FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

PROGRAMACIÓN IV

INFORME APP

Profesor

Carlos Andrés Guaita Ayala

Autores

Esteban Enríquez,Emilio Cabrear

Año

2023

1.De qué trata nuestro proyecto?

El proyecto esta basado en crear las bases para un sistema de administración de inventario para una empresa de artículos deportivos y en un futuro, poder desarrollar al 100 % este proyecto. Lo principal a presentar en este informe tiene que ver con el desarrollo de la APP y la implementación del manejo de inventario dentro de ella

2.Cómo lo haremos?

Se utilizará una API que será consumida por una aplicación basada en MAUI.NET el cual contará con diferentes vistas(xmls) para el personal encargado del inventario.

3.Clases/modelos a Implementar

Para esto hemos tomado en cuenta todos los actores que tendrán influencia dentro de la base de datos ya que estas serán las que se implementarán en todo el proceso

NOTA: ESTAS SERAN LAS PRINCIPALES CLASES A USAR PARA EL PROYECTO

Marca

```
namespace BochaStoreProyecto.Maui.Models
{

    public class Marca
    {

        public int idMarca { get; set; }

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

Se define los atributos que tendrá una marca que trabaja con la empresa

Producto

```
namespace BochaStoreProyecto.Maui.Models
{

    public class Producto
    {

        public int idProducto { get; set; }

        public int idProovedor { get; set; }

        public int idMarca { get; set; }

        public string nombreProducto { get; set; }

        public string descripcionProducto { get; set; }
        4 referencias
        public double precio { get; set; }
        4 referencias
        public int stock { get; set; }
        4 referencias
        public DateTime fechaCreacion { get; set; }

    }
}
```

Se define los atributos del producto el cual tendrá relación con un proveedor y una marca

Detalle de venta del producto

```
namespace BochaStoreProyecto.Maui.Models
{
    40 referencias
    public class Proovedor
    {
        3 referencias
        public int idProovedor { get; set; }

        4 referencias
        public string nombreProovedor { get; set; }

        4 referencias
        public int duracionContrato { get; set; }

        4 referencias
        public double precioImportacion { get; set; }
    }
}
```

Se define los atributos del proveedor el cual tendrá un contrato y se define con un precio de importación

Usuario

```
namespace BochaStoreProyecto.Maui.Models
{
    12 referencias
    public class Usuario
    {
        1 referencia
        public int idUsuario { get; set; }
        3 referencias
        public string username { get; set; }
        2 referencias
        public string password { get; set; }
    }
}
```

Se define los atributos de los usuarios que usaran el aplicativo móvil

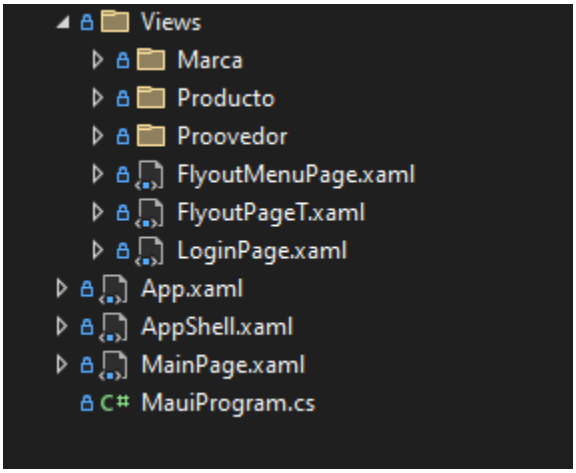
FlyoutPageItem

```
using System.Text;
using System.Threading.Tasks;

namespace BochaStoreProyecto.Maui.Models
{
    1 referencia
    public class FlyoutPageItem
    {
        0 referencias
        public string Title { get; set; }
        0 referencias
        public string IconSource { get; set; }
        1 referencia
        public Type TargetType { get; set; }
    }
}
```

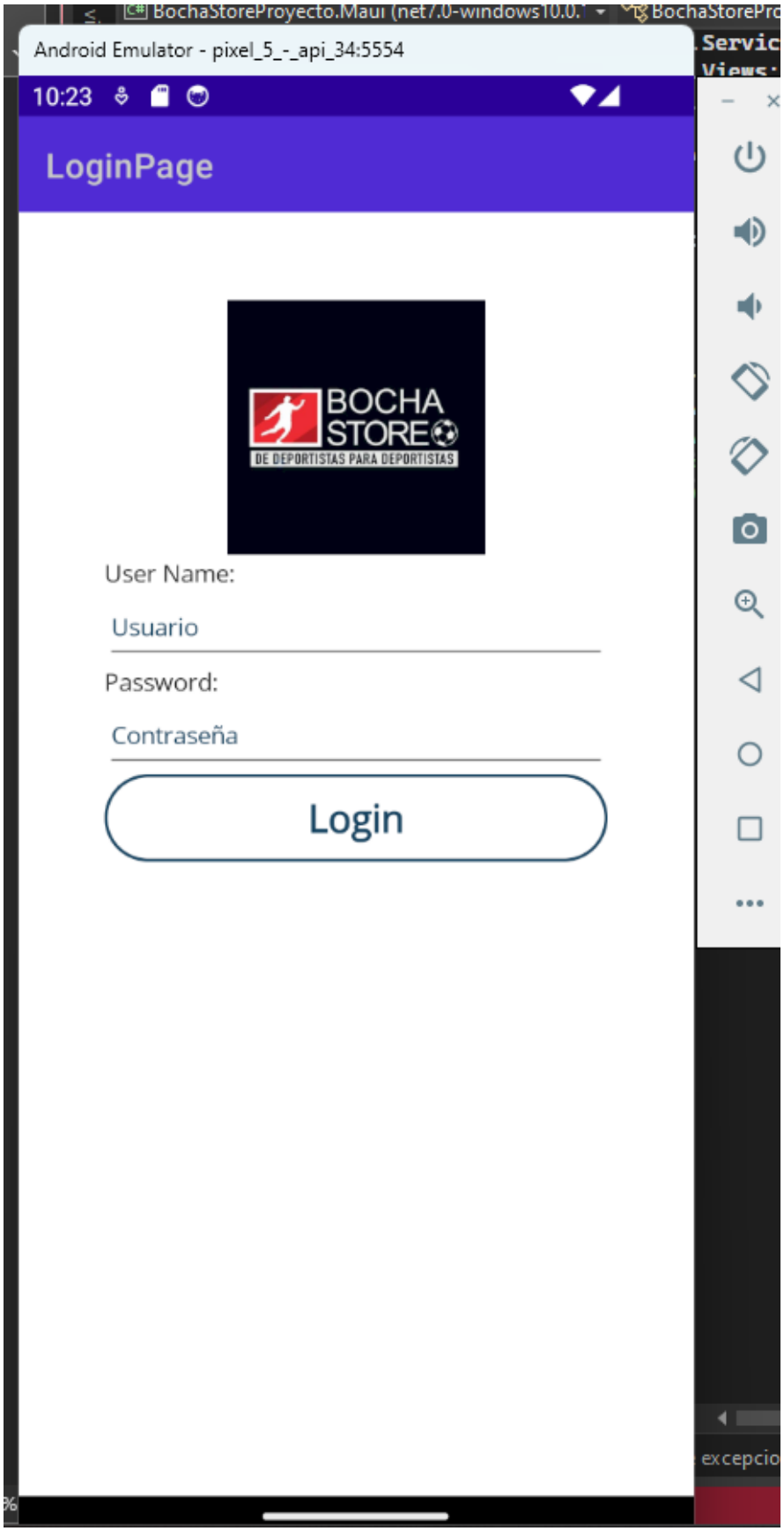
Esta clase solo se usará en el aplicativo movil, ya que ayudará a manejar de manera interactiva las diferentes vistas flotantes

4.Parte Gráfica

Dentro de la parte gráfica se implementará:



Login: Aquí los usuarios podrán loggearse para entrar a la página web


Android Emulator - pixel_5_-_api_34:5554
10:23
LoginPage
BOCHA STORE
DE DEPORTISTAS PARA DEPORTISTAS
User Name:
Usuario
Password:
Contraseña
Login

Parte XAML

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="BochaStoreProyecto.Maui.Views.LoginPage"
             Title="LoginPage">
    <StackLayout Margin="50" HorizontalOptions="Fill">
        <Image Source="logo_bochas.jpeg" WidthRequest="150" HeightRequest="150" />

        <Label Text="User Name:"></Label>

        <Entry x:Name="txtUserName"
               Placeholder="Usuario"
               TextColor="#000"
               PlaceholderColor="#154360"></Entry>

        <Label Text="Password: "></Label>

        <Entry x:Name="txtPassword"
               Placeholder="Contraseña"
               TextColor="#000"
               PlaceholderColor="#154360"></Entry>

        <Button x:Name="btnLogin"
                Text="Login"
                BackgroundColor="Transparent"
                TextColor="#154360"
                BorderColor="#154360"
                CornerRadius="50"
                BorderWidth="1.5"
                FontAttributes="Bold"
                FontSize="Large"
                Clicked="Login_Clicked"
                VerticalOptions="Center"></Button>
    </StackLayout>
</ContentPage>
```

Lo importante de esta parte es que se usa la imagen de la emprese con “Image” se da dos entrys para el username y contraseña y un botón para verificar el login

Parte XAML.cs

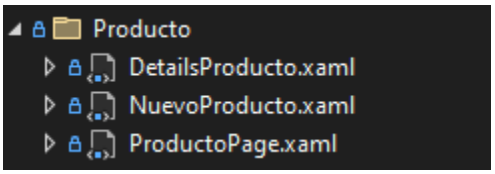
```
using BochaStoreProyecto.Maui.Models;
using BochaStoreProyecto.Maui.Services;
using BochaStoreProyecto.Maui.Views.Producto;
using CommunityToolkit.Maui.Alerts;

namespace BochaStoreProyecto.Maui.Views;

4 referencias
public partial class LoginPage : ContentPage
{
    private readonly APIService _APIService;
    0 referencias
    public LoginPage(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;
    }

    0 referencias
    private  async void Login_Clicked(object sender, EventArgs e)
    {
        string userName = txtUserName.Text;
        string password = txtPassword.Text;
        if(userName == null || password == null)
        {
            await DisplayAlert("Peligro", "Ingrese el usuario y la contraseña", "Ok");
            return;
        }
        Usuario usuario = await _APIService.Login(userName, password);
        if (usuario != null)
        {
            await Navigation.PushAsync(new NavigationPage(new FlyoutPageT(_APIService)));
        }
        else
        {
            await DisplayAlert("Warning", "Username or Password is incorrect", "Ok");
        }
    }
}
```

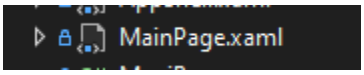
Se inicializa APIService para que pueda hacer uso de la Api y sus usuarios, el método Login_Clicked es el que verifica y da paso si el usuario existe, en esta parte se guarda el usuario y la contraseña el cual es verificado mediante

```
Usuario usuario = await _APIService.Login(userName, password);
```

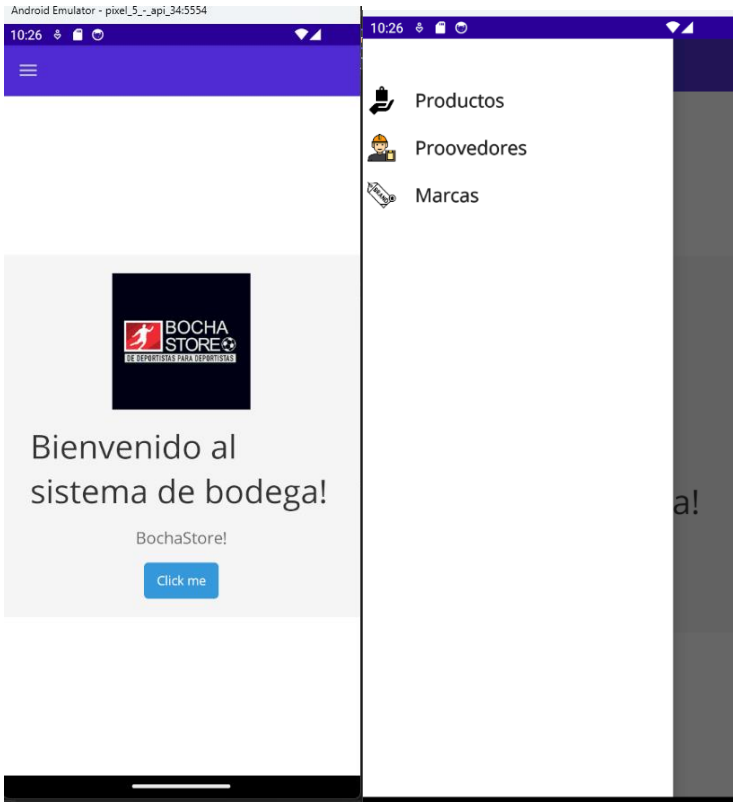
Esta entra en la API y verifica si el usuario existe y si existe se manda a la pagina main del LayOut,

A.PRODUCTOS





Main:Vista home donde los usuarios entran después de loggearse, donde se podrá interactuar con el LayOut



Parte XAML

```
x:Class="BochaStoreProyecto.Maui.MainPage">

<ScrollView>
    <VerticalStackLayout
        Spacing="15"
        Padding="30,20"
        BackgroundColor="#f5f5f5"
        VerticalOptions="Center">

        <Image
            Source="logo_bochas.jpeg"
            SemanticProperties.Description="Cute dot net bot waving hi to you!"
            HeightRequest="150"
            HorizontalOptions="Center">
            <Image.Margin>
                <OnPlatform x:TypeArguments="Thickness">
                    <On Platform="iOS" Value="0,0,0,20" />
                    <On Platform="Android" Value="0,0,0,0" />
                </OnPlatform>
            </Image.Margin>
        </Image>

        <Label
            Text="Bienvenido al sistema de bodega!"
            SemanticProperties.HeadingLevel="Level1"
            FontSize="36"
            TextColor="#333"
            HorizontalOptions="Center" />

        <Label
            Text="BochaStore!"
            SemanticProperties.HeadingLevel="Level2"
            SemanticProperties.Description="Welcome to dot net Multi-platform App UI"
            FontSize="18"
            TextColor="#666"
            HorizontalOptions="Center" />
```

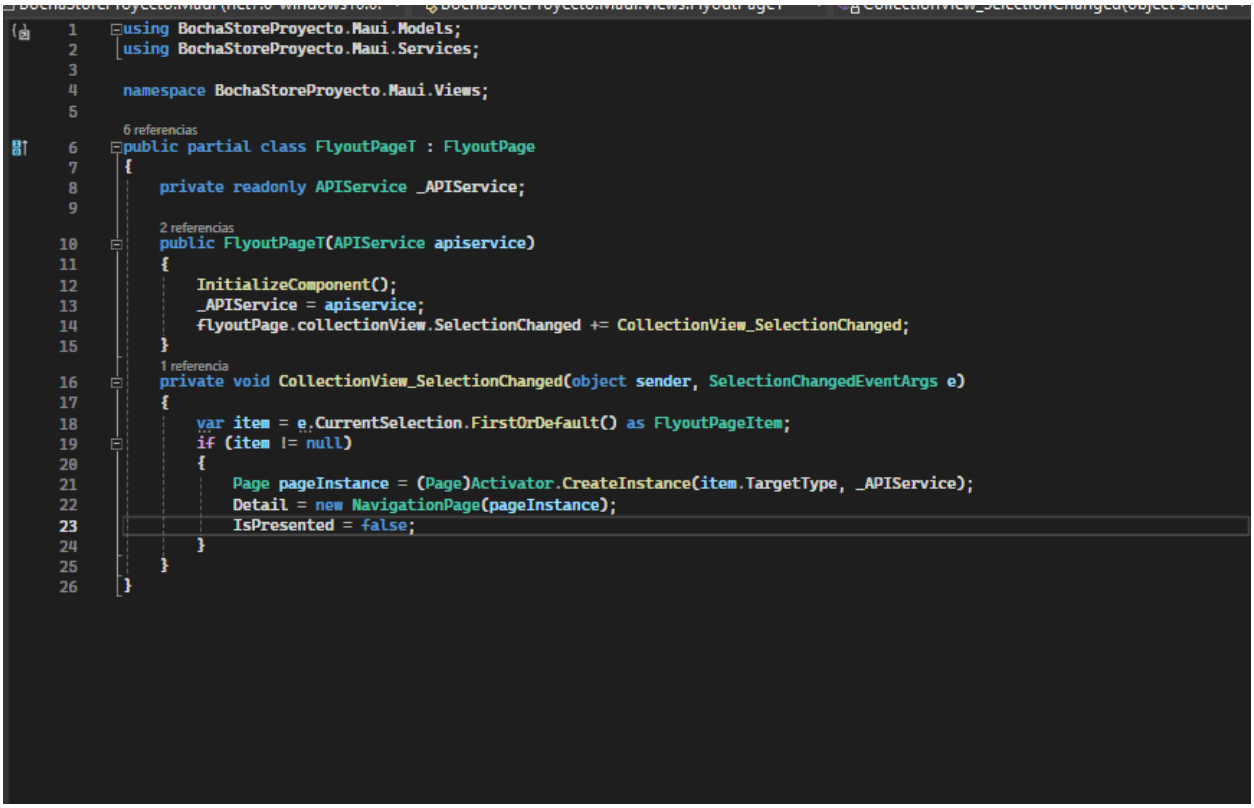
En resumen se muestra el logo de la empresa y una breve descripción del sistema de bodega

Flyout

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:BochaStoreProyecto.Maui.Views.Producto"
             xmlns:local1="clr-namespace:BochaStoreProyecto.Maui.Models"
             xmlns:local2="clr-namespace:BochaStoreProyecto.Maui.Views.Proovedor"
             xmlns:local3="clr-namespace:BochaStoreProyecto.Maui.Views.Marca"
             x:Class="BochaStoreProyecto.Maui.Views.FlyoutMenuPage"
             Padding="0,40,0,0"
             Title="FlyoutMenuPage">
    <CollectionView x:Name="collectionView"
                    x:FieldModifier="public"
                    SelectionMode="Single">
        <CollectionView.ItemsSource>
            <x:Array Type="{x:Type local1:FlyoutPageItem}">
                <local1:FlyoutPageItem Title="Productos"
                                       IconSource="icono_producto.png"
                                       TargetType="{x:Type local:ProductoPage}" />
                <local1:FlyoutPageItem Title="Proovedores"
                                       IconSource="proveedor_bocha.png"
                                       TargetType="{x:Type local2:ProovedorPage}" />
                <local1:FlyoutPageItem Title="Marcas"
                                       IconSource="marca_bocha.png"
                                       TargetType="{x:Type local3:MarcaPage}" />
            </x:Array>
        </CollectionView.ItemsSource>
        <CollectionView.ItemTemplate>
            <DataTemplate>
                <Grid Padding="5,10">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="30"/>
                        <ColumnDefinition Width="*" />
                    </Grid.ColumnDefinitions>
                    <Image Source="{Binding IconSource}" />
                    <Label Grid.Column="1"
                           Margin="20,0"
                           Text="{Binding Title}"
                           FontSize="20"
                           FontAttributes="Bold"
                           VerticalOptions="Center" />
                </Grid>
            </DataTemplate>
        </CollectionView.ItemTemplate>
    </CollectionView>
</ContentPage>
```

Aquí es la parte donde se muestra y redigirá a las paginas que sean seleccionadas en el flyout, son definidos como local1 local2 local3 ya que hace referencia a la ubicación en la que se encuentra cada pagina

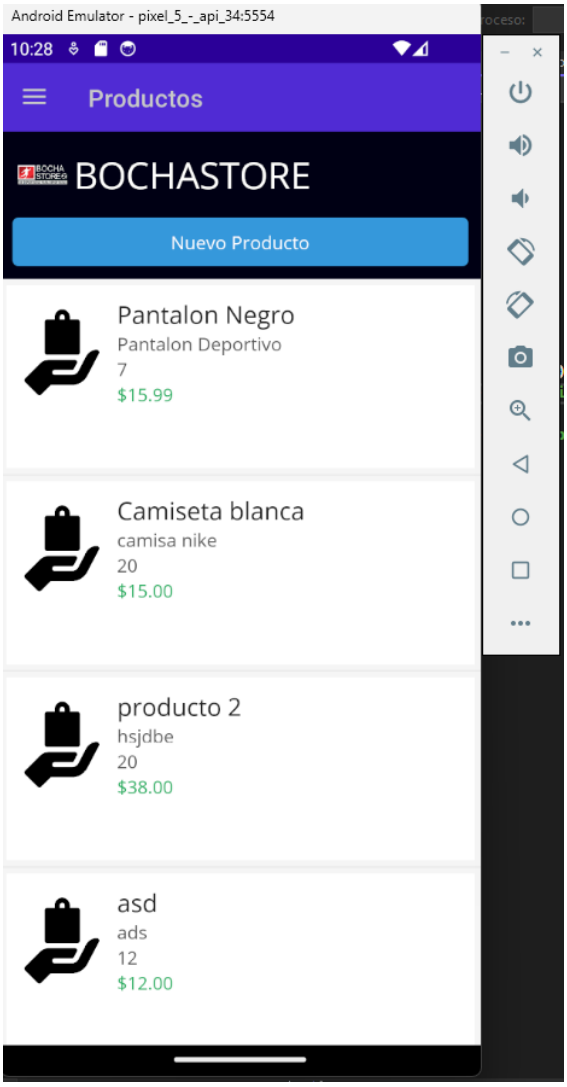
Parte XAML.CS



```
using BochaStoreProyecto.Maui.Models;
using BochaStoreProyecto.Maui.Services;

namespace BochaStoreProyecto.Maui.Views;

6 referencias
public partial class FlyoutPageT : FlyoutPage
{
    private readonly APIService _APIService;

    2 referencias
    public FlyoutPageT(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;
        flyoutPage.collectionView.SelectionChanged += CollectionView_SelectionChanged;
    }
    1 referencia
    private void CollectionView_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        var item = e.CurrentSelection.FirstOrDefault() as FlyoutPageItem;
        if (item != null)
        {
            Page pageInstance = (Page)Activator.CreateInstance(item.TargetType, _APIService);
            Detail = new NavigationPage(pageInstance);
            IsPresented = false;
        }
    }
}
```

Aquí es parte del flyout la cual se encarga de inicializar la API y pasarla dependiendo de la página seleccionada

ProductoPage:Esta es la parte gráfica que manejaran los de las bodega para poder agregar nuevos productos que han llegado o reducir el stock si es necesario



Aquí se podrá seleccionar el producto mediante un “itemSelected” el cual te llevará a la siguiente vista detella del producto(“DetailsProducto”) donde se podrá borrar u editar

Parte XAML

```xaml
<StackLayout BackgroundColor="#f5f5f5">
    <StackLayout BackgroundColor="#0e0f14" Padding="10" Spacing="10">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>

            <Image Source="logo_bochas.jpeg"
                   HeightRequest="50"
                   WidthRequest="50"
                   VerticalOptions="CenterAndExpand"/>
            <Label Grid.Column="1"
                   x:Name="Username"
                   Text="Productos"
                   FontSize="30"
                   TextColor="White"
                   VerticalOptions="CenterAndExpand"/>
        </Grid>
        <Button Text="Nuevo Producto"
                FontSize="15"
                BackgroundColor="#3498db"
                TextColor="White"
                Clicked="OnClickNuevoProducto"
                CornerRadius="5"
                HeightRequest="40"/>
    </StackLayout>

    <ListView x:Name="listaProductos"
              RowHeight="160"
              ItemSelected="OnClickShowDetails_ItemSelected">
        <ListView.ItemTemplate>
            <DataTemplate>
                <ViewCell>
                    <StackLayout Padding="10" BackgroundColor="White" Margin="5" >
                        <Grid>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="Auto"/>
                                <ColumnDefinition Width="*"/>
                            </Grid.ColumnDefinitions>

                            <Image Source="icono_producto.png"
                                   HeightRequest="70"
                                   WidthRequest="70"
                                   VerticalOptions="CenterAndExpand"/>
                            <StackLayout Grid.Column="1" Margin="10,0,0,0">
                                <Label Text="{Binding nombreProducto}" FontSize="20" />
                                <Label Text="{Binding descripcionProducto}" FontSize="15" TextColor="#555"/>
                                <Label Text="{Binding stock}" FontSize="15" TextColor="#555"/>
                                <Label Text="{Binding precio, StringFormat='${0:F2}'}" FontSize="15" TextColor="#27ae60"/>
                            </StackLayout>
                        </Grid>
                    </StackLayout>
                </ViewCell>
            </DataTemplate>
        </ListView.ItemTemplate>
    </ListView>
</StackLayout>
</ContentPage>

```

En resumen se presenta cada producto, con un Binding se busca las propiedades de los productos traídos y aparte con el uso de stringFormat se le brinda formato al precio para que tenga un “$” antes del precio, aparte se usa ItemSelected para dar funcionalidad si el usuario da un tap en el producto y asi te lleve a la pagina de detalle, también se define un Botón `<Button Text="Nuevo Producto"` el que llevará a a la pagina del nuevo producto.

Parte XAML.CS

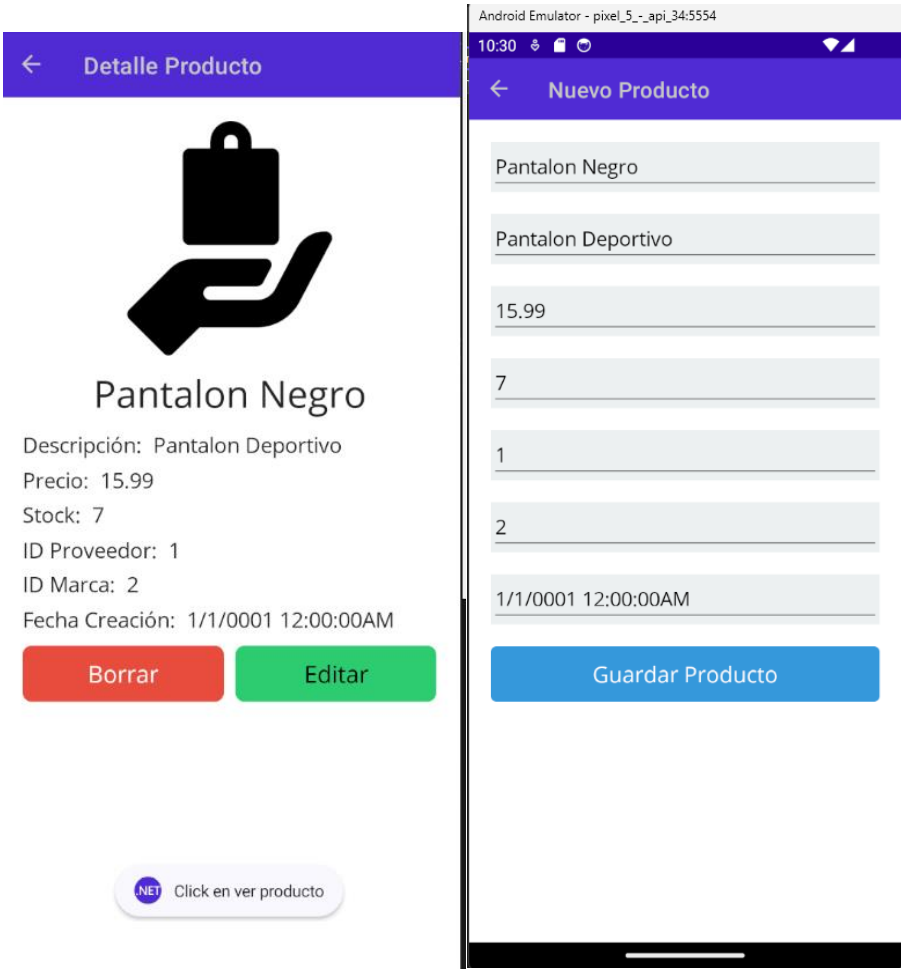
```
namespace BochaStoreProyecto.Maui.Views.Producto;

using CommunityToolkit.Maui.Alerts;
using CommunityToolkit.Maui.Core;
using BochaStoreProyecto.Maui.Services;
using System.Collections.ObjectModel;

using Producto = BochaStoreProyecto.Maui.Models.Producto;


public partial class ProductoPage : ContentPage
{
    ObservableCollection<Producto> products;
    private readonly APIService _APIService;

    public ProductoPage(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;
    }

    protected override async void OnAppearing()
    {
        base.OnAppearing();

        string username = "BOCHASTORE";
        Username.Text = username;
        List<Producto> ListaProducts = await _APIService.GetProductos();
        products = new ObservableCollection<Producto>(ListaProducts);
        listaProductos.ItemsSource = products;

    }

    0 referencias
    private async void OnClickNuevoProducto(object sender, EventArgs e)
    {
        var toast = Toast.Make("On Click Boton Nuevo Producto", ToastDuration.Short, 14);
        await toast.Show();
        await Navigation.PushAsync(new NuevoProducto(_APIService));

        //await Navigation.PushAsync(new NuevoProducto());
    }


    private async void OnClickShowDetails_ItemSelected(object sender, SelectedItemChangedEventArgs e)
    {
        var toast = CommunityToolkit.Maui.Alerts.Toast.Make("Click en ver producto", ToastDuration.Short, 14);

        await toast.Show();
        Producto producto = e.SelectedItem as Producto;
        await Navigation.PushAsync(new DetailsProducto(_APIService)
        {
            BindingContext = producto,
        });
    }
}
```

Se inicializa la API par apoder usarla, también se define que vamos a usar del producto, ya que tenemos una ambigüedad de nombres, en este caso se usara el producto de “models”

En el método OnAppearing lo que hace es traer la lista de productos , después se crea una lista de productos que es muy importante ser obvserbable para que cuando se haga algún cambio se muestre en la página.

En el método `OnClickNuevoProducto` lo que se hace es llevar a la pagina “NevoPrducto” y se le pasa el API inicializada

En el método `OnClickShowDetails_ItemSelected` , lo que se hace es que envia el producto con el binding a la pagina de detalle para asi obtener los datos



En la vista de editar se reutiliza la vista de “Nuevo Producto” ya que con el controlador que se explicará en el código xmls.cs que ayuda a saber si se debe editar el producto o crear uno nuevo.

Parte XAML

```
            Title="Detalle Producto">
    <StackLayout Padding="20" Spacing="10">
        <Image Source="icono_producto.png"
               HorizontalOptions="Center"
               HeightRequest="200"
               Aspect="AspectFit"/>

        <Label x:Name="Nombre"
               Text="Nombre"
               FontSize="32"
               HorizontalOptions="Center"/>

        <StackLayout Spacing="5">
            <StackLayout Orientation="Horizontal" Spacing="5">
                <Label Text="Descripción: " FontSize="18" VerticalOptions="Center"/>
                <Label x:Name="Descripcion" Text="Descripcion" FontSize="18" VerticalOptions="Center"/>
            </StackLayout>

            <StackLayout Orientation="Horizontal" Spacing="5">
                <Label Text="Precio: " FontSize="18" VerticalOptions="Center"/>
                <Label x:Name="Precio" Text="Precio" FontSize="18" VerticalOptions="Center"/>
            </StackLayout>

            <StackLayout Orientation="Horizontal" Spacing="5">
                <Label Text="Stock: " FontSize="18" VerticalOptions="Center"/>
                <Label x:Name="Stock" Text="Stock" FontSize="18" VerticalOptions="Center"/>
            </StackLayout>

            <StackLayout Orientation="Horizontal" Spacing="5">
                <Label Text="ID Proveedor: " FontSize="18" VerticalOptions="Center"/>
                <Label x:Name="idProveedor" Text="ID Proveedor" FontSize="18" VerticalOptions="Center"/>
            </StackLayout>

            <StackLayout Orientation="Horizontal" Spacing="5">
                <Label Text="ID Marca: " FontSize="18" VerticalOptions="Center"/>
                <Label x:Name="idMarca" Text="ID Marca" FontSize="18" VerticalOptions="Center"/>
            </StackLayout>

            <StackLayout Orientation="Horizontal" Spacing="5">
                <Label Text="Fecha Creación: " FontSize="18" VerticalOptions="Center"/>
                <Label x:Name="fechaCreacion" Text="Fecha Creación" FontSize="18" VerticalOptions="Center"/>
            </StackLayout>
        </StackLayout>

        <StackLayout Orientation="Horizontal" Spacing="10">
            <Button x:Name="Borrar"
                    Text="Borrar"
                    Clicked="Borrar_Clicked"
                    FontSize="20"
                    HorizontalOptions="FillAndExpand"
                    BackgroundColor="#e74c3c"/>
            <Button x:Name="Editar"
                    Text="Editar"
                    Clicked="Editar_Clicked"
                    FontSize="20"
                    HorizontalOptions="FillAndExpand"
                    TextColor="Black"
                    BackgroundColor="#2ecc71"/>
        </StackLayout>
    </StackLayout>
</ContentPage>
```

En resumen se presenta una imagen del producto, se tiene dos botones en color rojo(borrar) y verde(editar) y se da nombre a los labels que contendrán la información

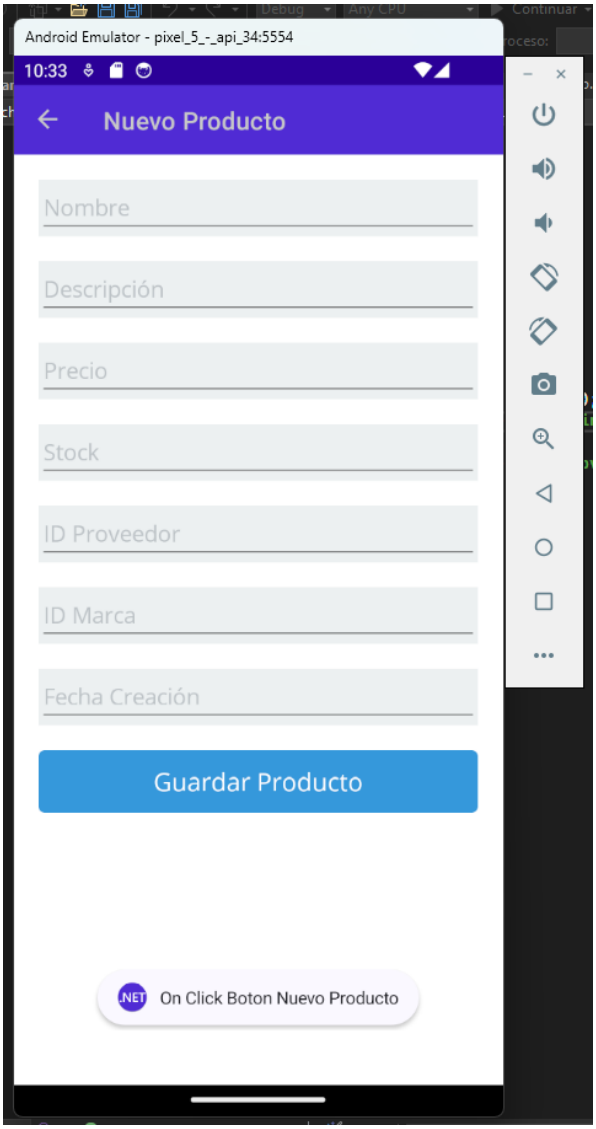
Parte XAML.CS

```
5 referencias
public partial class DetailsProducto : ContentPage
{
    private Producto _producto;
    private APIService _APIService;

    1 referencia
    public DetailsProducto(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;

    }
    0 referencias
    protected override void OnAppearing()
    {
        base.OnAppearing();
        _producto = BindingContext as Producto;
        Nombre.Text = _producto.nombreProducto;
        Descripcion.Text = _producto.descripcionProducto;
        Precio.Text = _producto.precio.ToString();
        Stock.Text = _producto.stock.ToString();
        idMarca.Text = _producto.idMarca.ToString();
        idProveedor.Text = _producto.idProovedor.ToString();
        fechaCreacion.Text = _producto.fechaCreacion.ToString();
    }
    0 referencias
    private async void Borrar_Clicked(object sender, EventArgs e)
    {
        await _APIService.DeleteProducto(_producto.idProducto);
        await Navigation.PopAsync();

    }

    0 referencias
    private async void Editar_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new NuevoProducto(_APIService)
        {
            BindingContext = _producto,
        });
    }
}
```

EL método OnAppering con ayuda del binding trae el producto y setea en los labels que son llamados por el nombre y se pone respectivamente las características
El método Borrar, llama a la api y usa el borrar para eliminar el producto de la BD

EL método editar manda el producto que se tiene a la página de `NuevoProducto`

Nuevo Producto:Esta es la vista que ayuda a crear un nuevo producto



Mediante Peaceholder se le indica que datos se deben llenar para crear un nuevo producto y para al final poder guardarlo dentro de la base de datos de Azure

Parte XAML

```
            Title="Nuevo Producto">
    <StackLayout Padding="20" Spacing="10">

        <Entry Placeholder="Nombre"
               x:Name="EntryNombre"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Entry Placeholder="Descripción"
               x:Name="EntryDescripcion"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Entry Placeholder="Precio"
               x:Name="EntryPrecio"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Entry Placeholder="Stock"
               x:Name="Entrystock"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Entry Placeholder="ID Proveedor"
               x:Name="EntryidProovedor"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Entry Placeholder="ID Marca"
               x:Name="EntryidMarca"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Entry Placeholder="Fecha Creación"
               x:Name="EntryfechaCreacion"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Button Text="Guardar Producto"
                FontSize="20"
                BackgroundColor="#3498db"
                TextColor="White"
                Clicked="OnClickGuardarNuevoProducto"
                CornerRadius="5"
                HeightRequest="50"/>
    </StackLayout>
</ContentPage>
```

Se tiene entrys que con placeholder se da una descripción de lo que se debe poner, además tiene un botón guardar

Parte XAML.CS

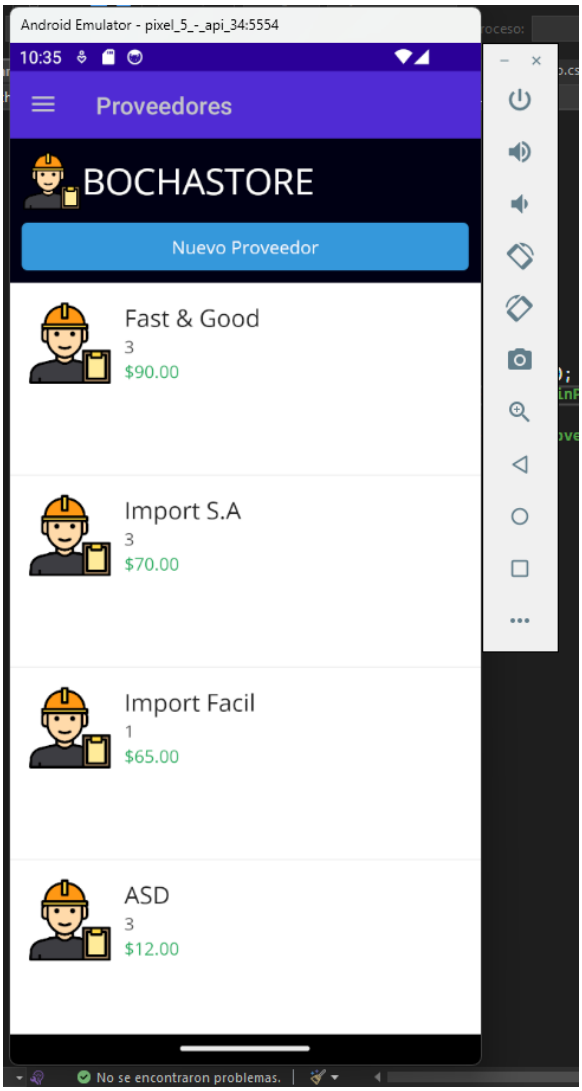
```
{
    private Producto _producto;
    private readonly APIService _APIService;
    2 referencias
    public NuevoProducto(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;

    }
    0 referencias
    protected override void OnAppearing()
    {
        base.OnAppearing();
        _producto = BindingContext as Producto;
        if (_producto != null)
        {
            EntryNombre.Text = _producto.nombreProducto;
            EntryDescripcion.Text = _producto.descripcionProducto;
            EntryPrecio.Text = _producto.precio.ToString();
            Entrystock.Text = _producto.stock.ToString();
            EntryidProovedor.Text = _producto.idProovedor.ToString();
            EntryidMarca.Text = _producto.idMarca.ToString();
            EntryfechaCreacion.Text = _producto.fechaCreacion.ToString();

        }
    }
    0 referencias
    private async void OnClickGuardarNuevoProducto(object sender, EventArgs e)
    {

        if (_producto != null)
        {

            _producto.nombreProducto = EntryNombre.Text;
            _producto.descripcionProducto = EntryDescripcion.Text;
            _producto.precio = double.Parse(EntryPrecio.Text);
            _producto.stock = Int32.Parse(Entrystock.Text);
            _producto.idProovedor = Int32.Parse(EntryidProovedor.Text);
            _producto.idMarca = Int32.Parse(EntryidMarca.Text);
            _producto.fechaCreacion = DateTime.Now;
            await _APIService.PutProducto(_producto.idProducto, _producto);
        }
        else
        {
            int id = Utils.Utils.ProductosList.Count + 1;

            Producto producto = new Producto
            {
                idProducto = 0,
                nombreProducto = EntryNombre.Text,
                descripcionProducto = EntryDescripcion.Text,
                precio = double.Parse(EntryPrecio.Text),
                stock = Int32.Parse(Entrystock.Text),
                idProovedor= Int32.Parse(EntryidProovedor.Text),
                idMarca= Int32.Parse(EntryidMarca.Text),
                fechaCreacion = DateTime.Now
            };
            //Utils.Utils.ProductosList.Add(producto);
            await _APIService.PostProducto(producto);
        }
        await Navigation.PopAsync();
    }
```

En el método OnAppering se valida si el producto no es traído por el método visto en la página de detalles, si este no es nulo entonces se setea directamente los atributos del producto en cada entry

En el método `OnClickGuardarNuevoProducto lo que hace es vlaidar si no es null para saber si tiene que editar o crear un nuevo producto.`

B.PROOVEDORES

ProovedorPage



Esta parte se muetra todos los proveedores creados y con la ayuda de un itemTapped te llevará a la vista “DetailsProovedor” donde se muestra todos los detalles que tiene y opciones de borrar u editar

Parte XAML

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="BochaStoreProyecto.Maui.Views.Proovedor.ProovedorPage"
             Title="Proveedores">
    <StackLayout>
        <StackLayout BackgroundColor="#0e0f14" Padding="10" Spacing="10">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <Image Source="proveedor_bocha.png"
                       HeightRequest="50"
                       WidthRequest="50"
                       VerticalOptions="CenterAndExpand"/>
                <Label Grid.Column="1"
                       x:Name="Username"
                       Text="Proveedores"
                       FontSize="30"
                       TextColor="White"
                       VerticalOptions="CenterAndExpand"/>
            </Grid>
            <Button Text="Nuevo Proveedor"
                    FontSize="15"
                    BackgroundColor="#3498db"
                    TextColor="White"
                    Clicked="OnClickNuevoProveedor"
                    CornerRadius="5"
                    HeightRequest="40"/>
        </StackLayout>

        <ListView x:Name="listaProveedores"
                  RowHeight="160"
                  ItemSelected="OnClickShowDetails_ItemSelected">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout Padding="10" BackgroundColor="White" Margin="5" >
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="Auto"/>
                                    <ColumnDefinition Width="*"/>
                                </Grid.ColumnDefinitions>

                                <Image Source="proveedor_bocha.png"
                                       HeightRequest="70"
                                       WidthRequest="70"
                                       VerticalOptions="CenterAndExpand"/>
                                <StackLayout Grid.Column="1" Margin="10,0,0,0">
                                    <Label Text="{Binding nombreProovedor}" FontSize="20" />
                                    <Label Text="{Binding duracionContrato}" FontSize="15" TextColor="#555"/>
                                    <Label Text="{Binding precioImportacion, StringFormat='${0:F2}'}" FontSize="15" TextColor="#27ae60"/>
                                </StackLayout>
                            </Grid>
                        </StackLayout>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

En resumen se presenta cada producto, con un Binding se busca las propiedades de los proveedores traídos y aparte con el uso de stringFormat se le brinda formato al precio para que tenga un “$” antes del precio de las importaciones, aparte se usa ItemSelected para dar funcionalidad si el usuario da un tap en el producto y asi te lleve a la pagina de detalle, también se define un Botón `<Button Text="Nuevo Proveedor"` el que llevará a la pagina del nuevo proovedor.

Parte XAML.CS

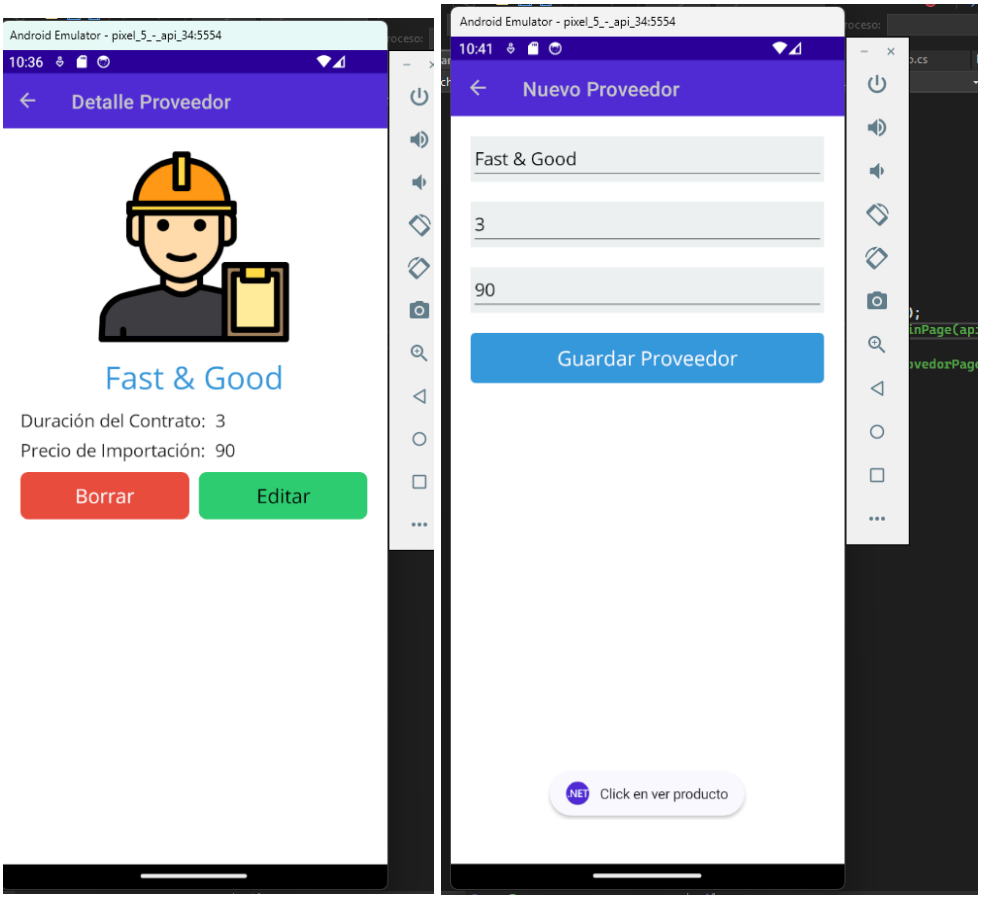
```
using BochaStoreProyecto.Maui.Services;
using CommunityToolkit.Maui.Alerts;
using CommunityToolkit.Maui.Core;
using System.Collections.ObjectModel;
using Proovedor = BochaStoreProyecto.Maui.Models.Proovedor;

4 referencias
public partial class ProovedorPage : ContentPage
{
    ObservableCollection<Proovedor> proovedores;
    private readonly APIService _APIService;

    0 referencias
    public ProovedorPage(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;
    }
    0 referencias
    protected override async void OnAppearing()
    {
        base.OnAppearing();

        string username = "BOCHASTORE";
        Username.Text = username;
        List<Proovedor> ListaProovedores = await _APIService.GetProovedor();
        proovedores = new ObservableCollection<Proovedor>(ListaProovedores);
        listaProveedores.ItemsSource = proovedores;

    }

    0 referencias
    private async void OnClickNuevoProveedor(object sender, EventArgs e)
    {
        var toast = Toast.Make("On Click Boton Nuevo Producto", ToastDuration.Short, 14);
        await toast.Show();
        await Navigation.PushAsync(new NuevoProovedor(_APIService));
    }

    0 referencias
    private async void OnClickShowDetails_ItemSelected(object sender, SelectedItemChangedEventArgs e)
    {
        var toast = CommunityToolkit.Maui.Alerts.Toast.Make("Click en ver producto", ToastDuration.Short, 14);

        await toast.Show();
        Proovedor producto = e.SelectedItem as Proovedor;
        await Navigation.PushAsync(new DetailsProovedor(_APIService)
        {
            BindingContext = producto,
        });
    }
}
```

Se inicializa la API par apoder usarla, también se define que vamos a usar del proveedor, ya que tenemos una ambigüedad de nombres, en este caso se usara el proovedor de “models”

En el método OnAppearing lo que hace es traer la lista de proveedores , después se crea una lista de proveedores que es muy importante ser obvserbable para que cuando se haga algún cambio se muestre en la página.

En el método `OnClickNuevoProvedor` lo que se hace es llevar a la pagina “NevoProveedor” y se le pasa el API inicializada

En el método `OnClickShowDetails_ItemSelected` , lo que se hace es que envia el proveedor con el binding a la pagina de detalle para asi obtener los datos



En la vista de editar se reutiliza la vista de “Nuevo Proovedor” ya que con el controlador que se explicará en las siguientes paginas ayuda a saber si se debe editar el proovedor o crear uno nuevo.

Parte XAML

```xaml
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="BochaStoreProyecto.Maui.Views.Proovedor.DetailsProovedor"
             Title="Detalle Proveedor">
    <StackLayout Padding="20" Spacing="10">
        <Image Source="proveedor_bocha.png"
               HorizontalOptions="Center"
               HeightRequest="200"
               Aspect="AspectFit"/>

        <Label x:Name="Nombre"
               Text="Nombre"
               FontSize="32"
               TextColor="#3498db"
               HorizontalOptions="Center"/>

        <StackLayout Spacing="5">
            <StackLayout Orientation="Horizontal" Spacing="5">
                <Label Text="Duración del Contrato: " FontSize="18" VerticalOptions="Center"/>
                <Label x:Name="duracionContrato" Text="Duración del Contrato" FontSize="18" VerticalOptions="Center"/>
            </StackLayout>

            <StackLayout Orientation="Horizontal" Spacing="5">
                <Label Text="Precio de Importación: " FontSize="18" VerticalOptions="Center"/>
                <Label x:Name="PrecioImportacion" Text="Precio de Importación" FontSize="18" VerticalOptions="Center"/>
            </StackLayout>
        </StackLayout>

        <StackLayout Orientation="Horizontal" Spacing="10">
            <Button x:Name="Borrar"
                    Text="Borrar"
                    Clicked="Borrar_Clicked"
                    FontSize="20"
                    HorizontalOptions="FillAndExpand"
                    BackgroundColor="#e74c3c"/>
            <Button x:Name="Editar"
                    Text="Editar"
                    Clicked="Editar_Clicked"
                    FontSize="20"
                    HorizontalOptions="FillAndExpand"
                    TextColor="Black"
                    BackgroundColor="#2ecc71"/>
        </StackLayout>
    </StackLayout>
</ContentPage>
```

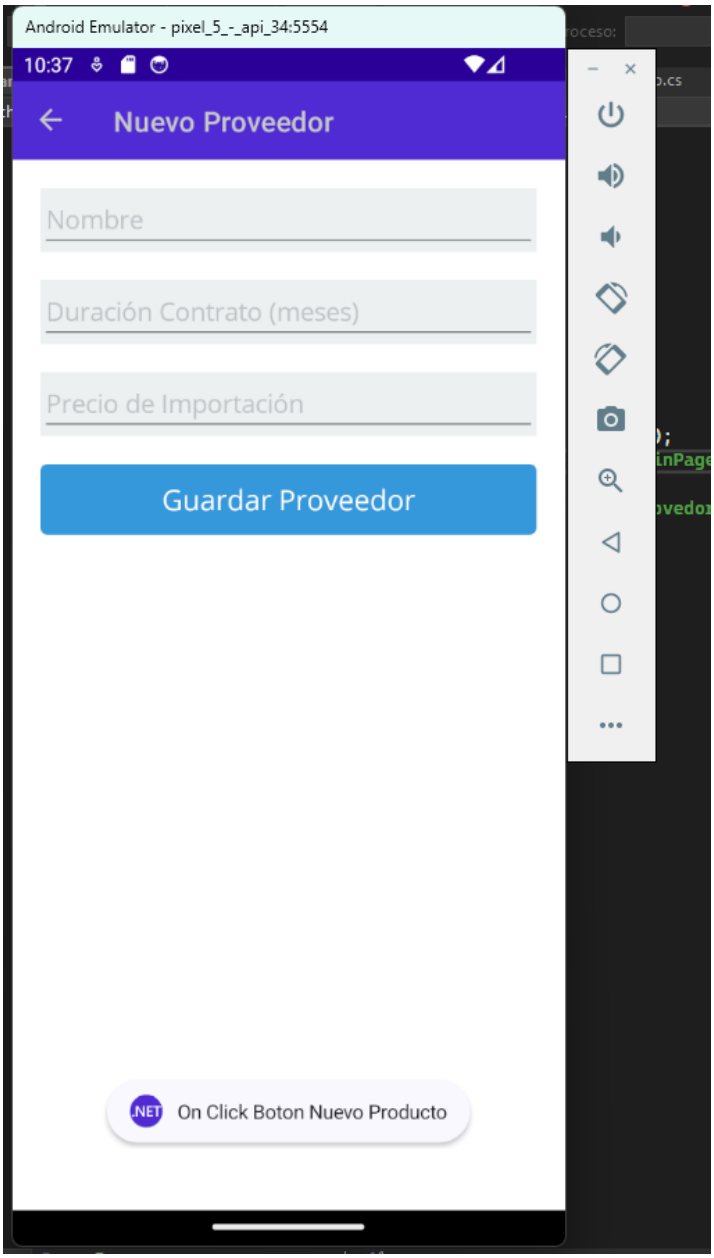
Parte XAML.CS

```
namespace BochaStoreProyecto.Maui.Views.Proovedor;

using CommunityToolkit.Maui.Core;
using BochaStoreProyecto.Maui.Services;
using Proovedor = BochaStoreProyecto.Maui.Models.Proovedor;


public partial class DetailsProovedor : ContentPage
{
    private Proovedor _proovedor;
    private APIService _APIService;

    public DetailsProovedor(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;

    }

    protected override void OnAppearing()
    {
        base.OnAppearing();
        _proovedor = BindingContext as Proovedor;
        Nombre.Text = _proovedor.nombreProovedor;
        PrecioImportacion.Text = _proovedor.precioImportacion.ToString();
        duracionContrato.Text = _proovedor.duracionContrato.ToString();
    }


    private async void Borrar_Clicked(object sender, EventArgs e)
    {
        await _APIService.DeleteProovedor(_proovedor.idProovedor);
        await Navigation.PopAsync();
    }


    private async void Editar_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new NuevoProovedor(_APIService)
        {
            BindingContext = _proovedor,
        });
    }
}
```

EL método OnAppering con ayuda del binding trae el proveedor y setea en los labels que son llamados por el nombre y se pone respectivamente las características
El método Borrar, llama a la api y usa el borrar para eliminar el proveedor de la BD

EL método editar manda el proveedor que se tiene a la página de `NuevoProducto`

NuevProovedor:Esta es la vista que ayuda a crear un nuevo proovedor



Mediante Peaceholder se le indica que datos se deben llenar para crear un nuevo proveedor y para al final poder guardarlo dentro de la base de datos de Azure

Parte XAML

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="BochaStoreProyecto.Maui.Views.Proovedor.NuevoProovedor"
             Title="Nuevo Proveedor">
    <StackLayout Padding="20" Spacing="10">

        <Entry Placeholder="Nombre"
               x:Name="EntryNombre"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Entry Placeholder="Duración Contrato (meses)"
               x:Name="EntryDuracionContrato"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"
               Keyboard="Numeric"/>

        <Entry Placeholder="Precio de Importación"
               x:Name="EntryprecioImportacion"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"
               Keyboard="Numeric"/>

        <Button Text="Guardar Proveedor"
                FontSize="20"
                BackgroundColor="#3498db"
                TextColor="White"
                Clicked="OnClickGuardarNuevoProducto"
                CornerRadius="5"
                HeightRequest="50"/>
    </StackLayout>
</ContentPage>

```

Se tiene entrys que con placeholder se da una descripción de lo que se debe poner, además tiene un botón guardar

Parte XAML.CS

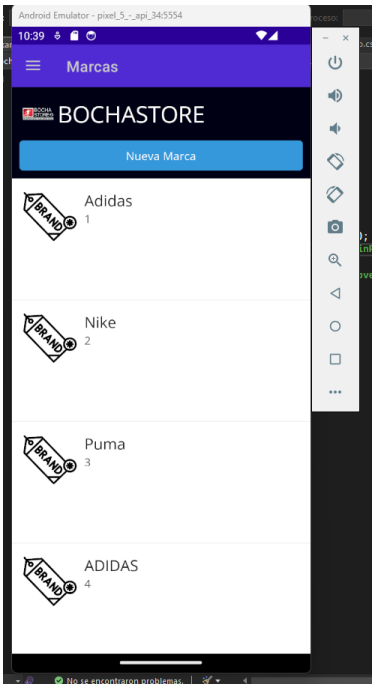
```
using BochaStoreProyecto.Maui.Services;
using Proovedor = BochaStoreProyecto.Maui.Models.Proovedor  ;

6 referencias
public partial class NuevoProovedor : ContentPage
{
    private Proovedor _proovedor;
    private readonly APIService _APIService;

    2 referencias
    public NuevoProovedor(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;

    }
    0 referencias
    protected override void OnAppearing()
    {
        base.OnAppearing();
        _proovedor = BindingContext as Proovedor;
        if (_proovedor != null)
        {
            EntryNombre.Text = _proovedor.nombreProovedor;
            EntryprecioImportacion.Text = _proovedor.precioImportacion.ToString();
            EntryDuracionContrato.Text = _proovedor.duracionContrato.ToString();

        }
    }

    0 referencias
    private async void OnClickGuardarNuevoProducto(object sender, EventArgs e)
    {
        if (_proovedor != null)
        {

            _proovedor.nombreProovedor = EntryNombre.Text;
            _proovedor.duracionContrato = Int32.Parse(EntryDuracionContrato.Text);
            _proovedor.precioImportacion = double.Parse(EntryprecioImportacion.Text);

            await _APIService.PutProovedor(_proovedor.idProovedor, _proovedor);

        }
        else
        {
            int id = Utils.Utils.ProovedoresList.Count + 1;

            Proovedor proovedor = new Proovedor
            {
                idProovedor = 0,//posible error
                nombreProovedor = EntryNombre.Text,
                duracionContrato = Int32.Parse(EntryDuracionContrato.Text),
                precioImportacion = double.Parse(EntryprecioImportacion.Text),
            };
            //Utils.Utils.ProductosList.Add(producto);
            await _APIService.PostProovedor(proovedor);
        }
        await Navigation.PopAsync();
    }
}
```

En el método OnAppering se valida si el producto no es traído por el método visto en la página de detalles, si este no es nulo entonces se setea directamente los atributos del producto en cada entry

En el método `OnClickGuardarNuevoProducto lo que hace es vlaidar si no es null para saber si tiene que editar o crear un nuevo producto.`

B.MARCAS

MarcaPage



Esta parte se muetra todos las marcas creadas y con la ayuda de un itemTapped te llevará a la vista “DetailsMarca” donde se muestra todos los detalles que tiene y opciones de borrar u editar

Parte XAML

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="BochaStoreProyecto.Maui.Views.Marca.MarcaPage"
             Title="Marcas">
    <StackLayout>
        <StackLayout BackgroundColor="#0e0f14" Padding="10" Spacing="10">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <Image Source="logo_bochas.jpeg"
                       HeightRequest="50"
                       WidthRequest="50"
                       VerticalOptions="CenterAndExpand"/>
                <Label Grid.Column="1"
                       x:Name="Username"
                       Text="Productos"
                       FontSize="30"
                       TextColor="White"
                       VerticalOptions="CenterAndExpand"/>
            </Grid>
            <Button Text="Nueva Marca"
                    FontSize="15"
                    BackgroundColor="#3498db"
                    TextColor="White"
                    Clicked="OnClickNuevaMarca"
                    CornerRadius="5"
                    HeightRequest="40"/>
        </StackLayout>

        <ListView x:Name="listaMarcas"
                  RowHeight="160"
                  ItemSelected="listaMarcas_ItemSelected">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout Padding="10" BackgroundColor="White" Margin="5" >
                            <Grid>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="Auto"/>
                                    <ColumnDefinition Width="*"/>
                                </Grid.ColumnDefinitions>

                                <Image Source="marca_bocha.png"
                                       HeightRequest="70"
                                       WidthRequest="70"
                                       VerticalOptions="CenterAndExpand"/>
                                <StackLayout Grid.Column="1" Margin="10,0,0,0">
                                    <Label Text="{Binding nombreMarca}" FontSize="20"/>
                                    <Label Text="{Binding idMarca}" FontSize="15" TextColor="#555"/>
                                </StackLayout>
                            </Grid>
                        </StackLayout>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
</ContentPage>
```

En resumen se presenta cada marca, con un Binding se busca las propiedades de las marcas traídos y aparte se usa ItemSelected para dar funcionalidad si el usuario da un tap en el producto y asi te lleve a la pagina de detalle, también se define un Botón `<Button Text="Nuevo Marca " ` el que llevará a a la pagina del nuevo producto.

Parte XAML.CS

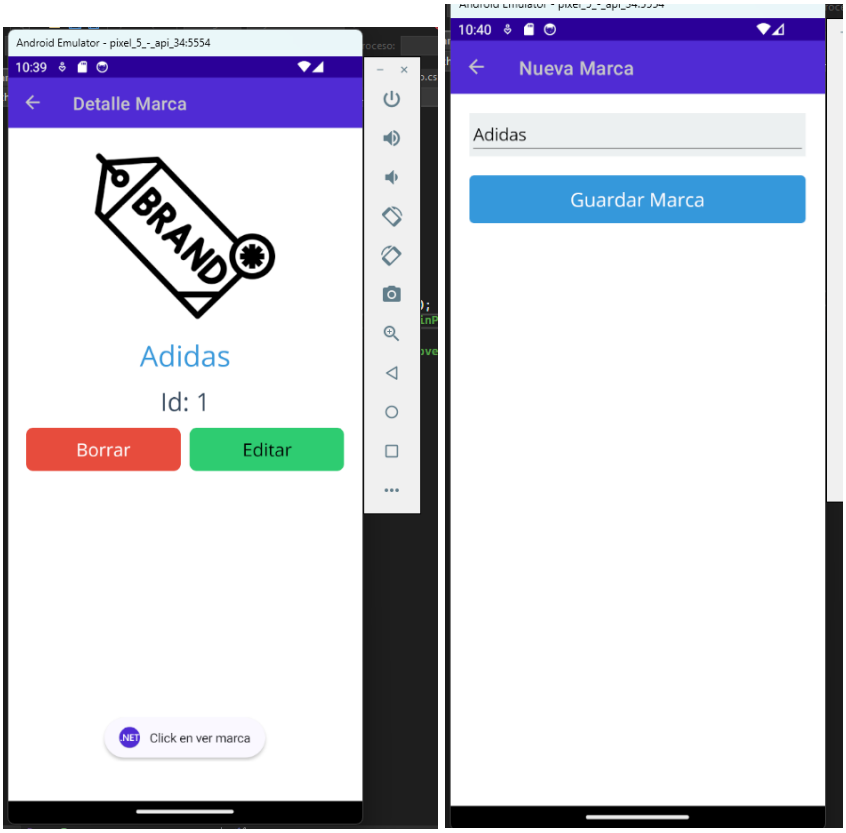
```
using BochaStoreProyecto.Maui.Services;
using CommunityToolkit.Maui.Alerts;
using CommunityToolkit.Maui.Core;
using System.Collections.ObjectModel;
using Marca = BochaStoreProyecto.Maui.Models.Marca;

4 referencias
public partial class MarcaPage : ContentPage
{
    ObservableCollection<Marca> marcas;
    private readonly APIService _APIService;
    0 referencias
    public MarcaPage(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;

    }
    0 referencias
    protected override async void OnAppearing()
    {
        base.OnAppearing();

        string username = "BOCHASTORE";
        Username.Text = username;
        List<Marca> ListaMarcas = await _APIService.GetMarca();
        marcas = new ObservableCollection<Marca>(ListaMarcas);
        listaMarcas.ItemsSource = marcas;

    }

    0 referencias
    private async void OnClickNuevaMarca(object sender, EventArgs e)
    {
        var toast = Toast.Make("On Click Boton Nueva marca", ToastDuration.Short, 14);
        await toast.Show();
        await Navigation.PushAsync(new NuevaMarca(_APIService));
    }
    0 referencias
    private async void listaMarcas_ItemSelected(object sender, SelectedItemChangedEventArgs e)
    {
        var toast = CommunityToolkit.Maui.Alerts.Toast.Make("Click en ver marca", ToastDuration.Short, 14);

        await toast.Show();
        Marca marca = e.SelectedItem as Marca;
        await Navigation.PushAsync(new DetailsMarca(_APIService)
        {
            BindingContext = marca,
        });
    }
}
```

Se inicializa la API par apoder usarla, también se define que vamos a usar de la marca, ya que tenemos una ambigüedad de nombres, en este caso se usara la Marca de “models”

En el método OnAppearing lo que hace es traer la lista de marcas , después se crea una lista de productos que es muy importante ser obvserbable para que cuando se haga algún cambio se muestre en la página.

En el método `OnClickNuevoaMarca` lo que se hace es llevar a la pagina “NevaMarca” y se le pasa el API inicializada

En el método `OnClickShowDetails_ItemSelected ,` lo que se hace es que envia la marca con el binding a la página de detalle para asi obtener los datos



En la vista de editar se reutiliza la vista de “NuevaMarca” ya que con el controlador que se explicará en las siguientes paginas ayuda a saber si se debe editar el proovedor o crear uno nuevo.

Parte XAML

```xml
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="BochaStoreProyecto.Maui.Views.Marca.DetailsMarca"
             Title="Detalle Marca">
    <StackLayout Padding="20" Spacing="10">
        <Image Source="marca_bocha.png"
               HorizontalOptions="Center"
               HeightRequest="200"
               Aspect="AspectFit"/>

        <Label x:Name="Nombre"
               Text="Nombre"
               FontSize="32"
               TextColor="#3498db"
               HorizontalOptions="Center"/>

        <Label x:Name="Id"
               Text="Id"
               FontSize="28"
               TextColor="#34495e"
               HorizontalOptions="Center"/>

        <StackLayout Orientation="Horizontal" Spacing="10">
            <Button x:Name="Borrar"
                    Text="Borrar"
                    Clicked="Borrar_Clicked"
                    FontSize="20"
                    HorizontalOptions="FillAndExpand"
                    BackgroundColor="#e74c3c"/>
            <Button x:Name="Editar"
                    Text="Editar"
                    Clicked="Editar_Clicked"
                    FontSize="20"
                    HorizontalOptions="FillAndExpand"
                    TextColor="Black"
                    BackgroundColor="#2ecc71"/>
        </StackLayout>
    </StackLayout>
</ContentPage>
```

En resumen se presenta una imagen del producto, se tiene dos botones en color rojo(borrar) y verde(editar) y se da nombre a los labels que contendrán la información

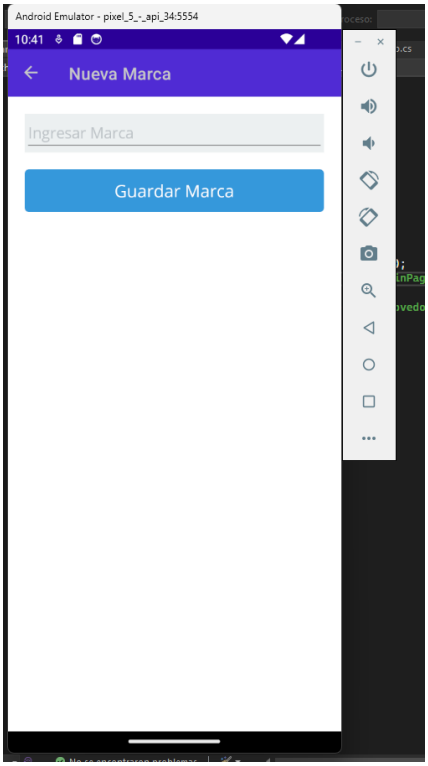
Parte XAML.CS

```
namespace BochaStoreProyecto.Maui.Views.Marca;

using BochaStoreProyecto.Maui.Services;
using Marca = BochaStoreProyecto.Maui.Models.Marca;


public partial class DetailsMarca : ContentPage
{
    private Marca _marca;
    private APIService _APIService;

    public DetailsMarca(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;

    }

    protected override void OnAppearing()
    {
        base.OnAppearing();
        _marca = BindingContext as Marca;
        Nombre.Text = _marca.nombreMarca;
        Id.Text = "Id: "+_marca.idMarca.ToString();
    }

    private async void Borrar_Clicked(object sender, EventArgs e)
    {
        await _APIService.DeleteMarca(_marca.idMarca);
        await Navigation.PopAsync();
    }


    private async void Editar_Clicked(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new NuevaMarca(_APIService)
        {
            BindingContext = _marca,
        });
    }
}
```

EL método OnAppering con ayuda del binding trae la marca y setea en los labels que son llamados por el nombre y se pone respectivamente las características
El método Borrar, llama a la api y usa el borrar para eliminar la marca de la BD

EL método editar manda el producto que se tiene a la página de `NuevaMarca`

NuevaMarca:Esta es la vista que ayuda a crear una nueva marca



Mediante Peaceholder se le indica que datos se deben llenar para crear una nueva marca y para al final poder guardarlo dentro de la base de datos de Azure

Parte XAML

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="BochaStoreProyecto.Maui.Views.Marca.NuevaMarca"
             Title="Nueva Marca">
    <StackLayout Padding="20" Spacing="10">
        <Entry Placeholder="Ingresar Marca"
               x:Name="EntryNombre"
               FontSize="18"
               BackgroundColor="#ecf0f1"
               PlaceholderColor="#bdc3c7"
               Margin="0,0,0,10"/>

        <Button Text="Guardar Marca"
                FontSize="20"
                BackgroundColor="#3498db"
                TextColor="White"
                Clicked="OnClickGuardarNuevaMarca"
                CornerRadius="5"
                HeightRequest="50"/>

    </StackLayout>
</ContentPage>
```

Parte XAML.CS

```
namespace BochaStoreProyecto.Maui.Views.Marca;

using BochaStoreProyecto.Maui.Services;
using Marca = BochaStoreProyecto.Maui.Models.Marca;


public partial class NuevaMarca : ContentPage
{

    private Marca _marca;
    private readonly APIService _APIService;


    public NuevaMarca(APIService apiservice)
    {
        InitializeComponent();
        _APIService = apiservice;

    }

    protected override void OnAppearing()
    {
        base.OnAppearing();
        _marca = BindingContext as Marca;
        if (_marca != null)
        {
            EntryNombre.Text = _marca.nombreMarca;
        }
    }


    private async void OnClickGuardarNuevaMarca(object sender, EventArgs e)
    {

        if (_marca != null)
        {

            _marca.nombreMarca = EntryNombre.Text;

            await _APIService.PutMarca(_marca.idMarca,_marca);
        }
        else
        {
            int id = Utils.Utils.ProductosList.Count + 1;

            Marca marca = new Marca
            {
                idMarca = 0,
                nombreMarca = EntryNombre.Text,
            };
            //Utils.Utils.ProductosList.Add(producto);
            await _APIService.PostMarca(marca);
        }
        await Navigation.PopAsync();
    }
}
```

En el método OnAppering se valida si la marca no es traído por el método visto en la página de detalles, si este no es nulo entonces se setea directamente los atributos de la marca en cada entry

En el método `OnClickGuardarNuevaMarca lo que hace es validar si no es null para saber si tiene que editar o crear un nuevo producto.`

Conclusiones

- La integración de una API para el manejo de la base de datos y la autenticación de usuarios es fundamental para garantizar la seguridad y la integridad de la información. En futuras etapas del proyecto, se deberá prestar especial atención a la implementación de medidas de seguridad adicionales, como el uso de tokens, para fortalecer la protección de datos sensibles
- La implementación de un sistema de manejo de inventario para una empresa de artículos deportivos a través de una aplicación móvil es una solución efectiva para mejorar la eficiencia en la gestión de productos y optimizar las operaciones internas de la empresa.
- Para las vistas se tendrá que comprobar el login de los usuarios y el tipo de Usuario, dependiendo de esto se redireccionará al usuario administrador a su vista respectiva y al cliente igual.
- La elección de MAUI.NET como plataforma para el desarrollo de la aplicación ofrece ventajas significativas al proporcionar una base de código única que puede ser utilizada en diferentes sistemas operativos, lo que simplifica el proceso de desarrollo y mantenimiento a largo plazo.

6.Link GITHUB

https://github.com/EstebanEr-03/ProyectoProgreso2BCHSTR.git