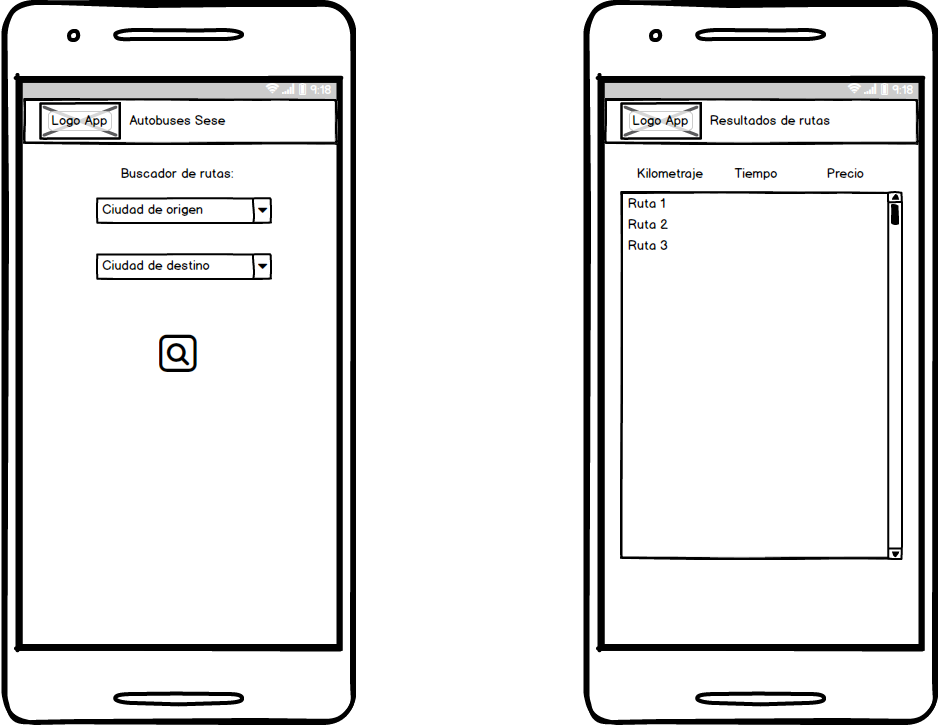
Documentación Xamarin

### Realizado por:

* Adrian Dobladez
* César Wilfrido Morales Oña

# Mockup:

Se crea un Mockup, mediante Balsamic, de la aplicación para plantear el proyecto:



# Cabecera de las páginas y navegación jerárquica:

Para la creación de varias pantallas primero se tiene que entender la navegación jerárquica y como funciona, en la [página de microsoft sobre navegación jerárquica y cabecera](https://docs.microsoft.com/es-es/xamarin/xamarin-forms/app-fundamentals/navigation/hierarchical#displaying-views-in-the-navigation-bar) se puede encontrar la información de como realizar todo el procedimiento.

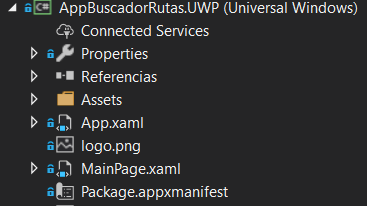
Para la creación de las cabeceras en las páginas se tiene que utilizar la propiedad *NavigationPage* y modificar el *ContentPage* con la propiedad *Content*.TitleView que se integra de la siguiente manera, negrita, en las páginas *MainPage* y *ResultPage*:

|  |
| --- |
| <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:AppBuscadorRutas"  x:Class="AppBuscadorRutas.MainPage">  **<NavigationPage.TitleView>**  …….  **</NavigationPage.TitleView>**  **<ContentPage.Content>**  ….....  **</ContentPage.Content>**  ….  </ContentPage> |

Dentro de *NavigationPage.TitleView* se añade una fotografía del logo de la aplicación

mediante la clase *Image* en la cual se dice hacer una integración de fotos mediante imágenes locales por lo que se debe usar el mismo nombre de archivo en todas las plataformas que en este caso es logo.png y debe ser un nombre de recurso de Android, para ello es preferible leer la [pagina de Microsoft sobre cómo añadir imágenes](https://docs.microsoft.com/es-es/xamarin/xamarin-forms/user-interface/images?tabs=windows).

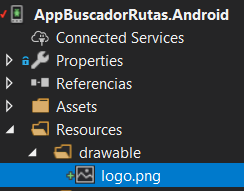
En el caso de UWP la imagen debe estar en el directorio raíz de la aplicación:



y en las propiedades de la imagen, avanzadas, acción de compilación se debe seleccionar la opción *Contenido*:



Para Android la imagen debe estar introducida en Resources, drawable:



En las propiedades de la imagen, avanzadas, acción de compilación se debe seleccionar la opción *AndroidResource*:



Se añade también un texto en la cacera mediante el método **label.**

Tanto **label** cómo **Image** estan contenidas dentro de un stack layout.

Para las transiciones entre las páginas se utiliza la clase NavigationPage la cual nos permite movernos entre las diferentes páginas.

Primero se debe crear la página raíz, para ello en la Clase App se debe añadir la línea de código en negrita:

|  |
| --- |
| public App()  {  InitializeComponent();  **MainPage = new NavigationPage(new MainPage());**  } |

En el XAML de MainPage se añade un botón y se genera su clase correspondiente la cual tenemos que pasar a una clase asíncrona e implementar la línea de código en negrita que mediante el método *PushAsync* introduce en la página *ResultPage* en la pila de páginas.

|  |
| --- |
| async void Button\_Clicked(object sender, EventArgs e)  {  **await Navigation.PushAsync(new ResultPage());**  } |

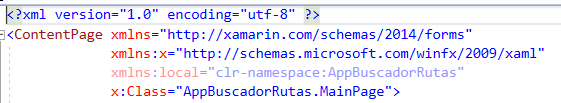
# Cliente API:

Para la comunicación con la API se crea una interface genérica, denominada IServicioAPI, que permitirá que el programa sea extensible y aplicable a cualquier modelo de datos. Dicha interface tiene declarado el método Obtener que representa a un GET, para obtener los datos de ciudades y las rutas:

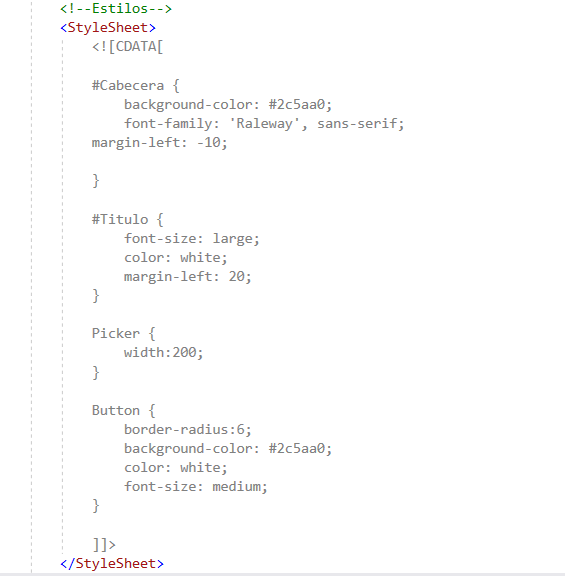
Se crea una clase, ServicioAPI, que implementa la interface IServicioAPI. Mediante el constructor ServicioAPI( string URL) se encapsula la dirección URL de la que se recuperará los datos necesarios para el programa.El método Obtener() es de tipo T que representa a la clase del modelo de datos que queremos devolver y este método realiza la recuperación de los datos.

Archivos a configurar:

* MainPage.xaml



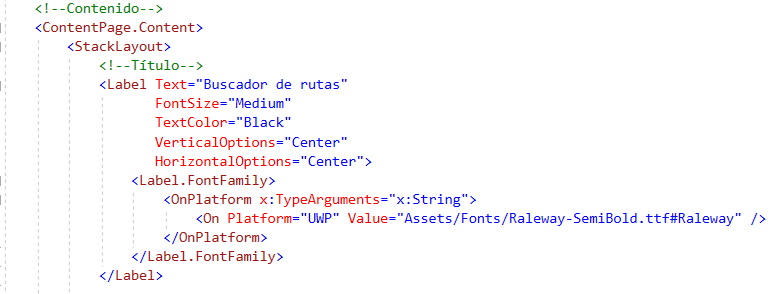
Aqui unicamente representaremos los datos con los que trabajaremos y los nombramos.



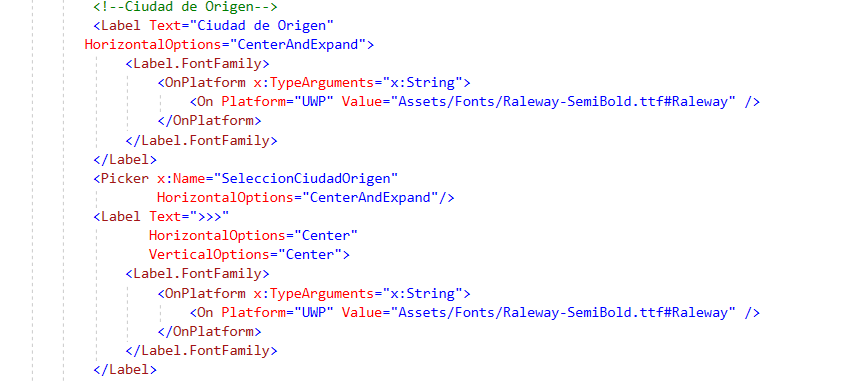
Continuamos con los estilos a utilizar dentro de la propia mainpage

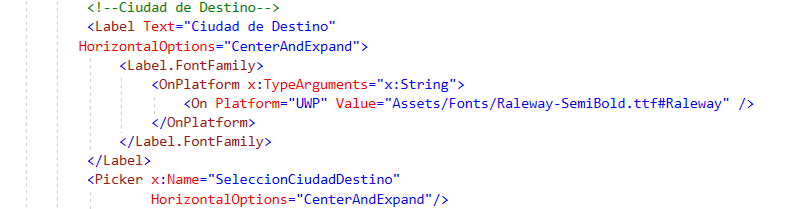


Cabecera dentro de la app, titleview es el tipo que hemos seleccionado y el resto muestra lo que hay en el interior.

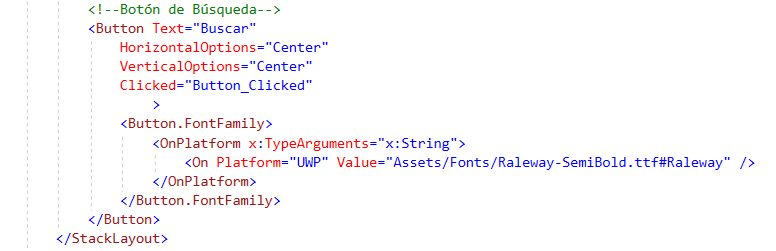


Dentro de la web comenzamos con un pequeño título para representar una descripción de lo que se hará a continuación.





Lista desplegable desde la API para seleccionar la ciudad de origen y lo mismo con la ciudad de destino



Por último un botón de búsqueda que nos enviara a la otra pagina y enviara los parámetros de vuelta a la base de datos.