**Estructura de BInding Genérico**

**La MainViewModel controla todo o gobierna todas las clases ViewModel**

**1.-loginPage.xaml**

BindingContext="{Binding Main, Source={StaticResource Locator}}">

2.-App.xaml

<Application.Resources>

<ResourceDictionary>

<!--Locator-->

<infra:InstanceLocator x:Key="Locator">

</infra:InstanceLocator>

</ResourceDictionary>

</Application.Resources>

**2.1.-** xmlns:infra="clr-namespace:MecShop.UIForms.Infrastructure"

**3.-Infrastructure/InstanceLocator.cs**

public InstanceLocator()

{

this.Main = new MainViewModel();

}

**Carga de datos en un Listview**

**1)Se establece los Binding en el ProductPage**

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="Shop.UIForms.Views.ProductsPage"

BindingContext="{Binding Main,Source={StaticResource Locator}}"

Title="Products">

<ContentPage.Content>

<StackLayout BindingContext="{Binding Products}"

Padding="8">

<ListView

HasUnevenRows="True"

ItemsSource="{Binding Products}">

</ListView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

**2)Se crea la clase Genérica para el patrón SINGLETON BaseViewModel**

public class BaseViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

protected void SetValue<T>(ref T backingField, T value, [CallerMemberName] string propertyName = null)

{

if (EqualityComparer<T>.Default.Equals(backingField, value))

{

return;

}

backingField = value;

OnPropertyChanged(propertyName);

}

}

}

**3) Implementamos en el ProductViewModel la BaseViewModel, el ApiService**

public class ProductsViewModel: BaseViewModel

{

private readonly ApiService apiService;

private ObservableCollection<Product> products;

public ObservableCollection<Product> Products

{

get { return this.products; }

set { this.SetValue(ref this.products, value); }

}

public ProductsViewModel()

{

this.apiService = new ApiService();

this.LoadProducts();

}

private async void LoadProducts()

{

var response = await this.apiService.GetListAsync<Product>(

"https://shopdemomectoys.azurewebsites.net",

"/api",

"/Products");

if (!response.IsSuccess)

{

await Application.Current.MainPage.DisplayAlert(

"Error",

response.Message,

"Accept");

return;

}

var myProducts = (List<Product>)response.Result;

//no pinto lista sino observablecolletions

this.Products = new ObservableCollection<Product>(myProducts);

}

}

4) Para llamar el ProductPage y cargar la lista de datos se debe de instanciar de memoria la MainViewModel usando el patrón Singleton y luego realizar un llamado al ProductPage .

MainViewModel.GetInstance().Products = new ProductsViewModel();

await Application.Current.MainPage.Navigation.PushAsync(new ProductsPage());