On Sale - App Móvil Parte I

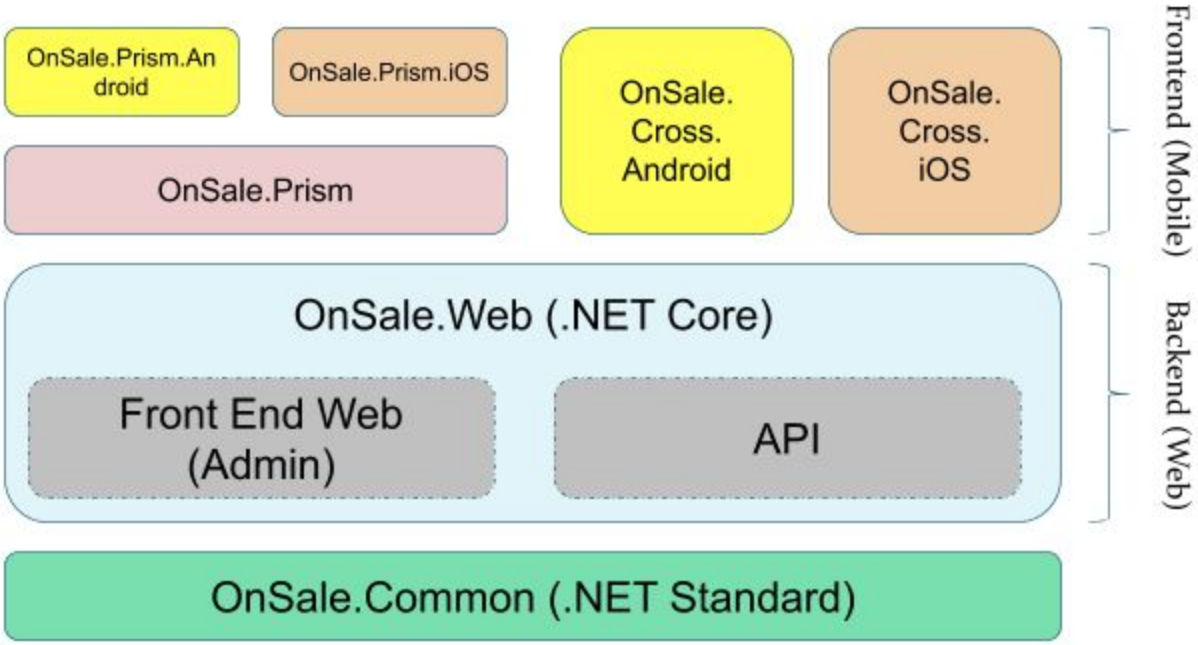
Juan Carlos Zuluaga Cardona
Medellín
2020

Tabla de contenido

Matriz de funcionalidad

Funcionalidad	**Web**		**App**
	Admin	**User**	**User**
Login	X	X	X
Registrarse como usuario		X	X
Modificar el perfil	X	X	X
Recordar contraseña	X	X	X
Administrar administradores	X		
Administrar países, departamentos, ciudades	X		
Administrar productos	X		
Ver y buscar productos		X	X
Agregar productos al carrito de compras		X	X
Confirmar orden		X	X
Administrar los pedidos	X		
Ver estado de mis pedidos		X	X

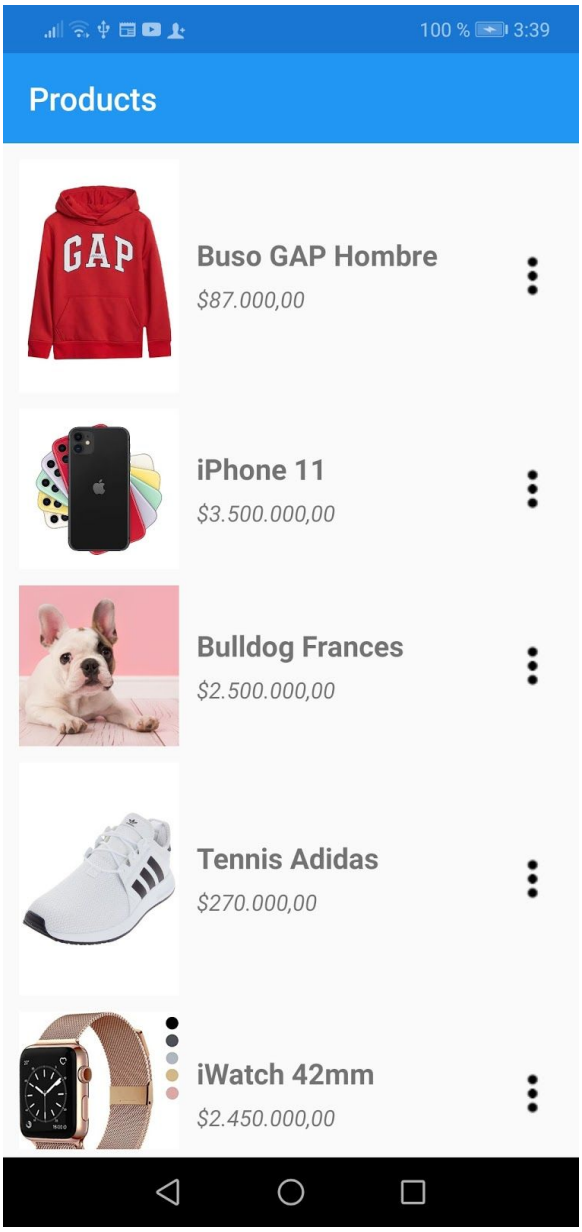
Arquitectura



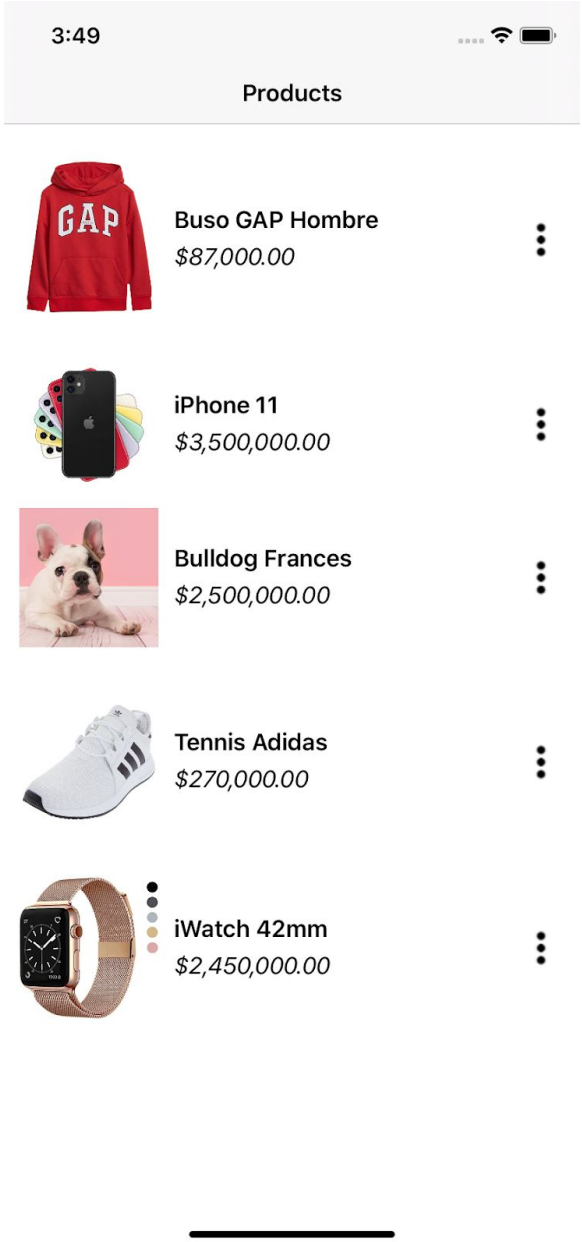
Creación de la aplicación Xamarin Forms + Prism

Vamos a crear esta primer pantalla:

Android

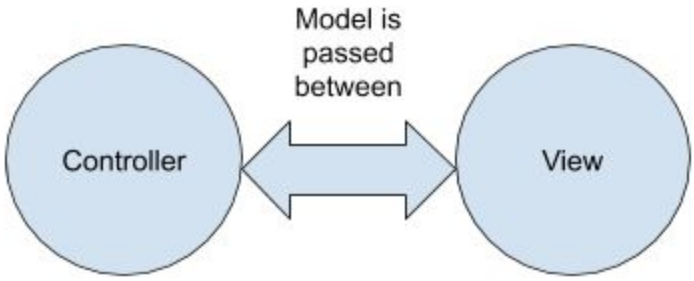


iOS

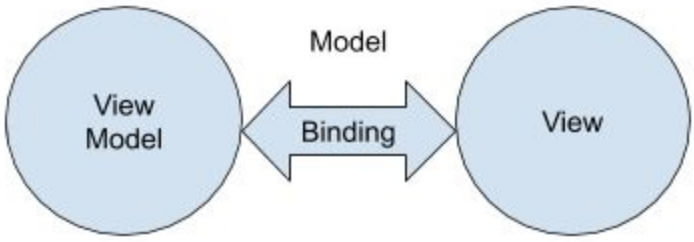


Pero primero, analicemos un poco la forma como vamos a trabajar:

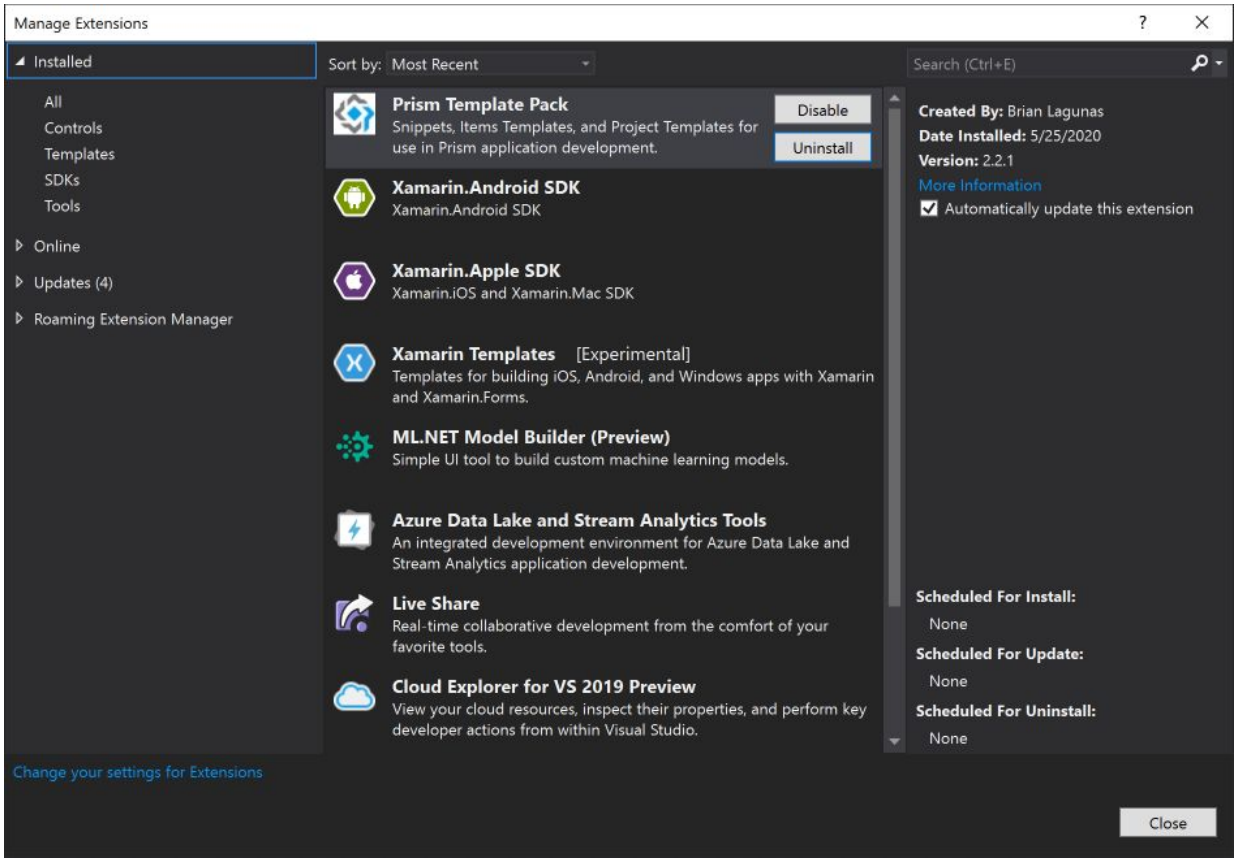
MVC - Backend



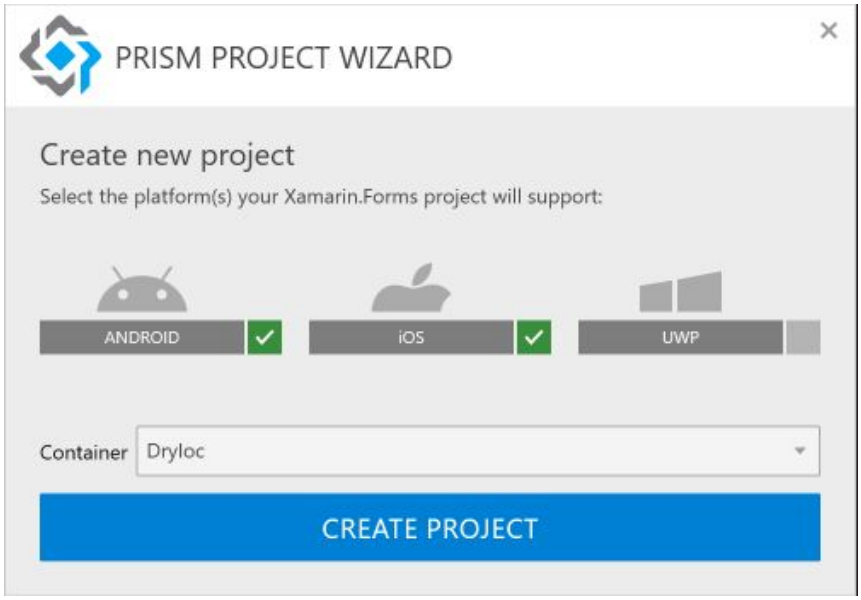
MVVM - Mobile



1. Lo primero que debes hacer es verificar que tu Visual Studio tenga instaladas las plantillas de Prism:



2. Creamos un nuevo proyecto **Prism Blank App (Xamarin.Forms)** y lo llamamos **OnSale.Prism**:



3. Verificamos que está corriendo el proyecto de “**Welcome to Xamarin Forms and Prism**”

4. Actualizamos los paquetes Nuget a las últimas versiones en los proyectos móviles.

5. Agregar el nuget **Xamarin.FFImageLoading.Forms** a los proyectos móviles.

6. Inicializar el **Xamarin.FFImageLoading.Forms** en Android:

```
global::Xamarin.Forms.Forms.Init(this, bundle);
FFImageLoading.Forms.Platform.CachedImageRenderer.Init(true);
LoadApplication(new App(new AndroidInitializer()));
```

7. Inicializar **Xamarin.FFImageLoading.Forms** en iO:

```
global::Xamarin.Forms.Forms.Init();
FFImageLoading.Forms.Platform.CachedImageRenderer.Init();
LoadApplication(new App(new iOSInitializer()));
```

8. Adicione el ícono **ic_more_vert.png** (https://romannurik.github.io/AndroidAssetStudio/index.html). Para el proyecto Android en **Resources/drawable** y para iOS en **Resources**.

9. En el proyecto **Common** cree la carpeta **Responses** y dentro de esta, la clase **Response**:

```
public class Response
{
    public bool IsSuccess { get; set; }

    public string Message { get; set; }

    public object Result { get; set; }
}
```

10. En el proyecto **Common** cree la carpeta **Services** y dentro de esta creas la interfaz: **IApiService**:

```
public interface IApiService
{
    Task<Response> GetListAsync<T>(string urlBase, string servicePrefix, string controller);
}
```

11. Continuamos con la implementación de la interfaz en **ApiService**:

```
public class ApiService : IApiService
{
    public async Task<Response> GetListAsync<T>(
        string urlBase,
        string servicePrefix,
```

```
        string controller)
    {
        try
        {
            HttpClient client = new HttpClient
            {
                BaseAddress = new Uri(urlBase),
            };

            string url = $"{servicePrefix}{controller}";
            HttpResponseMessage response = await client.GetAsync(url);
            string result = await response.Content.ReadAsStringAsync();

            if (!response.IsSuccessStatusCode)
            {
                return new Response
                {
                    IsSuccess = false,
                    Message = result,
                };
            }

            List<T> list = JsonConvert.DeserializeObject<List<T>>(result);
            return new Response
            {
                IsSuccess = true,
                Result = list
            };
        }
        catch (Exception ex)
        {
            return new Response
            {
                IsSuccess = false,
                Message = ex.Message
            };
        }
    }
}
```

12. Adicionamos la inyección del servicio creado en **App.xaml.cs**:

```
protected override void RegisterTypes(IContainerRegistry containerRegistry)
{
    containerRegistry.RegisterSingleton<IAppInfo, AppInfoImplementation>();
    containerRegistry.Register<IApiService, ApiService>();
    containerRegistry.RegisterForNavigation<NavigationPage>();
    containerRegistry.RegisterForNavigation<MainPage, MainPageViewModel>();
```

```
}
```

13. Corrija la ruta que habíamos dejado pendiente en las imágenes:

```
[Display(Name = "Image")]
public string ImageFullPath => ImageId == Guid.Empty
    ? $"https://onsalezulu.azurewebsites.net/images/noimage.png"
    : $"https://onsale.blob.core.windows.net/users/{ImageId}";
```

14. Adicionamos un diccionario de recursos para colocar la URL de nuestro servicio en el **App.xaml**:

```
<?xml version="1.0" encoding="utf-8" ?>
<prism:PrismApplication xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            xmlns:prism="http://prismlibrary.com"
            x:Class="OnSale.Prism.App">
  <Application.Resources>

        <ResourceDictionary>

            <!-- Parameters -->
            <x:String x:Key="UrlAPI">https://onsalezulu.azurewebsites.net</x:String>

        </ResourceDictionary>

    </Application.Resources>
</prism:PrismApplication>
```

15. Adicionar la **ProductsPage**:

```
<?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:prism="http://prismlibrary.com"
            prism:ViewModelLocator.AutowireViewModel="True"

xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.For
ms"
            x:Class="OnSale.Prism.Views.ProductsPage"
            Title="{Binding Title}">

    <StackLayout Padding="5">
        <CollectionView ItemsSource="{Binding Products}">
            <CollectionView.ItemsLayout>
                <GridItemsLayout Orientation="Vertical"/>
            </CollectionView.ItemsLayout>
            <CollectionView.ItemTemplate>
```

```
            <DataTemplate>
               <Grid>
                  <Grid.ColumnDefinitions>
                     <ColumnDefinition Width="Auto" />
                     <ColumnDefinition Width="*" />
                     <ColumnDefinition Width="Auto" />
                  </Grid.ColumnDefinitions>
                  <ffimageloading:CachedImage Grid.Column="0"
                                              Aspect="AspectFill"
                                              Source="{Binding ImageFullPath}"
                                              CacheDuration= "50"
                                              Margin="5"
                                              RetryCount= "3"
                                              RetryDelay= "600"
                                              WidthRequest="100"/>
                  <StackLayout Grid.Column="1"
                               VerticalOptions="Center">
                     <Label Text="{Binding Name}"
                            FontAttributes="Bold"
                            FontSize="Medium"
                            LineBreakMode="TailTruncation" />
                     <Label Text="{Binding Price, StringFormat='{0:C2}'}"
                            LineBreakMode="TailTruncation"
                            FontAttributes="Italic"
                            VerticalOptions="End" />
                  </StackLayout>
                  <Image Grid.Column="2"
                         Source="ic_more_vert"/>
               </Grid>
            </DataTemplate>
         </CollectionView.ItemTemplate>
      </CollectionView>
   </StackLayout>

</ContentPage>
```

16. Modificamos la **ProductsPageViewModel**:

```
public class ProductsPageViewModel : ViewModelBase
{
   private readonly INavigationService _navigationService;
   private readonly IApiService _apiService;
   private ObservableCollection<Product> _products;

   public ProductsPageViewModel(INavigationService navigationService, IApiService
apiService) : base(navigationService)
   {
      _navigationService = navigationService;
```

```
        _apiService = apiService;
        Title = "Products";
        LoadProductsAsync();
    }

    public ObservableCollection<Product> Products
    {
        get => _products;
        set => SetProperty(ref _products, value);
    }

    private async void LoadProductsAsync()
    {
        if (Connectivity.NetworkAccess != NetworkAccess.Internet)
        {
            await App.Current.MainPage.DisplayAlert("Error", "Check the internet connection.",
"Accept");
            return;
        }

        string url = App.Current.Resources["UrlAPI"].ToString();
        Response response = await _apiService.GetListAsync<Product>(
            url,
            "/api",
            "/Products");

        if (!response.IsSuccess)
        {
            await App.Current.MainPage.DisplayAlert(
                "Error",
                response.Message,
                "Accept");
            return;
        }

        List<Product> myProducts = (List<Product>)response.Result;
        Products = new ObservableCollection<Product>(myProducts);
    }
}
```

17. Modificamos la página de inicio para que inice por el **ProductsPage**, para ello modificamos el **App.xaml.cs**:

```
protected override async void OnInitialized()
{
    InitializeComponent();

    await NavigationService.NavigateAsync($"NavigationPage/{nameof(ProductsPage)}");
```

}

18. Modificamos el **AndroidedManifest** en Android para solicitarle permiso al SO de verificar el estado de la conexión a internet:

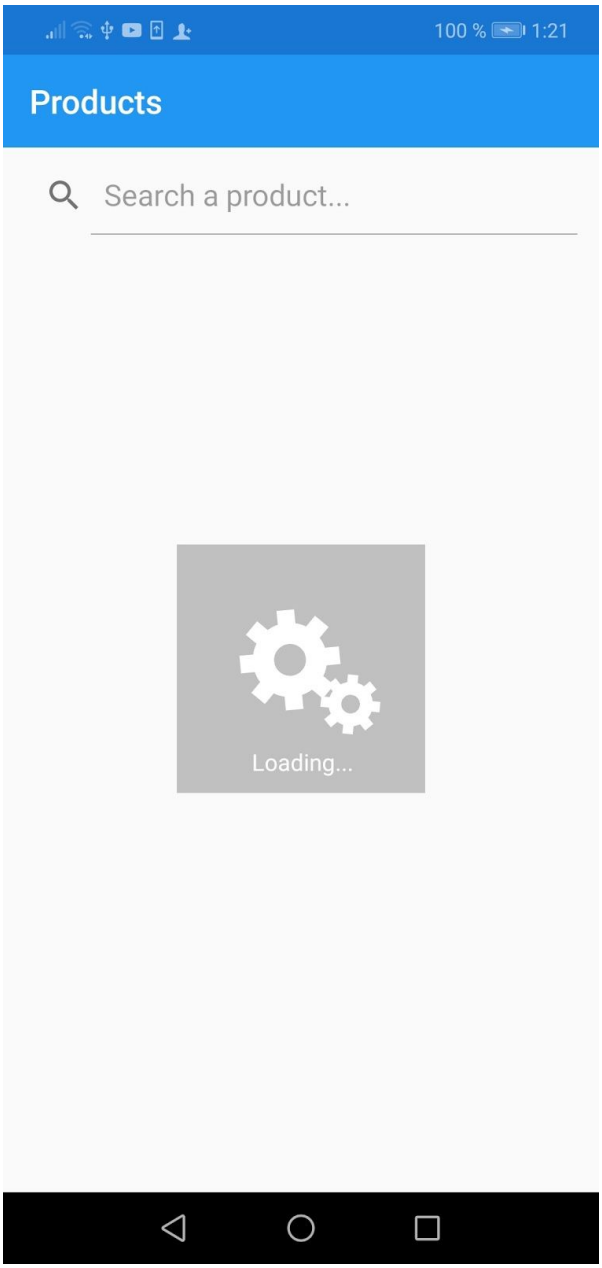
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="1" android:versionName="1.0" package="com.companyname.appname"
android:installLocation="auto">
	<uses-sdk android:minSdkVersion="21" android:targetSdkVersion="29" />
	<uses-permission android:name="android.permission.INTERNET" />
	<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
	<application android:label="OnSale.Prism.Android"
android:icon="@mipmap/ic_launcher"></application>
</manifest>
```

19. Probamos.

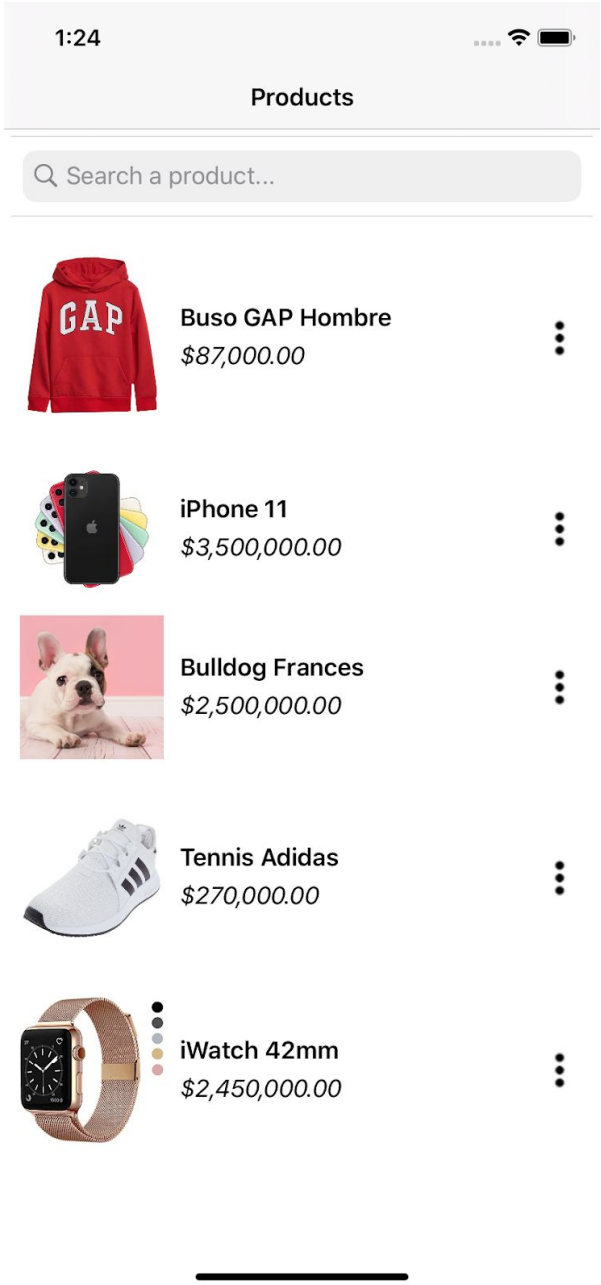
Mejorar la pantalla con un indicador de actividad y una búsqueda

Vamos a implementar estas mejoras:

Android



iOS



20. Primero debes obtener una licencia de **Syncfusion** en la página: https://www.syncfusion.com/account/downloads.

21. Adicione el NuGet **Syncfusion.Xamarin.SfBusyIndicator** a todos los proyectos móviles.

22. Adicione su licencia en **App.xaml.cs**:

```
protected override async void OnInitialized()
{
SyncfusionLicenseProvider.RegisterLicense("MjExMTY0QDMxMzcyZTM0MmUzMEc1K2xKbkhrV2RmMHByRXF6YUJDQlQ3RkxLZ3hxOVlyMHY0T1RiSUFEZUk9");
   InitializeComponent();
   await NavigationService.NavigateAsync($"NavigationPage/{nameof(ProductsPage)}");
}
```

23. Modificar el **MainActivity**:

```
global::Xamarin.Forms.Forms.Init(this, bundle);
new SfBusyIndicatorRenderer();
FFImageLoading.Forms.Platform.CachedImageRenderer.Init(true);
LoadApplication(new App(new AndroidInitializer()));
```

24. Modificar el **AppDelegate**:

```
global::Xamarin.Forms.Forms.Init();
FFImageLoading.Forms.Platform.CachedImageRenderer.Init();
new SfBusyIndicatorRenderer();
LoadApplication(new App(new iOSInitializer()));
```

25. Modificar la **ProductsPage**:

```
<?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:prism="http://prismlibrary.com"
             prism:ViewModelLocator.AutowireViewModel="True"
xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.Forms"
xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamarin.Forms.Core"
             ios:Page.UseSafeArea="true"
```

```
xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion
.SfBusyIndicator.XForms"
            x:Class="OnSale.Prism.Views.ProductsPage"
            Title="{Binding Title}">

    <AbsoluteLayout>
        <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"
                     AbsoluteLayout.LayoutFlags="All"
                     Padding="5">
            <CollectionView ItemsSource="{Binding Products}">
…
            </CollectionView>
        </StackLayout>
        <busyindicator:SfBusyIndicator AnimationType="Gear"
                                       AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"
                                       AbsoluteLayout.LayoutFlags="All"
                                       BackgroundColor="Silver"
                                       HorizontalOptions="Center"
                                       TextColor="White"
                                       IsBusy="{Binding IsRunning}"
                                       Title="Loading..."
                                       VerticalOptions="Center"
                                       ViewBoxWidth="80"
                                       ViewBoxHeight="80" />
    </AbsoluteLayout>

</ContentPage>
```

26. Modificar la **ProductsPageViewModel**:

```
private readonly INavigationService _navigationService;
private readonly IApiService _apiService;
private ObservableCollection<Product> _products;
private bool _isRunning;

public ProductsPageViewModel(INavigationService navigationService, IApiService apiService) :
base(navigationService)
{
    _navigationService = navigationService;
    _apiService = apiService;
    Title = "Products";
    LoadProductsAsync();
}
```

```
public bool IsRunning
{
    get => _isRunning;
    set => SetProperty(ref _isRunning, value);
}

public ObservableCollection<Product> Products
{
    get => _products;
    set => SetProperty(ref _products, value);
}

private async void LoadProductsAsync()
{
    if (Connectivity.NetworkAccess != NetworkAccess.Internet)
    {
        await App.Current.MainPage.DisplayAlert("Error", "Check the internet connection.",
"Accept");
        return;
    }

    IsRunning = true;
    string url = App.Current.Resources["UrlAPI"].ToString();
    Response response = await _apiService.GetListAsync<Product>(
        url,
        "/api",
        "/Products");
    IsRunning = false;

    if (!response.IsSuccess)
…
```

27. Probamos.

28. Ahora vamos con la barra de búsqueda. Modificar la **ProductsPage**:

```
...
<StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"
             AbsoluteLayout.LayoutFlags="All"
             Padding="5">
    <SearchBar Placeholder="Search a product..."
               SearchCommand="{Binding SearchCommand}"
```

```
            Text="{Binding Search}"/>
    <CollectionView ItemsSource="{Binding Products}">
...
```

29. Modificar la **ProductsPageViewModel**:

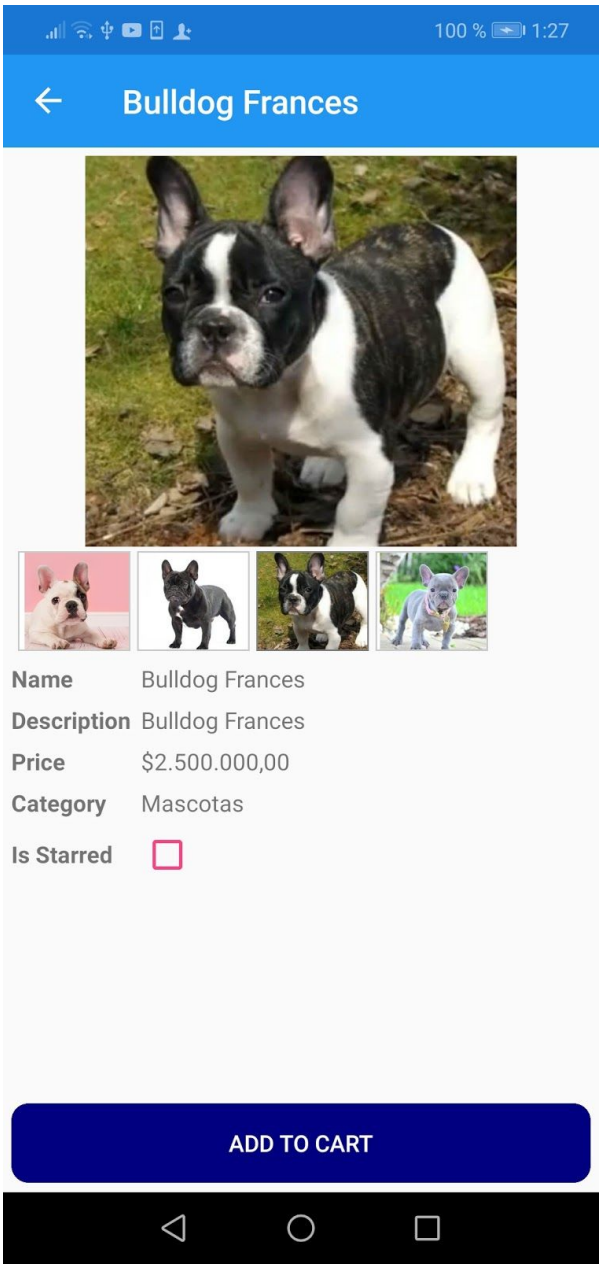
```
...
private readonly INavigationService _navigationService;
private readonly IApiService _apiService;
private ObservableCollection<Product> _products;
private bool _isRunning;
private string _search;
private List<Product> _myProducts;
private DelegateCommand _searchCommand;
...
public DelegateCommand SearchCommand => _searchCommand ?? (_searchCommand =
new DelegateCommand(ShowProducts));

public string Search
{
    get => _search;
    set
    {
        SetProperty(ref _search, value);
        ShowProducts();
    }
}
...
    _myProducts = (List<Product>)response.Result;
    ShowProducts();
}

private void ShowProducts()
{
    if (string.IsNullOrEmpty(Search))
    {
        Products = new ObservableCollection<Product>(_myProducts);
    }
    else
    {
        Products = new ObservableCollection<Product>(_myProducts
            .Where(p => p.Name.ToLower().Contains(Search.ToLower())));
    }
}
```

30. Probamos.

Navegar a otra página

Vamos a implementar navegar al detalle del producto.

Android



iOS

31. Agregamos la **ProductDetailPage**, inicialmente con este layout:

```
<?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:prism="http://prismlibrary.com"
            prism:ViewModelLocator.AutowireViewModel="True"

xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamari
n.Forms.Core"
            ios:Page.UseSafeArea="true"
            x:Class="OnSale.Prism.Views.ProductDetailPage"
            Title="{Binding Title}">

</ContentPage>
```

32. Modificar la **ProductDetailPageViewModel**, inicialmente con este código:

```
public class ProductDetailPageViewModel : ViewModelBase
{
    public ProductDetailPageViewModel(INavigationService navigationService) :
base(navigationService)
    {
        Title = "Product";
    }
}
```

33. En el proyecto **Prism** agregamos el folder **ItemViewModels** y dentro de este la clase **ProductItemViewModel**:

```
public class ProductItemViewModel : Product
{
    private readonly INavigationService _navigationService;
    private DelegateCommand _selectProductCommand;

    public ProductItemViewModel(INavigationService navigationService)
    {
        _navigationService = navigationService;
    }
```

```
    public DelegateCommand SelectProductCommand => _selectProductCommand ??
(_selectProductCommand = new DelegateCommand(SelectProductAsync));

    private async void SelectProductAsync()
    {
        await _navigationService.NavigateAsync(nameof(ProductDetailPage));
    }
}
```

34. Modificamos la **ProductsPage**:

```
...
<Grid>
    <Grid.GestureRecognizers>
        <TapGestureRecognizer Command="{Binding SelectProductCommand}"/>
    </Grid.GestureRecognizers>
    <Grid.ColumnDefinitions>
...
```

35. Modificar la **ProductsPageViewModel**:

```
...
private ObservableCollection<ProductItemViewModel> _products;
…
public ObservableCollection<ProductItemViewModel> Products
{
    get => _products;
    set => SetProperty(ref _products, value);
}
...
private void ShowProducts()
{
    if (string.IsNullOrEmpty(Search))
    {
        Products = new ObservableCollection<ProductItemViewModel>(_myProducts.Select(p =>
new ProductItemViewModel(_navigationService)
        {
            Category = p.Category,
            Description = p.Description,
            Id = p.Id,
            IsActive = p.IsActive,
            IsStarred = p.IsStarred,
            Name = p.Name,
```

```
            Price = p.Price,
            ProductImages = p.ProductImages
        })
            .ToList());
    }
    else
    {
        Products = new ObservableCollection<ProductItemViewModel>(_myProducts.Select(p =>
new ProductItemViewModel(_navigationService)
        {
            Category = p.Category,
            Description = p.Description,
            Id = p.Id,
            IsActive = p.IsActive,
            IsStarred = p.IsStarred,
            Name = p.Name,
            Price = p.Price,
            ProductImages = p.ProductImages
        })
            .Where(p => p.Name.ToLower().Contains(Search.ToLower()))
            .ToList());
    }
}
```

36. Probamos.

37. Ahora vamos a pasar el producto como parámetro. Modificamos la **ProductItemViewModel**:

```
private async void SelectProductAsync()
{
    NavigationParameters parameters = new NavigationParameters
    {
        { "product", this }
    };

    await _navigationService.NavigateAsync(nameof(ProductDetailPage), parameters);
}
```

38. Modificamos la **ProductDetailPageViewModel**:

```
private Product _product;
```

```
public ProductDetailPageViewModel(INavigationService navigationService) :
base(navigationService)
{
    Title = "Product";
}

public Product Product
{
    get => _product;
    set => SetProperty(ref _product, value);
}

public override void OnNavigatedTo(INavigationParameters parameters)
{
    base.OnNavigatedTo(parameters);

    if (parameters.ContainsKey("product"))
    {
        Product = parameters.GetValue<Product>("product");
        Title = Product.Name;
    }
}
```

39. Probamos.

40. Agruegamos el nuget **Syncfusion.Xamarin.SfRotator** en todos los proyectos prism.

41. Modificamos la **MainActivity**:

```
FFImageLoading.Forms.Platform.CachedImageRenderer.Init(true);
new SfBusyIndicatorRenderer();
new SfRotatorRenderer();
LoadApplication(new App(new AndroidInitializer()));
```

42. Modificamos el **AppDelegate**:

```
FFImageLoading.Forms.Platform.CachedImageRenderer.Init();
new SfBusyIndicatorRenderer();
new SfRotatorRenderer();
LoadApplication(new App(new iOSInitializer()));
return base.FinishedLaunching(app, options);
```

43. Modificamos la **ProductDetailPage**:

```
<?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:prism="http://prismlibrary.com"
            prism:ViewModelLocator.AutowireViewModel="True"

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

xmlns:syncfusion="clr-namespace:Syncfusion.SfRotator.XForms;assembly=Syncfusion.SfRotat
or.XForms"

xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.For
ms"
            x:Class="OnSale.Prism.Views.ProductDetailPage"
            Title="{Binding Title}">

    <StackLayout Padding="5">
        <ScrollView>
            <StackLayout>
                <syncfusion:SfRotator EnableAutoPlay="True"
                                      EnableLooping="True"
                                      HeightRequest="300"
                                      ItemsSource="{Binding Images}"
                                      NavigationDelay="5000"
                                      NavigationDirection="Horizontal"
                                      NavigationStripMode="Thumbnail"
                                      NavigationStripPosition="Bottom">
                    <syncfusion:SfRotator.ItemTemplate>
                        <DataTemplate>
                            <ffimageloading:CachedImage Aspect="AspectFit"
                                                        CacheDuration= "50"
                                                        DownsampleToViewSize = "true"
                                                        ErrorPlaceholder= "ErrorImage"
                                                        HeightRequest="300"
                                                        LoadingPlaceholder= "LoaderImage"
                                                        RetryCount= "3"
                                                        RetryDelay= "600"
                                                        Source="{Binding ImageFullPath}"/>
                        </DataTemplate>
                    </syncfusion:SfRotator.ItemTemplate>
```

```
            </syncfusion:SfRotator>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
                </Grid.RowDefinitions>
                <Label Grid.Row="0"
                       Grid.Column="0"
                       FontAttributes="Bold"
                       Text="Name"/>
                <Label Grid.Row="0"
                       Grid.Column="1"
                       Text="{Binding Product.Name}"/>
                <Label Grid.Row="1"
                       Grid.Column="0"
                       FontAttributes="Bold"
                       Text="Description"/>
                <Label Grid.Row="1"
                       Grid.Column="1"
                       Text="{Binding Product.Description}"/>
                <Label Grid.Row="2"
                       Grid.Column="0"
                       FontAttributes="Bold"
                       Text="Price"/>
                <Label Grid.Row="2"
                       Grid.Column="1"
                       Text="{Binding Product.Price, StringFormat='{0:C2}'}"/>
                <Label Grid.Row="3"
                       Grid.Column="0"
                       FontAttributes="Bold"
                       Text="Category"/>
                <Label Grid.Row="3"
                       Grid.Column="1"
                       Text="{Binding Product.Category.Name}"/>
                <Label Grid.Row="4"
                       Grid.Column="0"
```

```
                    FontAttributes="Bold"
                    Text="Is Starred"
                    VerticalOptions="Center"/>
                <CheckBox Grid.Row="4"
                      Grid.Column="1"
                      HorizontalOptions="Start"
                      IsEnabled="False"
                      IsChecked="{Binding Product.IsStarred}"/>
            </Grid>
        </StackLayout>
    </ScrollView>
    <Button BackgroundColor="Navy"
            Command="{Binding AddToCartBinding}"
            CornerRadius="10"
            Text="Add to cart"
            TextColor="White"
            VerticalOptions="EndAndExpand"/>
  </StackLayout>

</ContentPage>
```

44. Modificamos la **ProductDetailPageViewModel**:

```
…
private ObservableCollection<ProductImage> _images;
…
public ObservableCollection<ProductImage> Images
{
  get => _images;
  set => SetProperty(ref _images, value);
}
…
public override void OnNavigatedTo(INavigationParameters parameters)
{
  base.OnNavigatedTo(parameters);

  if (parameters.ContainsKey("product"))
  {
    Product = parameters.GetValue<Product>("product");
    Title = Product.Name;
    Images = new ObservableCollection<ProductImage>(Product.ProductImages);
  }
}
```

45. Probamos.

Multi Idioma en Xamarin Forms

1. Si no tienes el **ResX Manager Tool**, instalarlo desde: https://marketplace.visualstudio.com/items?itemName=TomEnglert.ResXManager

2. En el proyecto **Prism** crear el folder **Resources** y dentro de este crear los archivos de recursos de los diferentes idiomas:

Key	#	Comment ( )	Neutral	Portuguese [pt]	Spanish [es]
Accept	0		Accept	Aceitar	Aceptar
ConnectionError	1		Check the internet connection.	Verifique a ligação com a Internet.	Compruebe la conexión a Internet
Error	2		Error	Erro	Error

Ingles

```
<data name="Error" xml:space="preserve">
  <value>Error</value>
</data>
<data name="ConnectionError" xml:space="preserve">
  <value>Check the internet connection.</value>
</data>
<data name="Accept" xml:space="preserve">
  <value>Accept</value>
</data>
```

Español

```
<data name="Error" xml:space="preserve">
  <value>Error</value>
</data>
<data name="ConnectionError" xml:space="preserve">
  <value>Compruebe la conexión a Internet</value>
</data>
<data name="Accept" xml:space="preserve">
  <value>Aceptar</value>
</data>
```

Portuguez

```
<data name="Error" xml:space="preserve">
  <value>Erro</value>
</data>
<data name="Accept" xml:space="preserve">
  <value>Aceitar</value>
</data>
```

```
  <data name="ConnectionError" xml:space="preserve">
    <value>Verifique a ligação com a Internet.</value>
  </data>
```

3. En el proyecto **Common** cree el folder **Helpers**, dentro de este crear la interfaz **ILocalize**.

```
public interface ILocalize
{
    CultureInfo GetCurrentCultureInfo();

    void SetLocale(CultureInfo ci);
}
```

4. En el proyecto **Prism** cree el folder **Helpers**, dentro de este crear la clase **PlatformCulture**

```
public class PlatformCulture
{
    public string PlatformString { get; private set; }

    public string LanguageCode { get; private set; }

    public string LocaleCode { get; private set; }

    public PlatformCulture(string platformCultureString)
    {
        if (string.IsNullOrEmpty(platformCultureString))
        {
            throw new ArgumentException("Expected culture identifier", "platformCultureString"); //
in C# 6 use nameof(platformCultureString)
        }

        PlatformString = platformCultureString.Replace("_", "-"); // .NET expects dash, not
underscore
        int dashIndex = PlatformString.IndexOf("-", StringComparison.Ordinal);
        if (dashIndex > 0)
        {
            string[] parts = PlatformString.Split('-');
            LanguageCode = parts[0];
            LocaleCode = parts[1];
        }
        else
        {
            LanguageCode = PlatformString;
            LocaleCode = "";
        }
```

```
    }

    public override string ToString()
    {
        return PlatformString;
    }
}
```

5. En el proyecto **Prism** en la carpeta **Helpers** adicione la clase **Languages**:

```
public static class Languages
{
    static Languages()
    {
        CultureInfo ci = DependencyService.Get<ILocalize>().GetCurrentCultureInfo();
        Resource.Culture = ci;
        Culture = ci.Name;
        DependencyService.Get<ILocalize>().SetLocale(ci);
    }

    public static string Culture { get; set; }

    public static string Accept => Resource.Accept;

    public static string ConnectionError => Resource.ConnectionError;

    public static string Error => Resource.Error;
}
```

6. Implemente la interface en **iOS** en la carpeta **Implementations**.

```
[assembly: Dependency(typeof(OnSale.Prism.iOS.Implementations.Localize))]
namespace OnSale.Prism.iOS.Implementations
{
    public class Localize : ILocalize
    {
        public CultureInfo GetCurrentCultureInfo()
        {
            string netLanguage = "en";
            if (NSLocale.PreferredLanguages.Length > 0)
            {
                string pref = NSLocale.PreferredLanguages[0];
                netLanguage = iOSToDotnetLanguage(pref);
            }
            // this gets called a lot - try/catch can be expensive so consider caching or something
            CultureInfo ci = null;
            try
            {
```

```
            ci = new System.Globalization.CultureInfo(netLanguage);
        }
        catch (CultureNotFoundException)
        {
            // iOS locale not valid .NET culture (eg. "en-ES" : English in Spain)
            // fallback to first characters, in this case "en"
            try
            {
                string fallback = ToDotnetFallbackLanguage(new PlatformCulture(netLanguage));
                ci = new CultureInfo(fallback);
            }
            catch (CultureNotFoundException)
            {
                // iOS language not valid .NET culture, falling back to English
                ci = new CultureInfo("en");
            }
        }
        return ci;
    }

    public void SetLocale(CultureInfo ci)
    {
        Thread.CurrentThread.CurrentCulture = ci;
        Thread.CurrentThread.CurrentUICulture = ci;
    }

    private string iOSToDotnetLanguage(string iOSLanguage)
    {
        string netLanguage = iOSLanguage;
        //certain languages need to be converted to CultureInfo equivalent
        switch (iOSLanguage)
        {
            case "ms-MY":   // "Malaysian (Malaysia)" not supported .NET culture
            case "ms-SG":   // "Malaysian (Singapore)" not supported .NET culture
                netLanguage = "ms"; // closest supported
                break;
            case "gsw-CH":  // "Schwiizertüütsch (Swiss German)" not supported .NET culture
                netLanguage = "de-CH"; // closest supported
                break;
                // add more application-specific cases here (if required)
                // ONLY use cultures that have been tested and known to work
        }

        return netLanguage;
    }

    private string ToDotnetFallbackLanguage(PlatformCulture platCulture)
    {
```

```
            string netLanguage = platCulture.LanguageCode; // use the first part of the identifier (two
chars, usually);
            switch (platCulture.LanguageCode)
            {
                case "pt":
                    netLanguage = "pt-PT"; // fallback to Portuguese (Portugal)
                    break;
                case "gsw":
                    netLanguage = "de-CH"; // equivalent to German (Switzerland) for this app
                    break;
                    // add more application-specific cases here (if required)
                    // ONLY use cultures that have been tested and known to work
            }

            return netLanguage;
        }
    }
}
```

7. Adicione la lista de idiomas al **info.plist**.

```
<key>CFBundleLocalizations</key>
<array>
        <string>es</string>
        <string>pt</string>
</array>
<key>CFBundleDevelopmentRegion</key>
<string>en</string>
```

8. Implemente la interfaz en **Android** en la carpeta **Implementations**.

```
[assembly: Dependency(typeof(OnSale.Prism.Droid.Implementations.Localize))]
namespace OnSale.Prism.Droid.Implementations
{
    public class Localize : ILocalize
    {
        public CultureInfo GetCurrentCultureInfo()
        {
            string netLanguage = "en";
            Java.Util.Locale androidLocale = Java.Util.Locale.Default;
            netLanguage = AndroidToDotnetLanguage(androidLocale.ToString().Replace("_", "-"));
            // this gets called a lot - try/catch can be expensive so consider caching or something
            CultureInfo ci = null;
            try
            {
                ci = new CultureInfo(netLanguage);
            }
            catch (CultureNotFoundException)
```

```
        {
            // iOS locale not valid .NET culture (eg. "en-ES" : English in Spain)
            // fallback to first characters, in this case "en"
            try
            {
                string fallback = ToDotnetFallbackLanguage(new PlatformCulture(netLanguage));
                ci = new CultureInfo(fallback);
            }
            catch (CultureNotFoundException)
            {
                // iOS language not valid .NET culture, falling back to English
                ci = new CultureInfo("en");
            }
        }
        return ci;
    }

    public void SetLocale(CultureInfo ci)
    {
        Thread.CurrentThread.CurrentCulture = ci;
        Thread.CurrentThread.CurrentUICulture = ci;
    }

    private string AndroidToDotnetLanguage(string androidLanguage)
    {
        string netLanguage = androidLanguage;
        //certain languages need to be converted to CultureInfo equivalent
        switch (androidLanguage)
        {
            case "ms-BN":   // "Malaysian (Brunei)" not supported .NET culture
            case "ms-MY":   // "Malaysian (Malaysia)" not supported .NET culture
            case "ms-SG":   // "Malaysian (Singapore)" not supported .NET culture
                netLanguage = "ms"; // closest supported
                break;
            case "in-ID":  // "Indonesian (Indonesia)" has different code in  .NET
                netLanguage = "id-ID"; // correct code for .NET
                break;
            case "gsw-CH":  // "Schwiizertüütsch (Swiss German)" not supported .NET culture
                netLanguage = "de-CH"; // closest supported
                break;
                // add more application-specific cases here (if required)
                // ONLY use cultures that have been tested and known to work
        }
        return netLanguage;
    }

    private string ToDotnetFallbackLanguage(PlatformCulture platCulture)
    {
```

```
            string netLanguage = platCulture.LanguageCode; // use the first part of the identifier (two
chars, usually);
            switch (platCulture.LanguageCode)
            {
                case "gsw":
                    netLanguage = "de-CH"; // equivalent to German (Switzerland) for this app
                    break;
                    // add more application-specific cases here (if required)
                    // ONLY use cultures that have been tested and known to work
            }
            return netLanguage;
        }
    }
}
```

9. Modificar la **ProductsPageViewModel**:

```
private async void LoadProductsAsync()
{
    if (Connectivity.NetworkAccess != NetworkAccess.Internet)
    {
        await App.Current.MainPage.DisplayAlert(Languages.Error, Languages.ConnectionError,
Languages.Accept);
        return;
    }

    IsRunning = true;
    string url = App.Current.Resources["UrlAPI"].ToString();
    Response response = await _apiService.GetListAsync<Product>(
        url,
        "/api",
        "/Products");
    IsRunning = false;

    if (!response.IsSuccess)
    {
        await App.Current.MainPage.DisplayAlert(Languages.Error, response.Message,
Languages.Accept);
        return;
    }

    _myProducts = (List<Product>)response.Result;
    ShowProducts();
}
```

10. Ahora vamos a traducir los literales directamente en el XAML adicionamos la clase **TranslateExtension** in folder **Helpers** en el proyecto **Prism**:

```
[ContentProperty("Text")]
public class TranslateExtension : IMarkupExtension
{
    private readonly CultureInfo ci;
    private const string ResourceId = "OnSale.Prism.Resources.Resource";
    private static readonly Lazy<ResourceManager> ResMgr =
        new Lazy<ResourceManager>(() => new ResourceManager(
            ResourceId,
            typeof(TranslateExtension).GetTypeInfo().Assembly));

    public TranslateExtension()
    {
        ci = DependencyService.Get<ILocalize>().GetCurrentCultureInfo();
    }

    public string Text { get; set; }

    public object ProvideValue(IServiceProvider serviceProvider)
    {
        if (Text == null)
        {
            return "";
        }

        string translation = ResMgr.Value.GetString(Text, ci);

        if (translation == null)
        {
#if DEBUG
            throw new ArgumentException(
                string.Format(
                    "Key '{0}' was not found in resources '{1}' for culture '{2}'.",
                    Text, ResourceId, ci.Name), "Text");
#else
        translation = Text; // returns the key, which GETS DISPLAYED TO THE USER
#endif
        }

        return translation;
    }
}
```

11. Completamos los literales:

Ingles

```
  <data name="Loading" xml:space="preserve">
    <value>Loading...</value>
```

```
  </data>
  <data name="SearchProduct" xml:space="preserve">
    <value>Search a product...</value>
  </data>
  <data name="Name" xml:space="preserve">
    <value>Name</value>
  </data>
  <data name="Description" xml:space="preserve">
    <value>Description</value>
  </data>
  <data name="Price" xml:space="preserve">
    <value>Price</value>
  </data>
  <data name="Category" xml:space="preserve">
    <value>Category</value>
  </data>
  <data name="IsStarred" xml:space="preserve">
    <value>Is Starred</value>
  </data>
  <data name="AddToCart" xml:space="preserve">
    <value>Add to cart</value>
  </data>
  <data name="Product" xml:space="preserve">
    <value>Product</value>
  </data>
  <data name="Products" xml:space="preserve">
    <value>Products</value>
  </data>
```

Español

```
  <data name="Loading" xml:space="preserve">
    <value>Cargando...</value>
  </data>
  <data name="SearchProduct" xml:space="preserve">
    <value>Buscar un producto...</value>
  </data>
  <data name="Name" xml:space="preserve">
    <value>Nombre</value>
  </data>
  <data name="Description" xml:space="preserve">
    <value>Descripción</value>
  </data>
  <data name="Price" xml:space="preserve">
    <value>Precio</value>
  </data>
  <data name="Category" xml:space="preserve">
    <value>Categoría</value>
```

```
  </data>
  <data name="IsStarred" xml:space="preserve">
    <value>Tiene estrellas</value>
  </data>
  <data name="AddToCart" xml:space="preserve">
    <value>Agregar al carro</value>
  </data>
  <data name="Product" xml:space="preserve">
    <value>Producto</value>
  </data>
  <data name="Products" xml:space="preserve">
    <value>Productos</value>
  </data>
```

Portuguez

```
  <data name="SearchProduct" xml:space="preserve">
    <value>Pesquisar Produto...</value>
  </data>
  <data name="Name" xml:space="preserve">
    <value>Nome</value>
  </data>
  <data name="Description" xml:space="preserve">
    <value>Descrição</value>
  </data>
  <data name="Price" xml:space="preserve">
    <value>Preço</value>
  </data>
  <data name="Category" xml:space="preserve">
    <value>Categoria</value>
  </data>
  <data name="IsStarred" xml:space="preserve">
    <value>É Estrelado</value>
  </data>
  <data name="AddToCart" xml:space="preserve">
    <value>Adicionar ao carrinho</value>
  </data>
  <data name="Product" xml:space="preserve">
    <value>Produto</value>
  </data>
  <data name="Products" xml:space="preserve">
    <value>Produtos</value>
  </data>
```

12. Adicione los nuevos literales en la clase **Languages**:

```
public static string Loading => Resource.Loading;
```

```
public static string SearchProduct => Resource.SearchProduct;

public static string Name => Resource.Name;

public static string Description => Resource.Description;

public static string Price => Resource.Price;

public static string Category => Resource.Category;

public static string IsStarred => Resource.IsStarred;

public static string AddToCart => Resource.AddToCart;

public static string Product => Resource.Product;

public static string Products => Resource.Products;
```

13. Modificar la **ProductsPage:**

```
<?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:prism="http://prismlibrary.com"
            prism:ViewModelLocator.AutowireViewModel="True"

xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.For
ms"

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

xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion
.SfBusyIndicator.XForms"
            xmlns:i18n="clr-namespace:OnSale.Prism.Helpers"
            x:Class="OnSale.Prism.Views.ProductsPage"
            Title="{Binding Title}">

    <AbsoluteLayout>
        <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"
                    AbsoluteLayout.LayoutFlags="All"
                    Padding="5">
            <SearchBar Placeholder="{i18n:Translate SearchProduct}"
                      SearchCommand="{Binding SearchCommand}"
                      Text="{Binding Search}"/>
            <CollectionView ItemsSource="{Binding Products}">
                <CollectionView.ItemsLayout>
```

```
            <GridItemsLayout Orientation="Vertical"/>
          </CollectionView.ItemsLayout>
          <CollectionView.ItemTemplate>
            <DataTemplate>
                <Grid>
                  <Grid.GestureRecognizers>
                    <TapGestureRecognizer Command="{Binding
SelectProductCommand}"/>
                  </Grid.GestureRecognizers>
                  <Grid.ColumnDefinitions>
                  <ColumnDefinition Width="Auto" />
                  <ColumnDefinition Width="*" />
                  <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <ffimageloading:CachedImage Grid.Column="0"
                              Aspect="AspectFill"
                              Source="{Binding ImageFullPath}"
                              CacheDuration= "50"
                              Margin="5"
                              RetryCount= "3"
                              RetryDelay= "600"
                              WidthRequest="100"/>
                <StackLayout Grid.Column="1"
                       VerticalOptions="Center">
                  <Label Text="{Binding Name}"
                       FontAttributes="Bold"
                       FontSize="Medium"
                       LineBreakMode="TailTruncation" />
                  <Label Text="{Binding Price, StringFormat='{0:C2}'}"
                       LineBreakMode="TailTruncation"
                       FontAttributes="Italic"
                       VerticalOptions="End" />
                </StackLayout>
                <Image Grid.Column="2"
                     Source="ic_more_vert"/>
              </Grid>
            </DataTemplate>
          </CollectionView.ItemTemplate>
        </CollectionView>
      </StackLayout>
      <busyindicator:SfBusyIndicator AnimationType="Gear"
                       AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"
                         AbsoluteLayout.LayoutFlags="All"
                         BackgroundColor="Silver"
                         HorizontalOptions="Center"
                         TextColor="White"
                         IsBusy="{Binding IsRunning}"
                         Title="{i18n:Translate Loading}"
```

```
                                VerticalOptions="Center"
                                ViewBoxWidth="80"
                                ViewBoxHeight="80" />
    </AbsoluteLayout>

</ContentPage>
```

14. Modificar la **ProductDetailPage:**

```
<?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:prism="http://prismlibrary.com"
             prism:ViewModelLocator.AutowireViewModel="True"

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

xmlns:syncfusion="clr-namespace:Syncfusion.SfRotator.XForms;assembly=Syncfusion.SfRotat
or.XForms"

xmlns:ffimageloading="clr-namespace:FFImageLoading.Forms;assembly=FFImageLoading.For
ms"
             xmlns:i18n="clr-namespace:OnSale.Prism.Helpers"
             x:Class="OnSale.Prism.Views.ProductDetailPage"
             Title="{Binding Title}">

    <StackLayout Padding="5">
        <ScrollView>
            <StackLayout>
                <syncfusion:SfRotator EnableAutoPlay="True"
                                      EnableLooping="True"
                                      HeightRequest="300"
                                      ItemsSource="{Binding Images}"
                                      NavigationDelay="5000"
                                      NavigationDirection="Horizontal"
                                      NavigationStripMode="Thumbnail"
                                      NavigationStripPosition="Bottom">
                    <syncfusion:SfRotator.ItemTemplate>
                        <DataTemplate>
                            <ffimageloading:CachedImage Aspect="AspectFit"
                                                        CacheDuration= "50"
                                                        DownsampleToViewSize = "true"
                                                        ErrorPlaceholder= "ErrorImage"
                                                        HeightRequest="300"
                                                        LoadingPlaceholder= "LoaderImage"
```

```
                                RetryCount= "3"
                                RetryDelay= "600"
                                Source="{Binding ImageFullPath}"/>
            </DataTemplate>
        </syncfusion:SfRotator.ItemTemplate>
    </syncfusion:SfRotator>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Grid.Row="0"
               Grid.Column="0"
               FontAttributes="Bold"
               Text="{i18n:Translate Name}"/>
        <Label Grid.Row="0"
               Grid.Column="1"
               Text="{Binding Product.Name}"/>
        <Label Grid.Row="1"
               Grid.Column="0"
               FontAttributes="Bold"
               Text="{i18n:Translate Description}"/>
        <Label Grid.Row="1"
               Grid.Column="1"
               Text="{Binding Product.Description}"/>
        <Label Grid.Row="2"
               Grid.Column="0"
               FontAttributes="Bold"
               Text="{i18n:Translate Price}"/>
        <Label Grid.Row="2"
               Grid.Column="1"
               Text="{Binding Product.Price, StringFormat='{0:C2}'}"/>
        <Label Grid.Row="3"
               Grid.Column="0"
               FontAttributes="Bold"
               Text="{i18n:Translate Category}"/>
        <Label Grid.Row="3"
               Grid.Column="1"
               Text="{Binding Product.Category.Name}"/>
        <Label Grid.Row="4"
               Grid.Column="0"
```

```
                    FontAttributes="Bold"
                    Text="{i18n:Translate IsStarred}"
                    VerticalOptions="Center"/>
                <CheckBox Grid.Row="4"
                      Grid.Column="1"
                      HorizontalOptions="Start"
                      IsEnabled="False"
                      IsChecked="{Binding Product.IsStarred}"/>
            </Grid>
        </StackLayout>
    </ScrollView>
    <Button BackgroundColor="Navy"
            Command="{Binding AddToCartBinding}"
            CornerRadius="10"
            Text="{i18n:Translate AddToCart}"
            TextColor="White"
            VerticalOptions="EndAndExpand"/>
  </StackLayout>

</ContentPage>
```

15. Modificar la **ProductsPageViewModel**:

```
public ProductsPageViewModel(INavigationService navigationService, IApiService apiService) :
base(navigationService)
{
  _navigationService = navigationService;
  _apiService = apiService;
  Title = Languages.Products;
  LoadProductsAsync();
}
```

16. Modificar la **ProductDetailPageViewModel**:

```
public ProductDetailPageViewModel(INavigationService navigationService) :
base(navigationService)
{
  Title = Languages.Product;
}
```

17. Probar.

Adicionando colores y estilos

Como buenos desarrolladores somos terribles combinando colores. Recomiendo esta pagina para encontrar combinaciones de colores que son aprobadas por profesionales: https://color.adobe.com/es/explore

Para mi caso escojo estos colores y le damos un role a cada color.

Color Background / Color Font Inverse	Color Secondary	Color Primary	Color Danger / Color Accent	Color Font
#D9D9D9	#8C8C8C	#3E518C	#73221A	#0D0D0D

Un color puede tener varios roles.

1. Adicionamos estos colores al diccionario de recursos:

```
<ResourceDictionary>

    <!-- Parameters -->
    <x:String x:Key="UrlAPI">https://onsaleprepweb.azurewebsites.net</x:String>

    <!-- Colors -->
    <Color x:Key="ColorBackground">#D9D9D9</Color>
    <Color x:Key="ColorPrimary">#3E518C</Color>
    <Color x:Key="ColorSecondary">#8C8C8C</Color>
    <Color x:Key="ColorDanger">#73221A</Color>
    <Color x:Key="ColorAccent">#73221A</Color>
    <Color x:Key="ColorFont">#0D0D0D</Color>
    <Color x:Key="ColorFontInverse">#D9D9D9</Color>

</ResourceDictionary>
```

2. Adicionamos esta propiedad a todas las Pages:

```
BackgroundColor="{StaticResource ColorBackground}"
```

3. Modificamos el archivo **styles.xml** en **Android**:

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>

  <style name="MainTheme" parent="MainTheme.Base">
  </style>
  <!-- Base theme applied no matter what API -->
  <style name="MainTheme.Base" parent="Theme.AppCompat.Light.DarkActionBar">
    <!--If you are using revision 22.1 please use just windowNoTitle. Without android:-->
    <item name="windowNoTitle">true</item>
    <!--We will be using the toolbar so no need to show ActionBar-->
    <item name="windowActionBar">false</item>
    <!-- Set theme colors from
http://www.google.com/design/spec/style/color.html#color-color-palette -->
    <!-- colorPrimary is used for the default action bar background -->
    <item name=colorPrimary">#3E518C</item>
    <!-- colorPrimaryDark is used for the status bar -->
    <item name="colorPrimaryDark">#3E518C</item>
    <!-- colorAccent is used as the default value for colorControlActivated
         which is used to tint widgets -->
    <item name="colorAccent">#73221A</item>
    <!-- You can also set colorControlNormal, colorControlActivated
         colorControlHighlight and colorSwitchThumbNormal. -->
    <item name="windowActionModeOverlay">true</item>

    <item name="android:datePickerDialogTheme">@style/AppCompatDialogStyle</item>
  </style>

  <style name="AppCompatDialogStyle" parent="Theme.AppCompat.Light.Dialog">
    <item name="colorAccent">#73221A</item>
  </style>
</resources>
```

4. Adicionamos estos estilos al diccionario de recursos:

```
<!-- Styles -->
<Style TargetType="Button">
    <Setter Property="BackgroundColor" Value="{StaticResource ColorPrimary}" />
    <Setter Property="HorizontalOptions" Value="FillAndExpand" />
    <Setter Property="TextColor" Value="{StaticResource ColorFontInverse}" />
</Style>
```

```
<Style TargetType="Label">
   <Setter Property="TextColor" Value="{StaticResource ColorFont}" />
</Style>

<Style x:Key="SecondaryButton" TargetType="Button">
   <Setter Property="BackgroundColor" Value="{StaticResource ColorSecondary}" />
</Style>

<Style x:Key="DangerButton" TargetType="Button">
   <Setter Property="BackgroundColor" Value="{StaticResource ColorDanger}" />
</Style>
```

5. Modify the **ProductsPage**:

```
<busyindicator:SfBusyIndicator AnimationType="Gear"
                               AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"
                                  AbsoluteLayout.LayoutFlags="All"
                                  BackgroundColor="{StaticResource ColorAccent}"
                                  HorizontalOptions="Center"
                                  TextColor="{StaticResource ColorFontInverse}"
                                  IsBusy="{Binding IsRunning}"
                                  Title="{i18n:Translate Loading}"
                                  VerticalOptions="Center"
                                  ViewBoxWidth="80"
                                  ViewBoxHeight="80" />
```

6. Probamos.

Adicionando Icono & Splash

Android

1. Adicione una imagen para el Splash en la carpeta **drawable**, las dimensiones deben ser: 480 x 800 pixels o su equivalente. Para nuestro ejemplo vamos a usar: **onsale_splash.png**.

2. Adicione estas líneas a **styles.xml**.

```
 </style>
 <style name="Theme.Splash" parent="android:Theme">
```

```
        <item name="android:windowBackground">@drawable/onsale_splash</item>
        <item name="android:windowNoTitle">true</item>
  </style>
</resources>
```

3. En el proyecto Android adicione el **SplashActivity**.

```
[Activity(Theme = "@style/Theme.Splash", MainLauncher = true, NoHistory = true)]
public class SplashActivity : Activity
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);
        System.Threading.Thread.Sleep(1800);
        StartActivity(typeof(MainActivity));
    }
}
```

4. Modifique el **MainActivity** para cambiar la propiedad **MainLauncher** a **false**.

```
[Activity(Label = "On Sale", Icon = "@mipmap/ic_launcher", Theme = "@style/MainTheme",
MainLauncher = false, ConfigurationChanges = ConfigChanges.ScreenSize |
ConfigChanges.Orientation)]
```

5. Probamos.

6. Ahora vamos a cambiar el ícono de la aplicación. Vamos a https://romannurik.github.io/AndroidAssetStudio/ y personalizamos el ícono de la aplicación.

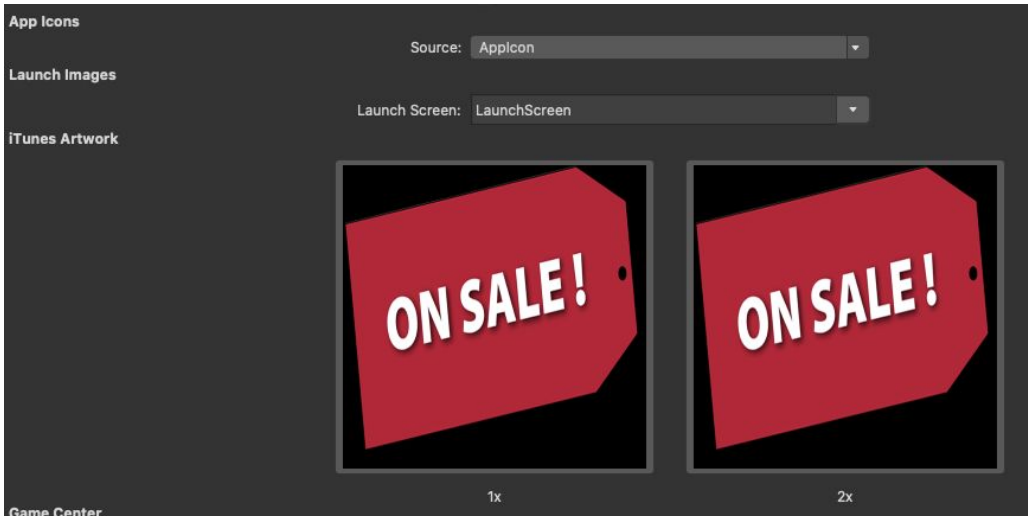
7. Cambiamos el nombre de la aplicación en propiedades del proyecto y probamos.

iOS

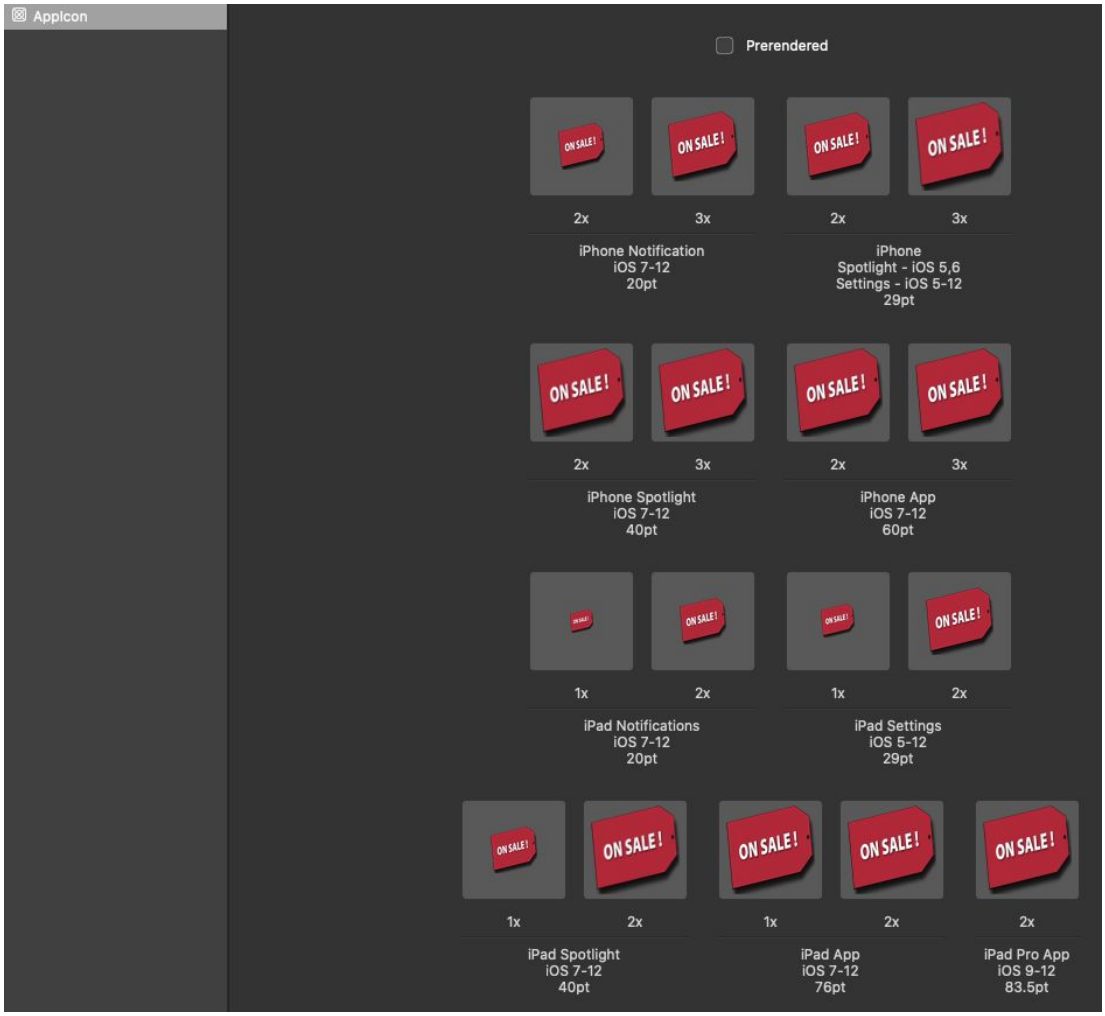
1. Agregamos la imagen del Splash en **Resources**.

2. Generamos los íconos en los tamaños de iOS en https://makeappicon.com

3. Editamos el info.plist y cambiamos el nombre de la aplicación y los íconos para mostrar en tienda, debes de asegurarte que App Icons sea Source: AppIcon:



4. Editamos los **Assets.xcassets** y colocamos los iconos correctos:



5. Probamos.

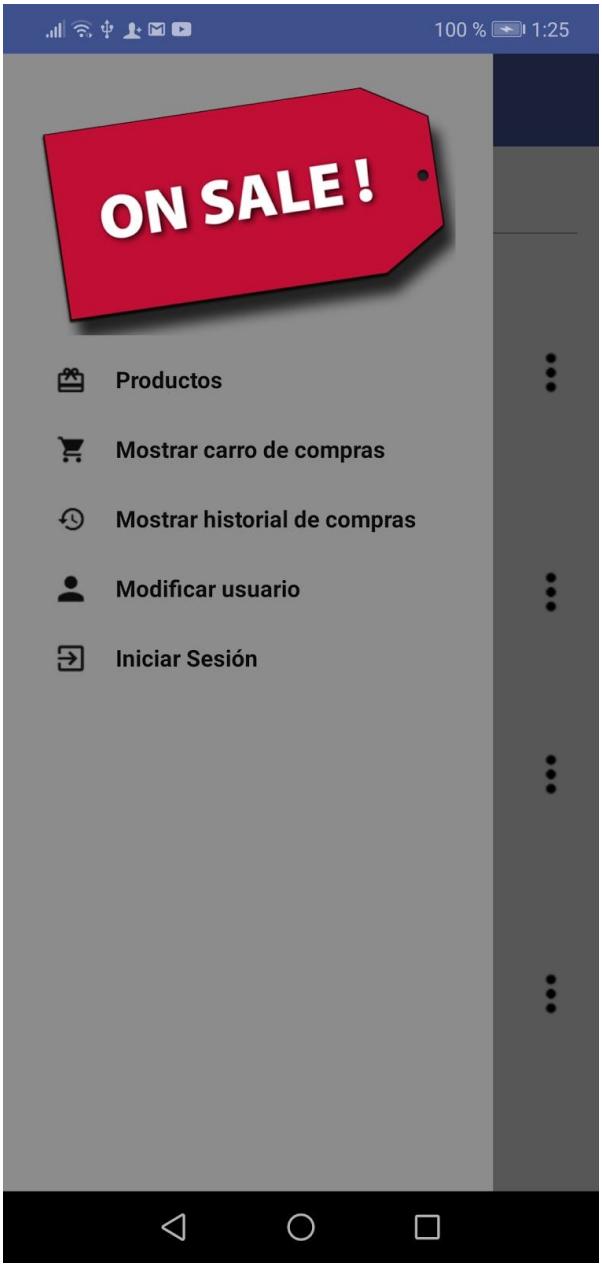
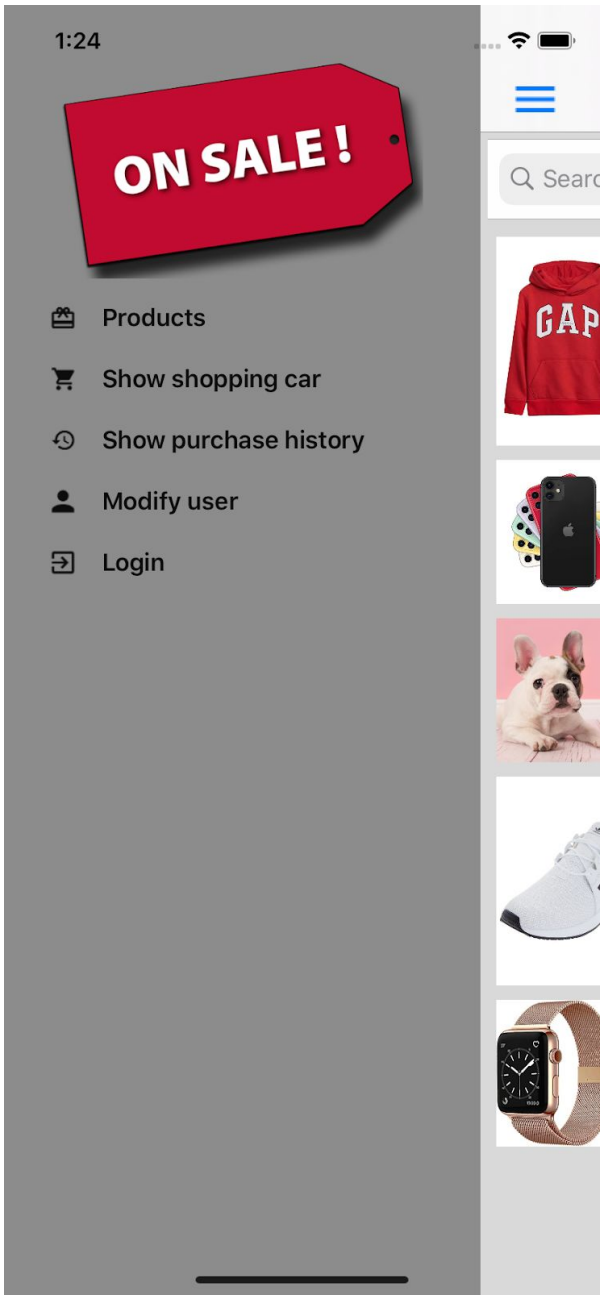
6. Ahora vamos con el Splash, editamos el **LaunchScreen.storyboard**:

7. En el info.plist colocamos como **Main Interface** al **LaunchScreen**.

8. Probamos.

Adicionando una master detail

Vamos a implementar un menú a nuestra aplicación

Android	iOS
	

1. Agregar los siguientes literales:

Ingles

```
<data name="Login" xml:space="preserve">
  <value>Login</value>
</data>
<data name="ModifyUser" xml:space="preserve">
  <value>Modify user</value>
</data>
<data name="ShowPurchaseHistory" xml:space="preserve">
  <value>Show purchase history</value>
</data>
<data name="ShowShoppingCar" xml:space="preserve">
  <value>Show shopping car</value>
</data>
```

Español

```
<data name="Login" xml:space="preserve">
  <value>Iniciar Sesión</value>
</data>
<data name="ModifyUser" xml:space="preserve">
  <value>Modificar usuario</value>
</data>
<data name="ShowPurchaseHistory" xml:space="preserve">
  <value>Mostrar historial de compras</value>
</data>
<data name="ShowShoppingCar" xml:space="preserve">
  <value>Mostrar carro de compras</value>
</data>
```

Portuguez

```
<data name="Login" xml:space="preserve">
  <value>Conecte-se</value>
</data>
<data name="ModifyUser" xml:space="preserve">
  <value>Modificar usuário</value>
</data>
<data name="ShowPurchaseHistory" xml:space="preserve">
  <value>Mostrar histórico de compras</value>
</data>
<data name="ShowShoppingCar" xml:space="preserve">
  <value>Mostrar carro de compras</value>
```

```
  </data>
```

2. Modificar **Languages**:

```
public static string Login => Resource.Login;

public static string ShowShoppingCar => Resource.ShowShoppingCar;

public static string ShowPurchaseHistory => Resource.ShowPurchaseHistory;

public static string ModifyUser => Resource.ModifyUser;
```

3. En el proyecto **Common** crea la carpeta **Models** y dentro de esta crea la clase **Menu**:

```
public class Menu
{
    public string Icon { get; set; }

    public string Title { get; set; }

    public string PageName { get; set; }

    public bool IsLoginRequired { get; set; }
}
```

4. Creamos la **LoginPage** inicialmente con este diseño:

```
<?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:prism="http://prismlibrary.com"
             prism:ViewModelLocator.AutowireViewModel="True"

xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfiguration.iOSSpecific;assembly=Xamari
n.Forms.Core"
             ios:Page.UseSafeArea="true"
             x:Class="OnSale.Prism.Views.LoginPage"
             BackgroundColor="{StaticResource ColorBackground}"
             Title="{Binding Title}">

</ContentPage>
```

5. Modificamos la **LoginPageViewModel** por:

```
public class LoginPageViewModel : ViewModelBase
{
    public LoginPageViewModel(INavigationService navigationService) : base(navigationService)
    {
        Title = Languages.Login;
    }
}
```

6. Adicione el ícono **ic_action_menu** para la master detail.

7. Creamos la **OnSaleMasterDetailPage**:

```
<?xml version="1.0" encoding="utf-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
                  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                  xmlns:prism="http://prismlibrary.com"
                  prism:ViewModelLocator.AutowireViewModel="True"
                  x:Class="OnSale.Prism.ViewModels.OnSaleMasterDetailPage">

    <MasterDetailPage.Master>
        <ContentPage BackgroundColor="{StaticResource ColorSecondary}"
                     IconImageSource="ic_action_menu"
                     Title="Menu">

            <ContentPage.Padding>
                <OnPlatform x:TypeArguments="Thickness">
                    <On Platform="Android, UWP">0</On>
                    <On Platform="iOS">0,20,0,0</On>
                </OnPlatform>
            </ContentPage.Padding>

            <StackLayout Padding="20">
                <Image HeightRequest="150"
                       Source="onsale"/>
                <ListView BackgroundColor="Transparent"
                          ItemsSource="{Binding Menus}"
                          HasUnevenRows="True"
                          SeparatorVisibility="None">
                    <ListView.ItemTemplate>
                        <DataTemplate>
                            <ViewCell>
```

```
                        <Grid>
                            <Grid.GestureRecognizers>
                                <TapGestureRecognizer Command="{Binding
SelectMenuCommand}"/>
                            </Grid.GestureRecognizers>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="Auto"></ColumnDefinition>
                                <ColumnDefinition Width="*"></ColumnDefinition>
                            </Grid.ColumnDefinitions>
                            <Image Grid.Column="0"
                                    HeightRequest="32"
                                Margin="5"
                                    Source="{Binding Icon}"
                                    WidthRequest="32"/>
                            <Label Grid.Column="1"
                                FontAttributes="Bold"
                                VerticalOptions="Center"
                                Text="{Binding Title}"/>
                        </Grid>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
  </ContentPage>
 </MasterDetailPage.Master>

</MasterDetailPage>
```

8. Creamos la **MenuItemViewModel**:

```
public class MenuItemViewModel : Menu
{
    private readonly INavigationService _navigationService;
    private DelegateCommand _selectMenuCommand;

    public MenuItemViewModel(INavigationService navigationService)
    {
        _navigationService = navigationService;
    }

    public DelegateCommand SelectMenuCommand => _selectMenuCommand ??
(_selectMenuCommand = new DelegateCommand(SelectMenuAsync));
```

```
    private async void SelectMenuAsync()
    {
        await
_navigationService.NavigateAsync($"/{nameof(OnSaleMasterDetailPage)}/NavigationPage/{Pa
geName}");
    }
}
```

9. Adicione las páginas con los layout y las viewmodel básicos: **ShowCarPage**, **ShowHistoryPage**, **ModifyUserPage**

10. Adicione los íconos que aparecen en el menú: **ic_card_giftcard**, **ic_shopping_cart**, **ic_history**, **ic_person**, **ic_exit_to_app**.

11. Modifique la **OnSaleMasterDetailPageViewModel**:

```
public class OnSaleMasterDetailPageViewModel : ViewModelBase
{
    private readonly INavigationService _navigationService;

    public OnSaleMasterDetailPageViewModel(INavigationService navigationService) :
base(navigationService)
    {
        _navigationService = navigationService;
        LoadMenus();
    }
    public ObservableCollection<MenuItemViewModel> Menus { get; set; }

    private void LoadMenus()
    {
        List<Menu> menus = new List<Menu>
        {
            new Menu
            {
                Icon = "ic_card_giftcard",
                PageName = $"{nameof(ProductsPage)}",
                Title = Languages.Products
            },
            new Menu
            {
                Icon = "ic_shopping_cart",
                PageName = $"{nameof(ShowCarPage)}",
```

```
            Title = Languages.ShowShoppingCar
        },
        new Menu
        {
            Icon = "ic_history",
            PageName = $"{nameof(ShowHistoryPage)}",
            Title = Languages.ShowPurchaseHistory,
            IsLoginRequired = true
        },
        new Menu
        {
            Icon = "ic_person",
            PageName = $"{nameof(ModifyUserPage)}",
            Title = Languages.ModifyUser,
            IsLoginRequired = true
        },
        new Menu
        {
            Icon = "ic_exit_to_app",
            PageName = $"{nameof(LoginPage)}",
            Title = Languages.Login
        }
    };

    Menus = new ObservableCollection<MenuItemViewModel>(
        menus.Select(m => new MenuItemViewModel(_navigationService)
        {
            Icon = m.Icon,
            PageName = m.PageName,
            Title = m.Title,
            IsLoginRequired = m.IsLoginRequired
        }).ToList());
  }
}
```

12. Cambiamos el inicio de la aplicación en el **App.xaml.cs**:

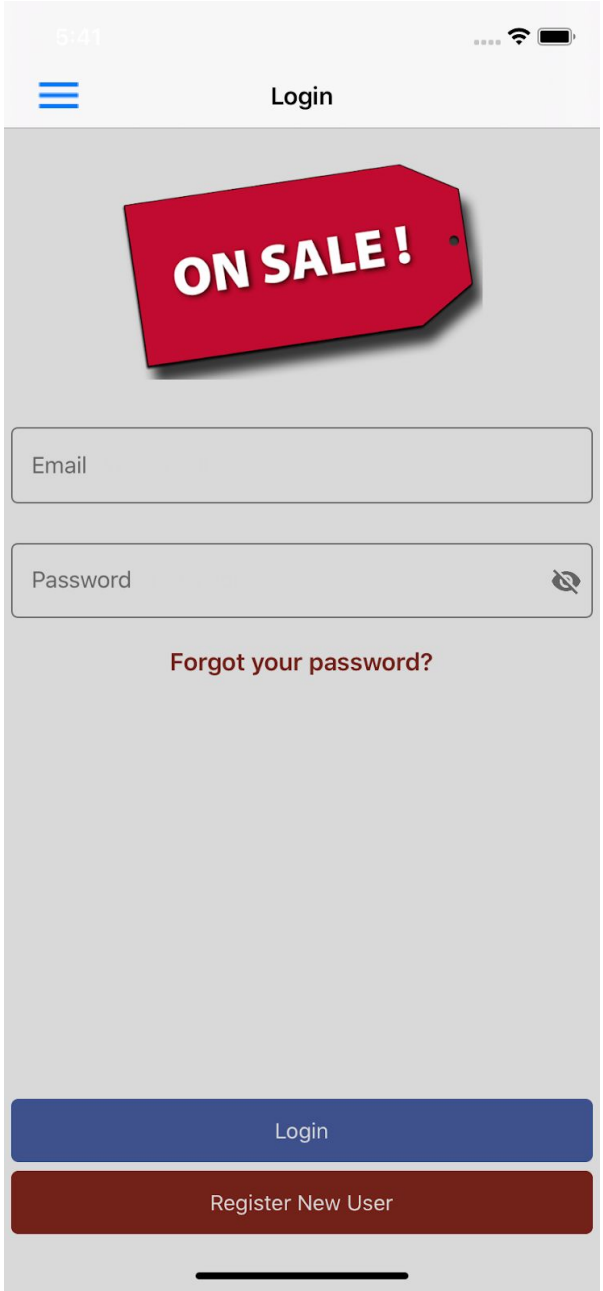
```
protected override async void OnInitialized()
{

SyncfusionLicenseProvider.RegisterLicense("MzAxMjQ2QDMxMzgyZTMyMmUzMEEwd1VtbUd
CVDlHdm1HYjhKNDBUNVk2WE4zdU1DMjFkY3BmNDZ6SXJXM2s9");
  InitializeComponent();
```

```
    await
NavigationService.NavigateAsync($"{nameof(OnSaleMasterDetailPage)}/NavigationPage/{nam
eof(ProductsPage)}");
}
```

13. Probemos.

Login

Vamos a implementar la opción de Login.

Android	iOS
	

1. Agregamos los siguientes literales:

Ingles

```
<data name="Email" xml:space="preserve">
  <value>Email</value>
</data>
<data name="EmailPlaceHolder" xml:space="preserve">
  <value>Enter your email...</value>
</data>
<data name="EmailError" xml:space="preserve">
  <value>You must enter a valid email.</value>
</data>
<data name="Password" xml:space="preserve">
  <value>Password</value>
</data>
<data name="PasswordPlaceHolder" xml:space="preserve">
  <value>Enter your password...</value>
</data>
<data name="PasswordError" xml:space="preserve">
  <value>You must enter your password.</value>
</data>
<data name="ForgotPassword" xml:space="preserve">
  <value>Forgot your password?</value>
</data>
<data name="Register" xml:space="preserve">
  <value>Register New User</value>
</data>
<data name="LoginError" xml:space="preserve">
  <value>Email or password wrong</value>
</data>
<data name="Logout" xml:space="preserve">
  <value>Logout</value>
</data>
```

Español

```
<data name="Email" xml:space="preserve">
  <value>Email</value>
</data>
<data name="EmailPlaceHolder" xml:space="preserve">
  <value>Enter your email...</value>
</data>
<data name="EmailError" xml:space="preserve">
  <value>You must enter a valid email.</value>
</data>
<data name="Password" xml:space="preserve">
  <value>Password</value>
</data>
<data name="PasswordPlaceHolder" xml:space="preserve">
```

```
  <value>Enter your password...</value>
</data>
<data name="PasswordError" xml:space="preserve">
  <value>You must enter your password.</value>
</data>
<data name="ForgotPassword" xml:space="preserve">
  <value>Forgot your password?</value>
</data>
<data name="Register" xml:space="preserve">
  <value>Register New User</value>
</data>
<data name="LoginError" xml:space="preserve">
  <value>Correo electrónico o contraseña incorrectos.</value>
</data>
<data name="Logout" xml:space="preserve">
  <value>Cerrar sesión</value>
</data>
```

Portuguez

```
<data name="Email" xml:space="preserve">
  <value>Email</value>
</data>
<data name="EmailPlaceHolder" xml:space="preserve">
  <value>Digite seu e-mail...</value>
</data>
<data name="EmailError" xml:space="preserve">
  <value>Você deve inserir um e-mail válido.</value>
</data>
<data name="Password" xml:space="preserve">
  <value>Senha</value>
</data>
<data name="PasswordPlaceHolder" xml:space="preserve">
  <value>Coloque sua senha...</value>
</data>
<data name="PasswordError" xml:space="preserve">
  <value>Você deve inserir sua senha.</value>
</data>
<data name="ForgotPassword" xml:space="preserve">
  <value>Você esqueceu sua senha?</value>
</data>
<data name="Register" xml:space="preserve">
  <value>Registrar Novo Usuário</value>
</data>
<data name="LoginError" xml:space="preserve">
  <value>Email ou senha incorretos.</value>
</data>
<data name="Logout" xml:space="preserve">
```

```
  <value>Fazer logoff</value>
 </data>
```

2. Modificamos **Languages**:

```
public static string Email => Resource.Email;

public static string EmailError => Resource.EmailError;

public static string EmailPlaceHolder => Resource.EmailPlaceHolder;

public static string Password => Resource.Password;

public static string PasswordError => Resource.PasswordError;

public static string PasswordPlaceHolder => Resource.PasswordPlaceHolder;

public static string ForgotPassword => Resource.ForgotPassword;

public static string LoginError => Resource.LoginError;

public static string Logout => Resource.Logout;
```

3. Agregamos el Nuget **Syncfusion.Xamarin.Core** a todos los proyectos Prism.

4. Inicializar el renderer para iOS en el **AppDelegate**:

```
new SfBusyIndicatorRenderer();
new SfRotatorRenderer();
SfTextInputLayoutRenderer.Init();
LoadApplication(new App(new iOSInitializer()));
```

5. Modificamos la **LoginPage**:

```
<?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:prism="http://prismlibrary.com"
             prism:ViewModelLocator.AutowireViewModel="True"

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

xmlns:busyindicator="clr-namespace:Syncfusion.SfBusyIndicator.XForms;assembly=Syncfusion.SfBusyIndicator.XForms"
             xmlns:i18n="clr-namespace:OnSale.Prism.Helpers"
```

```
xmlns:inputLayout="clr-namespace:Syncfusion.XForms.TextInputLayout;assembly=Syncfusion.
Core.XForms"
            x:Class="OnSale.Prism.Views.LoginPage"
            BackgroundColor="{StaticResource ColorBackground}"
            Title="{Binding Title}">

    <AbsoluteLayout>
        <StackLayout AbsoluteLayout.LayoutBounds="0,0,1,1"
                     AbsoluteLayout.LayoutFlags="All"
                     Padding="5">
            <ScrollView>
                <StackLayout>
                    <Image HeightRequest="150"
                           Margin="20"
                           Source="onsale"/>

                    <StackLayout VerticalOptions="CenterAndExpand">
                        <inputLayout:SfTextInputLayout Hint="{i18n:Translate Email}"
                                                       ContainerType="Outlined">
                            <Entry Placeholder="{i18n:Translate EmailPlaceHolder}"
                                   Keyboard="Email"
                                   Text="{Binding Email}" />
                        </inputLayout:SfTextInputLayout>

                        <inputLayout:SfTextInputLayout Hint="{i18n:Translate Password}"
                                                       EnablePasswordVisibilityToggle="true"
                                                       ContainerType="Outlined">
                            <Entry Placeholder="{i18n:Translate PasswordPlaceHolder}"
                                   IsPassword="True"
                                   Text="{Binding Password}" />
                        </inputLayout:SfTextInputLayout>
                    </StackLayout>

                    <Label FontAttributes="Bold"
                           HorizontalOptions="Center"
                           Text="{i18n:Translate ForgotPassword}"
                           TextColor="{StaticResource ColorAccent}"
                           VerticalOptions="CenterAndExpand">
                        <Label.GestureRecognizers>
                            <TapGestureRecognizer Command="{Binding ForgotPasswordCommand}"/>
                        </Label.GestureRecognizers>
                    </Label>

                </StackLayout>
            </ScrollView>
            <StackLayout VerticalOptions="EndAndExpand">
                <Button Command="{Binding LoginCommand}"
```

```
                IsEnabled="{Binding IsEnabled}"
                Text="{i18n:Translate Login}"/>
            <Button Command="{Binding RegisterCommand}"
                IsEnabled="{Binding IsEnabled}"
                Text="{i18n:Translate Register}"
                Style="{StaticResource DangerButton}"/>
        </StackLayout>
    </StackLayout>
    <busyindicator:SfBusyIndicator AnimationType="Gear"
                        AbsoluteLayout.LayoutBounds=".5,.5,.5,.5"
                            AbsoluteLayout.LayoutFlags="All"
                            BackgroundColor="{StaticResource ColorAccent}"
                            HorizontalOptions="Center"
                            TextColor="{StaticResource ColorFontInverse}"
                            IsBusy="{Binding IsRunning}"
                            Title="{i18n:Translate Loading}"
                            VerticalOptions="Center"
                            ViewBoxWidth="80"
                            ViewBoxHeight="80" />
  </AbsoluteLayout>

</ContentPage>
```

6. Modificamos la **LoginPageViewModel**:

```
public class LoginPageViewModel : ViewModelBase
{
  private bool _isRunning;
  private bool _isEnabled;
  private string _password;
  private DelegateCommand _loginCommand;
  private DelegateCommand _registerCommand;
  private DelegateCommand _forgotPasswordCommand;

  public LoginPageViewModel(INavigationService navigationService) : base(navigationService)
  {
    Title = Languages.Login;
    IsEnabled = true;
  }

  public DelegateCommand LoginCommand => _loginCommand ?? (_loginCommand = new
DelegateCommand(LoginAsync));

  public DelegateCommand RegisterCommand => _registerCommand ?? (_registerCommand
= new DelegateCommand(RegisterAsync));
```

```
    public DelegateCommand ForgotPasswordCommand => _forgotPasswordCommand ??
(_forgotPasswordCommand = new DelegateCommand(ForgotPasswordAsync));

    public bool IsRunning
    {
        get => _isRunning;
        set => SetProperty(ref _isRunning, value);
    }

    public bool IsEnabled
    {
        get => _isEnabled;
        set => SetProperty(ref _isEnabled, value);
    }

    public string Email { get; set; }

    public string Password
    {
        get => _password;
        set => SetProperty(ref _password, value);
    }

    private async void LoginAsync()
    {
        if (string.IsNullOrEmpty(Email))
        {
            await App.Current.MainPage.DisplayAlert(
                Languages.Error,
                Languages.EmailError,
                Languages.Accept);
            return;
        }

        if (string.IsNullOrEmpty(Password))
        {
            await App.Current.MainPage.DisplayAlert(
                Languages.Error,
                Languages.PasswordError,
                Languages.Accept);
            return;
        }
    }

    private void ForgotPasswordAsync()
    {
        //TODO: Pending
    }
```

```
    private void RegisterAsync()
    {
        //TODO: Pending
    }
}
```

7. Probamos las validaciones básicas. Luego continuamos con la lógica del login, como necesitamos almacenar el usuario logueado vamos a agregar un nuget para almacenar estos valores en persistencia. También vamos a necesitar el método para obtener el token.

8. Adicionamos la clase **TokenRequest** en **Common.Requets**:

```
public class TokenRequest
{
    public string Username { get; set; }

    public string Password { get; set; }
}
```

9. Adicionar la clase **UserResponse** en **Common.Responses**:

```
public class UserResponse
{
    public string Id { get; set; }

    public string Email { get; set; }

    public string PhoneNumber { get; set; }

    public string Document { get; set; }

    public string FirstName { get; set; }

    public string LastName { get; set; }

    public string Address { get; set; }

    public Guid ImageId { get; set; }

    public string ImageFullPath => ImageId == Guid.Empty
        ? $"https://onsalezulu.azurewebsites.net/images/noimage.png"
        : $"https://onsale.blob.core.windows.net/users/{ImageId}";

    public UserType UserType { get; set; }
```

```
    public City City { get; set; }

    public string FullName => $"{FirstName} {LastName}";

    public string FullNameWithDocument => $"{FirstName} {LastName} - {Document}";
}
```

10. Adicionamos la clase **TokenResponse** en **Common**

```
public class TokenResponse
{
    public string Token { get; set; }

    public UserResponse User { get; set; }

    public DateTime Expiration { get; set; }

    public DateTime ExpirationLocal => Expiration.ToLocalTime();
}
```

11. Adicionamos este método a la interfaz **IApiService**:

```
Task<Response> GetTokenAsync(string urlBase, string servicePrefix, string controller,
TokenRequest request);
```

12. Adicionamos la implementación en el **ApiService**:

```
public async Task<Response> GetTokenAsync(string urlBase, string servicePrefix, string
controller, TokenRequest request)
{
    try
    {
        string requestString = JsonConvert.SerializeObject(request);
        StringContent content = new StringContent(requestString, Encoding.UTF8,
"application/json");
        HttpClient client = new HttpClient
        {
            BaseAddress = new Uri(urlBase)
        };

        string url = $"{servicePrefix}{controller}";
        HttpResponseMessage response = await client.PostAsync(url, content);
        string result = await response.Content.ReadAsStringAsync();

        if (!response.IsSuccessStatusCode)
        {
            return new Response
```

```
            {
                IsSuccess = false,
                Message = result,
            };
        }

        TokenResponse token = JsonConvert.DeserializeObject<TokenResponse>(result);
        return new Response
        {
            IsSuccess = true,
            Result = token
        };
    }
    catch (Exception ex)
    {
        return new Response
        {
            IsSuccess = false,
            Message = ex.Message
        };
    }
}
```

13. Adicionar el NuGet **Xam.Plugins.Settings** en **Common**.

14. Adicionar la clase **Settings** en **Common.Helpers**:

```
public static class Settings
{
    private const string _token = "token";
    private const string _isLogin = "isLogin";
    private static readonly string _stringDefault = string.Empty;
    private static readonly bool _boolDefault = false;

    private static ISettings AppSettings => CrossSettings.Current;

    public static string Token
    {
        get => AppSettings.GetValueOrDefault(_token, _stringDefault);
        set => AppSettings.AddOrUpdateValue(_token, value);
    }

    public static bool IsLogin
    {
        get => AppSettings.GetValueOrDefault(_isLogin, _boolDefault);
```

```
        set => AppSettings.AddOrUpdateValue(_isLogin, value);
    }
}
```

15. En **Common** creamos la carpeta **Helpers** y dentro de esta la clase **Settings**:

```
public static class Settings
{
    private const string _token = "token";
    private const string _isLogin = "isLogin";
    private static readonly string _stringDefault = string.Empty;
    private static readonly bool _boolDefault = false;

    private static ISettings AppSettings => CrossSettings.Current;

    public static string Token
    {
        get => AppSettings.GetValueOrDefault(_token, _stringDefault);
        set => AppSettings.AddOrUpdateValue(_token, value);
    }

    public static bool IsLogin
    {
        get => AppSettings.GetValueOrDefault(_isLogin, _boolDefault);
        set => AppSettings.AddOrUpdateValue(_isLogin, value);
    }
}
```

16. Modificamos la **LoginPageViewModel**:

```
public class LoginPageViewModel : ViewModelBase
{
    private readonly INavigationService _navigationService;
    private readonly IApiService _apiService;
    private bool _isRunning;
    private bool _isEnabled;
    private string _password;
    private DelegateCommand _loginCommand;
    private DelegateCommand _registerCommand;
    private DelegateCommand _forgotPasswordCommand;

    public LoginPageViewModel(INavigationService navigationService, IApiService apiService)
        : base(navigationService)
    {
        _navigationService = navigationService;
        _apiService = apiService;
        Title = Languages.Login;
```

```
        IsEnabled = true;
    }

    public DelegateCommand LoginCommand => _loginCommand ?? (_loginCommand = new
DelegateCommand(LoginAsync));

    public DelegateCommand RegisterCommand => _registerCommand ?? (_registerCommand
= new DelegateCommand(RegisterAsync));

    public DelegateCommand ForgotPasswordCommand => _forgotPasswordCommand ??
(_forgotPasswordCommand = new DelegateCommand(ForgotPasswordAsync));

    public bool IsRunning
    {
        get => _isRunning;
        set => SetProperty(ref _isRunning, value);
    }

    public bool IsEnabled
    {
        get => _isEnabled;
        set => SetProperty(ref _isEnabled, value);
    }

    public string Email { get; set; }

    public string Password
    {
        get => _password;
        set => SetProperty(ref _password, value);
    }

    private async void LoginAsync()
    {
        if (string.IsNullOrEmpty(Email))
        {
            await App.Current.MainPage.DisplayAlert(
                Languages.Error,
                Languages.EmailError,
                Languages.Accept);
            return;
        }

        if (string.IsNullOrEmpty(Password))
        {
            await App.Current.MainPage.DisplayAlert(
                Languages.Error,
                Languages.PasswordError,
```

```
            Languages.Accept);
        return;
    }

    IsRunning = true;
    IsEnabled = false;

    if (Connectivity.NetworkAccess != NetworkAccess.Internet)
    {
        IsRunning = false;
        IsEnabled = true;
        await App.Current.MainPage.DisplayAlert(Languages.Error,
Languages.ConnectionError, Languages.Accept);
        return;
    }

    string url = App.Current.Resources["UrlAPI"].ToString();
    TokenRequest request = new TokenRequest
    {
        Password = Password,
        Username = Email
    };

    Response response = await _apiService.GetTokenAsync(url, "api",
"/Account/CreateToken", request);
    IsRunning = false;
    IsEnabled = true;

    if (!response.IsSuccess)
    {
        await App.Current.MainPage.DisplayAlert(Languages.Error, Languages.LoginError,
Languages.Accept);
        Password = string.Empty;
        return;
    }

    TokenResponse token = (TokenResponse)response.Result;
    Settings.Token = JsonConvert.SerializeObject(token);
    Settings.IsLogin = true;

    IsRunning = false;
    IsEnabled = true;

    await
_navigationService.NavigateAsync($"/{nameof(OnSaleMasterDetailPage)}/NavigationPage/{na
meof(ProductsPage)}");
    Password = string.Empty;
}
```

```
    private void ForgotPasswordAsync()
    {
        //TODO: Pending
    }

    private void RegisterAsync()
    {
        //TODO: Pending
    }
}
```

17. Modificamos la **OnSaleMasterDetailPage**:

```
<Image HeightRequest="150"
       Source="onsale"/>
<Label FontAttributes="Bold"
       FontSize="Large"
       Text="{Binding User.FullName}"/>
<ListView BackgroundColor="Transparent"
          ItemsSource="{Binding Menus}"
          HasUnevenRows="True"
          SeparatorVisibility="None">
```

18. Modificamos la **OnSaleMasterDetailPageViewModel**:

```
…
private UserResponse _user;
…
public OnSaleMasterDetailPageViewModel(INavigationService navigationService) :
base(navigationService)
{
    _navigationService = navigationService;
    LoadMenus();
    LoadUser();
}

public UserResponse User
{
    get => _user;
    set => SetProperty(ref _user, value);
}
…
private void LoadUser()
{
    if (Settings.IsLogin)
    {
```

```
        TokenResponse token =
JsonConvert.DeserializeObject<TokenResponse>(Settings.Token);
        User = token.User;
    }
}
…
new Menu
{
    Icon = "ic_exit_to_app",
    PageName = $"{nameof(LoginPage)}",
    Title = Settings.IsLogin ? Languages.Logout : Languages.Login
}
```

19. Modificamos la **MenuItemViewModel**:

```
private async void SelectMenuAsync()
{
    if (PageName == nameof(LoginPage) && Settings.IsLogin)
    {
        Settings.IsLogin = false;
        Settings.Token = null;
    }

    await
_navigationService.NavigateAsync($"/{nameof(OnSaleMasterDetailPage)}/NavigationPage/{Pa
geName}");
}
```

20. Probemos lo que llevamos.

21. Ahora mostremos la foto del usuario en el menú. Adicione el nuget **Xamarin.FFImageLoading.Transformations** en todos los proyectos Prism.

22. Modificamos la **OnSaleMasterDetailPage**:

```
xmlns:fftransformations="clr-namespace:FFImageLoading.Transformations;assembly=FFImage
Loading.Transformations"
...
<StackLayout Padding="20">
    <StackLayout Orientation="Horizontal">
        <Image HeightRequest="80"
               Source="onsale"/>
        <ffimageloading:CachedImage Aspect="AspectFill"
                                    Source="{Binding User.ImageFullPath}"
                                    CacheDuration= "50"
                                    HeightRequest="100"
                                    Margin="5"
```

```
                                RetryCount= "3"
                                RetryDelay= "600"
                                WidthRequest="100">
            <ffimageloading:CachedImage.Transformations>
                <fftransformations:CircleTransformation />
            </ffimageloading:CachedImage.Transformations>
        </ffimageloading:CachedImage>
    </StackLayout>
    <Label FontAttributes="Bold"
            FontSize="Large"
            Text="{Binding User.FullName}"/>
```

23. Probamos.