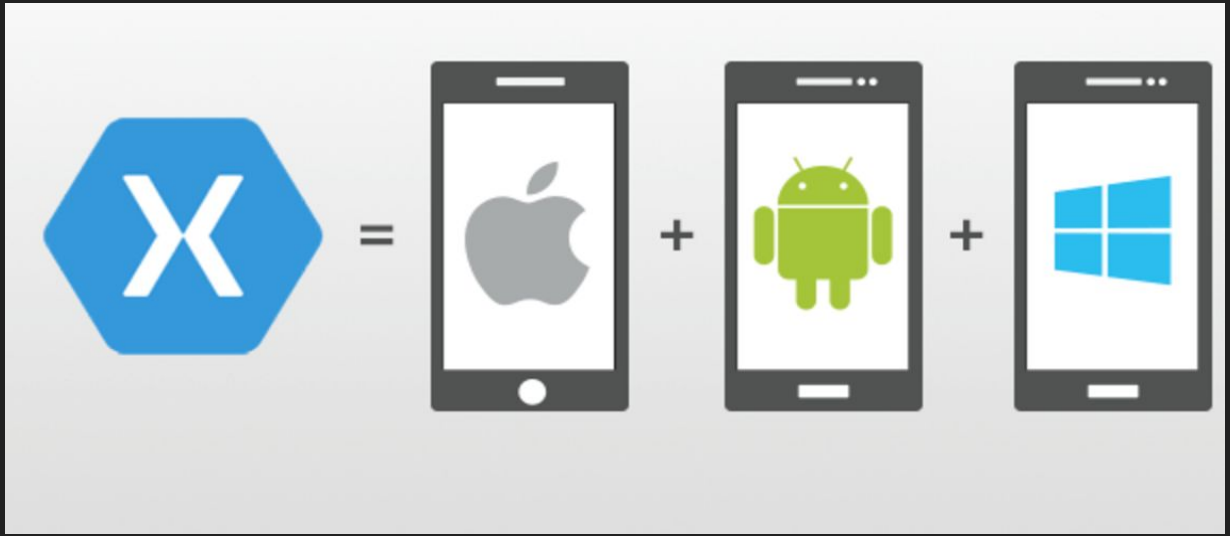
XAMARIN FORMS NOOB TO MASTER

By Rendy Del Rosario and Charlin Agramonte

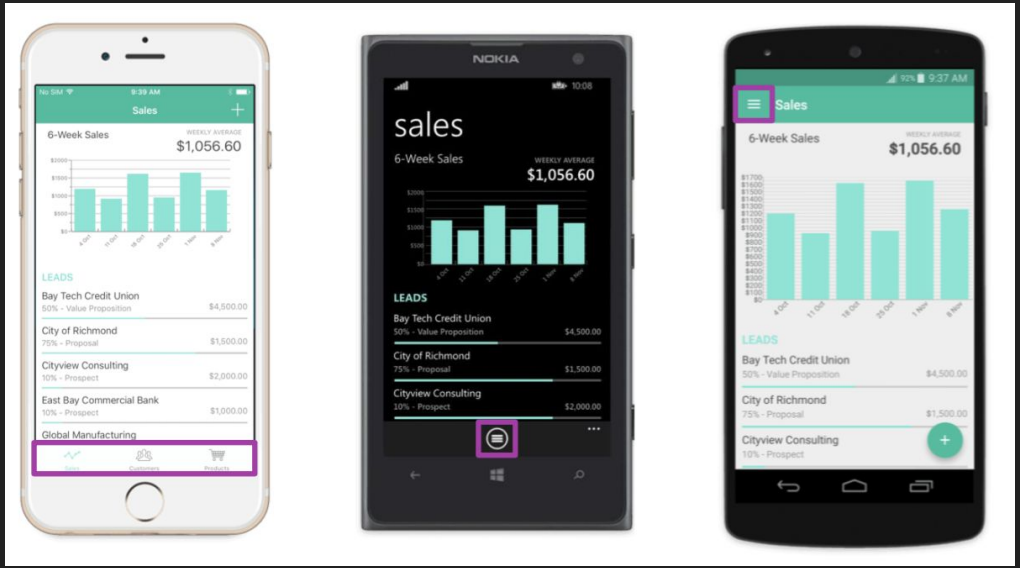
Objectives

- Understand what is Xamarin
- Understand what is Xamarin Forms
- Navigation/Pages/Controls/Layouts
- UI in Xamarin Forms (XAML/Code Behind)

What is Xamarin?

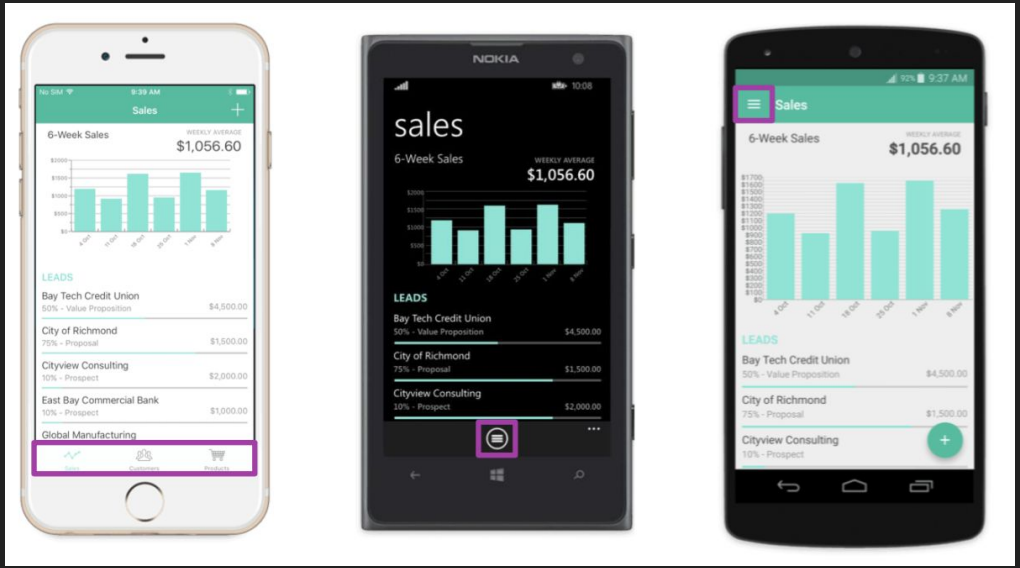


Traditional approach



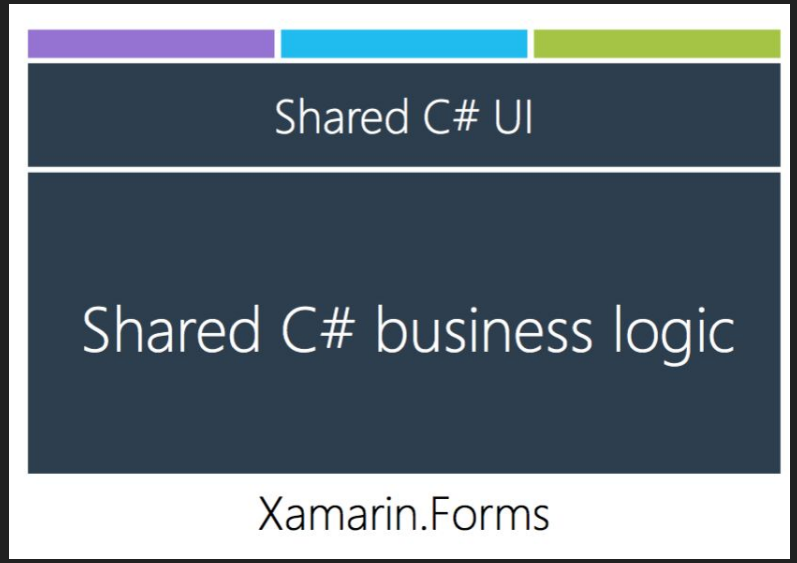
Traditionally, apps have separate code bases written in their native language, are built using native tools, and utilize platform-specific features

What is Xamarin?

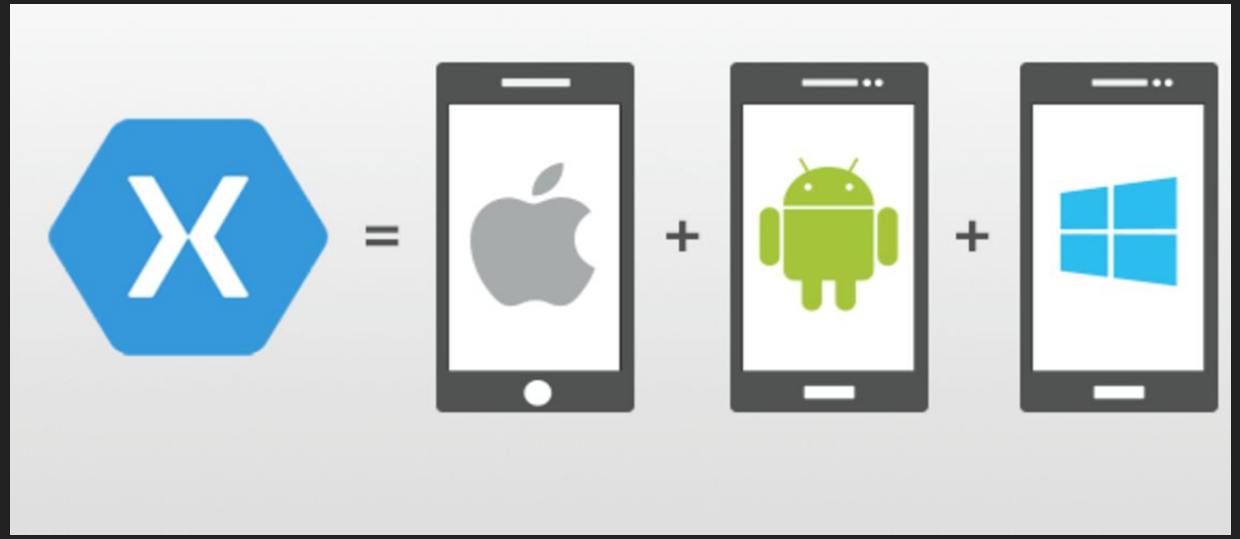


Xamarin is an app-development platform that lets you build apps for many operating systems from a single, shared code base

Xamarin development approaches

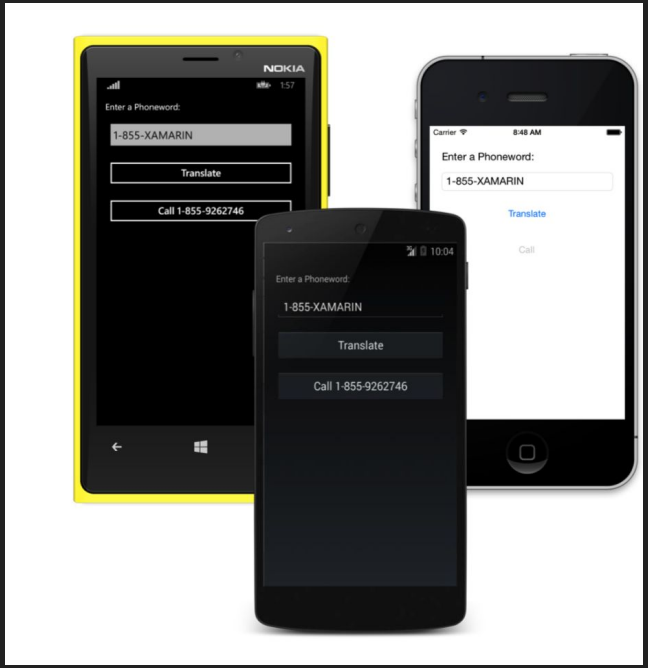


What is Xamarin Forms?

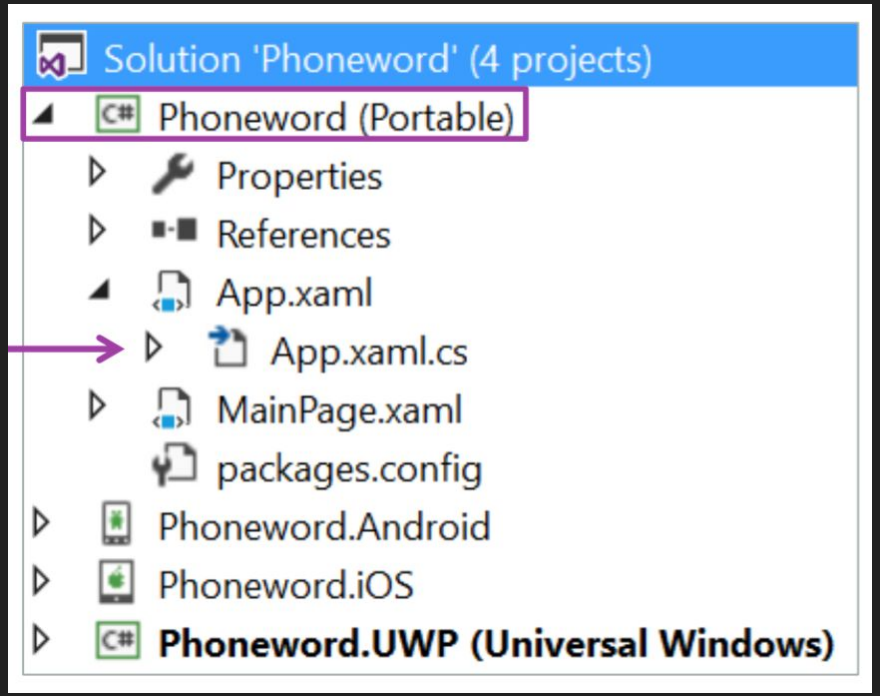


Xamarin Forms

Xamarin.Forms is a cross- platform UI framework to create mobile apps for Android /iOS/Windows.



Project structure



GROUP EXERCISE

Creating a Xamarin Forms application

Xamarin Forms Application

```
public class App : Application
{   ...
    protected override void OnStart() {}
    protected override void OnSleep() {}
    protected override void OnResume() {}
}
```

Use **OnStart** to initialize and/or reload your app's data

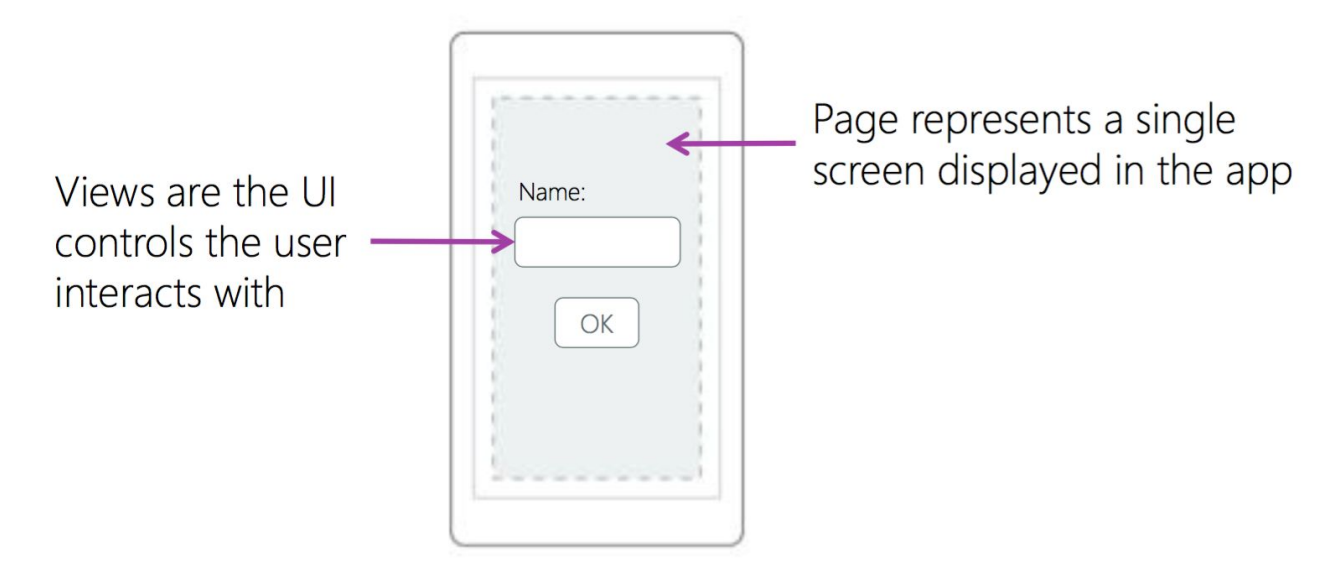
Use **OnSleep** to save changes or persist information

Use **OnResume** to refresh your displayed data

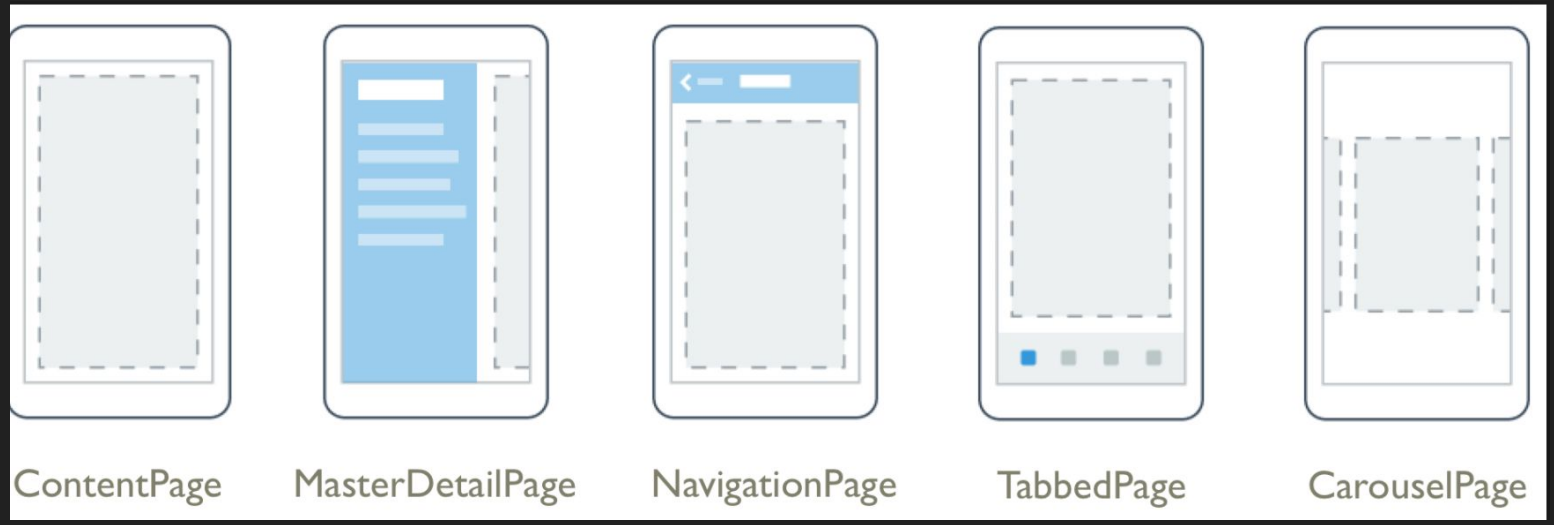
Application class provides lifecycle methods which can be used to manage persistence and refresh your data

Creating the application UI

Application UI is defined in terms of *pages* and *views.*



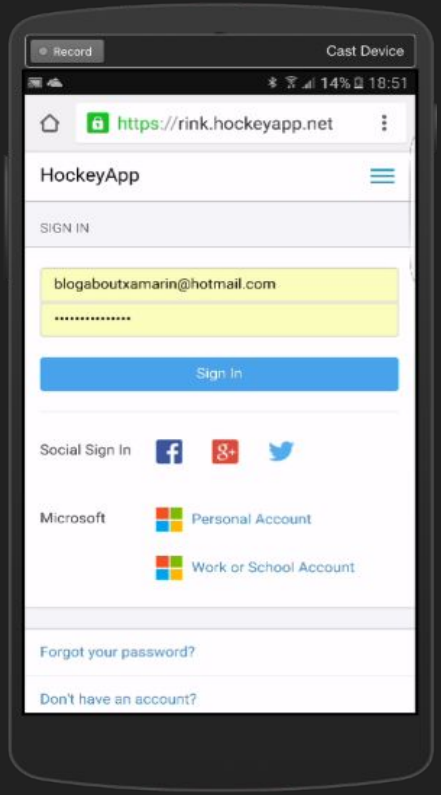
Pages



Page is an abstract class used to define a single screen of content.

Views

Label	Image	SearchBar
Entry	ProgressBar	ActivityIndicator
Button	Slider	OpenGLView
Editor	Stepper	WebView
DatePicker	Switch	ListView
BoxView	TimePicker	
Frame	Picker	



View is the base class for all visual controls, most standard controls are present

Views


Forms Gallery
ActivityIndicator


ActivityIndicator

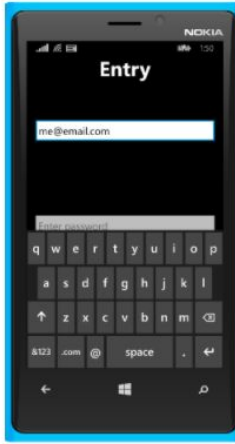

Forms Gallery
Label
Xamarin.Forms is a cross-platform natively backed UI toolkit abstraction that allows developers to easily create user interfaces that can be shared across Android, iOS, and Windows Phone.


Label
Xamarin.Forms is a cross-platform natively backed UI toolkit abstraction that allows developers to easily create user interfaces that can be shared across Android, iOS, and Windows Phone.


NOKIA
Label
Xamarin.Forms is a cross-platform natively backed UI toolkit abstraction that allows developers to easily create user interfaces that can be shared across Android, iOS, and Windows Phone.


Forms Gallery
Entry


Entry
itis@mymail.com


NOKIA
Entry
me@email.com

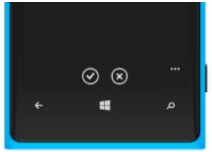

Forms Gallery
Image

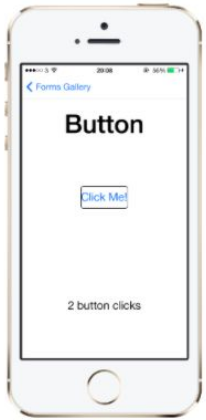

Image


NOKIA
Image


Forms Gallery
DatePicker


Mon, May 26, 2014
Done



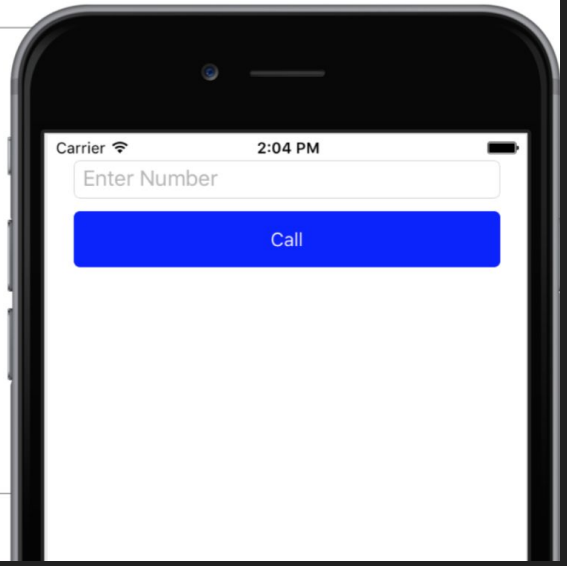

Forms Gallery
Button
Click Me!
2 button clicks


Button
Click Me!
3 button clicks


NOKIA
Button
Click Me!
3 button clicks

Visual adjustment

```
var numEntry = new Entry {
    Placeholder = "Enter Number",
    Keyboard = Keyboard.Numeric
};

var callButton = new Button {
    Text = "Call",
    BackgroundColor = Color.Blue,
    TextColor =  Color.White
};
```



Views utilize properties to adjust visual appearance and behavior

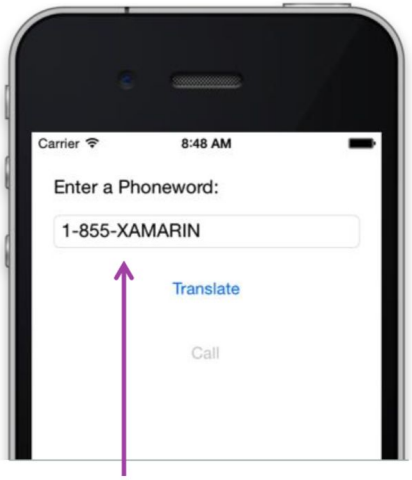
Providing Behavior

```
var testEntry = new Entry();
testEntry.TextChanged += (sender, e) => {

};

var buttonText = new Button();
buttonText.Clicked += (sender, e) => {

};
```

Controls use events to provide interaction behaviours.

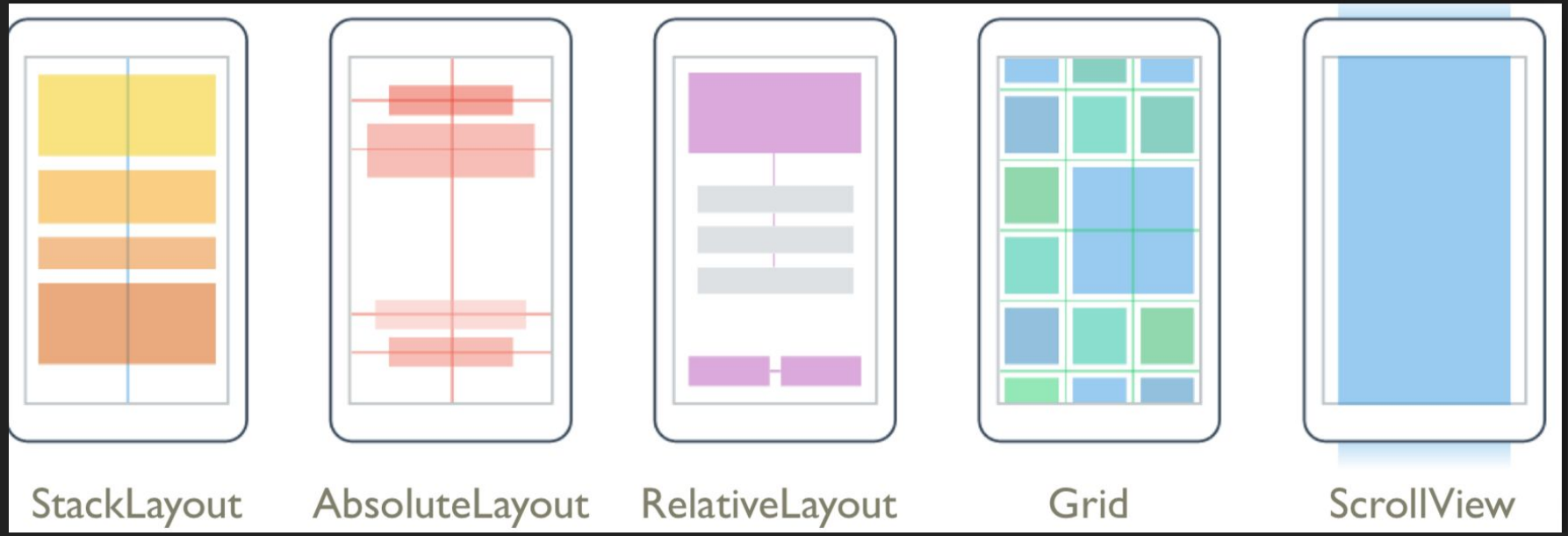
Organizing Content

Rather than specifying positions with coordinates (pixels, dips, etc.), you use layout containers to control how views are positioned relative to each other



For example, "stacking" views on top of each other with some spacing between them

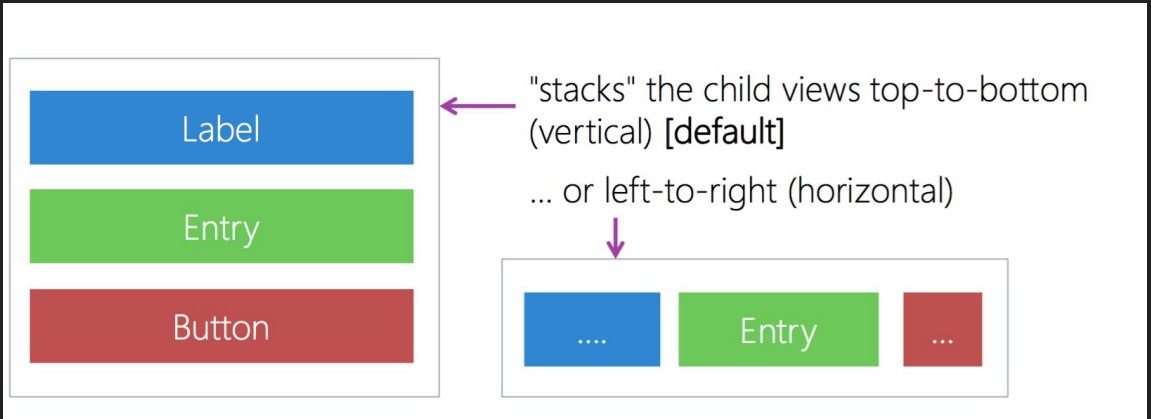
Layout containers



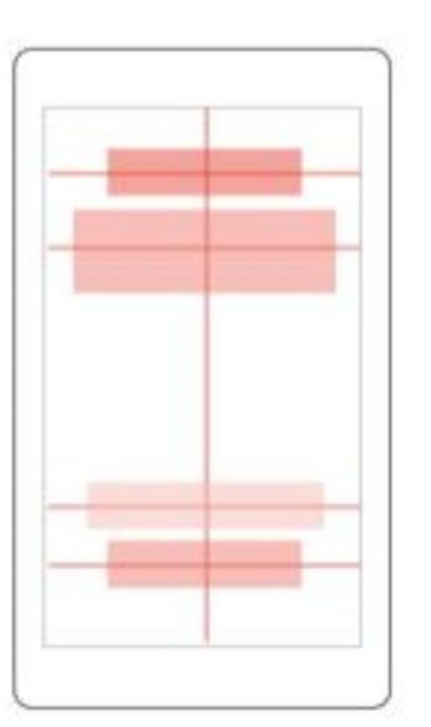
Organize child elements based on specific rules.

StackLayout

`StackLayout` places children top-to-bottom (default) or left-to- right based on `Orientation` property setting



AbsoluteLayout

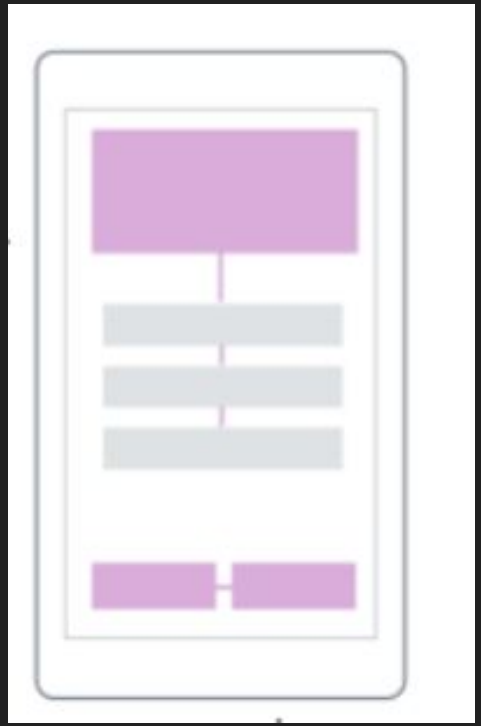


`AbsoluteLayout` places children in absolute requested positions based on anchors and bounds

```
var layout = new AbsoluteLayout();
...
// Stretch image across entire container
layout.Children.Add(fillImage, new Rectangle (0, 0, 1, 1),
                    AbsoluteLayoutFlags.All );
```

Here we "fill" the container with an image
[0,0] – [1,1]

RelativeLayout



`RelativeLayout` allows you to position child views relative to two other views, or to the panel itself using constraint-based rules

```
var layout = new RelativeLayout();
...
layout.Children.Add(label,
     Constraint.RelativeToParent(
           parent => (0.5 * parent.Width) - 25),    // X
     Constraint.RelativeToView(button,
           (parent, sibling) => sibling.Y + 5),     // Y
     Constraint.Constant(50),                       // Width
     Constraint.Constant(50));                      // Height
```

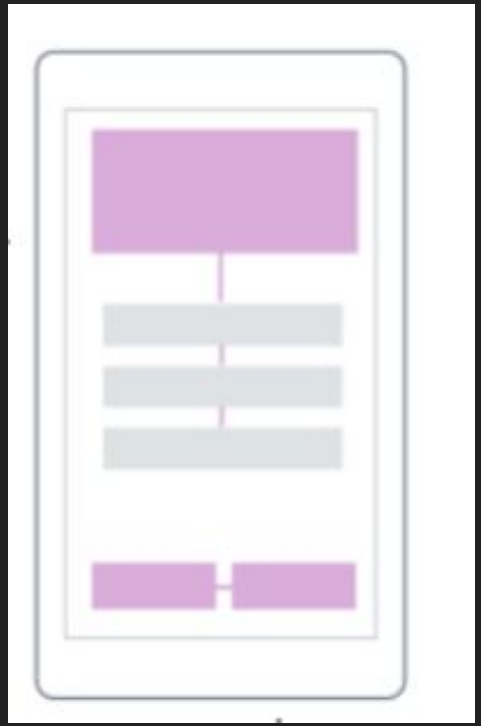
ScrollView

`ScrollView` scrolls a single piece of content (which is normally a layout container)

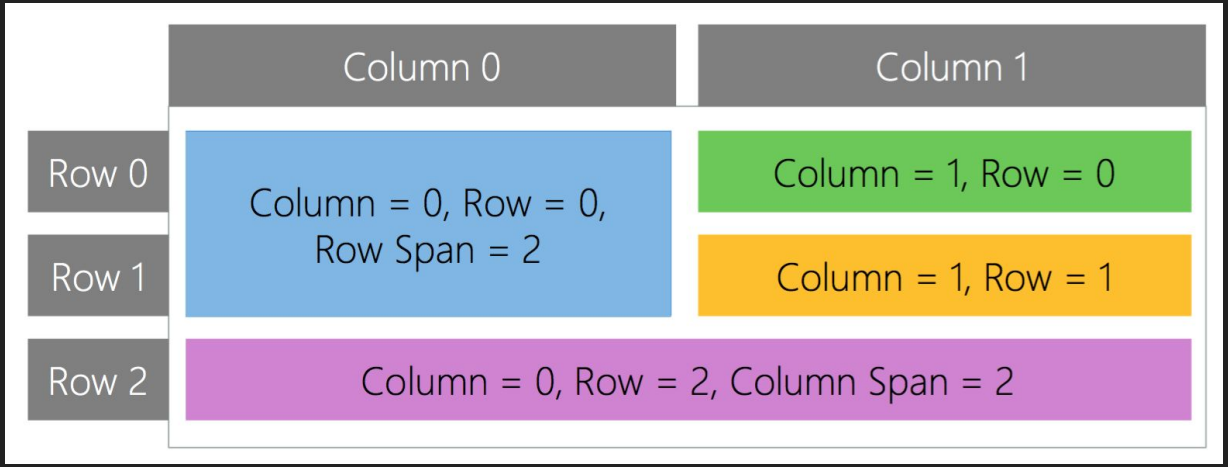
```
<ScrollView>
   <StackLayout>
      <BoxView Color="Silver" HeightRequest="100" />
      <BoxView Color="Blue"   HeightRequest="200" />
      <BoxView Color="Gray"   HeightRequest="300" />
      <BoxView Color="Navy"   HeightRequest="200" />
   </StackLayout>
</ScrollView>
```

Wrap a `ScrollView` around a single element to add scrolling

Grid



`Grid` is a layout panel used to create rows and columns of views, children identify specific column, row and span



How to create a Grid?

- Defining the Grid

```
var grid = new Grid();
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength(2, GridUnitType.Star) });
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength (1, GridUnitType.Star) });
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength(200)});
grid.ColumnDefinitions.Add (new ColumnDefinition{ Width = new GridLength (200) });
```

- Add Childrens

```
controlGrid.Children.Add (new Button { Text = "C", Style = darkerButton }, 0, 1);
controlGrid.Children.Add (new Button { Text = "+/-", Style = darkerButton }, 1, 1);
```

What is GridLenght?

```
public struct GridLength
{   ...
    public GridUnitType GridUnitType { get; }
    public double       Value        { get; }
}
```

Units can be: **Absolute**, **Auto**, **Star**

Absolute

```
var row = new RowDefinition() { Height = new GridLength(100) };
```

```
<RowDefinition Height="100" />
```

Value is in platform-independent units

Star

```
var row = new RowDefinition() { Height = new GridLength(2.5, GridUnitType.Star) };
```

```
<RowDefinition Height="2.5*" />
```

XAML type converter uses `*` instead of the **Star** used in code.
Note: `"1*"` and `"*"` are equivalent in XAML.

Auto

```
var row = new RowDefinition() {Height = new GridLength(1, GridUnitType.Auto)};
```

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

Value is irrelevant for **Auto**, it is typical to use 1 as the value when creating in code

Grid - Add Childrens

```
var grid = new Grid();
int row, column;
...
grid.Children.Add(label, column, row);

grid.Children.Add(button, column, column+1, row, row+2);
```

Yields a **ColumnSpan** of 1

Yields a **RowSpan** of 2

Grid - Code

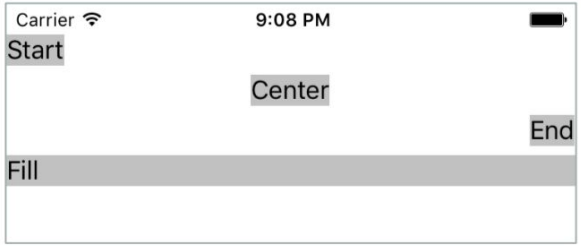
```
var grid = new Grid();

        grid.RowDefinitions.Add(new RowDefinition { Height = new GridLength(0.4, GridUnitType.Auto) });

        grid.RowDefinitions.Add(new RowDefinition { Height = new GridLength(0.5, GridUnitType.Auto) });

        grid.RowDefinitions.Add(new RowDefinition { Height = new GridLength(0.1, GridUnitType.Auto) });

        grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(200) });

//Add Childrens

        grid.Children.Add(new BoxView() { BackgroundColor = Color.Red, HorizontalOptions = LayoutOptions.FillAndExpand}, 0,0);

        grid.Children.Add(new BoxView() { BackgroundColor = Color.Green, HorizontalOptions = LayoutOptions.FillAndExpand }, 0,
1);

        grid.Children.Add(new BoxView() { BackgroundColor = Color.Yellow, HorizontalOptions = LayoutOptions.FillAndExpand }, 0,
2);
```

LayoutOptions

```
public class View : ...
{
    public LayoutOptions HorizontalOptions { get; set; }
    public LayoutOptions VerticalOptions   { get; set; }
    ...
}
```

Alignment

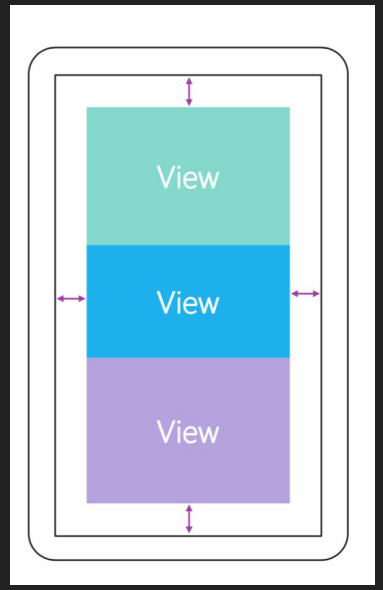
```
<StackLayout>
   <Label Text="Start"  HorizontalOptions="Start"  BackgroundColor="Silver" />
   <Label Text="Center" HorizontalOptions="Center" BackgroundColor="Silver" />
   <Label Text="End"    HorizontalOptions="End"    BackgroundColor="Silver" />
   <Label Text="Fill"   HorizontalOptions="Fill"   BackgroundColor="Silver" />
</StackLayout>
```



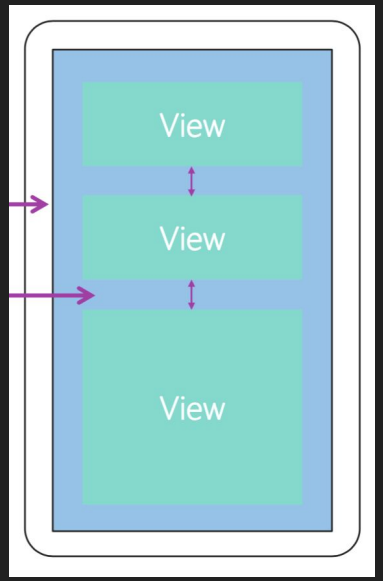
Expansion

```
<StackLayout Orientation="Vertical">
   <Label ... VerticalOptions="StartAndExpand"  />
   <Label ... VerticalOptions="CenterAndExpand" />
   <Label ... VerticalOptions="EndAndExpand"    />
   <Label ... VerticalOptions="FillAndExpand"   />
</StackLayout>
```

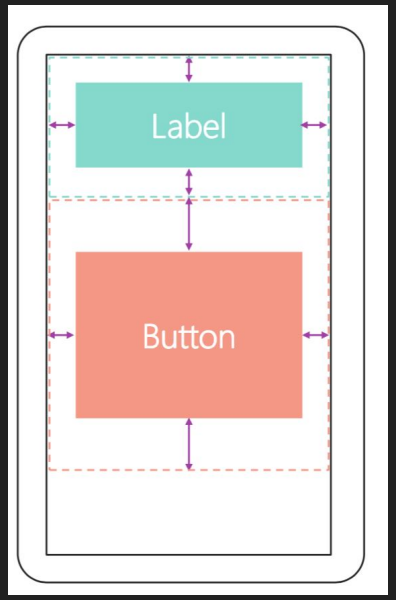
Properties



`Padding:` `Padding` adds distance between the inside edges of a layout container and its children (only available in layouts)



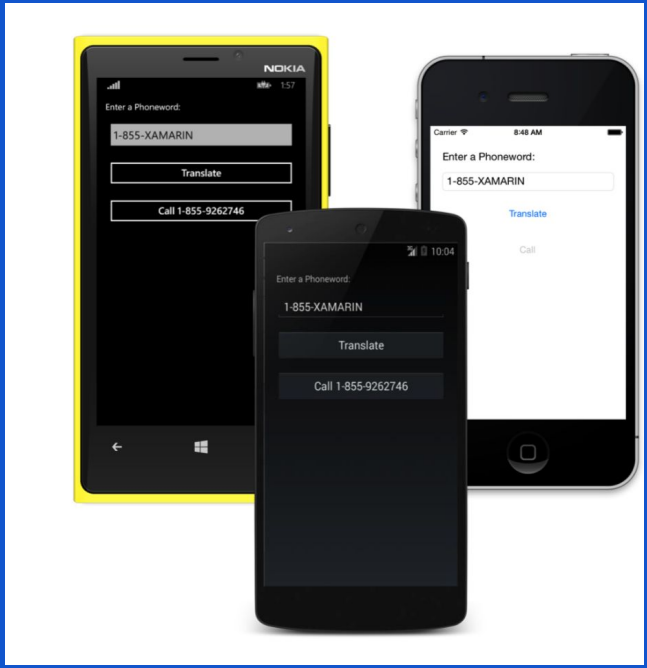
`Spacing:` The `Spacing` property of `StackLayout` and controls the distance between child elements



`Margin:` Add distance for a view

GROUP EXERCISE

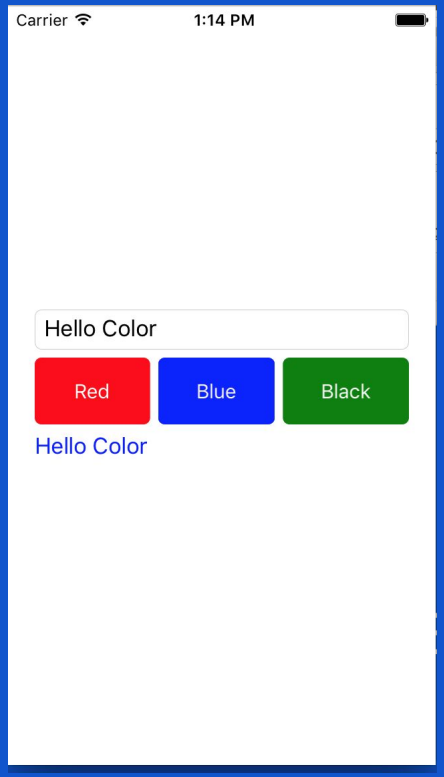
Creating a basic UI



INDIVIDUAL EXERCISE

Creating a Text color procesador

TIME: 25 MINUTES



Using Platform Features

`Device.OpenUri`
to launch external apps based on a URL scheme

`Page.DisplayAlert`
to show simple alert messages

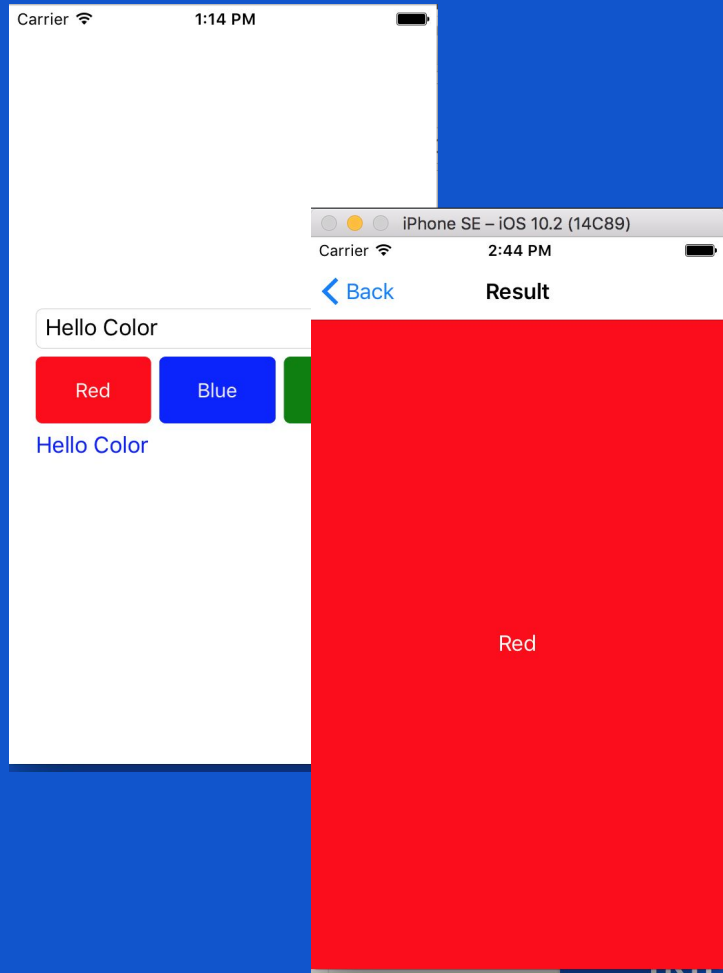
Timer management using **`Device.StartTimer`**

Navigation

- PushAsync
- PopAsync
- PushModalAsync
- PopModalAsync
- RemovePage
- PopToRootAsync

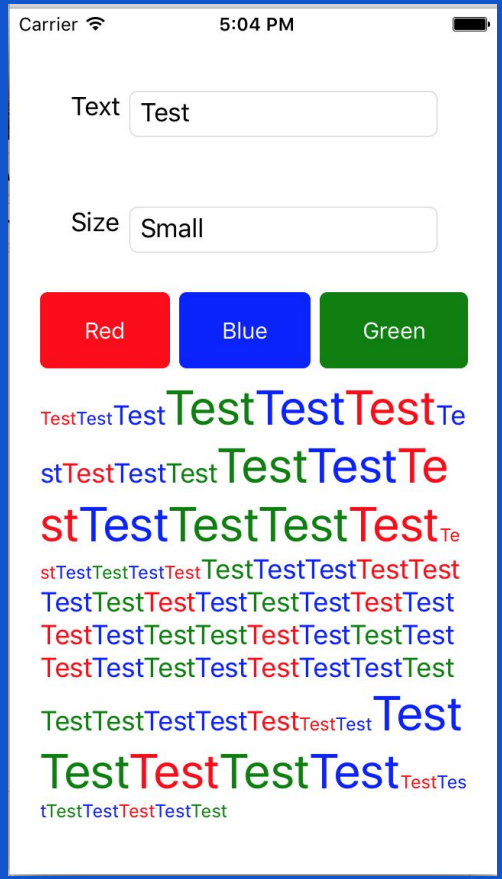
HOMEWORK PRACTICE

Creating Navigation Options to Our Color procesador



HOMEWORK PRACTICE

Colorful Text editor



THANK YOU!!!

By Rendy Del Rosario and Charlin Agramonte