DESARROLLO DE INTERFACES

Interfaz con ICommands

Gestión de Tareas entre páginas

Marcos Zahonero

Índice

Introducción

En esta actividad veremos una gestión de tareas como la reciente actividad pero con la particularidad que en este caso utilizaremos ViewModels para todo el tema de gestión de información y todo lo que tenga que ver con las páginas, también haremos uso de los ICommands para gestionar varios botones y darles un uso funcional.

También hacemos uso de conocimientos anteriores para volverlos a poner uso como por ejemplo el Shell, para navegar por páginas gracias a él y también el uso de Triggers, para cambiar su aspecto al estar completada la tarea o no, esto nos da más margen de decoración y varias posibles opciones a realizar.

Tiempo utilizado

He utilizado 7 horas, y para hacer el documento y vídeos aproximadamente 1 hora y media más, en total serían 8 horas y media, esta actividad fue compleja de entender su funcionamiento y sus conocimientos necesarios, y también volver a recordar conocimientos como Shell y Triggers.

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

- Realizar una interfaz con ICommands en MAUI.
- Entender el funcionamiento de como usarlos ICommand.
- Entender el funcionamiento de los Binding y utilizarlos.
- Diseñar una interfaz interesante de ver.
- Ser capaz de traspasar información entre una página y otra.

Desarrollo de la Actividad

Esta aplicación consta de 2 páginas (**MainPage** y **AddItemNewWindow**), con los siguientes elementos:

- 3 Label
- 1 Entrys
- 3 Buttons
- 1 CollectionView
- 2 Checkboxs
- 2 ViewModels (MainPageViewModel y AddItemNewWindowViewModel)

Página – MainPage

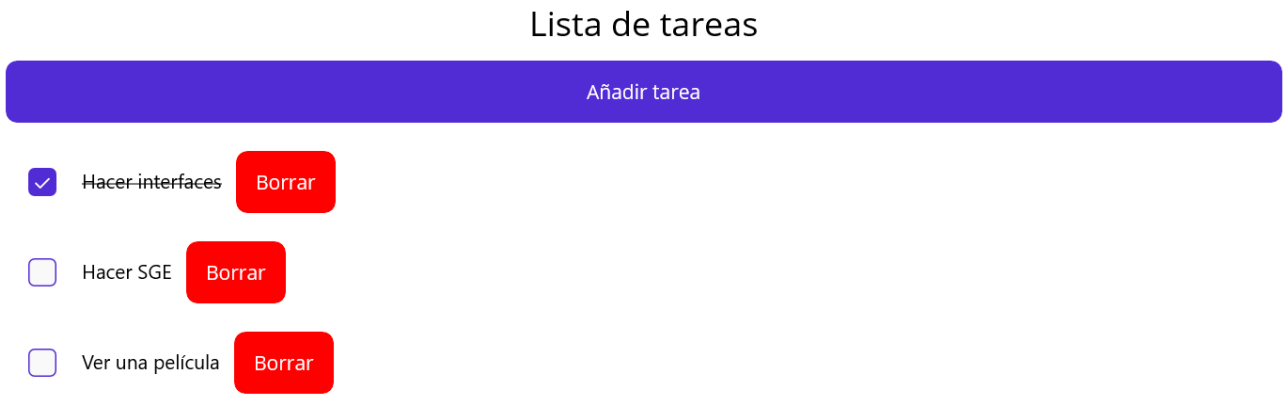
Pantalla inicial

En este momento no contiene ninguna tarea por hacer, debes añadirla, haciendo clic en el botón “Añadir tarea” que lanzará el Command que lleva implementado, que este servirá para lanzar un método que he preparado para mandarlo a otra página y así volver con una tarea.

Pantalla con unas tareas añadidas

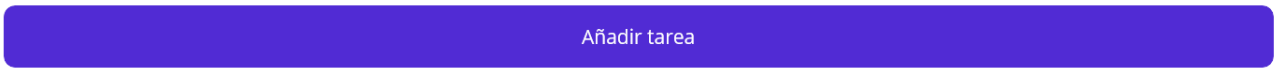
Aquí podemos ver que ha cambiado la pantalla, esto se debe a que he añadido unas tareas para ver como reacciona la página principal, lo que ocurre es que al añadir una tarea la devuelve y la añade al CollectionView, que se encargará de mostrar esa tarea por pantalla con una plantilla que consta de:

- **1 Checkbox** para cambiar su valor de completado o no, el cual tiene un trigger que en el caso que este marcado el nombre de la tarea se tachará.
- **1 Label** que muestra el nombre de la tarea.
- **1 Button** que lanza un Command llamado `DeleteItemCommand` que está enlazado con la clase Todoitem y accede a la lista de tareas y la borra.



Botón de “Añadir tarea”

Al hacer clic en este botón como bien he dicho anteriormente lanzará un ICommand.

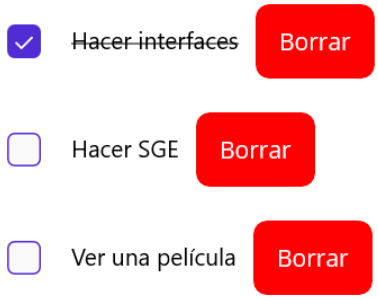


Se llama `AddItemCommandNewWindow`, y como ya dije lanza un método, este es más simple ya que solo cambia de página a `AddItemNewWindow` usando la navegación con shell, y insertando sus correspondientes rutas, así sería para cambiar de página:

```
await Shell.Current.GoToAsync($"AddItemNewWindow");
```

CollectionView<Todoitem> listaTareas

Esta **CollectionView** hace posible que se vean las tareas que hay en una lista, para que identifiquemos más, te muestro donde se encuentra en la MainPage:



Este contiene tareas, pero esas tareas son realmente de una clase llamada Todoitem, la cuál hablaré más tarde, ahora solo explicaré que tiene **2 atributos**, un **boolean** que se cambia con su checkbox y un **String** para el nombre de la tarea, el botón viene extra para solo borrarlo, pero no esta dentro de la clase **Todoitem.cs**.

Página – AddItemNewWindow

Esta página se accede gracias al botón del MainPage de “Añadir tarea”, sirve únicamente para devolver una tarea y que esta se añada a la lista de tareas que contiene el MainPage y la muestre por pantalla.

Pantalla inicial

Esta pantalla no tiene nada fuera de lo común, su EditText para introducir el nombre de la tarea y una checkbox para saber si ya está completada o no.

AddItemNewWindow

Botón “Añadir”

Este botón sirve para enviarnos de vuelta al MainPage, pero lo hará mandando una tarea y esta se añadirá en la MainPage.

Botón “Cancelar”

Este botón sirve únicamente (por el momento), para mandarnos de vuelta al MainPage, pero sin mandar ninguna información.

Todoitem.cs

Esta clase es la que maneja la **lista de tareas**, el **CollectionView<TodoItem>** que hemos visto en la MainPage.

Esta clase contiene 2 atributos:

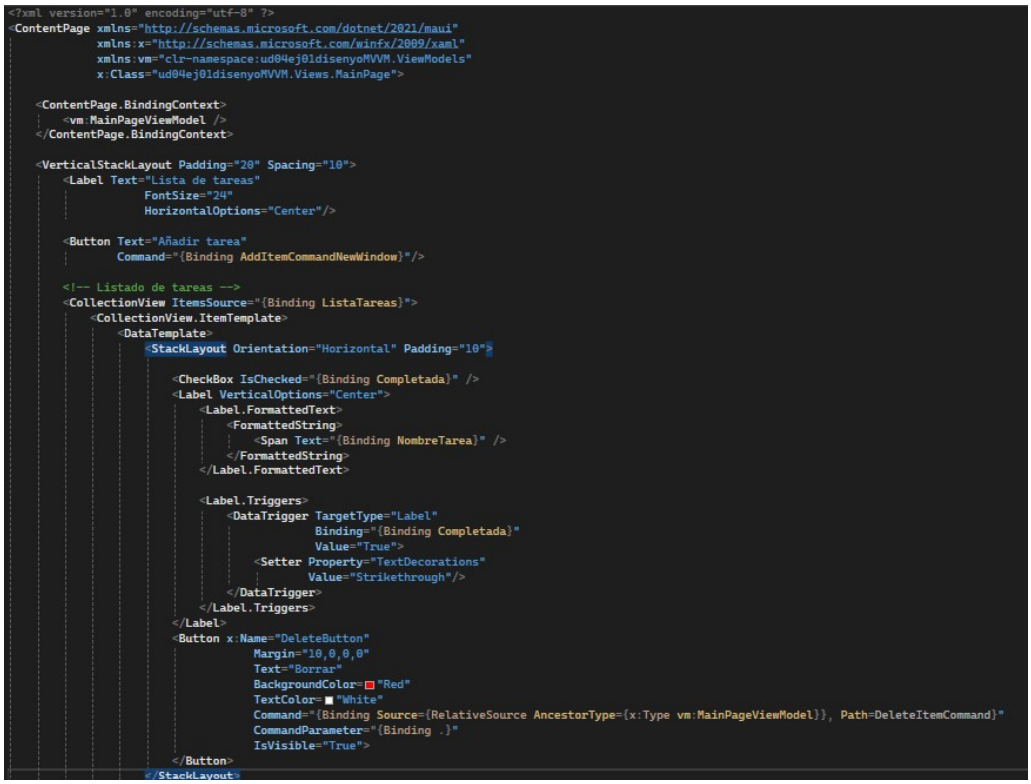
```
String _nombreTarea → Nombre de la tarea
bool _completada → Indica si está completada o no
```

El valor de `_completada` se gestiona al crear la tarea, al igual que con el `_nombreTarea` pero en el caso de `_completada` se puede llegar a cambiar su valor al ya crear la tarea, desde el MainPage dándole clic a su checkbox correspondiente.

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:vm="clr-namespace:ud04ej01disenyoMVVM.ViewModels"
             x:Class="ud04ej01disenyoMVVM.Views.MainPage">

    <ContentPage.BindingContext>
        <vm:MainPageViewModel />
    </ContentPage.BindingContext>

    <VerticalStackLayout Padding="20" Spacing="10">
        <Label Text="Lista de tareas"
               FontSize="24"
               HorizontalOptions="Center"/>

        <Button Text="Añadir tarea"
                Command="{Binding AddItemCommandNewWindow}"/>

        <!-- Listado de tareas -->
        <CollectionView ItemsSource="{Binding ListaTareas}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <StackLayout Orientation="Horizontal" Padding="10">

                        <CheckBox IsChecked="{Binding Completada}" />
                        <Label VerticalOptions="Center">
                            <Label.FormattedText>
                                <FormattedString>
                                    <Span Text="{Binding NombreTarea}" />
                                </FormattedString>
                            </Label.FormattedText>

                            <Label.Triggers>
                                <DataTrigger TargetType="Label"
                                             Binding="{Binding Completada}"
                                             Value="True">
                                    <Setter Property="TextDecorations"
                                            Value="Strikethrough"/>
                                </DataTrigger>
                            </Label.Triggers>
                        </Label>
                        <Button x:Name="DeleteButton"
                                Margin="10,0,0,0"
                                Text="Borrar"
                                BackgroundColor="Red"
                                TextColor="White"
                                Command="{Binding Source={RelativeSource AncestorType={x:Type vm:MainPageViewModel}}, Path=DeleteItemCommand}"
                                CommandParameter="{Binding .}"
                                IsVisible="True">
                        </Button>
                    </StackLayout>
```

MainPage.xaml.cs

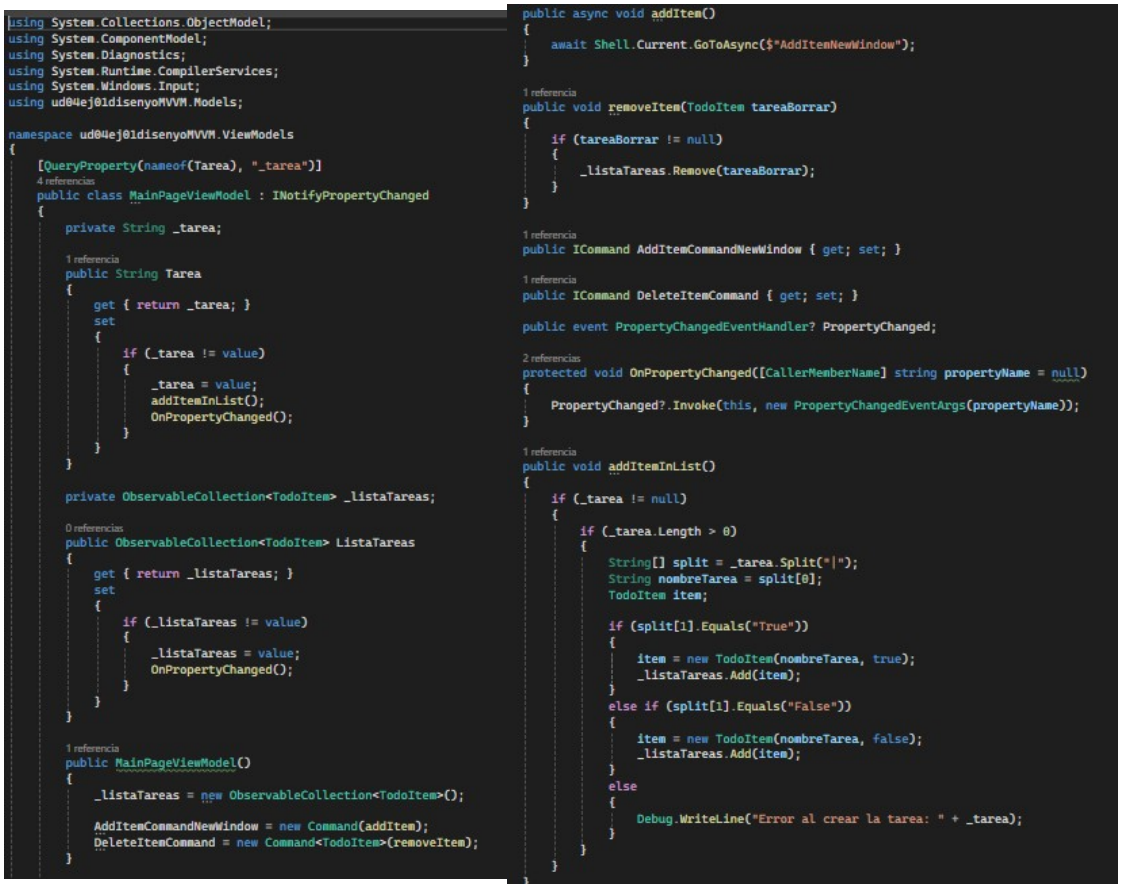
```cs
using System.Collections.ObjectModel;
using ud04ej01disenyoMVVM.Models;
using ud04ej01disenyoMVVM.ViewModels;

namespace ud04ej01disenyoMVVM.Views
{
    [QueryProperty(nameof(TodoItem), "ListaTareas")]
    7 referencias
    public partial class MainPage : ContentPage
    {

        private ObservableCollection<TodoItem> _listaTareas;
        0 referencias
        public ObservableCollection<TodoItem> ListaTareas
        {
            get { return _listaTareas; }
            set
            {
                if (_listaTareas != value)
                {
                    _listaTareas = value;
                    OnPropertyChanged();
                }
            }
        }
        0 referencias
        public MainPage()
        {
            InitializeComponent();

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

MainPageViewModel.cs



```
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Diagnostics;
using System.Runtime.CompilerServices;
using System.Windows.Input;
using ud04ej01disenyoMVVM.Models;

namespace ud04ej01disenyoMVVM.ViewModels
{
    [QueryProperty(nameof(Tarea), "_tarea")]
    4 referencias
    public class MainPageViewModel : INotifyPropertyChanged
    {
        private String _tarea;

        1 referencia
        public String Tarea
        {
            get { return _tarea; }
            set
            {
                if (_tarea != value)
                {
                    _tarea = value;
                    addItemInList();
                    OnPropertyChanged();
                }
            }
        }

        private ObservableCollection<TodoItem> _listaTareas;

        0 referencias
        public ObservableCollection<TodoItem> ListaTareas
        {
            get { return _listaTareas; }
            set
            {
                if (_listaTareas != value)
                {
                    _listaTareas = value;
                    OnPropertyChanged();
                }
            }
        }

        1 referencia
        public MainPageViewModel()
        {
            _listaTareas = new ObservableCollection<TodoItem>();

            AddItemCommandNewWindow = new Command(addItem);
            DeleteItemCommand = new Command<TodoItem>(removeItem);
        }

        public async void addItem()
        {
            await Shell.Current.GoToAsync($"AddItemNewWindow");
        }

        1 referencia
        public void removeItem(TodoItem tareaBorrar)
        {
            if (tareaBorrar != null)
            {
                _listaTareas.Remove(tareaBorrar);
            }
        }

        1 referencia
        public ICommand AddItemCommandNewWindow { get; set; }

        1 referencia
        public ICommand DeleteItemCommand { get; set; }

        public event PropertyChangedEventHandler? PropertyChanged;

        2 referencias
        protected void OnPropertyChanged([CallerMemberName] string propertyName = null)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }

        1 referencia
        public void addItemInList()
        {
            if (_tarea != null)
            {
                if (_tarea.Length > 0)
                {
                    String[] split = _tarea.Split("|");
                    String nombreTarea = split[0];
                    TodoItem item;

                    if (split[1].Equals("True"))
                    {
                        item = new TodoItem(nombreTarea, true);
                        _listaTareas.Add(item);
                    }
                    else if (split[1].Equals("False"))
                    {
                        item = new TodoItem(nombreTarea, false);
                        _listaTareas.Add(item);
                    }
                    else
                    {
                        Debug.WriteLine("Error al crear la tarea: " + _tarea);
                    }
                }
            }
        }
```

AddItemNewWindow

AddItemNewWindow.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:vm="clr-namespace:ud04ej01disenyoMVVM.ViewModels"
             x:Class="ud04ej01disenyoMVVM.Views.AddItemNewWindow"
             Title="AddItemNewWindow">

    <ContentPage.BindingContext>
        <vm:AddItemNewWindowViewModel/>
    </ContentPage.BindingContext>

    <VerticalStackLayout
        Padding="15"
        Spacing="10">
        <Label
            Text="Nueva tarea"
            FontSize="Large"
            VerticalOptions="Center"
            HorizontalOptions="Center" />

        <Entry
            Text="{Binding NombreTarea}"
            Placeholder="Añade tarea"
            x:Name="eNombreTarea"/>

        <HorizontalStackLayout>
            <CheckBox x:Name="checkCompleted" IsChecked="{Binding IsCompleted}"/>
            <Label Text="Completada"
                   VerticalTextAlignment="Center"/>
        </HorizontalStackLayout>

        <Button Text="Añadir" Command="{Binding addTareaCommand}"/>
        <Button Text="Cancelar" Command="{Binding cancelarTareaCommand}"/>
    </VerticalStackLayout>
</ContentPage>
```

AddItemNewWindow.cs

```
using ud04ej01disenyoMVVM.ViewModels;

namespace ud04ej01disenyoMVVM.Views;

7 referencias
public partial class AddItemNewWindow : ContentPage
{
    0 referencias
    public AddItemNewWindow()
    {
        InitializeComponent();

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

AddItemNewWindowViewModel.cs

```
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Windows.Input;

namespace ud04ej01disenyoMVVM.ViewModels
{
    4 referencias
    public class AddItemNewWindowViewModel
    {
        private string _nombreTarea;
        private bool _isCompleted;
        2 referencias
        public string NombreTarea
        {
            get { return _nombreTarea; }
            set
            {
                if (_nombreTarea != value)
                {
                    _nombreTarea = value;
                    OnPropertyChanged(nameof(NombreTarea));
                }
            }
        }
        1 referencia
        public bool IsCompleted
        {
            get { return _isCompleted; }
            set
            {
                if (_isCompleted != value)
                {
                    _isCompleted = value;
                }
            }
        }

        1 referencia
        public ICommand addTareaCommand { get; set; }
        1 referencia
        public ICommand cancelarTareaCommand { get; set; }

        1 referencia
        public AddItemNewWindowViewModel()
        {
            addTareaCommand = new Command(addTarea);
            cancelarTareaCommand = new Command(cancelarTarea);
        }
```

```
        1 referencia
        public void addTarea()
        {
            String tarea = "";

            String nombreTarea = NombreTarea;
            bool completed = IsCompleted;

            if (nombreTarea.Trim().Equals(""))
            {
                tarea = null;
            } else
            {
                tarea = nombreTarea + "|" + completed;
            }

            goBack(tarea);
        }

        1 referencia
        public async void cancelarTarea()
        {
            // Lo dejo así por si llegamos a querer hacer algo especial
            await Shell.Current.GoToAsync($"///MainPage");
        }

        1 referencia
        public async void goBack(String tarea)
        {
            await Shell.Current.GoToAsync($"///MainPage?_tarea={tarea}");
        }


        public event PropertyChangedEventHandler PropertyChanged;

        1 referencia
        protected void OnPropertyChanged([CallerMemberName] string propertyName = nu
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName)
        }
```

Clases extras

Todoitem.cs

```
12 referencias
public class TodoItem : INotifyPropertyChanged
{
    private String _nombreTarea;
    private bool _completada;

    0 referencias
    public String NombreTarea
    {
        get { return _nombreTarea; }
        set
        {
            if (_nombreTarea != value)
            {
                _nombreTarea = value;
                OnPropertyChanged();
            }
        }
    }

    0 referencias
    public bool Completada
    {
        get { return _completada; }
        set
        {
            if (_completada != value)
            {
                _completada = value;
                OnPropertyChanged();
            }
        }
    }

    2 referencias
    public TodoItem(String nombreTarea, bool completada)
    {
        _nombreTarea = nombreTarea;
        _completada = completada;
    }


    public event PropertyChangedEventHandler? PropertyChanged;

    2 referencias
    protected void OnPropertyChanged([CallerMemberName] string propertyName = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
```

Resultado Final – Vista Móvil

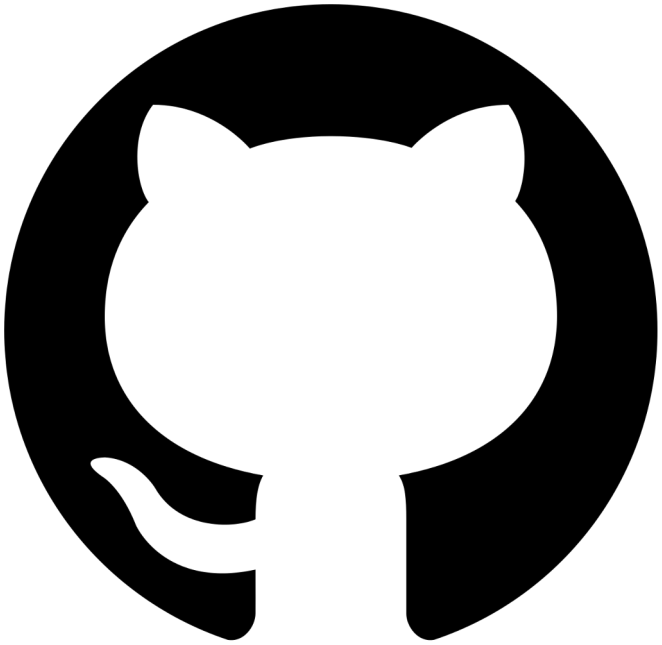
Resultado Final – Vista Ordenador

GitHub

 **He cambiado el nombre del repositorio así que si has estado tenido problemas con mis actividades al corregir será por eso, en esta actividad está corregido para el resto del curso.** 

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



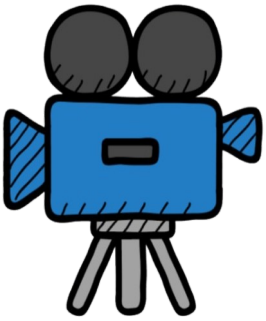
Conclusión

En conclusión, los ViewModels son una molestia de aprenderlos, pero puedes llegar a darle un gran uso, los ICommand parecían más difíciles de lo que parecían pero son más útiles y fáciles de lo que puedes esperas, ha sido una buena actividad por su conocimiento adquirido y su idea principal.

Problemas y/o sugerencias en la actividad

1# - Mi extra a la actividad

Aquí estamos de nuevo, 2 vídeos, uno de su funcionamiento, añadiendo, actualizando, borrando y viendo los posibles errores que da y como reacciona la interfaz para avisarte y poco más, el otro vídeo de explicativo se basará sobre todo en el código principal donde hablaré de como he gestionado para seleccionar el usuario y así poder hacer las funciones que solicita el ejercicio.



Vídeo funcionamiento: https://youtu.be/3EJgtiqHrho
Video explicativo de código: https://youtu.be/7i_r2cY0NXw

Webgrafía

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

- CollectionsView:
https://learn.microsoft.com/es-es/dotnet/maui/user-interface/controls/collectionview/

- 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