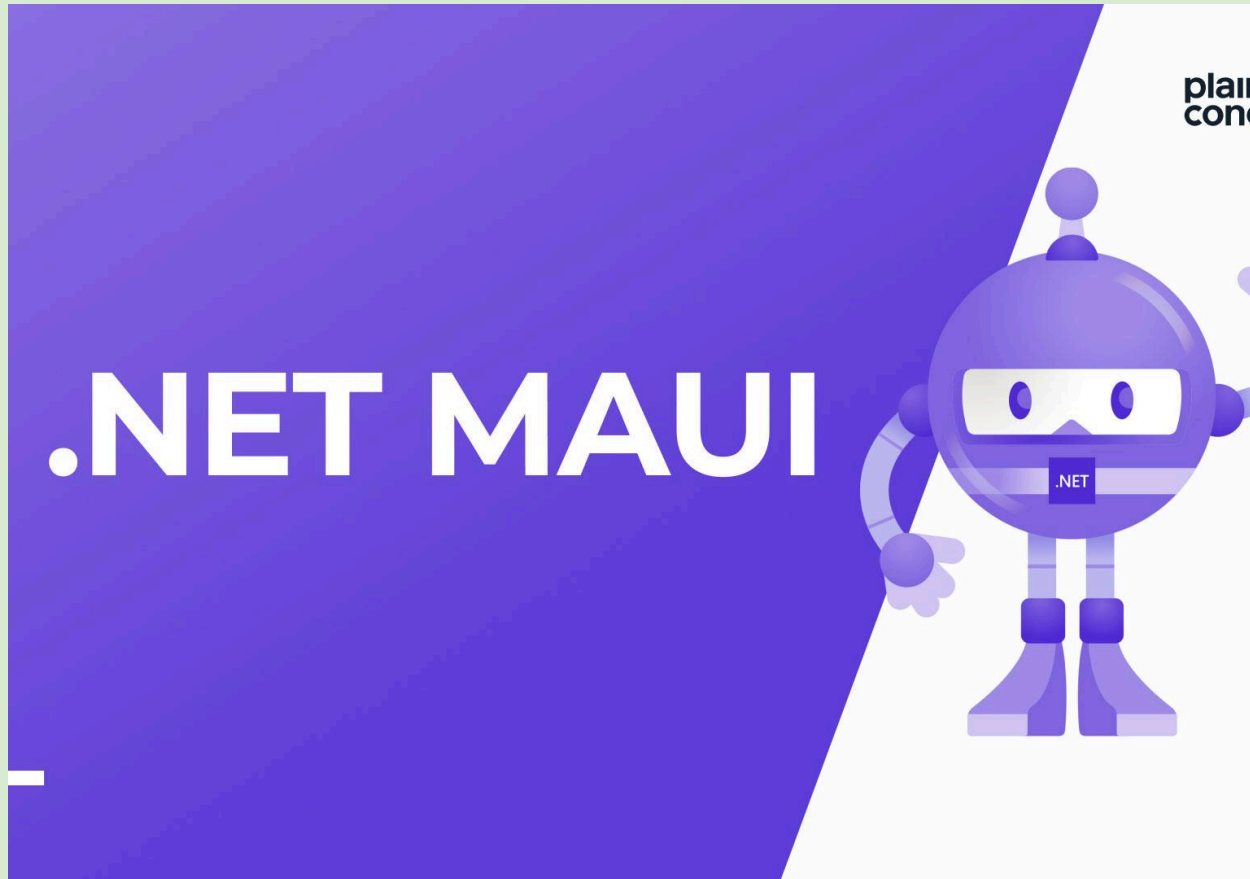


UNIDAD 06 ACTIVIDAD 02




Azael Morell Martínez

19/01/2025

Desarrollo de Interfaces

ÍNDICE:

[ÍNDICE:](#)

[INTRODUCCIÓN](#) 

[OBJETIVOS](#) 

[PLANTEAMIENTOS PREVIOS](#) 

[DIAGRAMA DE GANTT:](#) 

[RESULTADOS](#) 

[VISTA GENERAL DEL PROYECTO:](#)

[SUGERENCIA:](#)

[EL CÓDIGO .XAML:](#)

[EL CÓDIGO .XAML.CS:](#)

[PROBLEMA ENCONTRADO](#) 

[SUGERENCIAS PARA HACER LA APP](#) 

[CONCLUSIÓN](#) 

[WEBGRAFÍA:](#) 

INTRODUCCIÓN

En este proyecto, he desarrollado una aplicación de lista de tareas utilizando .NET MAUI, implementando el patrón de diseño Model-View-ViewModel (MVVM). El objetivo principal de este proyecto era crear una aplicación que permita a los usuarios añadir, visualizar y eliminar tareas de manera sencilla y eficiente. Al seguir el patrón MVVM, hemos asegurado una separación clara entre la lógica de negocio, la presentación y la interfaz de usuario.

OBJETIVOS

- A. Implementar una clase `TodoItem` para representar cada tarea con propiedades para el nombre y el estado de completado.
- B. Crear un `MainPageViewModel` que gestione una colección observable de tareas y proporcione comandos para añadir y eliminar tareas.
- C. Diseñar una página de interfaz de usuario (`MainPage.xaml`) que muestre la lista de tareas y permita al usuario interactuar con ella mediante bindings y comandos.
- D. Implementar una vista modelo adicional (`AddItemNewWindowViewModel`) para gestionar la adición de nuevas tareas en una página separada.
- E. Configurar la navegación entre páginas utilizando el Shell de .NET MAUI para una experiencia de usuario fluida.

PLANTEAMIENTOS PREVIOS

Antes de comenzar, era fundamental tener una comprensión sólida de:

- La estructura y funcionamiento de .NET MAUI.
- Cómo trabajar con XAML para diseñar la interfaz de usuario.
- La implementación de métodos asíncronos en C#.

DIAGRAMA DE GANTT:

<u>Tarea</u>	<u>1-3 días</u>	<u>4-7 días</u>	<u>más de una semana</u>
Desarrollo del código y de la interfaz	xxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxx		
Desarrollo de la documentación	xxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxx		
Testing del funcionamiento	xxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxx		

RESULTADOS 💪

VISTA GENERAL DEL PROYECTO:

✨ Estoy súper emocionado de compartir el diseño de mi app de tareas. La he llevado a cabo en .NET MAUI y ha quedado genial. ¡Déjame contarte más al respecto!

📝 Diseño de la App:

Interfaz Sencilla 🎨: El fondo tiene un color gris oscuro, que le da un aspecto profesional y sobrio.

Título 📋: Arriba, el título dice "Lista de tareas" en letras blancas sobre un fondo morado. Muy claro y directo.

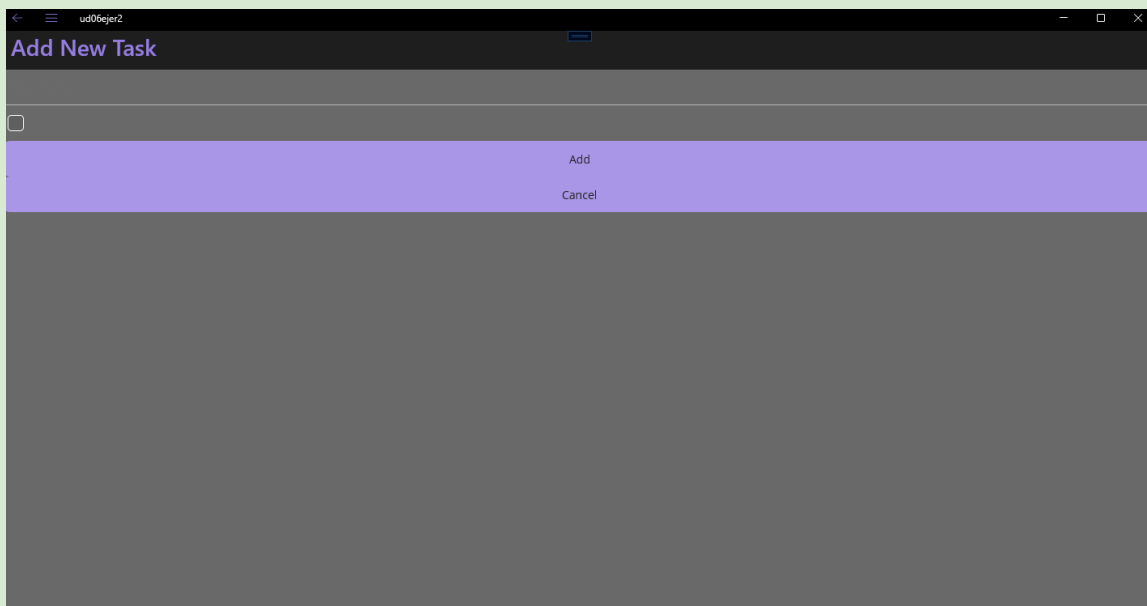
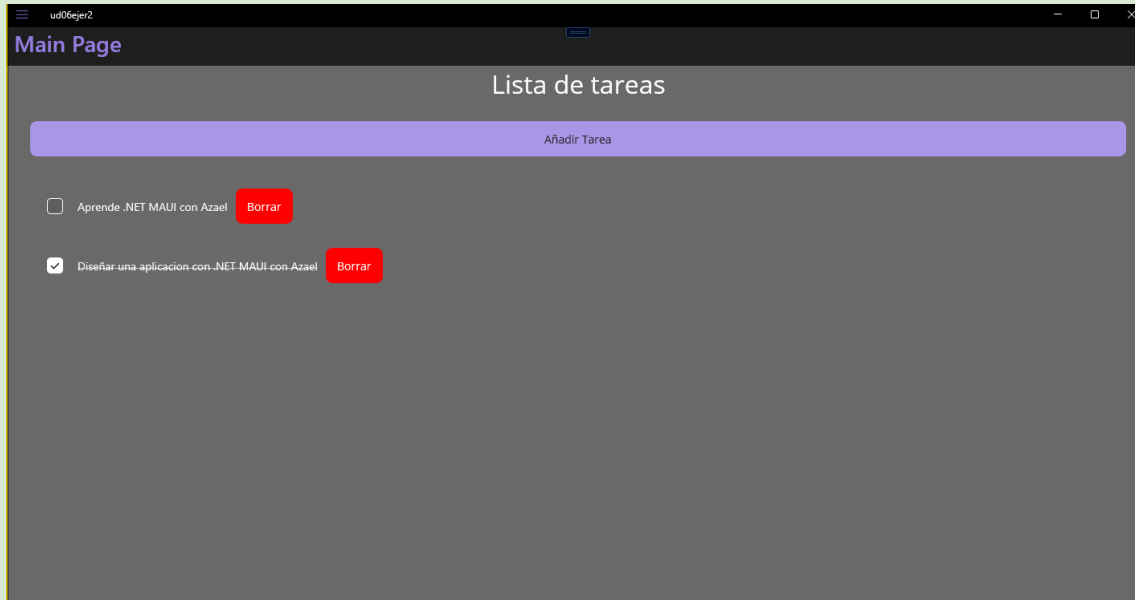
Lista de Tareas 📝: Justo debajo, hay una lista de tareas con opciones para marcar como completadas y botones rojos para eliminar cada tarea.

Botones Funcionales ☀️:

- Botón rojo que dice "Remove" ❌ para eliminar tareas.
- Botón morado que dice "Add Task" ➕ para agregar nuevas tareas.

Página de Nueva Tarea 📝: Al hacer clic en "Add Task", se abre una nueva página con un título que dice "Add New Task" en letras blancas sobre un fondo morado.

Opciones de Tarea 📌: En esta página, hay opciones para "Add" ➕ o "Cancel" ❌ la nueva tarea.



SUGERENCIA:

SI QUIERES VER EL FUNCIONAMIENTO DE ESTA APP. RECOMIENDO VER EL VIDEO LLAMADO “TestingAPP_ListaTareas” QUE ESTÁ SUBIDO EN ESTE MISMO REPOSITORIO DE GITHUB.

EL CÓDIGO .XAML:

He desarrollado una aplicación de gestión de tareas utilizando .NET MAUI. La interfaz de usuario está diseñada para ser sencilla y eficiente, con un fondo de color aqua que le da un toque fresco y moderno.

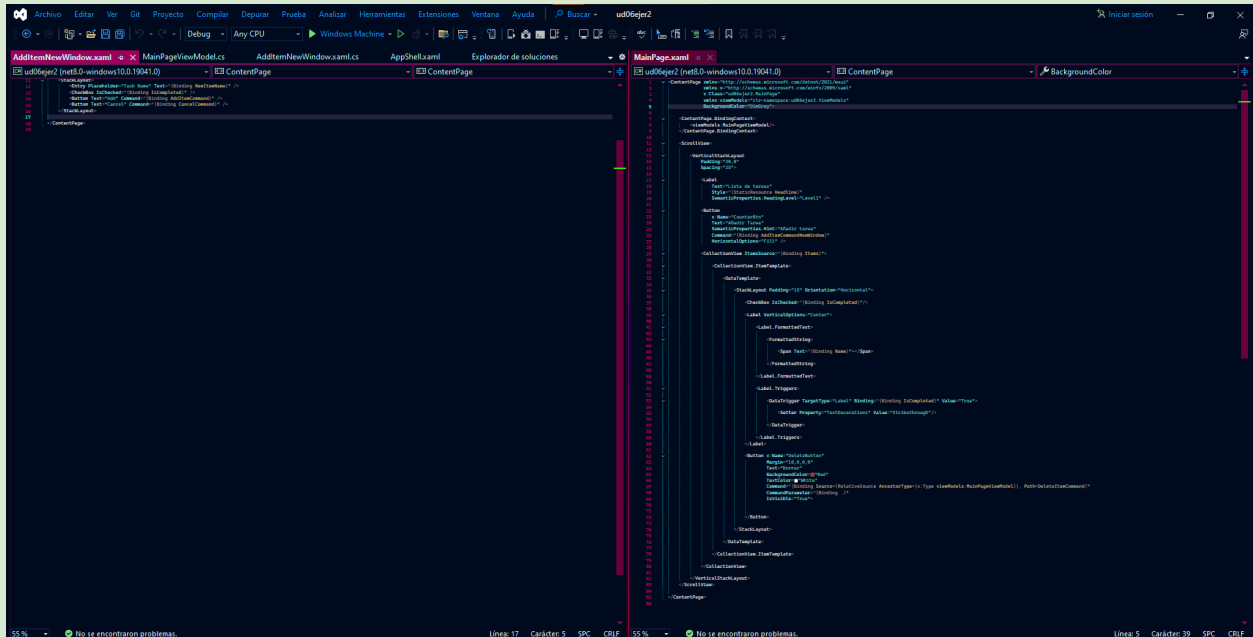
El encabezado de la aplicación, titulado "Lista de tareas", se presenta en texto blanco sobre un fondo morado, proporcionando un contraste claro y profesional. 📝 En la parte superior de la interfaz, se incluye un botón de adición de tareas, definido por la etiqueta `<Button>` y con el texto "Añadir Tarea". ➕ Este botón permite a los usuarios abrir una nueva página donde pueden introducir los detalles de una nueva tarea.

Justo debajo del botón de adición de tareas, se encuentra una lista de tareas, estructurada mediante la etiqueta `<CollectionView>`. 📋 Esta lista muestra cada tarea con un checkbox para marcar su estado de completado, un label que muestra el nombre de la tarea y un botón rojo para eliminar la tarea seleccionada. ✖ Los elementos de la lista están definidos mediante un `DataTemplate`, lo que facilita la personalización de la apariencia de cada tarea.

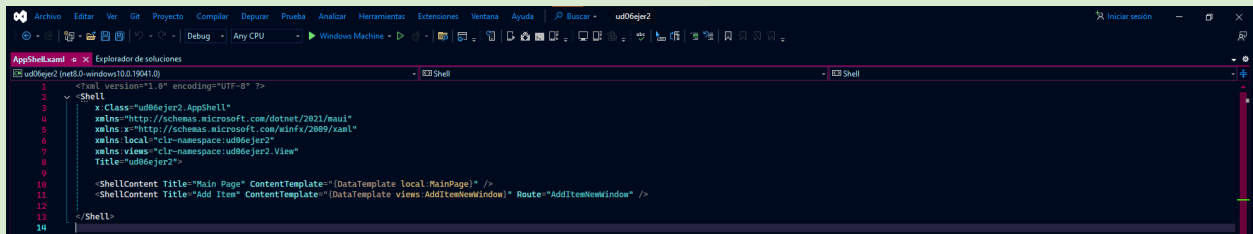
Además, la aplicación incluye una página de nueva tarea donde los usuarios pueden introducir el nombre de la tarea y su estado de completado. 📌 Esta página está diseñada con una interfaz intuitiva que incluye un campo de entrada para el nombre de la tarea (definido por la etiqueta `<Entry>`) y un checkbox para el estado de completado. También se proporcionan dos botones: uno para añadir la tarea y otro para cancelar la acción.

La estructura de navegación de la aplicación está definida en el archivo `AppShell.xaml`, utilizando la etiqueta `<ShellContent>` para especificar las rutas y el contenido de cada página. 🇪🇸 Esto asegura una navegación fluida y coherente entre las diferentes secciones de la aplicación. 🚀

AddItemNewWindow.xaml y MainPage.xaml



AppShell.xaml



EL CÓDIGO .XAML.CS:

Descripción Técnica:

Bibliotecas Importadas :

- System: Para manejar eventos y tipos básicos como EventArgs.
- System.Collections.ObjectModel: Para manejar colecciones observables que reflejan cambios automáticamente en la UI.
- System.Linq: Para métodos útiles en la manipulación de colecciones.
- System.Windows.Input: Para utilizar ICommand en la aplicación.

Configuración Inicial :

- Se define una clase TodoItem que representa un elemento de tarea con propiedades como Name y IsCompleted.
- Se crea una clase MainPageViewModel que maneja la lógica de negocio y comandos para la página principal.
- Se define una clase AddItemNewWindowViewModel para manejar la lógica de agregar nuevas tareas.

Constructor de la Clase MainPageViewModel :

- Se inicializan los componentes definidos en el archivo XAML.
- Se crea una colección observable de tareas (ObservableCollection<TodoItem>) que se actualiza automáticamente en la UI.
- Se definen comandos para agregar y eliminar tareas (AddItemCommandNewWindow y DeleteItemCommand).
- Se vincula el contexto de datos de la página a la clase MainPageViewModel.

Constructor de la Clase AddItemNewWindowViewModel :

- Se inicializan los componentes definidos en el archivo XAML.
- Se define un comando AddItemCommand para agregar un nuevo elemento de tarea y un comando CancelCommand para cancelar la acción.
- Se utiliza QueryProperty para pasar parámetros entre páginas.

Carga de Tareas :

- El método LoadTasks se encarga de obtener todas las tareas almacenadas.
- Se limpia la colección de tareas actual y se añaden las nuevas tareas obtenidas a la colección observable.

Agregar Tareas :

- El método AddTask verifica que el campo de entrada no esté vacío antes de añadir una nueva tarea.
- Crea un nuevo objeto de tarea (TodoItem) y lo añade a la colección observable.
- Navega de regreso a la página principal después de agregar la tarea.

Actualizar Tareas :

- El método UpdateTask verifica que haya una tarea seleccionada y que el campo de entrada no esté vacío.
- Actualiza el nombre de la tarea seleccionada y refleja los cambios en la fuente de datos.
- Recarga la lista de tareas para reflejar las actualizaciones.

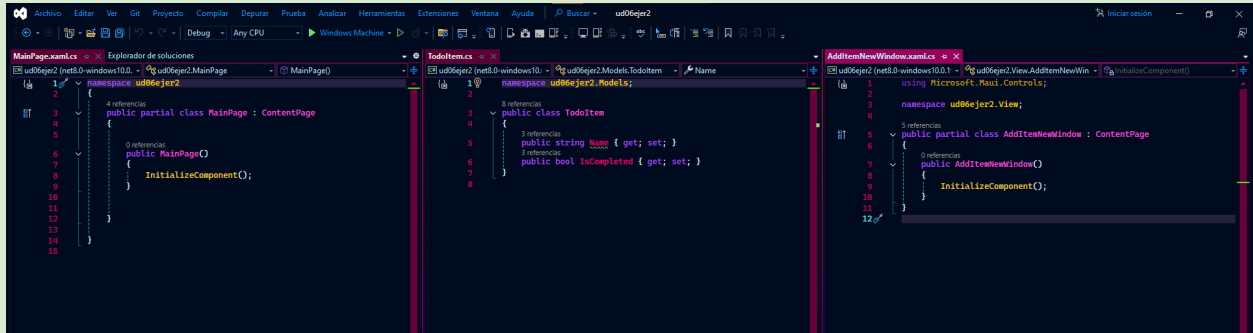
Eliminar Tareas :

- El método DeleteTask verifica que haya una tarea seleccionada.
- Elimina la tarea de la colección observable y actualiza la interfaz de usuario para reflejar la eliminación de la tarea.

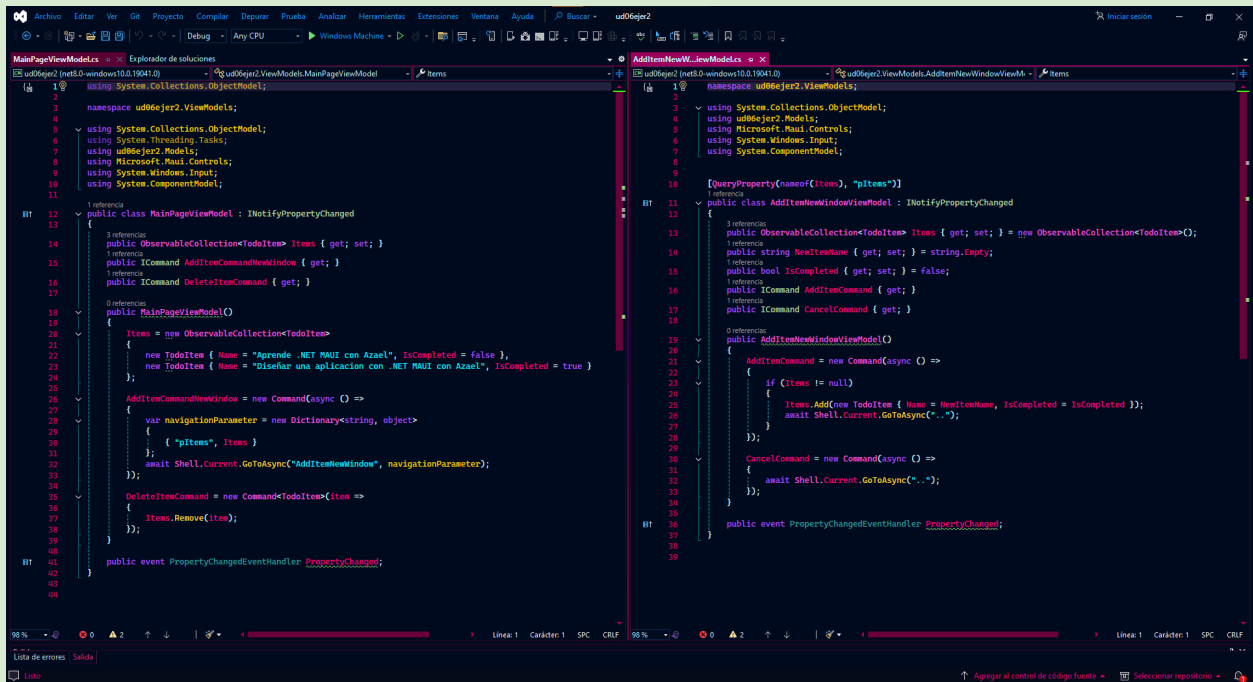
Selección de Tareas :

- El evento OnTaskSelected se dispara al seleccionar una tarea en la lista.
- Muestra el nombre de la tarea seleccionada en el campo de entrada para facilitar su actualización o eliminación.

TodoItem.cs, AddItemNewWindow.xaml.cs y MainPage.xaml.cs



AddItemNewWindowViewModel.xaml.cs y MainPageViewModel.xaml.cs



PROBLEMA ENCONTRADO 🚧

Uno de los principales problemas encontrados durante el desarrollo del proyecto fue la configuración de las rutas de navegación en el AppShell. Aunque la mayoría del código estaba bien estructurado y organizado, la navegación entre páginas no funcionaba correctamente debido a errores en la definición y registro de rutas.

El error más común fue la omisión de la ruta en el archivo AppShell.xaml.cs, lo que impedía que la aplicación encontrara la página AddItemNewWindow. Para solucionar este problema, fue necesario asegurarse de que la ruta estaba correctamente registrada en el archivo AppShell.xaml.cs:

```
Routing.RegisterRoute("AddItemNewWindow", typeof(AddItemNewWindow));
```

Además, también fue importante verificar que las rutas estuvieran correctamente definidas en el archivo AppShell.xaml, asegurando que la navegación estuviera configurada adecuadamente:

SUGERENCIAS PARA HACER LA APP 💡

Validación de entrada de datos:

- Descripción: Implementa una validación para asegurar que los usuarios no puedan añadir tareas vacías o con nombres duplicados.

Uso de CollectionView en lugar de ListView:

- Descripción: Cambiar de ListView a CollectionView puede ofrecer una mejor performance y más flexibilidad en la presentación de la lista de tareas.

Manejo de errores y excepciones:

- Descripción: Implementar un manejo adecuado de errores y excepciones para mejorar la robustez de la aplicación.

CONCLUSIÓN

En resumen, este proyecto ha sido una excelente oportunidad para explorar las capacidades de .NET MAUI y el patrón MVVM. Aunque encontré algunos desafíos en la configuración de la navegación, logré superarlos y crear una aplicación funcional y eficiente. Este proyecto no solo ha mejorado mis habilidades técnicas, sino que también ha proporcionado una base sólida para futuros desarrollos en .NET MAUI.

WEBGRAFÍA:

La documentación ha sido extraída de los recursos colocados por mi profesor Oscar Garcia en la plataforma educativa de Aules.

Enlaces MUY útiles:

<https://learn.microsoft.com/en-us/dotnet/maui/>

<https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/model-view-viewmodel>

<https://github.com/dotnet/maui-samples>