

Xamarin Inteligente - Módulo 2

Humberto Jaimes - humberto@humbertojaimes.net

Creando interfaces XAML

En este ejercicio vamos a trabajar con tres de los patrones de diseño que se pueden utilizar cuando queremos introducir lógica específica de las plataformas (sensores, hardware de terceros, APIs de Google) en Xamarin utilizando una PCL para el proyecto Core.

Requerimientos

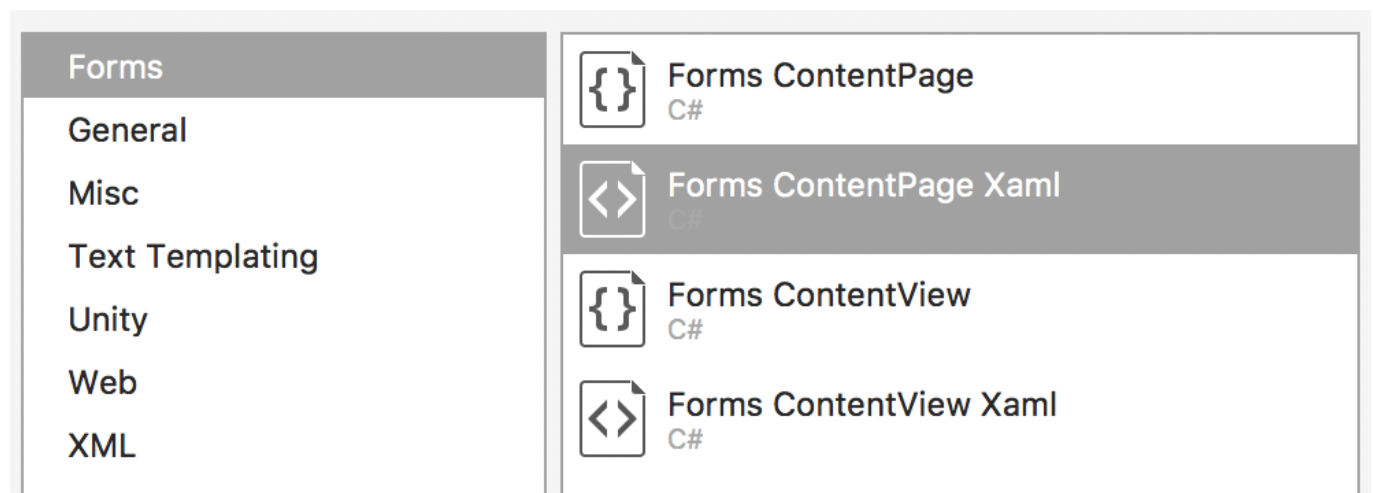
- Crear un proyecto en blanco de Xamarin.Forms
- El proyecto debe usar .NET Standard
- Crear una carpeta "Views" dentro del proyecto

Creando una página XAML para el Login

Nuestras apps están conformadas por una o más pantallas, en el caso de Xamarin.Forms cada pantalla se representa mediante una página XAML, en temas posteriores revisaremos los diferentes tipos de páginas, a lo largo de este ejercicio solo usaremos la más simple, la cual es la `ContentPage`.

A la hora de generarla no olvides usar la que es con **XAML** y como buena práctica los archivos llevan el sufijo **Page** por ejemplo: `LoginPage`, `ProfilePage`, `MainPage`.

Para crear las páginas agrega un nuevo elemento a la carpeta **Views** y selecciona la opción "Forms `ContentPage` XAML" dale por nombre "LoginPage"



Después de crear la página tendrás algo parecido a esto

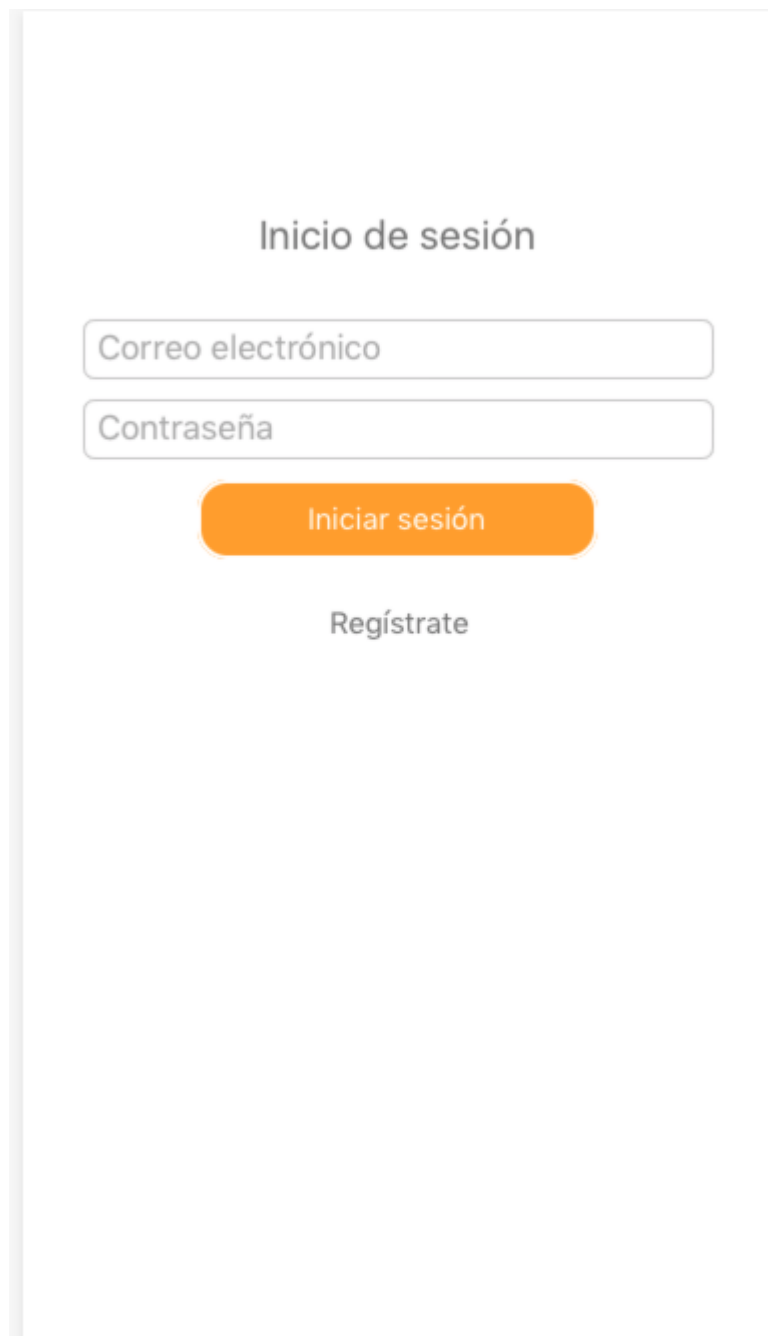
```
1 <?xml version="1.0" encoding="UTF-8"?>
2   <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Pages_Demo.LoginPage">
5     <ContentPage.Content>
6     </ContentPage.Content>
7 </ContentPage>
8
```

Recuerda que algunos elementos XAML pueden contener hijos directamente sin necesidad de poner las etiquetas con las propiedades, ese es el caso de la propiedad `<ContentPage.Content>`, en algunas versiones del IDE no la genera, sin embargo, la propiedad `Content` es la propiedad por defecto de la `ContentPage` por lo que el código anterior es el equivalente a este.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2   <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="Pages_Demo.LoginPage">|
5 </ContentPage>
6
```

Estructurando nuestros elementos

1. Ahora vamos a comenzar a crear la pantalla que verá el usuario al interactuar con nuestra app, el resultado final será algo como esto.

A login form UI mockup. At the top, the text "Inicio de sesión" is centered. Below it are two input fields: "Correo electrónico" and "Contraseña". Under the password field is an orange button with the text "Iniciar sesión". At the bottom, the text "Regístrate" is centered.

El primer paso es crear un StackLayout que ordene nuestros elementos de arriba hacia abajo, este StackLayout contendrá los siguientes elementos:

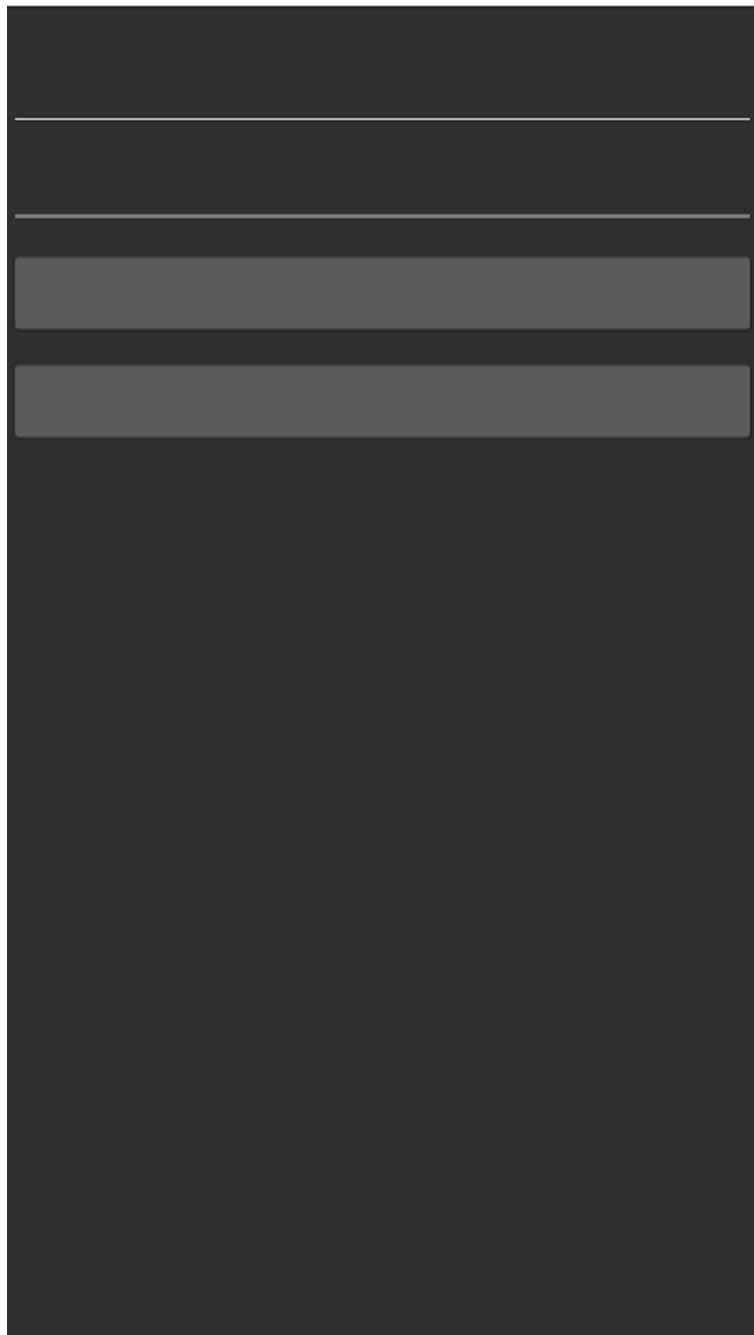
- Label para la descripción
- Entry para el nombre de usuario
- Entry para la contraseña
- Botón para iniciar sesión
- Botón para el registró
- ActivityIndicator para mostrar al usuario cuando la app este realizando algún proceso

El código es el siguiente:

```
<StackLayout>
  <Label />
  <Entry />
  <Entry />
```

```
<Button/>  
<Button/>  
<ActivityIndicator/>  
</StackLayout>
```

EL resultado es el siguiente



2. Hasta este punto solo tenemos el cascaron de nuestra página de Login, ahora comenzaremos a trabajar con los tamaños y posiciones para lograr el resultado que deseamos.

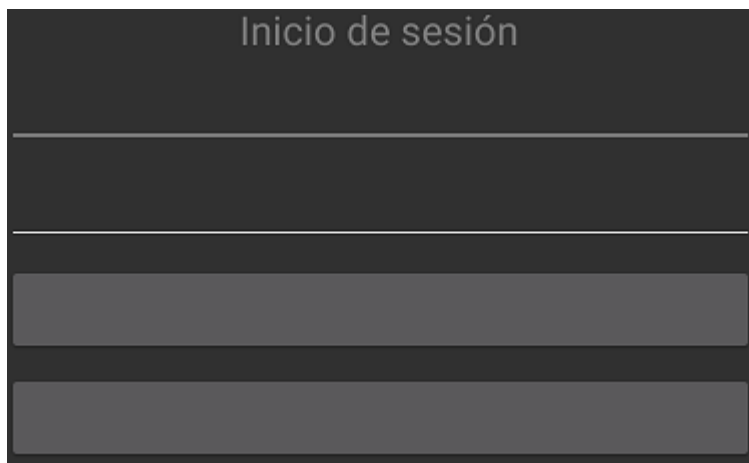
Vamos a comenzar con el Label, este Label tendrá las siguientes propiedades:

- **Text** = Inicio de Sesión -> Texto descriptivo
- **FontSize** = 20 -> Tamaño de la fuente
- **TextColor** = Gray -> Color del Texto
- **HorizontalOptions** = Center -> Alineación del elemento

En código el resultado es el siguiente:

```
<Label Text="Inicio de sesión" TextColor="Gray" FontSize="20"  
HorizontalOptions="Center"/>
```

Y la pantalla se convierte en esto



3. Ahora a nuestro primer Entry le pondremos un texto descriptivo para indicarle al usuario que debe introducir su correo electrónico y al segundo Entry le pondremos un indicador de que es el campo para la contraseña, esto lo hacemos a través de la propiedad Placeholder

```
<Entry Placeholder="Correo electrónico"/>  
<Entry Placeholder="Contraseña" />
```

Al aplicar este cambios nuestros Entry están estéticamente listos para mostrarse sin embargo para mejorar la usabilidad de nuestra pantalla podemos asignarles algunas propiedades que hacen sentido a su funcionalidad.

La primera de ellas es la propiedad **Keyboard** con la que podemos definir el teclado que se le mostrará al usuario dependiendo del tipo de entrada que esperemos, los valores aceptados son los siguientes:

- Default
- Text
- Chat
- Url
- Email
- Telephone
- Numeric

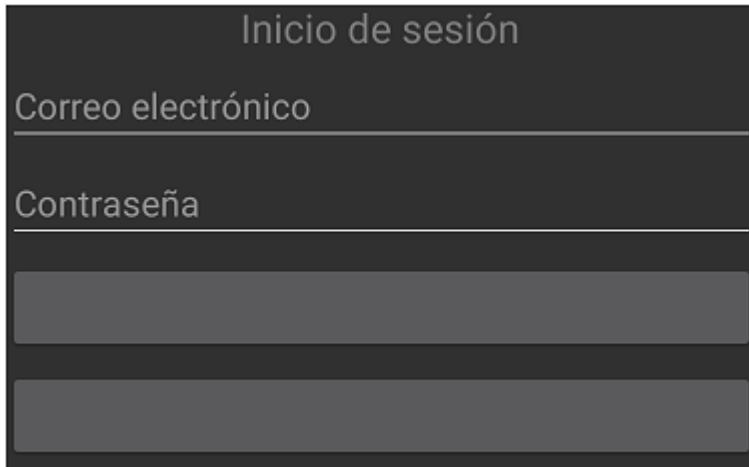
En nuestro caso para el primer Entry podemos hacer uso del valor Email.

```
<Entry Placeholder="Correo electrónico" Keyboard="Email"/>
```

La funcionalidad del segundo Entry es la de recibir una contraseña por lo que necesitamos que la entrada de texto no se muestre, para eso existe una propiedad **IsPassword** esta acepta los valores de True o False, en este caso lo asignaremos como True para el segundo Entry

```
<Entry Placeholder="Contraseña" IsPassword="true" />
```

Al modificar nuestros Entry el resultado en el Previewer es el siguiente:



4. Ahora es el turno de los botones, el primero lo queremos con este estilo

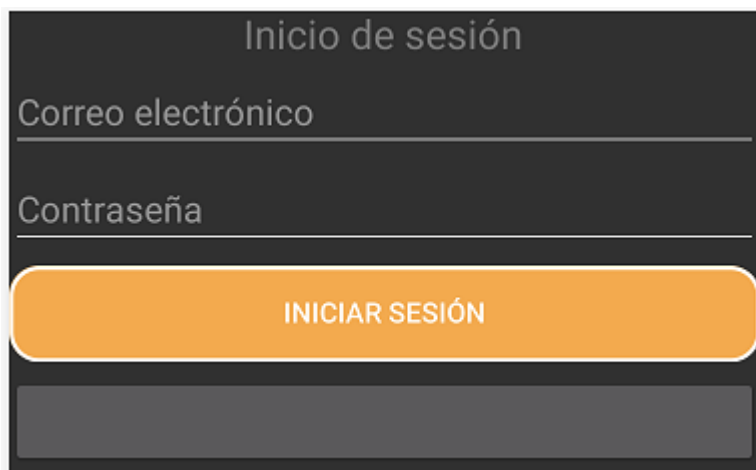


Para eso haremos uso de varias propiedades, empecemos por la parte visual. Para esto asignaremos las siguientes propiedades

- **Text** = Iniciar Sesión -> Texto a mostrar
- **TextColor** = White -> Color del texto
- **BackgroundColor** = #FFA733 -> Color de fondo del botón
- **BorderRadius** = 15 -> Tamaño de la curva del borde
- **BorderColor** = White -> Color del borde
- **BorderWidth** = 2 -> Ancho del borde

```
<Button  
Text="Iniciar sesión"  
TextColor="White"  
BackgroundColor="#FFA733"  
BorderRadius="15"  
BorderColor="White"  
BorderWidth="2"/>
```

Nuestro botón queda de la siguiente manera

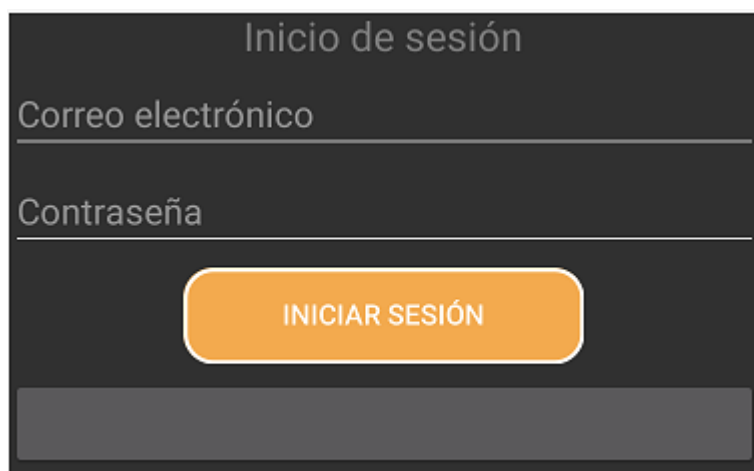


Ahora haremos el botón más pequeño y lo centraremos, para eso usaremos las propiedades `WidthRequest` y `HorizontalOptions`, de modo que el código ahora quede así

```
<Button
    Text="Iniciar sesión"
    TextColor="White"
    BackgroundColor="#FFA733"
    BorderRadius="15"
    BorderColor="White"
    BorderWidth="2"
    WidthRequest="200"
    HorizontalOptions="Center"/>
```

Nota: Recuerda que Xamarin.Forms asigna los tamaños de acuerdo al espacio disponible y que por defecto ***HorizontalOptions*** tiene un valor de ***Fill***, por lo que si quitas la línea `HorizontalOptions="Center"` puedes notar que se ignora el valor de 200 en la propiedad ***WidthRequest***

Ahora el botón se ve de este modo:

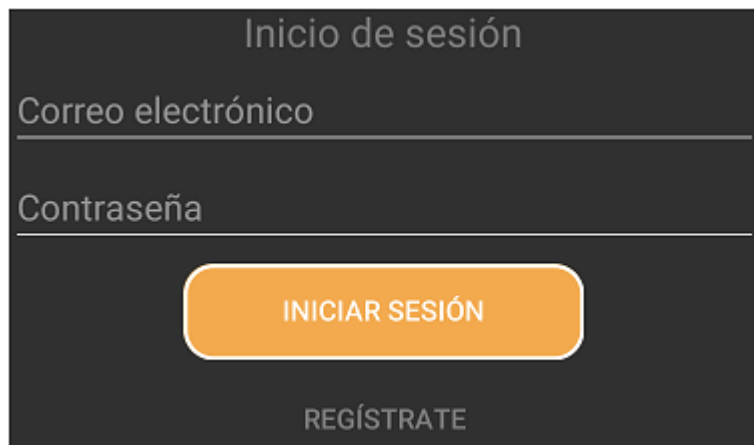


Para el siguiente botón el estilo es más parecido a un `Label`, para lograr eso asignaremos solo tres propiedades.

- `Text` = Regístrate -> Texto a mostrar
- `TextColor` = Gray -> Color del texto
- `BackgroundColor` = Transparent -> Color de fondo del botón

```
<Button Text="Regístrate" TextColor="Gray" BackgroundColor="Transparent" />
```

El resultado es el siguiente:

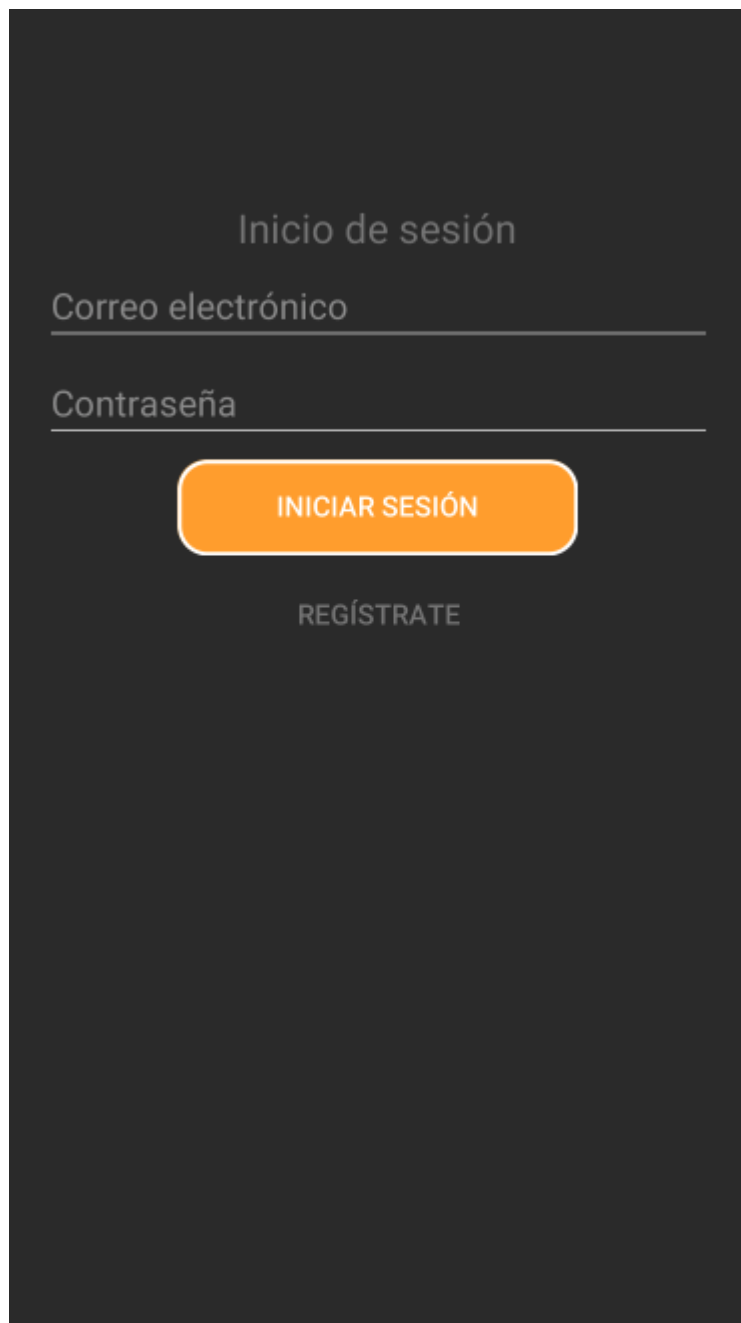


5. Finalmente centraremos nuestro ActivityIndicator, el cual usaremos en el futuro.

Ya tenemos nuestros elementos con el estilo visual que queríamos ahora solo hace falta acomodar en un mejor lugar todos los elementos, hay muchas maneras de hacerlo, sin embargo, para este ejemplo usaremos la propiedad `Padding` del `StackLayout`, al final debe quedar de este modo

```
<StackLayout Padding="20,100,20,0">
```

Al hacer el cambio el Previewer debe mostrar la siguiente pantalla para cada plataforma



En iOS existe una configuración especial, la cual nos permite que los elementos no se superpongan sobre la barra superior.

Para poder activarla es necesario agregar las siguientes líneas dentro de nuestra página. Justo en la parte donde se declaran los namespaces y atributos de la ContentPage.

```
xmlns:ios="clr-
namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin
.Forms.Core"
    ios:Page.UseSafeArea="true"
```

Aplicando el cambio la página quedaría

```
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="XamarinInteligenteRefractored.Views.OrderPage"
    xmlns:ios="clr-
namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin
.Forms.Core"
    ios:Page.UseSafeArea="true">
```

Creando una página XAML para el Registro

1. Vamos a crear una página en blanco llamada SignUpPage, donde el resultado final debe ser el siguiente

INFORMACIÓN DEL USUARIO

Nombre

Dirección

Teléfono de contacto

INFORMACIÓN DE LA CUENTA

Correo Electrónico

Contraseña

2. Para crear esta página haremos uso de un control llamado TableView, el cual nos permite crear una estructura de lista de elementos dentro de un Scroll y a diferencia de un ListView este no cuenta con el concepto de ItemsSource por lo que agregaremos los elementos como hijos de la TableView.

Primero definiremos que campos queremos que el usuario introduzca, los cuales son los siguientes:

- El nombre del usuario
- Su dirección física
- Su teléfono
- Su correo electrónico
- Su contraseña

Esos campos los dividiremos en dos categorías

- Información del usuario: Nombre, dirección y teléfono
- Información de la cuenta: Correo electrónico y contraseña

3. Ahora comenzaremos a crear nuestro XAML, comenzamos poniendo el TableView con algunas propiedades y con sus secciones definidas

```
<TableView Intent="Form" HasUnevenRows="true">
  <TableSection Title="Información del usuario">
  </TableSection>
  <TableSection Title="Información de la cuenta">
  </TableSection>
</TableView>
```

La propiedad Intent nos permite definir la funcionalidad de la TableView, en este caso la usaremos para un formulario por eso le dimos el valor de Form, los valores aceptados son los siguientes:

- Data
- Form
- Menu
- Settings

Estos valores solo modifican algunos aspectos visuales del TableView y no su comportamiento.

La propiedad HasUnevenRows sirve para indicar si las celdas de la tabla van a tener una altura variable dependiendo de su contenido o de lo que definamos, para que la tabla se vea correctamente por eso la ponemos como true.

Una TableView puede dividir el contenido en secciones para lograr esto podemos hacer uso de las etiquetas TableSection, en nuestro caso definimos dos la de información del usuario y la de información de cuenta.

4. Ahora es turno de comenzar a poner nuestros controles, las TableSection solo soportan elementos del tipo Cell, los tipos de Cell disponibles son los siguientes:

- EntryCell
- SwitchCell
- TextCell
- ImageCell
- ViewCell

Como nosotros queremos una estructura personalizada usaremos la de tipo ViewCell la cual puede aceptar como hijos otro control o contenedor, en nuestro caso será un contenedor de tipo StackLayout.

Todos los ViewCell para esta pantalla de registro están basados en el siguiente:

```
<ViewCell >
  <StackLayout Padding="5,5,5,5" Orientation="Horizontal">
    <Entry Placeholder="Nombre"
HorizontalOptions="FillAndExpand" />    <Image HeightRequest="30"
WidthRequest="30" />
  </StackLayout>
</ViewCell>
```

Dentro del ViewCell tenemos los siguientes elementos:

- StackLayout para apilar dos elementos de forma horizontal
- Entry para recibir el dato del usuario
- Image para mostrar un icono dependiendo de si la entrada es correcta o no (para uso futuro)

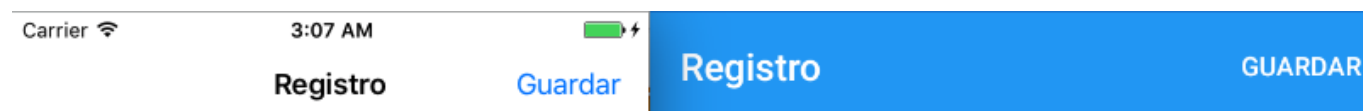
Llenando nuestra TableView debemos llegar al siguiente resultado:

```
<TableView Intent="Form" HasUnevenRows="true" >
  <TableSection Title="Información del usuario">
    <ViewCell >
      <StackLayout Padding="5,5,5,5"
Orientation="Horizontal">
        <Entry Placeholder="Nombre"
HorizontalOptions="FillAndExpand" />
        <Image HeightRequest="30" WidthRequest="30" />
      </StackLayout>
    </ViewCell>
    <ViewCell>
      <StackLayout Padding="5,5,5,5"
Orientation="Horizontal">
        <Entry Placeholder="Dirección"
HorizontalOptions="FillAndExpand" />
        <Image HeightRequest="30" WidthRequest="30" />
      </StackLayout>
    </ViewCell>
    <ViewCell>
      <StackLayout Padding="5,5,5,5"
Orientation="Horizontal">
        <Entry Keyboard="Telephone" Placeholder="Teléfono
de contacto" HorizontalOptions="FillAndExpand" />
        <Image HeightRequest="30" WidthRequest="30" />
      </StackLayout>
    </ViewCell>
  </TableSection>
  <TableSection Title="Información de la cuenta">
    <ViewCell>
      <StackLayout Padding="5,5,5,5"
Orientation="Horizontal">
        <Entry Keyboard="Email" Placeholder="Correo
Electrónico" HorizontalOptions="FillAndExpand" />
        <Image HeightRequest="30" WidthRequest="30" />
      </StackLayout>
    </ViewCell>
    <ViewCell>
      <StackLayout Padding="5,5,5,5"
Orientation="Horizontal">
        <Entry Placeholder="Contraseña" IsPassword="true"
HorizontalOptions="FillAndExpand" />
        <Image HeightRequest="30" WidthRequest="30" />
      </StackLayout>
    </ViewCell>
  </TableSection>
</TableView>
```

```
</TableSection>
</TableView>
```

Aplicando los conceptos vistos anteriormente estamos haciendo uso de la propiedad KeyBoard e IsPassword en algunos Entry.

4. Adicional a la Tabla agregaremos un botón para que el usuario pueda generar su cuenta, en este caso el botón lo pondremos en la parte superior de la página, lo que es el ActionBar en Android y el NavigationBar en iOS

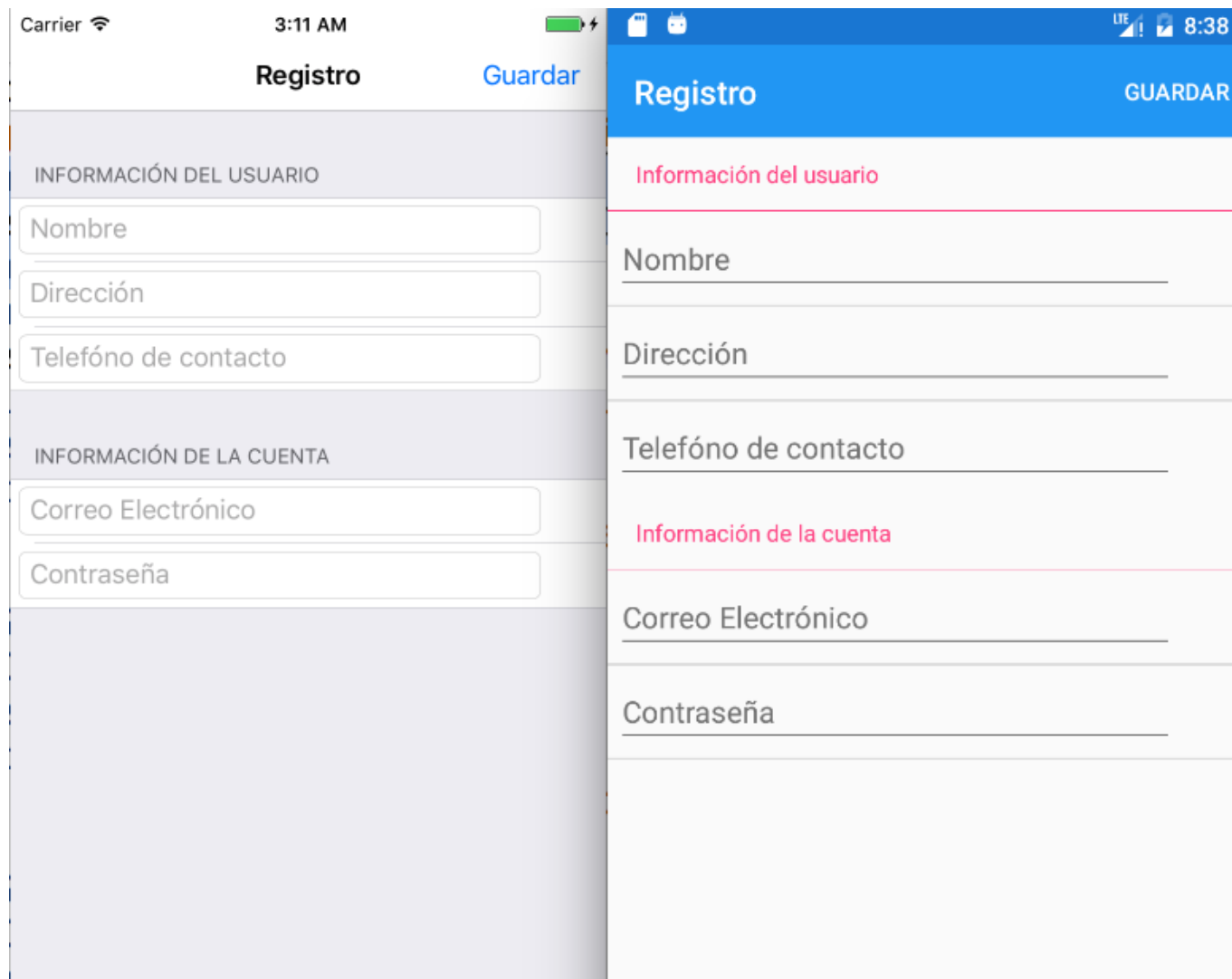


Para lograr esto usaremos la propiedad ToolbarItems de la ContentPage a la cual le agregaremos el ToolBarItem con el texto Guardar

```
<ContentPage.ToolbarItems>
    <ToolBarItem Text="Guardar" Priority="1" Order="Primary" />
</ContentPage.ToolbarItems>
```

Nota: Este cambio no se ve en el Xaml Previewer

Al finalizar los pasos anteriores llegamos al siguiente resultado



Creando una página XAML para levantar pedidos

1. Crearemos nuestra tercera página Xaml en blanco a la que llamaremos "OrderPage", la funcionalidad de esta página es la de crear una orden de compra por lo cual debe cumplir con las siguientes funciones:

- Búsqueda de productos
- Mostrar los resultados de la búsqueda
- Permitir seleccionar la cantidad de productos a comprar
- Agregar los productos a la orden
- Mostrar la lista de productos seleccionados para compra
- Permitir eliminar productos de la compra
- Finalizar la compra

El resultado final debe ser parecido a esta pantalla:

2. Para este ejemplo usaremos un Grid para acomodar los elementos, el Grid estará compuesto por 8 fila y 5 columnas con diferentes medidas, en código nuestro Grid queda de esta manera

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="40" />
    <RowDefinition Height="15" />
    <RowDefinition Height="1*" />
    <RowDefinition Height="40" />
    <RowDefinition Height="15" />
    <RowDefinition Height="2*" />
    <RowDefinition Height="40" />
    <RowDefinition Height="10" />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="10" />
```



```

        <ColumnDefinition />
        <ColumnDefinition Width="90" />
        <ColumnDefinition Width="130" />
        <ColumnDefinition Width="10" />
    </Grid.ColumnDefinitions>
</Grid>

```

Repasando los tres casos para definir medidas, nuestro Grid toma sus medidas de la siguiente manera:

- Si Width o Height no están definidos la medida por defecto es 1*
- El * indica una proporción del espacio disponible, esa proporción mide el equivalente a dividir el espacio disponible entre la suma de medidas con *.
- Si Width o Height tienen un valor sin asterisco el valor se convierte en la medida que maneje la plataforma dpi en el caso de Android y puntos en el caso de iOS.

A partir de este punto todo el código que se muestre para esta página se debe poner dentro del Grid.

3. Para cumplir con la función de búsqueda usaremos un control que nos ofrece Xamarin.Forms llamado `SearchBar`, este control ya tiene la apariencia de un control que sirve para buscar. La forma de definirlo en nuestro XAML es la siguiente.

```
<SearchBar Grid.ColumnSpan="5" Placeholder="Buscar producto" />
```

El `Placeholder` funciona igual que en los campos de tipo `Entry` y con la propiedad adjunta `ColumnSpan` le indicamos a nuestro control que debe abarcar 5 columnas, como no definimos un valor para las propiedades adjuntas `Column` y `Row`, el control se ubica por defecto en la fila 0 y la columna 0. Recuerda que los índices de las columnas comienzan en 0.

4. Para mostrar los resultados de la búsqueda usaremos un control de tipos `ListView`, y encima le pondremos un `Label` con un encabezado

```

<Label Grid.Column="1" Grid.Row="1" Text="Productos" />
<ListView Grid.Column="1" Grid.Row="2" Grid.ColumnSpan="3" >
    <ListView.ItemTemplate>
        <DataTemplate>
            <ImageCell />
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>

```

En este caso si definimos las propiedades adjuntas `Row` y `Column` para ambos controles, los posicionamos en la columna 1 debido a que la columna 0 solo mide 10 y la estamos usando como margen, lo mismo sucede con la columna 4.

En el caso del `label` no es necesario que ocupe más de una columna, sin embargo, en el caso de la `ListView` si le indicamos que abarque 3 columnas para que aproveche el espacio disponible.

Al igual que el TableView las ListView muestran sus filas mediante elementos de tipo Cell y tiene soporte para los mismos mencionados anteriormente, en este caso haremos uso de uno de los que ofrece Xamarin.Forms que es el de tipo ImageCell, el cual nos permite mostrar una fila con una pequeña imagen y dos textos. El Cell debe definirse dentro de un elemento DataTemplate que se asigna a la propiedad ItemTemplate de la ListView.

5. Después de la búsqueda el usuario puede definir la cantidad de productos a comprar, para eso usaremos los siguientes controles:

- Entry para mostrar el número.
- Stepper para incrementar o disminuir el número mostrado en el Entry
- Bóton para agregar a la orden la cantidad seleccionada

En código nuestro XAML queda de la siguiente manera

```
<Entry Grid.Column="1" Grid.Row="3" WidthRequest="60"
IsEnabled="false" />
<Stepper Grid.Column="2" Grid.Row="3" VerticalOptions="Center"
HorizontalOptions="End" />
<Button Grid.Column="3" Grid.Row="3" TextColor="White"
BackgroundColor="#FFA733" BorderRadius="15" Text="Agregar" />
```

En este caso todos los elementos se encuentran dentro de la fila 3 con la diferencia de que cada uno esta en una columna diferente. A ninguno de los elementos le indicamos su alto o ancho debido a que estamos aprovechando que las filas y columnas ya lo tienen definido.

6. La siguiente funcionalidad de nuestra página es mostrar los productos a comprar y permitir eliminarlos. Este es el XAML que usaremos

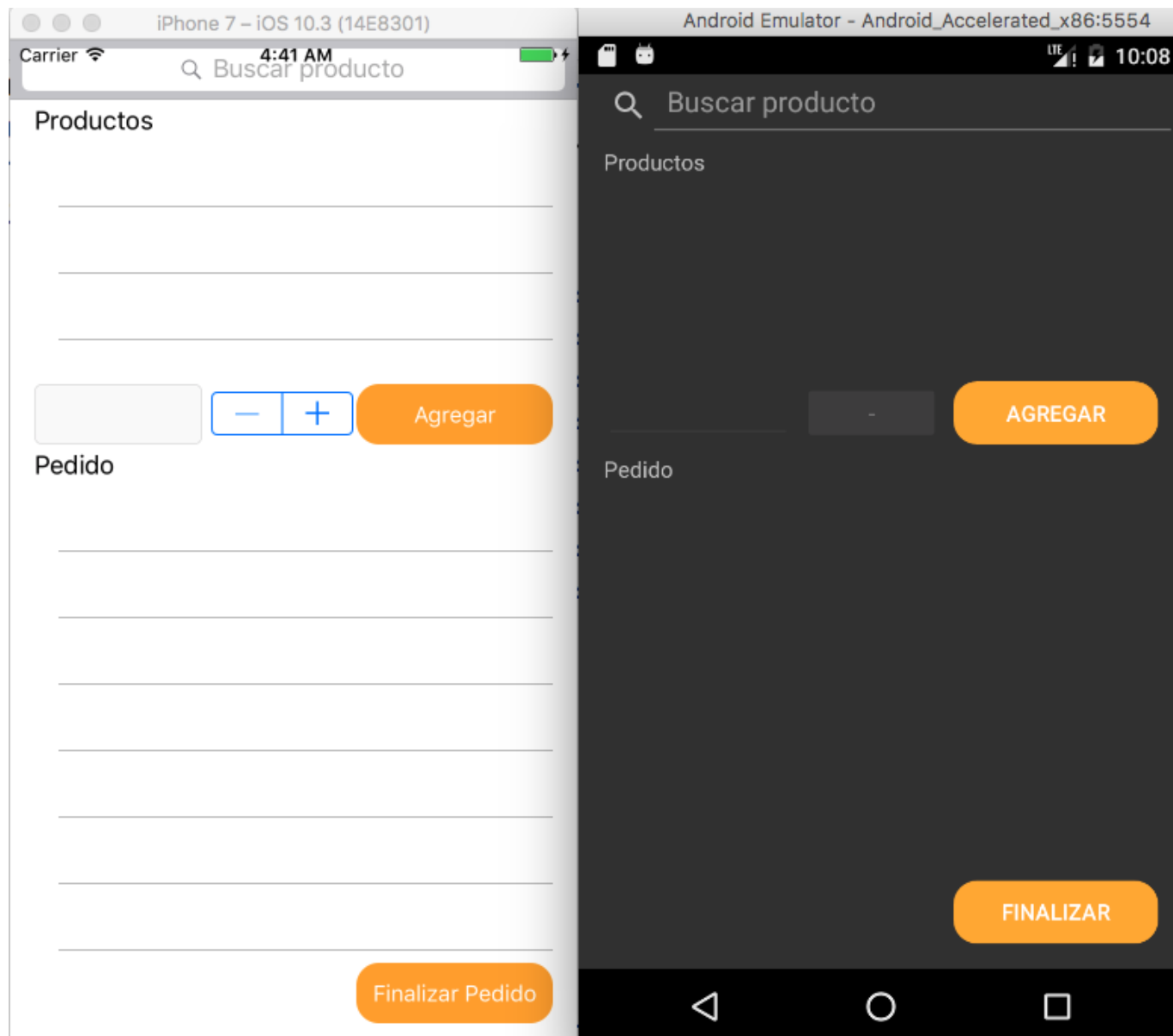
```
<Label Grid.Column="1" Grid.Row="4" Text="Pedido" />
<ListView Grid.Column="1" Grid.ColumnSpan="3" Grid.Row="5">
  <ListView.ItemTemplate>
    <DataTemplate>
      <ImageCell>
        <ViewCell.ContextActions>
          <MenuItem Text="Eliminar"/>
        </ViewCell.ContextActions>
      </ImageCell>
    </DataTemplate>
  </ListView.ItemTemplate>
</ListView>
```

El primer Label solo es descriptivo, en el caso de la ListView, hacemos uso del DataTemplate al igual que en la funcionalidad de resultados de búsqueda sin embargo aquí hay una diferencia y es que asignamos la propiedad ContextActions, la cual permite que le pongamos una acción a los elementos de la lista la cual se muestra dependiendo de la plataforma, el en caso de Android se muestra al dejar presionado unos segundo el elemento y en el caso de iOS se muestra al desplazar hacia la izquierda el elemento.

7. Finalmente agregamos el botón para finalizar el pedido

```
<Button Grid.Column="3" Grid.Row="6" TextColor="White"
BackgroundColor="#FFA733" BorderRadius="15" Text="Finalizar Pedido" />
```

Con esto debemos tener ya nuestra pantalla de este modo

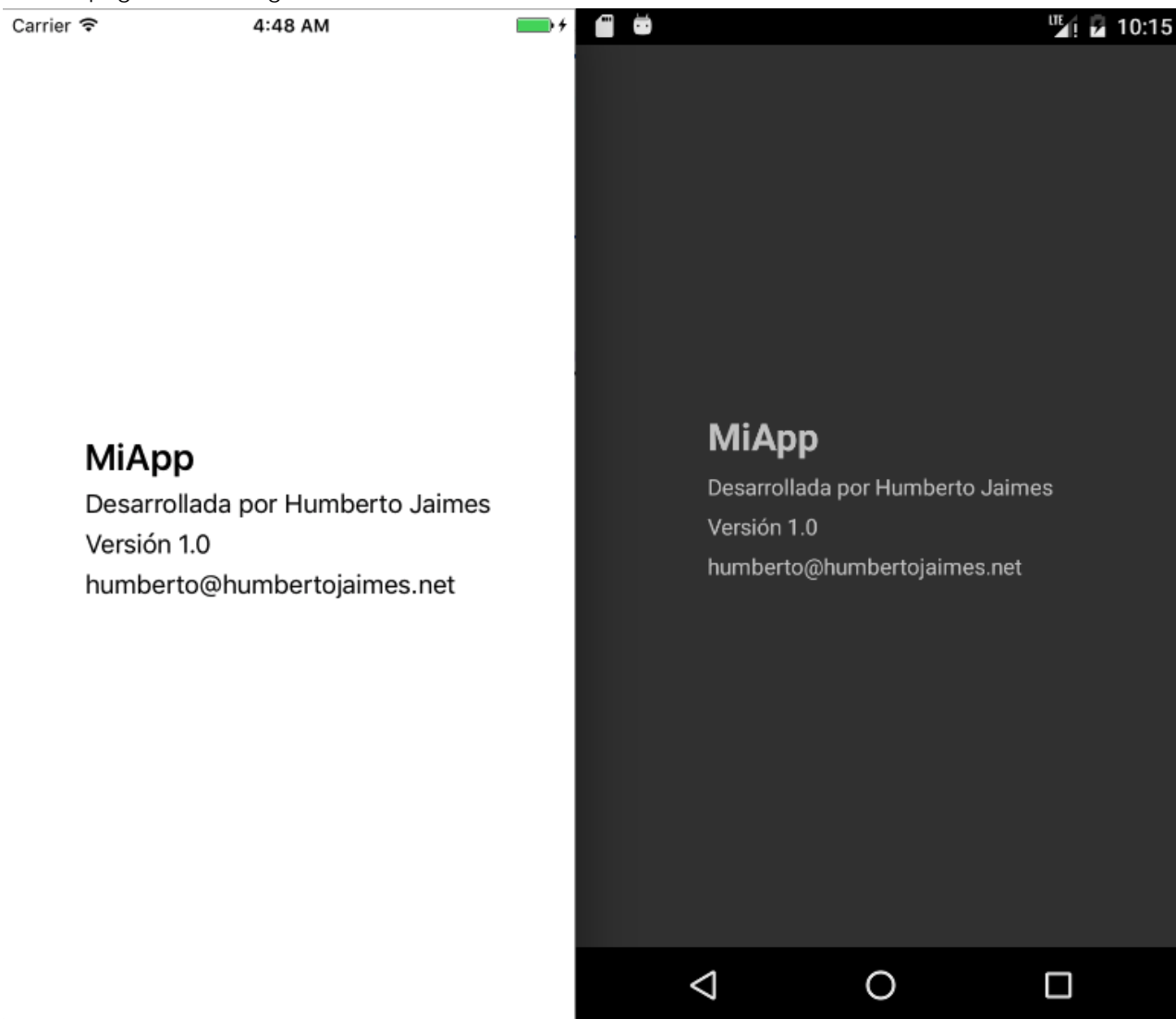


Creando las páginas XAML Restantes

1. Para que puedas practicar trata de crear las siguientes páginas usando los conceptos vistos anteriormente. Toma en cuenta las siguientes recomendaciones.

- No uses muchos contenedores uno dentro de otro (Ejemplo: Crear un Grid usando StackLayouts).
- Trata de usar medidas proporcionales en lugar de medidas fijas.
- Apóyate de las propiedades Margin y Padding y de las LayoutOptions para acomodar los elementos.
- Mfractor en su versión gratuita te puede mostrar algunos errores dentro de XAML
- Activa XAMLC para no tener errores de XAML en tiempo de ejecución.

2. Crea la página AboutPage



Ayuda: Solo contiene Labels

3. Página NextClientPage

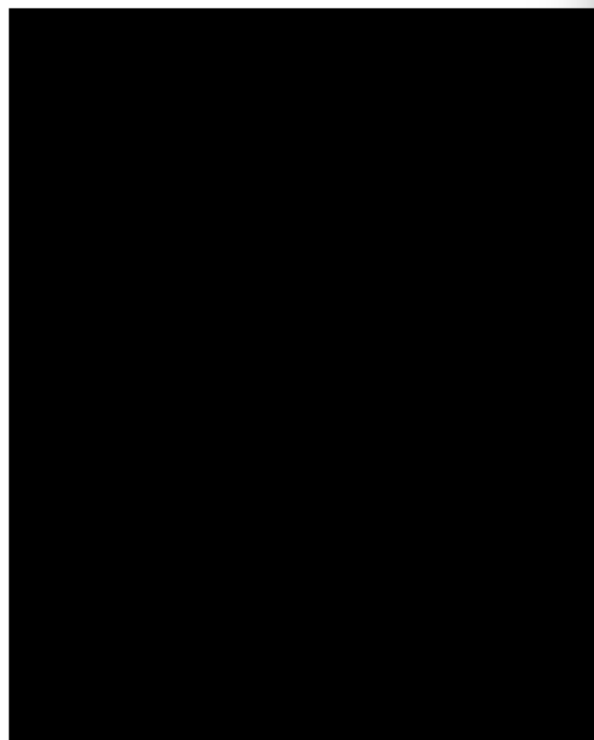
Carrier 4:49 AM

Nombre

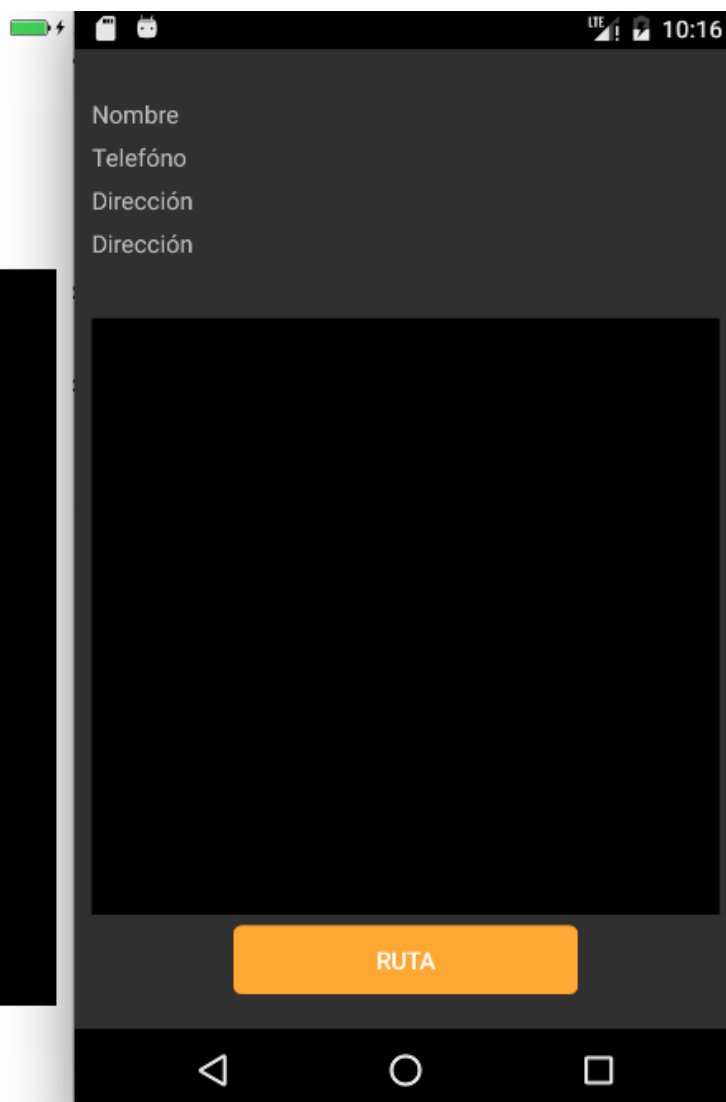
Teléfono

Dirección

Dirección

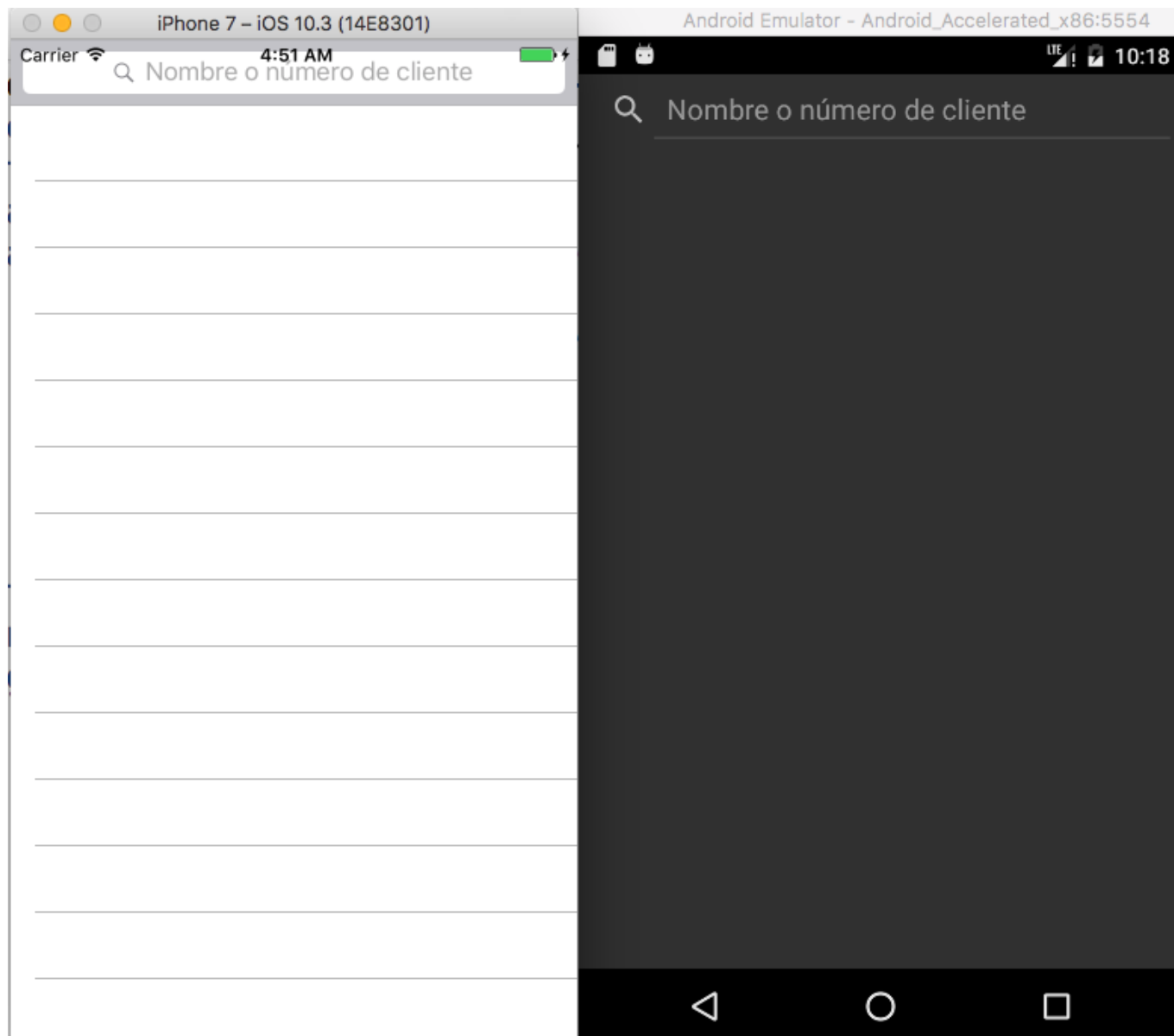


Ruta



Ayuda: Esta conformada por Labels y un BoxView con Background en Negro (En un futuro cambiara por un mapa)

4. Página ClientSearchPage



Ayuda: Solo contiene un SearchBar y un ListView que mostrara el nombre del cliente

Opcional

Trata de corregir los pequeños detalles que encuentre en las páginas y compártelos en el grupo (Ejemplo: La barra superior de iOS tapa parte del SearchBar)