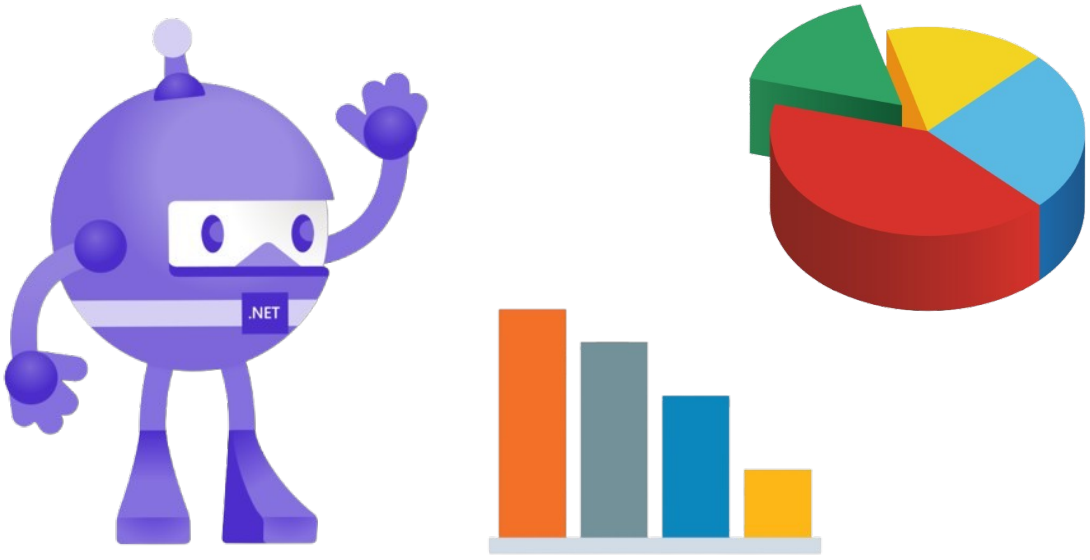
DESARROLLO DE INTERFACES

Gráficos en MAUI



Marcos Zahonero

Índice

Introducción

En esta actividad veremos como utilizar gráficos gracias a la librería de Microcharts en MAUI, será también utilizando un poco los conceptos ya vistos anteriormente de pasar de pantallas traspasando información para así poder mostrarla en los gráficos.

Tiempo utilizado

He utilizado 4 horas, y para hacer el documento y vídeos aproximadamente 1 hora, en total serían 5 horas utilizadas para el trabajo.

Material utilizado

- El portátil de clase, con mi ordenador personal de casa.
- Visual Studio.

Mi ordenador de casa

Componente	Nombre
Sistema Operativo	Windows 11 Pro
Procesador	AMD Ryzen 5 2600X
Tarjeta gráfica	NVIDIA GeForce GTX 1650 4GB
Memoria RAM	16 GB
Placa base	B450M-A PRO MAX
Disco duro 1#	KIOXIA EXCERIA G2 SSD 1 TB
Disco duro 2#	ST1000DM010-2EP102 HDD 1 TB

Portátil de clase

Componente	Nombre
Sistema Operativo	Windows 10 Pro
Procesador	Intel(R) Core(TM) i5-7300U @ 2.60GHz 2.71 GHz
Tarjeta gráfica	*(Integrada)*
Modelo portátil	HP EliteBook 840 G5
Memoria RAM	16 GB
Placa base	HP 83B2
Disco duro	Disco duro HDD 256GB

Objetivos de la práctica

- Conocer la librería de Microcharts
- Entender el funcionamiento de como usar los ICommand.
- Entender el funcionamiento de los Binding y utilizarlos.
- Realizar el traspaso de información entre pantallas correctamente.
- Diseñar una interfaz interesante de ver.

Desarrollo de la Actividad

Esta aplicación consta de 2 páginas (MainPage y GráficoPage con los siguientes elementos:

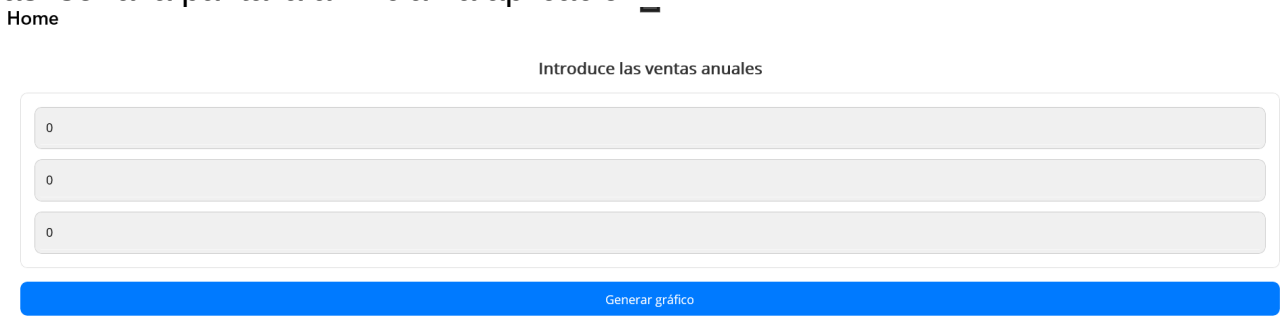
- 3 Entrys
- 2 Frames
- 2 Labels
- 2 Button
- 3 ChartViews
- 2 ViewModels (Grafico y MainPage)
- 1 Model (Vendes.cs)

Páginas

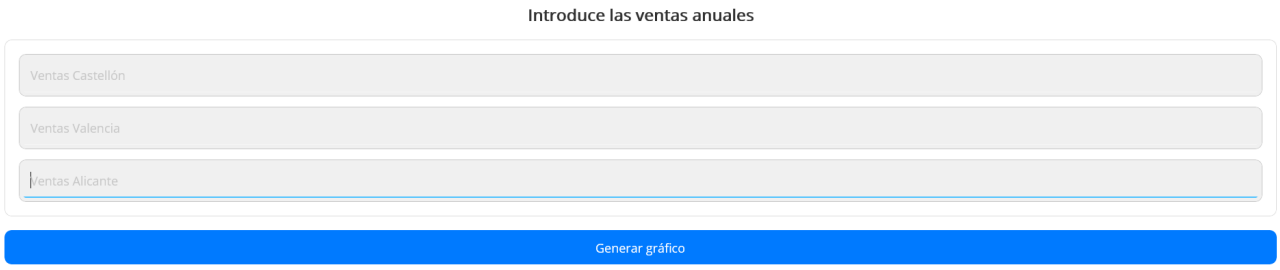
Página - MainPage

Pantalla inicial

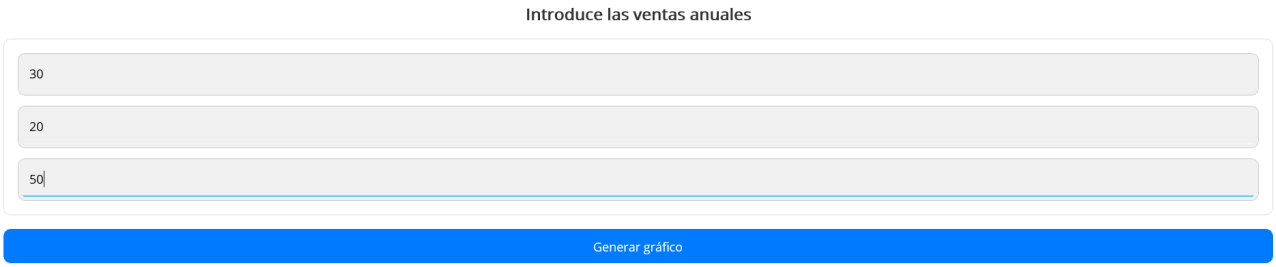
En este momento no contiene ningún número puesto ni ninguno calculado, así que así sería la pantalla al iniciar la aplicación.



Pantalla sin datos



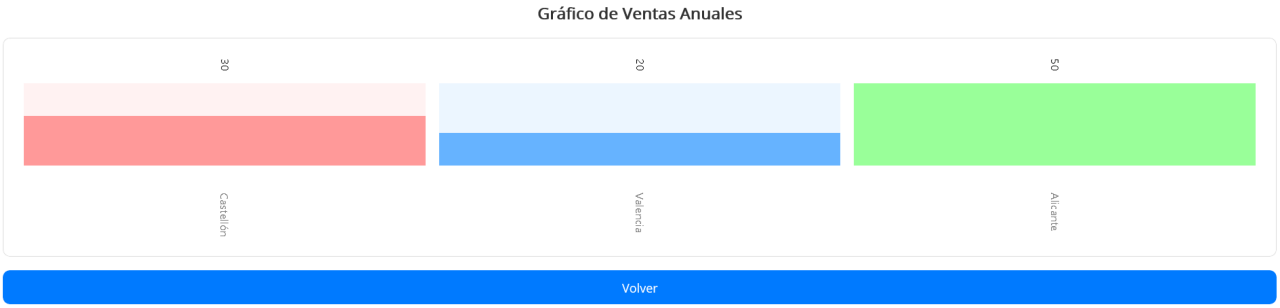
Pantalla con datos



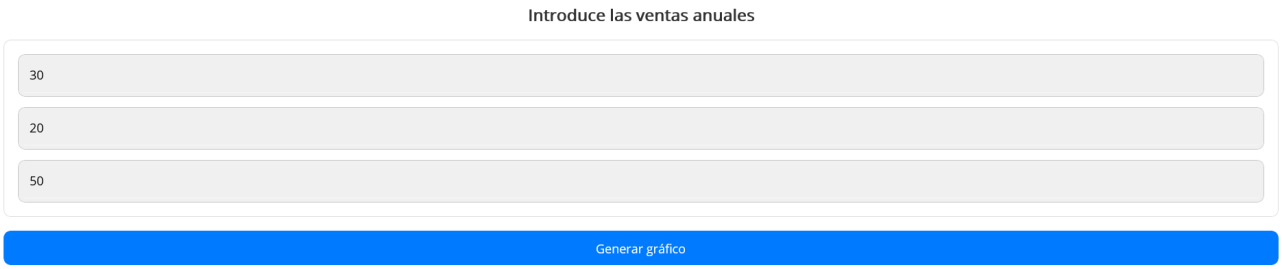
Página – GraficoPage

Pantalla inicial

Esta pantalla se genera con estos datos debido a la anterior captura con los datos insertados, tienes un botón para volver hacia atrás para cambiar los datos



Y como puedes observar podrás modificar los datos y volver a generar el gráfico:



ViewModels

MainPageViewModel

En este ViewModel he guardado toda la información obviamente de los entrys, obteniendo las variables después con sus variables públicas, el resto ha sido crear un ICommand que “generara los gráficos”, que realmente lo que hace es llevarte a otra página pasando los parámetros insertados en los Entrys:

```
private async void GenerarGrafico()
{
    var ventasAnuales = new Vendes
    {
        Castello = Castello,
        Valencia = Valencia,
        Alicante = Alicante
    };

    string datosVentas = $"{Castello}|{Valencia}|{Alicante}";
    await Shell.Current.GoToAsync($"//grafico?Ventas={datosVentas}");
}
```

GraficoViewModel

Al recibir datos bien sabemos lo que debemos utilizar, y es el **QueryProperty**.

```
[QueryProperty(nameof(Ventas), "Ventas")]
```

En este caso contengo 2 variables públicas, un Chart que contendrá los 3 gráficos y el String de ventas, que contiene básicamente lo que le pasamos desde el MainPage, que aprovechando la actividad probé algo distinto para hacerlo más entretenido de ver, iremos por partes.

Todo se concentra en el set, es decir, cuando el programa recibe el elemento **Ventas** desde el MainPage el recorrido que hará será el siguiente:

```
public GraficoViewModel() -> public string Ventas set {}
```

¿Qué quiere decir esto? Que podemos usar esto a nuestro favor viendo la estructura del proyecto, ¿Cuál es la estructura del proyecto? Es que desde la MainPage siempre pasamos ese valor, que se debe tratar de x forma siempre, pues entonces podemos hacer que el set del propio Ventas sea especial, veamos como.

Al principio algo simple, si el valor que le pasamos no es null ni esta vacío podrá continuar.

```
if (!string.IsNullOrEmpty(value))
```

Después pasamos a algo más complejo, y es el uso (realmente no es tan necesario pero es más profesional) del `out int` (**enlace**), veamos:

Aquí al principio dividimos los datos que hemos enviado por el carácter que decidimos en su momento para dividirlo y después nos aseguramos que son 3 y a su vez nos aseguramos que los valores que hemos pasado son int, usando el `int.TryParse` esto asegura que esos valores son int, para evitar excepciones, después tenemos el out int, que lo que hace es que el valor de su izquierda lo que hará será en el caso de castello cojera el datos[0] su valor, e inicializará la variable castello con ese nombre y guardará el dato y así con el resto.

```
var datos = value.Split("|");
if (datos.Length == 3 &&
    int.TryParse(datos[0], out int castello) &&
    int.TryParse(datos[1], out int valencia) &&
    int.TryParse(datos[2], out int alacant))
{
```

Después ya viene la parte de los gráficos, en este caso lo que nos pide la actividad es que sea un BarChart con unos colores en específico.

Construimos el objeto Chart pasandole una lista de Entries, que contendrá ChartEntrys que se crearán a partir de los ints que hemos pasado, luego los labels para poner el texto que pondrá abajo, su valor y su color correspondiente.

```
Chart = new BarChart
{
    Entries = new[]
    {
        new ChartEntry(castello) { Label = "Castellón", ValueLabel = castello.ToString(), Color = SKColor.Parse("#ff9999") },
        new ChartEntry(valencia) { Label = "Valencia", ValueLabel = valencia.ToString(), Color = SKColor.Parse("#66b3ff") },
        new ChartEntry(alacant) { Label = "Alicante", ValueLabel = alacant.ToString(), Color = SKColor.Parse("#99ff99") }
    }
};
```

Y así es como obtendríamos los gráficos, ya que esta variable Chart se enlaza con un Binding con el XAML y se mostrará.

Cambios o archivos extras

MauiProgram.cs

En este caso habrá que tocar un poco este archivo tan poco modificado en nuestras actividades ya que debemos avisar a MAUI que vamos a utilizar la librería de Microcharts, es por eso que añadimos la siguiente línea “

“ `.UseMicrocharts()` “

```
namespace ud08EjemploMicroCharts
{
    4 referencias
    public static class MauiProgram
    {
        4 referencias
        public static MauiApp CreateMauiApp()
        {
            var builder = MauiApp.CreateBuilder();
            builder
                .UseMauiApp<App>()
                .UseMicrocharts()
```

Aquí normalmente es lo necesario para realizar la actividad, pero debido a unos problemas

Vendes.cs

Esta clase contiene los valores que insertamos en el MainPage en su momento.

```
1 referencia
public class Vendes
{
    1 referencia
    public int Castello { get; set; }
    1 referencia
    public int Valencia { get; set; }
    1 referencia
    public int Alicante { get; set; }
}
```

Códigos de los archivos

MainPage

MainPage.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"
             xmlns:micro="clr-namespace:Microcharts.Maui;assembly=Microcharts.Maui"
             x:Class="ud08EjemploMicroCharts.MainPage">

    <ScrollView>
        <VerticalStackLayout Padding="30" Spacing="20">

            <Label Text="Introduce las ventas anuales"
                   FontSize="24"
                   FontAttributes="Bold"
                   TextColor="#333"
                   HorizontalOptions="Center" />

            <Frame Padding="20"
                   BackgroundColor="White"
                   CornerRadius="10"
                   HasShadow="True"
                   BorderColor="#DDD">

                <VerticalStackLayout Spacing="15">

                    <Frame Padding="5" BackgroundColor="#F0F0F0" CornerRadius="10">
                        <Entry Placeholder="Ventas Castellón"
                               Keyboard="Numeric"
                               Text="{Binding Castello}"
                               FontSize="18"
                               HeightRequest="50"/>
                    </Frame>

                    <Frame Padding="5" BackgroundColor="#F0F0F0" CornerRadius="10">
                        <Entry Placeholder="Ventas Valencia"
                               Keyboard="Numeric"
                               Text="{Binding Valencia}"
                               FontSize="18"
                               HeightRequest="50"/>
                    </Frame>

                    <Frame Padding="5" BackgroundColor="#F0F0F0" CornerRadius="10">
                        <Entry Placeholder="Ventas Alicante"
                               Keyboard="Numeric"
                               Text="{Binding Alicante}"
                               FontSize="18"
                               HeightRequest="50"/>
                    </Frame>

                </VerticalStackLayout>
            </Frame>

            <Button Text="Generar gráfico"
                    Command="{Binding GenerarGraficoCommand}"
                    BackgroundColor="#007AFF"
                    TextColor="White"
                    FontSize="18"
                    CornerRadius="10"
                    HeightRequest="50"
                    HorizontalOptions="Fill" />

        </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```

MainPage.xaml.cs

```csharp
using ud08EjemploMicroCharts.ViewModels;

namespace ud08EjemploMicroCharts
{
    5 referencias
    public partial class MainPage : ContentPage
    {

        0 referencias
        public MainPage()
        {
            InitializeComponent();

            BindingContext = new MainPageViewModel();
        }
    }
}
```

GraficoPage

GraficoPage.xaml

```xaml
<?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:microcharts="clr-namespace:Microcharts.Maui;assembly=Microcharts.Maui"
             x:Class="ud08EjemploMicroCharts.GraficoPage"
             Title="GraficoPage">

    <VerticalStackLayout Padding="20" Spacing="20">
        <Label Text="Gráfico de Ventas Anuales"
               FontSize="24"
               FontAttributes="Bold"
               TextColor="#333"
               HorizontalOptions="Center" />

        <Frame Padding="10"
               BackgroundColor="White"
               CornerRadius="10"
               HasShadow="True"
               BorderColor="#DDD">
            <microcharts:ChartView x:Name="ChartView"
                                   HeightRequest="300"
                                   Chart="{Binding Chart}" />
        </Frame>

        <Button Text="Volver"
                Clicked="Volver_Clicked"
                BackgroundColor="#007AFF"
                TextColor="White"
                FontSize="18"
                CornerRadius="10"
                HeightRequest="50"
                HorizontalOptions="Fill" />

    </VerticalStackLayout>
</ContentPage>
```

GraficoPage.xaml.cs

```cs
using ud08EjemploMicroCharts.ViewModels;

namespace ud08EjemploMicroCharts
{
    5 referencias
    public partial class GraficoPage : ContentPage
    {
        0 referencias
        public GraficoPage()
        {
            InitializeComponent();
            BindingContext = new GraficoViewModel();
        }

        0 referencias
        private async void Volver_Clicked(object sender, EventArgs e)
        {
            await Shell.Current.GoToAsync("//MainPage");
        }
    }
}
```

ViewModels

MainPageViewModel.cs

```
public class MainPageViewModel : INotifyPropertyChanged
{
    private int _castello;
    3 referencias
    public int Castello
    {
        get => _castello;
        set
        {
            if (_castello != value)
            {
                _castello = value;
                OnPropertyChanged(nameof(Castello));
            }
        }
    }

    private int _valencia;
    3 referencias
    public int Valencia
    {
        get => _valencia;
        set
        {
            if (_valencia != value)
            {
                _valencia = value;
                OnPropertyChanged(nameof(Valencia));
            }
        }
    }

    private int _alicante;
    3 referencias
    public int Alicante
    {
        get => _alicante;
        set
        {
            if (_alicante != value)
            {
                _alicante = value;
                OnPropertyChanged(nameof(Alicante));
            }
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;
    3 referencias
    protected void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
```

```
    1 referencia
    public ICommand GenerarGraficoCommand { get; }

    1 referencia
    public MainPageViewModel()
    {
        GenerarGraficoCommand = new Command(GenerarGrafico);
    }

    1 referencia
    private async void GenerarGrafico()
    {
        var ventasAnuales = new Vendes
        {
            Castello = Castello,
            Valencia = Valencia,
            Alicante = Alicante
        };

        string datosVentas = $"{Castello}|{Valencia}|{Alicante}";
        await Shell.Current.GoToAsync($"//grafico?Ventas={datosVentas}");
    }
}
```

GraficoViewModel.cs

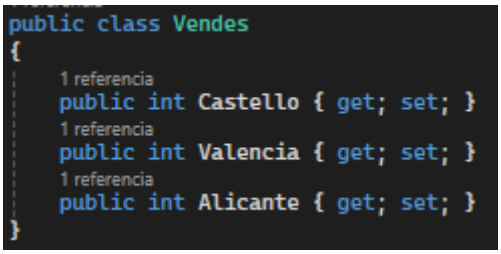
```
namespace ud08EjemploMicroCharts.ViewModels
{
    [QueryProperty(nameof(Ventas), "Ventas")]
    1 referencia
    public class GraficoViewModel : INotifyPropertyChanged
    {
        private Chart _chart;
        2 referencias
        public Chart Chart
        {
            get => _chart;
            set
            {
                _chart = value;
                OnPropertyChanged(nameof(Chart));
            }
        }

        private string? _ventas;
        1 referencia
        public string Ventas
        {
            get { return _ventas; }
            set
            {
                if (!string.IsNullOrEmpty(value))
                {
                    var datos = value.Split("|");
                    if (datos.Length == 3 &&
                        int.TryParse(datos[0], out int castello) &&
                        int.TryParse(datos[1], out int valencia) &&
                        int.TryParse(datos[2], out int alacant))
                    {
                        Chart = new BarChart
                        {
                            Entries = new[]
                            {
                                new ChartEntry(castello) { Label = "Castellón", ValueLabel = castello.ToString(), Color = SKColor.Parse("#ff9999") },
                                new ChartEntry(valencia) { Label = "Valencia", ValueLabel = valencia.ToString(), Color = SKColor.Parse("#66b3ff") },
                                new ChartEntry(alacant) { Label = "Alicante", ValueLabel = alacant.ToString(), Color = SKColor.Parse("#99ff99") }
                            }
                        };
                    }
                }
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;
        1 referencia
        protected void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

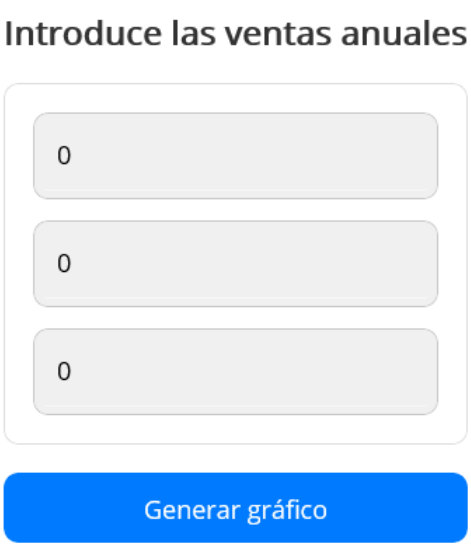
Clases extras

(Models) Vendes.cs



Resultados finales

Resultado Final – Vista Móvil

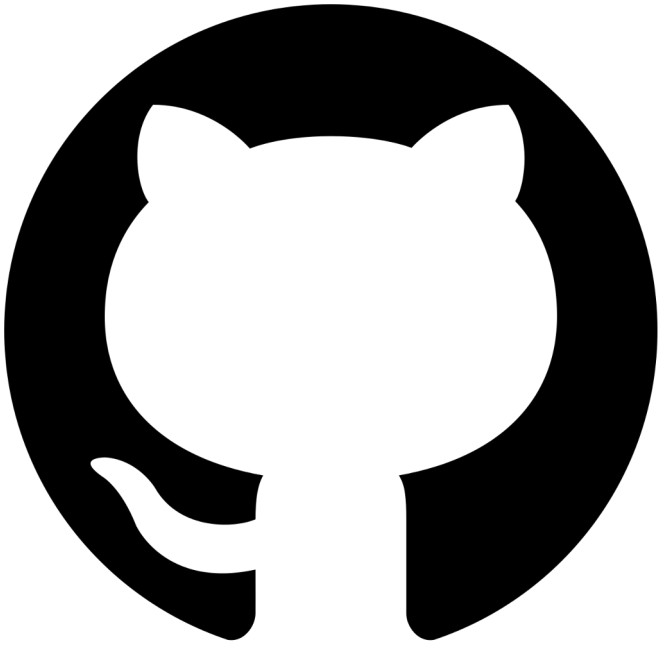


Resultado Final – Vista Ordenador

GitHub

En esta actividad he añadido GitHub para añadir el proyecto completo, junto a una descripción en el README.md para añadir una breve explicación y preview de cada actividad.

Enlace: Maek0s/2DAM_DesarrolloInterfaces



Conclusión

En conclusión, la actividad ha sido relativamente fácil y sinceramente he aprovechado para complicarme un poco la vida para probar cosas como la que hice en el GraficoViewModel.cs con la variable Ventas, el cual quedo bastante ordenado y visible, lo único complejo ha sido encontrar como hacer un charEntry y la interfaz, aunque debido al proyecto y anteriores trabajos me ha costado menos hacer una interfaz bonita y simple de ver.

Problemas y/o sugerencias en la actividad

1# - Mi extra a la actividad

Volvemos a los 2 vídeos por actividad para finalizar con esta actividad optativa.

Vídeo del funcionamiento: https://youtu.be/4EUrQeiLU3Q
Vídeo de explicativo: https://youtu.be/ZLXQDnQ9LK4

2# - Problema con SkiaSharp

No tengo exactamente el error pero llego un momento realizando la actividad que el proyecto no se ejecutaba directamente, pero si tengo el código de error, que fue el **"código 3221226356 (0xc0000374)"**, es debido a las incompatibilidades o dependencias con SkiaSharp y Microcharts, ya que estuve utilizando .NET 9 y es posible que fuera por la versión, ya que al utilizar el ejemplo las dependencias estarían menos actualizadas, y también me aconsejo añadir una línea en **MauiProgram.cs** poniendo la librería de SkiaSharp, que no estaba puesta así que puse "`.UseSkiaSharp()`" debajo de la librería de Microcharts y actualizado SkiaSharp y volvió a funcionar todo correctamente.

Webgrafía

- https://visualstudio.microsoft.com/es/vs/community/
- https://dotnet.microsoft.com/es-es/apps/maui

- ICommand:
https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/data-binding/commanding?view=net-maui-9.0

- ViewModels:
https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

- Gráficos (Wiki Microcharts):
https://github.com/microcharts-dotnet/Microcharts/wiki