"Lista de Tareas" en .NET MAUI

MVVM EN .NET MAUI

Piero Zavala Chira | Desenvolupament d'interfícies | 19/01/2025

Introducción

PROPÓSITO DEL PROYECTO

El objetivo principal del proyecto es desarrollar una aplicación de lista de tareas que permita a los usuarios gestionar sus pendientes de manera eficiente. Los usuarios podrán añadir, eliminar y marcar tareas como completadas, lo que mejora la organización personal y profesional. Este proyecto también busca demostrar el uso del patrón MVVM en .NET MAUI, proporcionando una solución escalable y mantenible.

TECNOLOGÍAS UTILIZADAS

- **.NET MAUI**: Framework multiplataforma para construir aplicaciones nativas.
- **C#**: Lenguaje de programación utilizado para la lógica del negocio.
- **XAML**: Lenguaje de marcado para diseñar la interfaz de usuario.
- **ObservableCollection**: Para gestionar la lista de tareas de forma reactiva.
- **MVVM**: Patrón de diseño utilizado para separar la lógica de la interfaz.
- **Procesador**: 12th Gen Intel(R) Core(TM) i5-12400F 2.50 GHz
- **RAM instalada**: 16,0 GB (15,8 GB usable)
- **Tipo de sistema**: Sistema operativo de 64 bits, procesador basado en x64

Contenido

Organización de la Estructura

PRIMERA PARTE: INTRODUCCIÓN GENERAL

Se contextualiza el proyecto, explicando el propósito y las tecnologías utilizadas. Esta sección establece las bases del proyecto, proporcionando una visión clara del objetivo y los beneficios esperados.

SEGUNDA PARTE: DISEÑO CONCEPTUAL

Se presenta el diseño de la interfaz y las funcionalidades principales de la aplicación. Esta sección incluye esquemas y justificaciones del diseño visual y funcional.

TERCERA PARTE: DESARROLLO TÉCNICO

Se incluyen fragmentos de código comentados que explican la implementación del proyecto. Esta sección está orientada a desarrolladores que desean comprender cómo reproducir y adaptar el proyecto.

CUARTA PARTE: RESULTADOS

Se describen los resultados obtenidos, pruebas realizadas y errores solucionados. Esta parte también incluye una evaluación del cumplimiento de los objetivos iniciales y un resumen de los aprendizajes

Diseño de la Solución

DIAGRAMA O BOCETO INICIAL

Se diseñó un wireframe(Plano pantalla) simple que muestra:

- Una pantalla principal con una lista de tareas.
- Un botón para añadir nuevas tareas.
- Opciones para marcar tareas como completadas o eliminarlas.

EXPLICACIÓN DEL DISEÑO

- StackLayout: Se utilizó para organizar los elementos verticalmente.
- CollectionView: Permite mostrar la lista de tareas de forma eficiente.
- Triggers: Cambian dinámicamente el estilo del texto cuando se marca como completada.
- Colores y fuentes: Se seleccionaron colores contrastantes y una fuente clara para mejorar la accesibilidad.

DISPOSICIÓN ADAPTATIVA

La interfaz se adapta a diferentes tamaños de pantalla utilizando diseños responsivos de .NET MAUI.

Desarrollo Técnico

FRAGMENTOS DE CÓDIGO COMENTADOS

Modelo **TodoItem**

```
public class TodoItem : INotifyPropertyChanged
{
    private string _title;
    private bool _isCompleted;

    public string Title
    {
        get => _title;
        set
        {
            _title = value;
            OnPropertyChanged();
        }
    }

    public bool IsCompleted
    {
        get => _isCompleted;
        set
        {
            _isCompleted = value;
            OnPropertyChanged();
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;

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

Vista **MainPage.xaml**

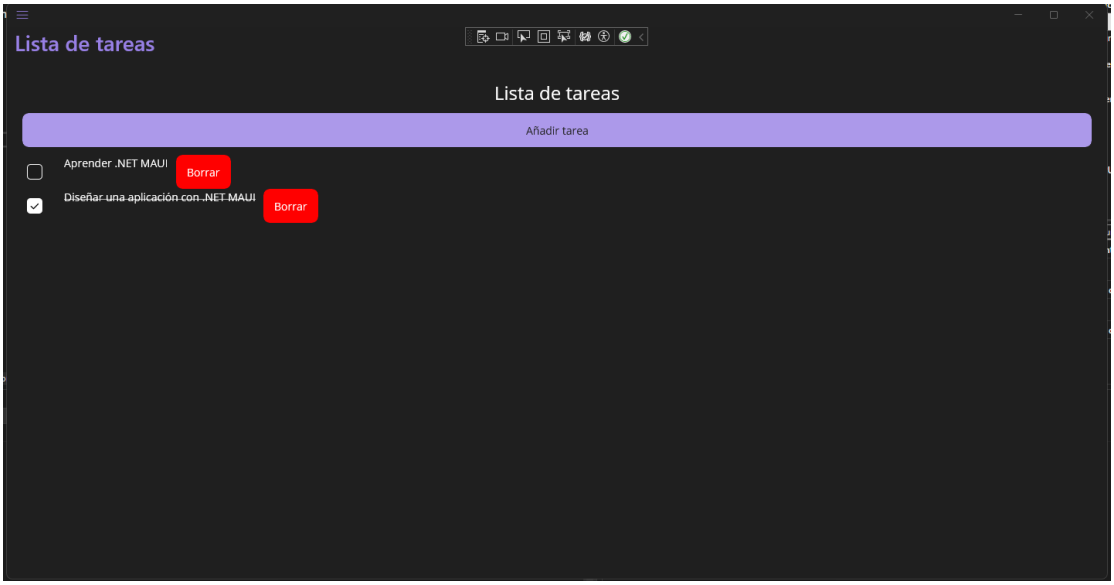
```xaml
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:vm="clr-namespace:TaskListApp.ViewModels"
             x:Class="TaskListApp.Views.MainPage"
             Title="Lista de tareas">

    <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}" />

        <CollectionView ItemsSource="{Binding Items}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <HorizontalStackLayout Spacing="10">
                        <!-- CheckBox para marcar como completado -->
                        <CheckBox IsChecked="{Binding IsCompleted}" />

                        <!-- Label que cambia el estilo del texto según el estado -->
                        <Label Text="{Binding Title}">
                            <Label.Triggers>
                                <!-- Cambia el texto a tachado si está completado -->
                                <DataTrigger TargetType="Label"
                                     Binding="{Binding IsCompleted}"
                                     Value="true">
                                    <Setter Property="TextDecorations" Value="Strikethrough" />
                                </DataTrigger>
                                <!-- Cambia el texto a normal si no está completado -->
                                <DataTrigger TargetType="Label"
                                     Binding="{Binding IsCompleted}"
                                     Value="false">
                                    <Setter Property="TextDecorations" Value="None" />
                                </DataTrigger>
                            </Label.Triggers>
                        </Label>

                        <!-- Botón para borrar la tarea -->
                        <Button Text="Borrar"
                        BackgroundColor="Red"
                        TextColor="White"
                        Command="{Binding Source={RelativeSource AncestorType={x:Type vm:MainPageViewModel}},
Path=DeleteItemCommand}"
                                CommandParameter="{Binding .}" />
                    </HorizontalStackLayout>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>

    </VerticalStackLayout>
</ContentPage>
```

PASOS DE IMPLEMENTACIÓN

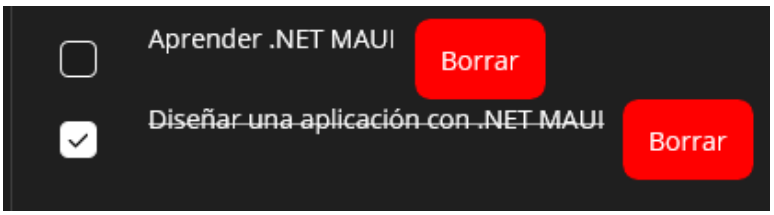
1. Crear el modelo **TodoItem.**
2. Configurar el **ViewModel** con la lista de tareas y los comandos.
3. Diseñar la interfaz en **MainPage.xaml**.
4. Implementar los comandos para añadir y eliminar tareas.
5. Probar la aplicación en dispositivos reales y emuladores.

CAPTURAS DE PANTALLA

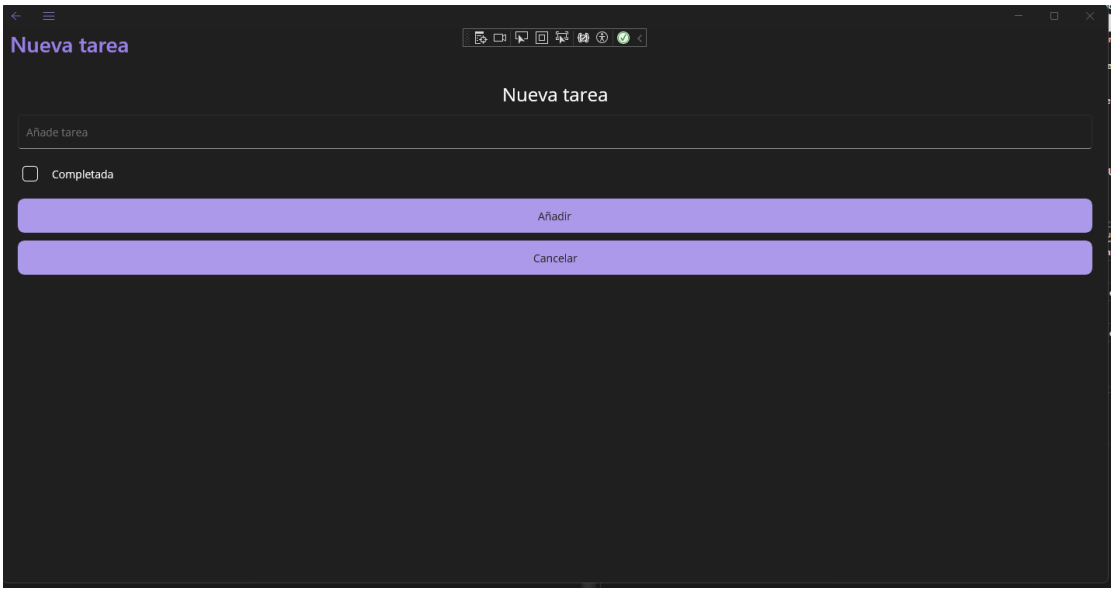
1. Pantalla principal mostrando tareas.



2. Tarea marcada como completada (con texto tachado).



3. Pantalla de añadir una nueva tarea.



Funcionalidades Avanzadas

- **Validación de datos**: Evitar que se agreguen tareas vacías.
- **Navegación**: Uso de **Shell.GoToAsync** para pasar datos entre pantallas.
- **Accesibilidad**: Diseño compatible con lectores de pantalla.

Resultados y Pruebas

RESULTADOS OBTENIDOS

- Una aplicación funcional y responsiva.
- Uso efectivo del patrón MVVM para separar la lógica y la interfaz.

PRUEBAS REALIZADAS

- Dispositivos Android y Windows.
- Pruebas de interacción con el usuario para agregar, eliminar y marcar tareas.

ERRORES Y CORRECCIONES

1. Problemas iniciales con el **BindingContext**. Solucionado configurándolo correctamente.
2. Error al registrar rutas en el **AppShell**. Resuelto con **Routing.RegisterRoute().**

Conclusión

VALORACIÓN PERSONAL

El proyecto ayudó a consolidar conocimientos en .NET MAUI y el patrón MVVM. Se mejoró la habilidad para estructurar código y diseñar interfaces adaptativas.

IMPACTO DEL PROYECTO

Esta aplicación sirve como base para proyectos más complejos y ofrece una herramienta práctica para la gestión personal de tareas.

Repositorio y Bibliografía

REPOSITORIO GITHUB

https://github.com/CevicheConAji/TaskListApp

FUENTES DE INFORMACIÓN

- Documentación oficial de .NET MAUI
- Blogs y foros de programación relacionados con XAML y MVVM.