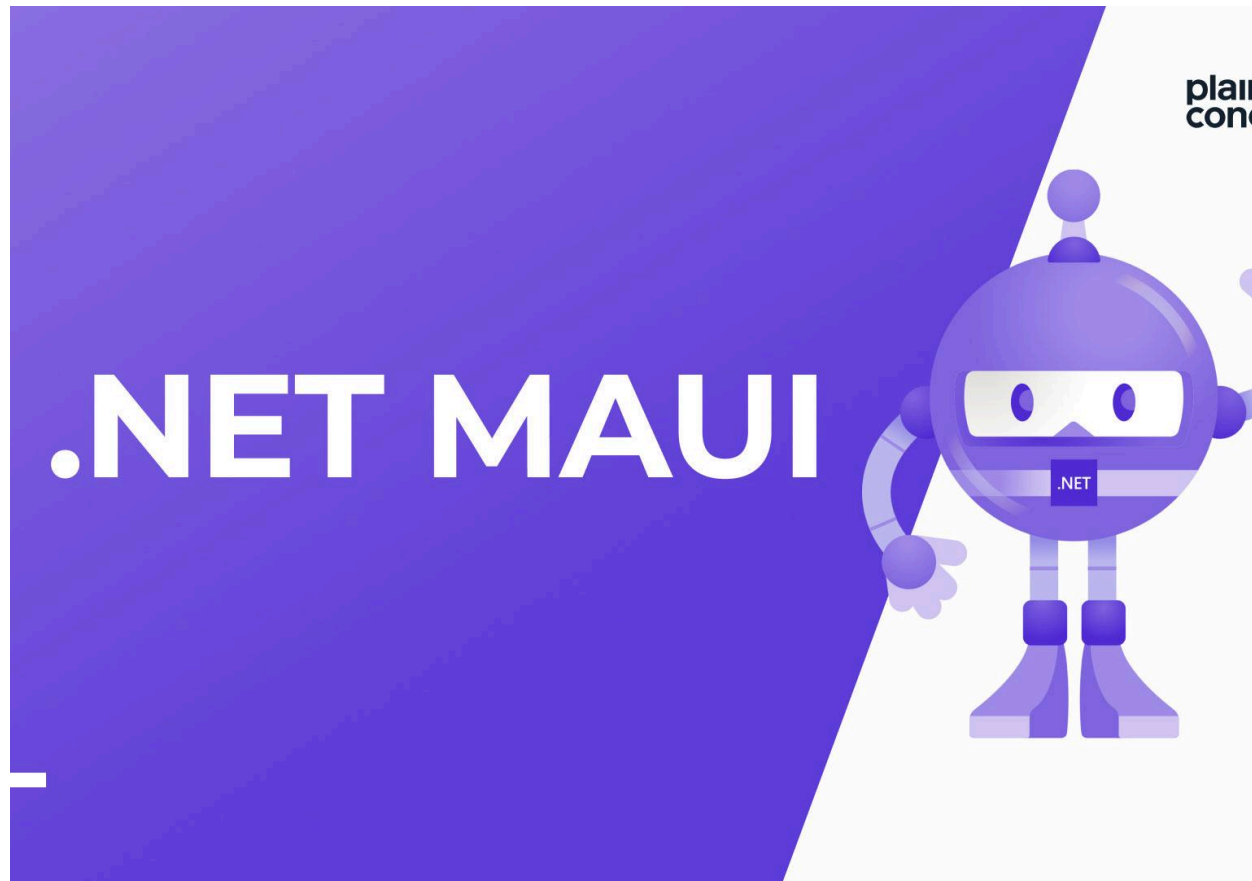


# UNIDAD 02 ACTIVIDAD 03



**Azael Morell Martínez**

20/10/2024

Desarrollo de Interfaces

# ÍNDICE

---

[ÍNDICE !\[\]\(529949c2c3dadbaa4e538e8c643454bc\_img.jpg\)](#)

[INTRODUCCIÓN !\[\]\(3dfb8d66e81160ad61421a3452093d1b\_img.jpg\)](#)

[OBJETIVOS !\[\]\(99f58673407353e96a019fbca558fd72\_img.jpg\)](#)

[DIAGRAMA DE GANTT: !\[\]\(0f848bbd71cef6b345273b16f905912a\_img.jpg\)](#)

[PLANTEAMIENTOS PREVIOS](#)

[RESULTADOS !\[\]\(a870788d6ed9b8fd294b7654a8c8526b\_img.jpg\)](#)

[PÁGINA 1 \(Matricular Curso\)](#)

[.XAML](#)

[.XAML.CS](#)

[Todo el .XAML.CS](#)

[PÁGINA 2 \(Seleccionar Curso\):](#)

[.XAML](#)

[.XAML.CS](#)

[PÁGINA 3 \(Seleccionar Método de Pago\):](#)

[.XAML](#)

[.XAML.CS](#)

[RESULTADOS FINALES: !\[\]\(eabd9f9ababee93effadc3b380fe65fd\_img.jpg\)](#)

[CONCLUSIÓN !\[\]\(83bbbd261710c59db0214aa27b2edc0d\_img.jpg\)](#)

[SUGERENCIAS !\[\]\(166772600a13ad0a433053f90fe45649\_img.jpg\) !\[\]\(98e7c6b0160281a12fbadc337ff6b6c3\_img.jpg\)](#)

[WEBGRAFÍA !\[\]\(291e070cef6c4d5e78fefe4696ef53be\_img.jpg\)](#)

## INTRODUCCIÓN 👁👁

---

El proyecto consiste en desarrollar una aplicación en .NET MAUI que permita simular el proceso de registro básico. La aplicación consta de tres páginas que guían al usuario desde la selección de un curso hasta la selección de un método de pago y finalmente el cálculo del monto total pagado en base a los datos seleccionados. A través de este ejercicio, explorará varios aspectos básicos del desarrollo con .NET MAUI, incluida la navegación entre páginas, el manejo de eventos y la implementación de lógica condicional. Este enfoque estructurado facilita el aprendizaje y mejora las habilidades para desarrollar interfaces intuitivas y funcionales.

## OBJETIVOS 🎯

---

- Diseño de interfaz de usuario en .NET MAUI: cree tres páginas completamente funcionales y bien organizadas con una interfaz amigable y visualmente atractiva.
- Implementar navegación y gestión de datos entre páginas: desarrollar lógica para transferir información entre páginas para que los cursos seleccionados y los métodos de pago se reflejen en la página de Datos de registro.
- Controlar la activación del botón: Configurar el botón “Calcular Precio” para que se active únicamente al seleccionar curso y método de pago, aplicando una lógica que mejora la usabilidad de la aplicación.
- Realizar cálculo de condiciones: Calcula el precio final del curso en función del método de pago seleccionado. Si el usuario opta por pagar con tarjeta podrá disfrutar de un 10% de descuento.

## DIAGRAMA DE GANTT:

Tarea	1-3 días	4-7 días	más de una semana
Desarrollo del código y de la interfaz		XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXX	
Desarrollo de la documentación		XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXX	
Pruebas del funcionamiento	XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXX		

## PLANTEAMIENTOS PREVIOS

Para abordar este proyecto de manera efectiva, es importante tener en cuenta los siguientes aspectos previos:

1. **Conocimiento básico de .NET MAUI:** Es útil conocer el entorno de trabajo de .NET MAUI, sus componentes principales, y el flujo de navegación entre páginas.
2. **Manejo de eventos en .NET MAUI:** Familiarizarse con el manejo de eventos y comandos en las aplicaciones MAUI, lo cual permite capturar las interacciones del usuario y desencadenar la lógica correspondiente.
3. **Binding de datos entre páginas:** En este proyecto se deben actualizar las etiquetas con datos de las páginas de selección, por lo que el entendimiento de los `BindingContext` y los mecanismos para transferir datos entre páginas es fundamental.
4. **Formatos de diseño XAML:** La creación de las páginas requiere entender y usar XAML para diseñar una interfaz que incluya etiquetas, botones, y otros controles visuales.

### PÁGINA 1 (Matricular Curso)

He creado una página en una aplicación .NET MAUI llamada "Datos de Matrícula." Aquí te explico cada parte del código:

1. Encabezado y espacios de nombres: He definido la versión XML y los espacios de nombres necesarios para trabajar con .NET MAUI y XAML. También he especificado la clase a la que pertenece esta página y su título, "Datos de Matrícula". 📖
2. Fondo de la página: He configurado un fondo con un degradado lineal que va del color #3E3E3E al negro (#000000). 🎨
3. Diseño vertical principal: He usado un VerticalStackLayout con relleno y espaciado de 20 unidades. Dentro de este diseño, hay una etiqueta centrada con el texto "🎃 Datos de Matrícula! 🎃", tamaño de fuente 24, en negrita y color de texto #FFA500. ✍️
4. Marco con información del curso: He añadido un Frame con relleno de 15 unidades, radio de esquina de 10, color de fondo #222222 y sombra habilitada. Dentro, hay un StackLayout con espaciado de 15 unidades que contiene varias etiquetas para mostrar el nombre del curso, el precio inicial y la forma de pago. Estos valores se enlazan a propiedades de datos (Binding). 📦
5. Botones de acción: He definido tres botones con diferentes textos y acciones de clic (Clicked). Cada botón tiene un margen, color de fondo, color de texto, radio de esquina, tamaño de fuente y altura específicos. 🖱️
6. Etiqueta para mostrar el precio final: Finalmente, he añadido una etiqueta con nombre lblPrecioFinal para mostrar el precio calculado, centrada y con color de texto #FFA500. 💰

# .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"
              x:Class="UD2EJER5Azael.DatosMatricula"
              Title="Datos de Matrícula">
    <ContentPage.Background>
        <LinearGradientBrush>
            <GradientStop Color="#3E3E3E" Offset="0.0" />
            <GradientStop Color="#000000" Offset="1.0" />
        </LinearGradientBrush>
    </ContentPage.Background>
    <VerticalStackLayout Padding="20" Spacing="20">
        <Label Text="👋 ¡Datos de Matrícula! 👋" FontSize="24" FontAttributes="Bold"
            HorizontalOptions="Center" TextColor="#FFA500" />
        <Frame Padding="15" CornerRadius="10" BackgroundColor="#222222" HasShadow="True">
            <StackLayout Spacing="15">
                <Label Text="Curso" FontSize="Medium" TextColor="#FFA500" />
                <Label x:Name="lblCurso" Text="{Binding NombreCurso}" FontSize="Medium" TextColor="White"
                    />
                <Label Text="Precio inicial" FontSize="Medium" TextColor="#FFA500" />
                <Label x:Name="lblPrecio" Text="{Binding PrecioCurso}" FontSize="Medium"
                    TextColor="White" />
                <Label Text="Forma de pago" FontSize="Medium" TextColor="#FFA500" />
                <Label x:Name="lblFormaPago" Text="{Binding FormaPago}" FontSize="Medium"
                    TextColor="White" />
            </StackLayout>
        </Frame>
        <Button Text="Seleccionar curso" Clicked="OnSeleccionarCursoClicked" Margin="0,10,0,0"
            BackgroundColor="#FFA500" TextColor="Black" CornerRadius="25" FontSize="Medium" HeightRequest="50" />
        <Button Text="Seleccionar forma de pago" Clicked="OnSeleccionarFormaPagoClicked" Margin="0,5,0,0"
            BackgroundColor="#FFA500" TextColor="Black" CornerRadius="25" FontSize="Medium" HeightRequest="50" />
        <Button Text="Calcular precio" x:Name="btnCalcularPrecio" IsEnabled="True"
            Clicked="OnCalcularPrecioClicked" Margin="0,20,0,0" BackgroundColor="#FFC107" TextColor="Black"
            CornerRadius="25" FontSize="Medium" HeightRequest="50" />
        <Label x:Name="lblPrecioFinal" Text="" FontSize="Large" TextColor="#FFA500" Margin="0,10,0,0"
            HorizontalOptions="Center" />
    </VerticalStackLayout>
</ContentPage>
```

# .XAML.CS

Declaración de la clase y propiedades 📄:

```
namespace UD2EJERSAzael;

[QueryProperty(nameof(NombreCurso), "nombreCurso")]
[QueryProperty(nameof(PrecioCurso), "precioCurso")]
[QueryProperty(nameof(FormaPago), "formaPago")]
public partial class DatosMatricula : ContentPage
{
    private string _nombreCurso, _precioCurso, _formaPago;
```

- Definí el espacio de nombres UD2EJERSAzael.
- Utilicé atributos QueryProperty para enlazar propiedades de la clase con parámetros de consulta.
- Declaré tres variables privadas: \_nombreCurso, \_precioCurso y \_formaPago.

## Propiedades públicas 📖:

```
public string NombreCurso
{
    get => _nombreCurso;

    set { _nombreCurso = value; OnPropertyChanged(); lblCurso.Text
= $"curso: {NombreCurso}"; VerificarBotonCalcular(); }
}

public string PrecioCurso
{
    set { _precioCurso = value; OnPropertyChanged(); lblPrecio.Text
= $"precio inicial: {PrecioCurso}"; VerificarBotonCalcular(); }
}

public string FormaPago
{
    get => _formaPago;

    set { _formaPago = value; OnPropertyChanged();
lblFormaPago.Text = $"metodo de pago: {FormaPago}";
VerificarBotonCalcular(); }
}
```

- Definí las propiedades NombreCurso, PrecioCurso y FormaPago con sus respectivos getters y setters.
- Cada setter actualiza la variable privada correspondiente, llama a OnPropertyChanged() para notificar cambios, actualiza etiquetas (lblCurso, lblPrecio, lblFormaPago) y llama a VerificarBotonCalcular().



## Constructor y método OnAppearing 🚀:

```
public DatosMatricula()  
{  
    InitializeComponent();  
    BindingContext = this;  
}  
  
protected override void OnAppearing()  
{  
    base.OnAppearing();  
    lblCurso.Text = $"curso: {NombreCurso}";  
    lblPrecio.Text = $"precio inicial: {PrecioCurso}";  
    lblFormaPago.Text = $"metodo de pago: {FormaPago}";  
    VerificarBotonCalcular();  
}
```

- El constructor inicializa los componentes y establece el contexto de enlace a la propia clase.
- OnAppearing se sobrescribe para actualizar las etiquetas cuando la página aparece.




## Métodos de eventos y lógica de cálculo

```
private async void OnSeleccionarCursoClicked(object sender,
EventArgs e)
    => await
Shell.Current.GoToAsync($"Curso?nombreCurso={NombreCurso}&precioCur
so={PrecioCurso}&formaPago={FormaPago}");

private async void OnSeleccionarFormaPagoClicked(object sender,
EventArgs e)
    => await
Shell.Current.GoToAsync($"FormaPago?nombreCurso={NombreCurso}&preci
oCurso={PrecioCurso}&formaPago={FormaPago}");

private void VerificarBotonCalcular()
    => btnCalcularPrecio.IsEnabled =
!string.IsNullOrEmpty(NombreCurso) &&
!string.IsNullOrEmpty(PrecioCurso) &&
!string.IsNullOrEmpty(FormaPago);

private void OnCalcularPrecioClicked(object sender, EventArgs e)
{
    if (double.TryParse(PrecioCurso.Replace("€", "").Trim(), out
double precioInicial))
        lblPrecioFinal.Text = $"Precio final: {(FormaPago == "Con
tarjeta" ? precioInicial * 0.9 : precioInicial):N2} €";
    else
        lblPrecioFinal.Text = "Error al calcular el precio.";
}
```

- OnSeleccionarCursoClicked y OnSeleccionarFormaPagoClicked son métodos asincrónicos que navegan a otras páginas con parámetros de consulta. 
- VerificarBotonCalcular habilita o deshabilita el botón btnCalcularPrecio según si las propiedades requeridas están llenas. 
- OnCalcularPrecioClicked intenta convertir el precio a un número y calcula el precio final aplicando un descuento si la forma de pago es "Con tarjeta". Si la conversión falla, muestra un mensaje de error. 

```

namespace UD2EJER5Azael;

[QueryProperty(nameof(NombreCurso), "nombreCurso")]
[QueryProperty(nameof(PrecioCurso), "precioCurso")]
[QueryProperty(nameof(FormaPago), "formaPago")]
public partial class DatosMatricula : ContentPage
{
    private string _nombreCurso, _precioCurso, _formaPago;

    public string NombreCurso
    {
        get => _nombreCurso;
        set { _nombreCurso = value; OnPropertyChanged(); lblCurso.Text = $"curso: {NombreCurso}";
        VerificarBotonCalcular(); }
    }

    public string PrecioCurso
    {
        get => _precioCurso;
        set { _precioCurso = value; OnPropertyChanged(); lblPrecio.Text = $"precio inicial:
{PrecioCurso}"; VerificarBotonCalcular(); }
    }

    public string FormaPago
    {
        get => _formaPago;
        set { _formaPago = value; OnPropertyChanged(); lblFormaPago.Text = $"metodo de pago:
{FormaPago}"; VerificarBotonCalcular(); }
    }

    public DatosMatricula()
    {
        InitializeComponent();
        BindingContext = this;
    }

    protected override void OnAppearing()
    {
        base.OnAppearing();
        lblCurso.Text = $"curso: {NombreCurso}";
        lblPrecio.Text = $"precio inicial: {PrecioCurso}";
        lblFormaPago.Text = $"metodo de pago: {FormaPago}";
        VerificarBotonCalcular();
    }

    private async void OnSeleccionarCursoClicked(object sender, EventArgs e)
    => await Shell.Current.GoToAsync($"Curso?nombreCurso={NombreCurso}&precioCurso=
{PrecioCurso}&formaPago={FormaPago}");

    private async void OnSeleccionarFormaPagoClicked(object sender, EventArgs e)
    => await Shell.Current.GoToAsync($"FormaPago?nombreCurso={NombreCurso}&precioCurso=
{PrecioCurso}&formaPago={FormaPago}");

    private void VerificarBotonCalcular()
    => btnCalcularPrecio.IsEnabled = !string.IsNullOrEmpty(NombreCurso) &&
    !string.IsNullOrEmpty(PrecioCurso) && !string.IsNullOrEmpty(FormaPago);

    private void OnCalcularPrecioClicked(object sender, EventArgs e)
    {
        if (double.TryParse(PrecioCurso.Replace("€", "").Trim(), out double precioInicial))
            lblPrecioFinal.Text = $"Precio final: {(FormaPago == "Con tarjeta" ? precioInicial * 0.9 :
precioInicial):N2} €";
        else
            lblPrecioFinal.Text = "Error al calcular el precio.";
    }
}

```

1. Encabezado y espacios de nombres 📄: He definido la versión XML y los espacios de nombres necesarios para trabajar con .NET MAUI y XAML. También he especificado la clase a la que pertenece esta página y su título, "Curso".
2. Fondo de la página 🎨: He configurado un fondo con un degradado lineal que va del color azul (#08A2B2) al naranja (#FF4500).
3. Diseño vertical principal 📏: He usado un `VerticalStackLayout` con relleno de 20 unidades. Dentro de este diseño, hay una etiqueta centrada con el texto "Selecciona un curso", tamaño de fuente grande y color de texto naranja. Añadí un `BoxView` para crear espacio entre los elementos.
4. Diseño horizontal con botones de imagen 🖼️: He añadido un `HorizontalStackLayout` centrado horizontalmente con un espaciado de 20 unidades. Contiene dos `Frame` que encapsulan `ImageButton` con imágenes de cursos (`cursor1.jfif` y `curso2.jfif`), con sombras y eventos de clic.
5. Imagen al final de la página 🖼️: Finalmente, he añadido una imagen al final de la página con la fuente `calabazasin fondo.png`, centrada horizontalmente y alineada al final verticalmente.

# .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"
              x:Class="UD2EJER5Azael.Cursor"
              Title="Curso">
    <ContentPage.Background>
        <LinearGradientBrush>
            <GradientStop Color="#8A2BE2" Offset="0.0" />
            <GradientStop Color="#FF4500" Offset="1.0" />
        </LinearGradientBrush>
    </ContentPage.Background>
    <VerticalStackLayout Padding="20">
        <Label Text="Selecciona un curso" FontSize="Large" TextColor="Orange"
HorizontalOptions="Center" />
        <BoxView HeightRequest="20" BackgroundColor="Transparent" />
        <HorizontalStackLayout HorizontalOptions="Center" Spacing="20">
            <Frame Padding="0" BackgroundColor="Transparent" BorderColor="Orange" CornerRadius="20"
HasShadow="True">
                <ImageButton Source="curso1.jfif" Clicked="OnCurso1Clicked" WidthRequest="250"
HeightRequest="250">
                    <ImageButton.Shadow>
                        <Shadow Brush="Black" Opacity="0.4" Offset="4,4" Radius="10"/>
                    </ImageButton.Shadow>
                </ImageButton>
            </Frame>
            <Frame Padding="0" BackgroundColor="Transparent" BorderColor="Orange" CornerRadius="20"
HasShadow="True">
                <ImageButton Source="curso2.jfif" Clicked="OnCurso2Clicked" WidthRequest="250"
HeightRequest="250">
                    <ImageButton.Shadow>
                        <Shadow Brush="Black" Opacity="0.4" Offset="4,4" Radius="10"/>
                    </ImageButton.Shadow>
                </ImageButton>
            </Frame>
        </HorizontalStackLayout>
        <Image Source="calabazasinfondo.png" WidthRequest="50" HeightRequest="50"
HorizontalOptions="Center" VerticalOptions="End" />
    </VerticalStackLayout>
</ContentPage>
```

## .XAML.CS

Declaración de la clase y propiedades 🛠️: He creado una clase parcial llamada Curso que hereda de ContentPage. Dentro de esta clase, he definido un constructor que inicializa los componentes de la página.

Eventos de clic 🚀: He añadido dos métodos asincrónicos, OnCurso1Clicked y OnCurso2Clicked, que se ejecutan cuando se hace clic en los botones correspondientes. Estos métodos utilizan Shell.Current.GoToAsync para navegar a otra página, pasando parámetros de consulta con los valores de nombreCurso y precioCurso.

```
namespace UD2EJER5Azael;

public partial class Curso : ContentPage
{
    public Curso()
    {
        InitializeComponent();
    }

    private async void OnCurso1Clicked(object sender, EventArgs e)
    {
        await Shell.Current.GoToAsync($"..?nombreCurso=Curso
1&precioCurso=100");
    }

    private async void OnCurso2Clicked(object sender, EventArgs e)
    {
        await Shell.Current.GoToAsync($"..?nombreCurso=Curso
2&precioCurso=150");
    }
}
```

### PÁGINA 3 (Seleccionar Método de Pago):

---

Fondo de la página 🎨: He configurado un fondo con un degradado lineal que va del color naranja (#FF6F00) al púrpura oscuro (#4A148C) y negro (#000000).

Diseño vertical principal 📏: He usado un `VerticalStackLayout` con relleno de 20 unidades. Dentro de este diseño, hay una etiqueta centrada con el texto "Selecciona una forma de pago", tamaño de fuente grande y color de texto naranja. Añadí un `BoxView` para crear espacio entre los elementos.

Diseño horizontal con botones de imagen 🖼️: He añadido un `HorizontalStackLayout` centrado horizontalmente con un espaciado de 30 unidades. Contiene dos `Frame` que encapsulan `ImageButton` con imágenes de formas de pago (`efectivo.jfif` y `tarjeta.jfif`), con sombras y eventos de clic.

Imagen al final de la página 🖼️: Finalmente, he añadido una imagen al final de la página con la fuente `calabazasinfondo.png`, centrada horizontalmente y alineada al final verticalmente.

# .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"
  x:Class="UD2EJER5Azael.FormaPago"
  Title="Forma de Pago">
  <ContentPage.Background>
    <LinearGradientBrush>
      <GradientStop Color="#FF6F00" Offset="0.0" />
      <GradientStop Color="#4A148C" Offset="0.5" />
      <GradientStop Color="#000000" Offset="1.0" />
    </LinearGradientBrush>
  </ContentPage.Background>
  <VerticalStackLayout Padding="20">
    <Label Text="Selecciona una forma de pago" FontSize="Large" TextColor="Orange"
HorizontalOptions="Center" />
    <BoxView HeightRequest="20" BackgroundColor="Transparent" />
    <HorizontalStackLayout Spacing="30" VerticalOptions="Center" HorizontalOptions="Center">
      <Frame CornerRadius="20" Padding="10" BackgroundColor="#333" HasShadow="True">
        <ImageButton Source="efectivo.jfif" Clicked="OnPagoEfectivoClicked" WidthRequest="100"
HeightRequest="100" />
      </Frame>
      <Frame CornerRadius="20" Padding="10" BackgroundColor="#333" HasShadow="True">
        <ImageButton Source="tarjeta.jfif" Clicked="OnPagoTarjetaClicked" WidthRequest="100"
HeightRequest="100" />
      </Frame>
    </HorizontalStackLayout>
    <Image Source="calabazasin fondo.png" WidthRequest="50" HeightRequest="50"
HorizontalOptions="Center" VerticalOptions="End" />
  </VerticalStackLayout>
</ContentPage>
```



## .XAML.CS

Declaración de la clase y propiedades 🛠️: He creado una clase parcial llamada FormaPago que hereda de ContentPage. Dentro de esta clase, he definido dos propiedades públicas, NombreCurso y PrecioCurso, y he utilizado el atributo QueryProperty para enlazarlas con parámetros de consulta.

Constructor de la clase 🚀: He añadido un constructor que inicializa los componentes de la página con InitializeComponent().

Eventos de clic 🎯: He definido dos métodos asincrónicos privados, OnPagoEfectivoClicked y OnPagoTarjetaClicked, que se ejecutan cuando se hace clic en los botones correspondientes. Estos métodos utilizan Shell.Current.GoToAsync para navegar a una nueva página, pasando parámetros de consulta con los valores de NombreCurso, PrecioCurso y formaPago.

```
namespace UD2EJER5Azael;

[QueryProperty(nameof(NombreCurso), "nombreCurso")]
[QueryProperty(nameof(PrecioCurso), "precioCurso")]
public partial class FormaPago : ContentPage
{
    public string NombreCurso { get; set; }
    public string PrecioCurso { get; set; }

    public FormaPago()
    {
        InitializeComponent();
    }

    private async void OnPagoEfectivoClicked(object sender, EventArgs e)
    {
        // Reenvía los datos que ya se encuentran en las propiedades de la página
        await Shell.Current.GoToAsync($"..?nombreCurso={Uri.EscapeDataString(NombreCurso)}&precioCurso={Uri.EscapeDataString(PrecioCurso)}&formaPago=Al contado");
    }

    private async void OnPagoTarjetaClicked(object sender, EventArgs e)
    {
        await Shell.Current.GoToAsync($"..?nombreCurso={Uri.EscapeDataString(NombreCurso)}&precioCurso={Uri.EscapeDataString(PrecioCurso)}&formaPago=Con tarjeta");
    }
}
```

## RESULTADOS FINALES:

UDZERSAzel

Datos de Matrícula

¡Datos de Matrícula!

Curso

curso: Curso 1

Precio inicial

precio inicial: 100

Forma de pago

metodo de pago: Al contado

Seleccionar curso

Seleccionar forma de pago

Calcular precio

Precio final: 100,00 €

UDZERSAzel

Datos de Matrícula

¡Datos de Matrícula!

Curso

curso: Curso 2

Precio inicial

precio inicial: 150

Forma de pago

metodo de pago: Con tarjeta

Seleccionar curso

Seleccionar forma de pago

Calcular precio

Precio final: 135,00 €

## CONCLUSIÓN

---

Este proyecto me ha permitido desarrollar una aplicación básica, pero completa, en .NET MAUI que combina diseño de interfaz, navegación y lógica. Al completarlo, he afianzado mi conocimiento sobre cómo estructurar aplicaciones en MAUI y he ganado experiencia en el diseño de interfaces de usuario que mejoran la experiencia del usuario al interactuar con la app. Además, la construcción de una lógica de habilitación y cálculo en función de las selecciones me ha enseñado a manejar flujos de usuario de manera efectiva.

## SUGERENCIAS

---

En el desarrollo de esta aplicación, he seguido varias prácticas recomendadas para mejorar la estructura y legibilidad del código. Una de estas prácticas ha sido la de nombrar las variables de acuerdo a su visibilidad y propósito. Las variables locales comienzan en minúsculas, mientras que las variables públicas empiezan con un guion bajo y están en mayúscula, lo que facilita identificar rápidamente su alcance en el código.

Para la navegación entre páginas, he utilizado `QueryProperty` para enlazar datos entre ellas, asegurándome de que las propiedades estén correctamente asignadas y enlazadas, como en el siguiente ejemplo: `[QueryProperty(nameof(seleccionarCurso), "curs")]`. Esto permite una comunicación clara entre las páginas y una actualización automática de los valores en los Labels.

Además, he implementado el "App Shell" para organizar todas las páginas de la aplicación, facilitando el acceso a cada una mediante "routing". Este shell utiliza `ContentTemplate`, `DataTemplate` y `Route` para definir las rutas de cada página y establecer una estructura clara y modular.

Finalmente, he usado `Binding` en los Labels de la página principal para que la información seleccionada en las páginas de "Curso" y "Forma de Pago" se refleje automáticamente sin necesidad de actualizar manualmente los valores. Esto permite que los datos se sincronicen de manera fluida, mejorando la experiencia del usuario y la eficiencia del código.

Aquí tienes una webgrafía con enlaces más útiles que vi para consultar la información clave sobre .NET MAUI, Shell, Binding, Routing y mejores prácticas de código en .NET:

1. **Documentación oficial de .NET MAUI**

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

La documentación oficial de Microsoft para .NET MAUI proporciona guías detalladas sobre el desarrollo de aplicaciones con MAUI, incluyendo configuración inicial, control de interfaz y navegación entre páginas.

2. **Uso de App Shell en .NET MAUI**

<https://learn.microsoft.com/es-es/dotnet/maui/fundamentals/shell/>

Esta sección explica el funcionamiento del App Shell, sus propiedades y cómo estructurar aplicaciones que usan navegación basada en rutas, ContentTemplate, DataTemplate y rutas personalizadas.

3. **Bindings en .NET MAUI**

<https://learn.microsoft.com/es-es/dotnet/maui/fundamentals/data-binding/>

El binding es fundamental para conectar propiedades de la interfaz con datos en el código. Aquí puedes aprender cómo aplicar Binding en etiquetas (Labels), botones y otros controles para lograr que el contenido de la interfaz se actualice dinámicamente.

4. **Navegación con rutas y QueryProperty**

<https://learn.microsoft.com/es-es/dotnet/maui/fundamentals/shell/navigation>

Este recurso ofrece una introducción completa a la navegación en MAUI, mostrando cómo usar rutas, pasar parámetros entre páginas y aplicar QueryProperty para enlazar propiedades en diferentes vistas.

5. **Buenas prácticas de código en C#: Convenciones de nombrado**

<https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/coding-style/coding-conventions>

Microsoft ofrece estas guías de estilo de código para C# que incluyen convenciones de nombres para variables, clases y métodos, lo cual es esencial para mantener la consistencia en el código.