

.NET MAUI

.NET Multi-platform App UI

Mobile app development

Native

Apps specifiek ontwikkeld voor één platform, met behulp van platform-specifieke programmeertalen en ontwikkelomgevingen.

Hybrid

Mobiele apps gebaseerd op webtechnologieën die draaien binnen een native container (bijvoorbeeld een ingebouwde browser in de app).

Web

Apps die via een mobiele browser worden uitgevoerd, gebouwd met webtechnologieën (HTML, CSS, JavaScript).

Cross-platform native

Apps die worden ontwikkeld met frameworks die echte native interfaces genereren vanuit één gedeelde codebasis.

Native

Apps specifiek ontwikkeld voor één platform, met behulp van platform-specifieke programmeertalen en ontwikkelomgevingen.

Voorbeelden

- iOS: Swift, Objective-C (via Xcode)
- Android: Kotlin, Java (via Android Studio)

Voordelen

- Beste prestaties
- Optimale toegang tot alle platformfuncties
- Meest vloeiende gebruikerservaring

Nadelen

- Aparte codebasis per platform
- Meer kosten en tijd nodig voor ontwikkeling en onderhoud

Hybrid

Mobiele apps gebaseerd op webtechnologieën die draaien binnen een native container (bijvoorbeeld een ingebouwde browser in de app).

Voorbeelden

- Apache Cordova
- Ionic Framework

Voordelen

- Eén gedeelde codebasis voor meerdere platformen
- Snelle ontwikkelcyclus

Nadelen

- Lagere prestaties vergeleken met native
- Beperkte native API-integratie, afhankelijk van plugins

Web

Apps die via een mobiele browser worden uitgevoerd, gebouwd met webtechnologieën (HTML, CSS, JavaScript).

Voorbeelden

- Progressive Web Apps (PWA's)

Voordelen

- Eén codebasis voor alle platformen (ook webbrowser)
- Snelle updates zonder installatie

Nadelen

- Beperkte toegang tot native functionaliteiten
- Minder vloeiende gebruikerservaring en lagere prestaties

Cross-platform native

Apps die worden ontwikkeld met frameworks die echte native interfaces genereren vanuit één gedeelde codebasis.

Voorbeelden

- .NET MAUI
- Flutter
- React Native
- Xamarin (voorganger van MAUI)

Voordelen

- Native prestaties (vergelijkbaar met puur native)
- Eén gedeelde codebasis voor meerdere platformen (Android, iOS, Windows, macOS)
- Gebruik van echte native UI-elementen, waardoor apps niet van native apps te onderscheiden zijn

Nadelen

- Minder directe controle vergeleken met puur native
- Afhankelijkheid van het gekozen framework

.NET MAUI

Multi-Platform App UI

Één UI framework om voor elk platform in te zetten.

Single project structuur

Één project voor Android, iOS, Windows en MacOS.

Xamarin.Forms

Xamarin.Forms is vernieuwd en uitgebreid naar .NET MAUI.

Één gedeelde codebase

Één codebase in uitsluitend C# en XAML (zæməl); Extensible Application Markup Language

Basisbegrippen

XAML

XAML is een op XML gebaseerde taal voor het definiëren van gebruikersinterfaces in .NET MAUI-apps. Hiermee beschrijf je visueel en gestructureerd hoe schermen opgebouwd worden, los van je app-logica.

Layouts

Layouts bepalen hoe visuele elementen zoals knoppen, labels en invoervelden binnen een scherm worden gerangschikt. Ze zorgen ervoor dat je interface zich aanpast aan verschillende schermgroottes en oriëntaties.

Navigatie

Navigatie regelt hoe gebruikers van het ene scherm naar het andere gaan binnen je app, bijvoorbeeld via tabs of menu's. Met behulp van AppShell in .NET MAUI kun je navigatie eenvoudig definiëren en beheren.

Views

Views zijn visuele componenten zoals knoppen, invoervelden, labels of sliders waarmee gebruikers interactie hebben. Ze bieden functionaliteit aan je app en zorgen voor de communicatie tussen gebruiker en applicatie.

Basisbegrippen

XAML

XAML is een op XML gebaseerde taal voor het definiëren van gebruikersinterfaces in .NET MAUI-apps. Hiermee beschrijf je visueel en gestructureerd hoe schermen opgebouwd worden, los van je app-logica.

Layouts

Layouts bepalen hoe visuele elementen zoals knoppen, labels en invoervelden binnen een scherm worden gerangschikt. Ze zorgen ervoor dat je interface zich aanpast aan verschillende schermgroottes en oriëntaties.

Navigatie

Navigatie regelt hoe gebruikers van het ene scherm naar het andere gaan binnen je app, bijvoorbeeld via tabs of menu's. Met behulp van AppShell in .NET MAUI kun je navigatie eenvoudig definiëren en beheren.

Views

Views zijn visuele componenten zoals knoppen, invoervelden, labels of sliders waarmee gebruikers interactie hebben. Ze bieden functionaliteit aan je app en zorgen voor de communicatie tussen gebruiker en applicatie.

XAML

Deze voorbeelden illustreren hoe je met XAML helder, compact en gestructureerd gebruikersinterfaces kunt beschrijven.

```
<Label Text="Welkom bij .NET MAUI!"  
FontSize="24"  
HorizontalOptions="Center"  
VerticalOptions="Center"/>
```

```
<Button Text="Klik mij!"  
Clicked="OnButtonClicked"  
BackgroundColor="#0d6efd"  
TextColor="White"  
Padding="10"/>
```

```
<Image Source="dotnet_bot.png"  
WidthRequest="200"  
HeightRequest="200"  
HorizontalOptions="Center"  
VerticalOptions="Center"/>
```

Basisbegrippen

XAML

XAML is een op XML gebaseerde taal voor het definiëren van gebruikersinterfaces in .NET MAUI-apps. Hiermee beschrijf je visueel en gestructureerd hoe schermen opgebouwd worden, los van je applicatie-logica.

Layouts

Layouts bepalen hoe visuele elementen zoals knoppen, labels en invoervelden binnen een scherm worden gerangschikt. Ze zorgen ervoor dat je interface zich aanpast aan verschillende schermgroottes en oriëntaties.

Navigatie

Navigatie regelt hoe gebruikers van het ene scherm naar het andere gaan binnen je app, bijvoorbeeld via tabs of menu's. Met behulp van AppShell in .NET MAUI kun je navigatie eenvoudig definiëren en beheren.

Views

Views zijn visuele componenten zoals knoppen, invoervelden, labels of sliders waarmee gebruikers interactie hebben. Ze bieden functionaliteit aan je app en zorgen voor de communicatie tussen gebruiker en applicatie.

StackLayout

Voor het 'stapelen' van elementen in horizontale of verticale richting.

```
<StackLayout Orientation="Vertical" Spacing="10">
    <Label Text="Eerste element"/>
    <Label Text="Tweede element"/>
    <Button Text="Klik hier"/>
</StackLayout>
```

```
<StackLayout Orientation="Vertical"
            Padding="20"
            Spacing="10">
    <Entry Placeholder="Gebruikersnaam"/>
    <Entry Placeholder="Wachtwoord"
          IsPassword="True"/>
    <Button Text="Inloggen"
           BackgroundColor="#198754"
           TextColor="White"/>
</StackLayout>
```

GridLayout

*Om het scherm te verdelen in rijen en kolommen.
Elk element kan hiermee in een ‘vak’ worden
geplaatst.*



```
<Grid RowDefinitions="Auto,Auto"  
      ColumnDefinitions="*,*"  
      RowSpacing="10"  
      ColumnSpacing="10">  
  <Label Text="Naam:"  
         Grid.Row="0"  
         Grid.Column="0"/>  
  <Entry Placeholder="Vul naam in"  
        Grid.Row="0"  
        Grid.Column="1"/>  
  <Label Text="Leeftijd:"  
         Grid.Row="1"  
         Grid.Column="0"/>  
  <Entry Placeholder="Vul leeftijd in"  
        Grid.Row="1"  
        Grid.Column="1"/>  
</Grid>
```

FlexLayout

De meest responsive layout met flexibele uitlijning.
Vergelijkbaar met de werking van flexbox in CSS.

```
<FlexLayout Direction="Row"
             Wrap="Wrap"
             JustifyContent="SpaceAround"
             AlignItems="Center">
    <Button Text="Knop 1"/>
    <Button Text="Knop 2"/>
    <Button Text="Knop 3"/>
</FlexLayout>
```

```
<FlexLayout Direction="Column"
             AlignItems="Start"
             Padding="20">
    <Label Text="Naam:"/>
    <Entry Placeholder="Voer je naam in"/>
    <Label Text="E-mail:"/>
    <Entry Placeholder="Voer je e-mailadres in"/>
    <Button Text="Verstuur"
           BackgroundColor="#0d6efd"/>
</FlexLayout>
```

AbsoluteLayout

De layout met de meeste vrijheid voor precies positioneren maar ook de grootste kans op mismatches.



```
<AbsoluteLayout>
    <BoxView Color="Blue"
        AbsoluteLayout.LayoutBounds="20,20,100,100"
        AbsoluteLayout.LayoutFlags="None"/>
    <Label Text="Hallo MAUI"
        AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
        AbsoluteLayout.LayoutFlags="PositionProportional"/>
</AbsoluteLayout>
```

AbsoluteLayout

- **LayoutBounds** "*x, y, width, height*": coördinaten en grootte van het element.
- **LayoutFlags** bepaalt hoe die waarden worden geïnterpreteerd:
 - *None*: vaste waarden voor positie en grootte.
 - *PositionProportional*: x en y zijn percentages van de breedte en hoogte van de container (0.5 = 50%).
 - *SizeProportional*: breedte en/of hoogte zijn proportioneel aan de grootte van de container.
 - *All*: positie én grootte zijn proportioneel.

```
<AbsoluteLayout>
    <!-- Blauw vierkant met vaste positie en grootte -->
    <BoxView Color="Blue"
        AbsoluteLayout.LayoutBounds="50,100,100,100"
        AbsoluteLayout.LayoutFlags="None"/>
    <!-- Label gecentreerd in het midden van het scherm -->
    <Label Text="Hallo vanuit het midden!"
        FontSize="18"
        TextColor="White"
        BackgroundColor="DarkSlateGray"
        Padding="10"
        AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
        AbsoluteLayout.LayoutFlags="PositionProportional"/>
    <!-- Rode knop die 80% breed is en onderin het scherm staat -->
    <Button Text="Doorgaan"
        BackgroundColor="Red"
        TextColor="White"
        AbsoluteLayout.LayoutBounds="0.1,1,0.8,50"
        AbsoluteLayout.LayoutFlags="All"/>
</AbsoluteLayout>
```

AbsoluteLayout

LayoutBounds en **LayoutFlags** zijn sterk aan elkaar gekoppeld. Ze bepalen of **x**, **y** en/of **width**, **height** verhoudingsgewijs bepaald moeten worden of als *absolute* waardes.



```
<AbsoluteLayout>
    <!-- Blauw vierkant met vaste positie en grootte -->
    <BoxView Color="Blue"
        AbsoluteLayout.LayoutBounds="50,100,100,100"
        AbsoluteLayout.LayoutFlags="None"/>
    <!-- Label gecentreerd in het midden van het scherm -->
    <Label Text="Hallo vanuit het midden!"
        FontSize="18"
        TextColor="White"
        BackgroundColor="DarkSlateGray"
        Padding="10"
        AbsoluteLayout.LayoutBounds="0.5,0.5,-1,-1"
        AbsoluteLayout.LayoutFlags="PositionProportional"/>
    <!-- Rode knop die 80% breed is en onderin het scherm staat -->
    <Button Text="Doorgaan"
        BackgroundColor="Red"
        TextColor="White"
        AbsoluteLayout.LayoutBounds="0.1,1,0.8,50"
        AbsoluteLayout.LayoutFlags="All"/>
</AbsoluteLayout>
```

Basisbegrippen

XAML

XAML is een op XML gebaseerde taal voor het definiëren van gebruikersinterfaces in .NET MAUI-apps. Hiermee beschrijf je visueel en gestructureerd hoe schermen opgebouwd worden, los van je app-logica.

Layouts

Layouts bepalen hoe visuele elementen zoals knoppen, labels en invoervelden binnen een scherm worden gerangschikt. Ze zorgen ervoor dat je interface zich aanpast aan verschillende schermgroottes en oriëntaties.

Navigatie

Navigatie regelt hoe gebruikers van het ene scherm naar het andere gaan binnen je app, bijvoorbeeld via tabs of menu's. Met behulp van AppShell in .NET MAUI kun je navigatie eenvoudig definiëren en beheren.

Views

Views zijn visuele componenten zoals knoppen, invoervelden, labels of sliders waarmee gebruikers interactie hebben. Ze bieden functionaliteit aan je app en zorgen voor de communicatie tussen gebruiker en applicatie.

Shell

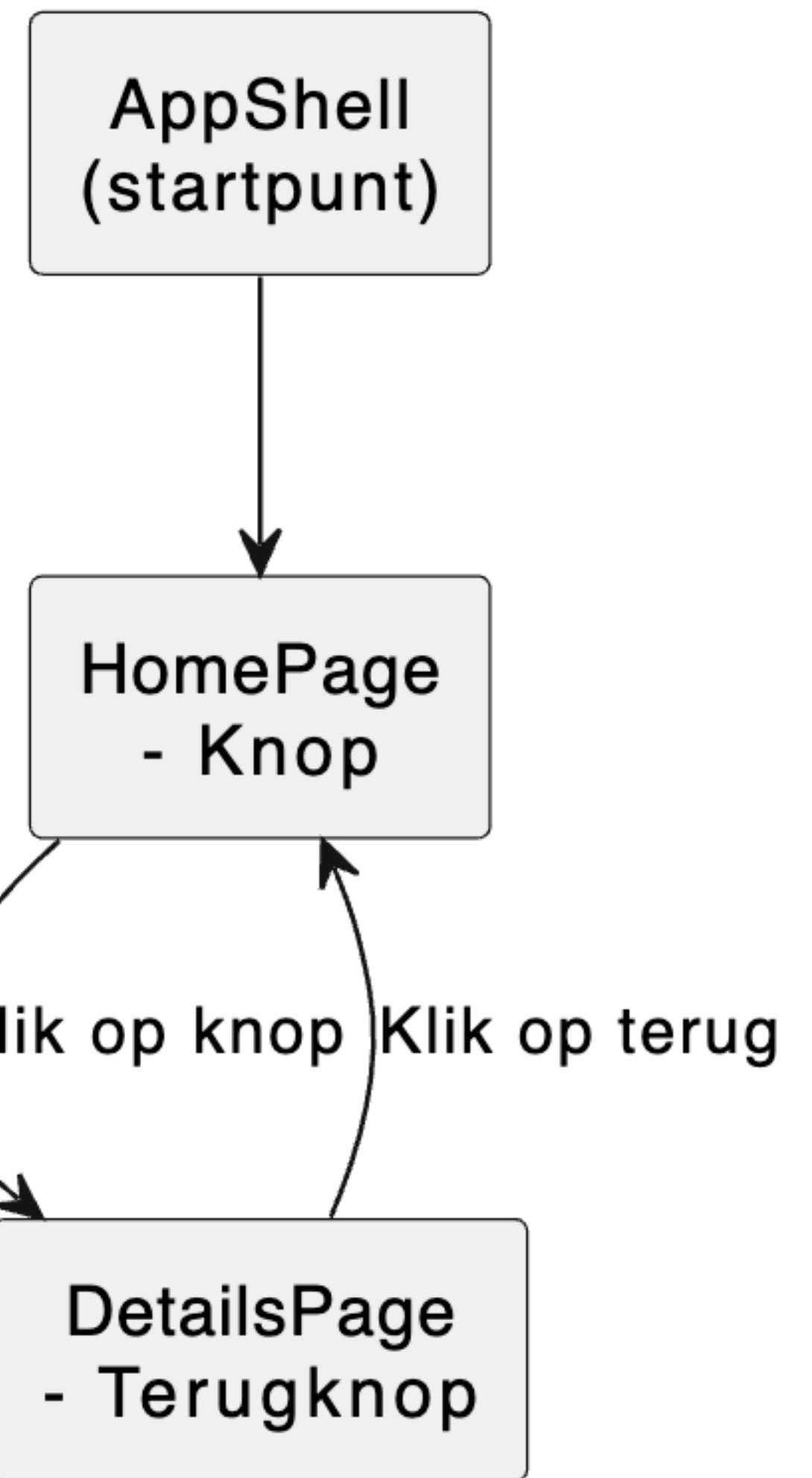
Dé (moderne) manier om navigatie en structuur in een .NET MAUI-app te organiseren.

Principes

- Eenvoudige declaratieve structuur in XAML
- Ondersteunt tabs, fly-out menu's, en routenavigatie
- Relatief weinig code
- Automatisch beheer van navigatiegeschiedenis en back-button

Shell

Dé (moderne) manier om navigatie en structuur in een .NET MAUI-app te organiseren.



AppShell.xaml(.cs)

Het registreren van de routing en het koppelen van het juiste template.

```
<Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:local="clr-namespace:DemoApp"
       x:Class="DemoApp.AppShell">
    <!-- Home direct zichtbaar -->
    <ShellContent Title="Home"
                  ContentTemplate="{DataTemplate local:HomePage}" />
</Shell>
```

```
public partial class AppShell : Shell
{
    public AppShell()
    {
        InitializeComponent();
        Routing.RegisterRoute(nameof(DetailsPage), typeof(DetailsPage));
    }
}
```

Routing.RegisterRoute(...)	Tells the Shell framework: "This route points to this page."
nameof(DetailsPage)	This becomes the route name, automatically using the class name "DetailsPage" as a string.
typeof(DetailsPage)	This tells Shell: when someone navigates to this route, load this Page class.

HomePage.xaml(.cs)

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="DemoApp.HomePage"
    Title="Home">
    <StackLayout VerticalOptions="Center"
        HorizontalOptions="Center">
        <Label Text="Welkom op de homepagina!"
            FontSize="24"
            HorizontalOptions="Center"/>
        <Button Text="Ga naar details"
            Clicked="OnGoToDetailsClicked"/>
    </StackLayout>
</ContentPage>
```

```
private async void OnGoToDetailsClicked(object sender, EventArgs e)
{
    await Shell.Current.GoToAsync(nameof(DetailsPage));
}
```

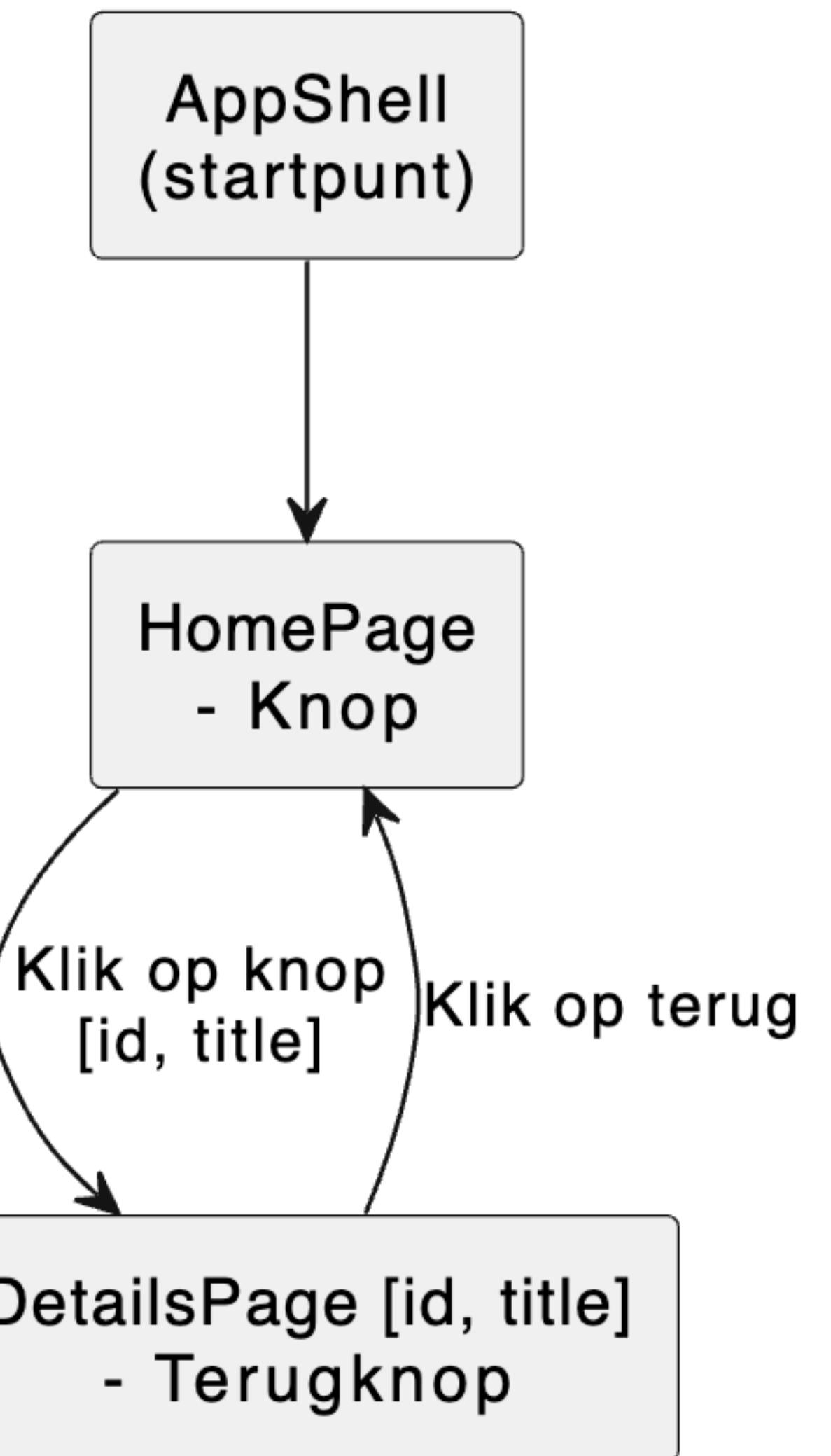
DetailsPage.xaml(.cs)

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="DemoApp.DetailsPage"
    Title="Details">
    <StackLayout VerticalOptions="Center"
        HorizontalOptions="Center">
        <Label Text="Dit is de detailpagina."
            FontSize="24" />
        <Button Text="Terug"
            Clicked="OnBackClicked"/>
    </StackLayout>
</ContentPage>
```

```
private async void OnBackClicked(object sender, EventArgs e)
{
    await Shell.Current.GoToAsync("../"); // Gaat terug naar vorige pagina
}
```

Shell met parameters

Simpele parameters doorgeven



HomePage.xaml(.cs)

Dit keer met navigeren naar een specifieke detailpagina met het ID als parameter.

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="DemoApp.HomePage"
    Title="Home">
    <StackLayout VerticalOptions="Center"
        HorizontalOptions="Center">
        <Label Text="Welkom op de homepagina!"
            FontSize="24"
            HorizontalOptions="Center"/>
        <Button Text="Ga naar details"
            Clicked="OnGoToDetailsClicked"/>
    </StackLayout>
</ContentPage>
```

```
private async void OnGoToDetailsClicked(object sender, EventArgs e)
{
    int id = 42;
    string title = "Meer informatie";
    await Shell.Current.GoToAsync($"{nameof(DetailsPage)}?id={id}&title={Uri.EscapeDataString(title)}");
}
```

DetailsPage.xaml(.cs)

Het attribuut `x:Name` gebruiken we om een naam te geven aan een UI-element, zodat je er in je code-behind toegang toe hebt.

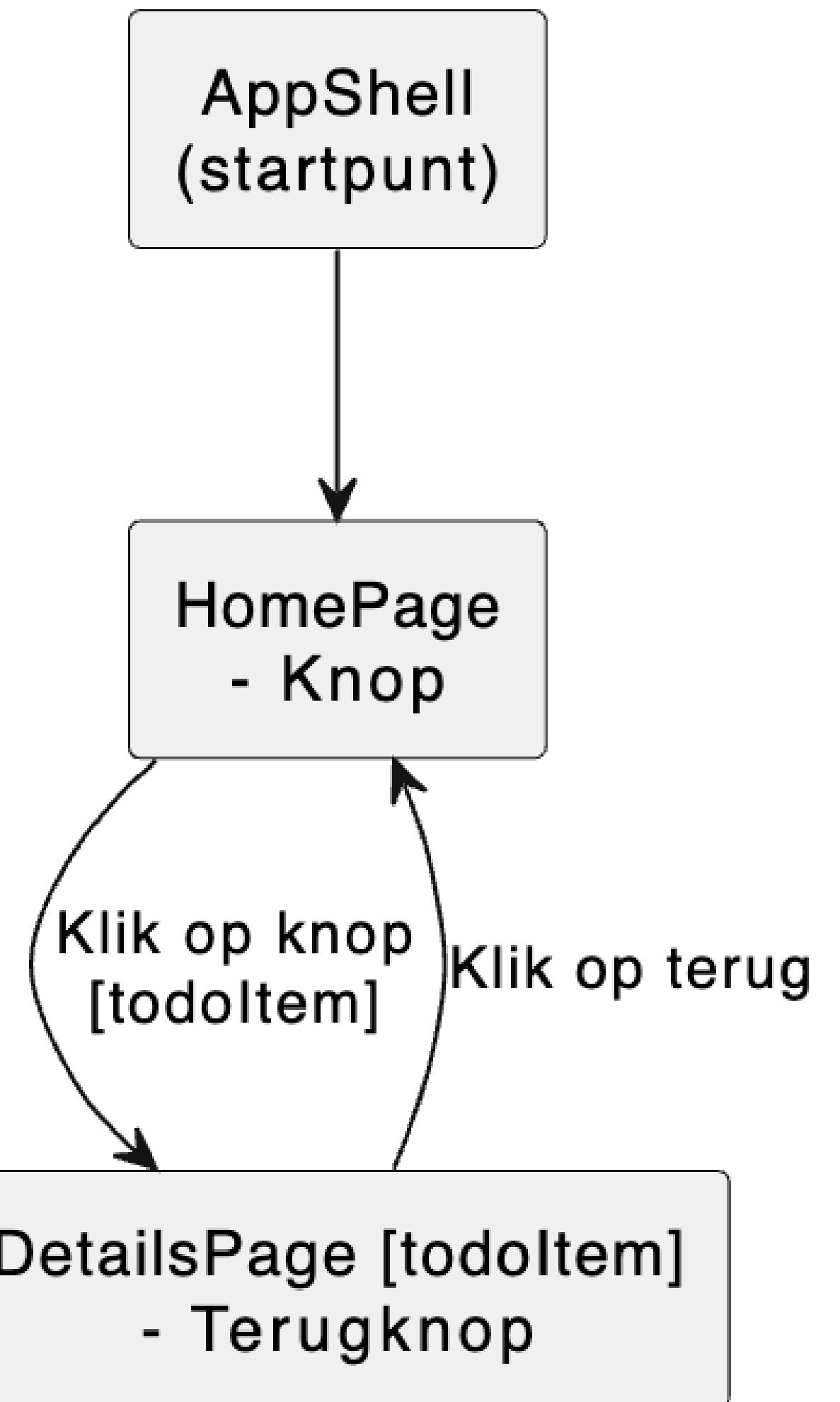
```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="DemoApp.DetailsPage"
    Title="Details">

    <VerticalStackLayout Padding="20">
        <Label x:Name="IdLabel" FontSize="20"/>
        <Label x:Name="TitleLabel" FontSize="20"/>
    </VerticalStackLayout>
</ContentPage>
```

```
[QueryProperty(nameof(Id), "id")]
[QueryProperty(nameof>Title), "title")]
public partial class DetailsPage : ContentPage
{
    public string Id { get; set; }
    public string Title { get; set; }
    public DetailsPage()
    {
        InitializeComponent();
    }
    protected override void OnAppearing()
    {
        base.OnAppearing();
        IdLabel.Text = $"ID: {Id}";
        TitleLabel.Text = $"Titel: {Title}";
    }
}
```

Shell met complexe objecten

Complexe objecten doorgeven kan ook



Basisbegrippen

XAML

XAML is een op XML gebaseerde taal voor het definiëren van gebruikersinterfaces in .NET MAUI-apps. Hiermee beschrijf je visueel en gestructureerd hoe schermen opgebouwd worden, los van je app-logica.

Layouts

Layouts bepalen hoe visuele elementen zoals knoppen, labels en invoervelden binnen een scherm worden gerangschikt. Ze zorgen ervoor dat je interface zich aanpast aan verschillende schermgroottes en oriëntaties.

Navigatie

Navigatie regelt hoe gebruikers van het ene scherm naar het andere gaan binnen je app, bijvoorbeeld via tabs of menu's. Met behulp van AppShell in .NET MAUI kun je navigatie eenvoudig definiëren en beheren.

Views

Views zijn visuele componenten zoals knoppen, invoervelden, labels of sliders waarmee gebruikers interactie hebben. Ze bieden functionaliteit aan je app en zorgen voor de communicatie tussen gebruiker en applicatie.

Pages

ContentPage

ContentPage toont één enkele weergave en is het meest gebruikte paginatype.

NavigationPage

NavigationPage biedt een hiërarchische navigatie-ervaring waarbij je voorwaarts en achterwaarts door pagina's kunt navigeren.

FlyoutPage

FlyoutPage is een pagina die twee gerelateerde pagina's beheert: een flyoutpagina die items toont, en een detailpagina die details toont over deze items.

TabbedPage

TabbedPage bestaat uit een reeks pagina's die via tabbladen aan de boven- of onderkant van de pagina toegankelijk zijn. Elk tabblad laadt zijn eigen pagina.

Views

Een heel arsenaal aan elementen om je pagina mee vorm te geven.

View	Description
ActivityIndicator	ActivityIndicator uses an animation to show that the app is engaged in a lengthy activity, without giving any indication of progress. For more information, see ActivityIndicator .
BlazorWebView	BlazorWebView enables you to host a Blazor web app in your .NET MAUI app. For more information, see BlazorWebView .
Border	Border is a container control that draws a border, background, or both, around another control. For more information, see Border .
BoxView	BoxView draws a rectangle or square, of a specified width, height, and color. For more information, see BoxView .
Button	Button displays text and responds to a tap or click that directs an app to carry out a task. For more information, see Button .
CarouselView	CarouselView displays a scrollable list of data items, where users swipe to move through the collection. For more information, see CarouselView .
CheckBox	CheckBox enables you to select a boolean value using a type of button that can either be checked or empty. For more information, see CheckBox .
CollectionView	CollectionView displays a scrollable list of selectable data items, using different layout specifications. For more information, see CollectionView .
ContentView	ContentView is a control that enables the creation of custom, reusable controls. For more information, see ContentView .
DatePicker	DatePicker enables you to select a date with the platform date picker. For more information, see DatePicker .
Editor	Editor enables you to enter and edit multiple lines of text. For more information, see Editor .
Ellipse	Ellipse displays an ellipse or circle. For more information, see Ellipse .
Entry	Entry enables you to enter and edit a single line of text. For more information, see Entry .
Frame	Frame is used to wrap a view or layout with a border that can be configured with color, shadow, and other options. For more information, see Frame .
GraphicsView	GraphicsView is a graphics canvas on which 2D graphics can be drawn using types from the

Zelf aan de slag

Bepaal voor je eigen applicatie de **Navigatiestructuur**,
de **Layouts** en de **Views**



<https://tinyurl.com/maui-controls>

MVVM

Model-View-ViewModel



```
<StackLayout Spacing="15" Padding="20">
    <Entry x:Name="NaamEntry" Placeholder="Voer je naam in"/>
    <Button Text="Verzenden" Clicked="OnSendClicked"/>
    <Label x:Name="StatusLabel" Text="Status: wacht op
    </StackLayout>
```



'Traditioneel'

XAML met daarin een clickevent welke in de Codebehind wordt aangeroepen.

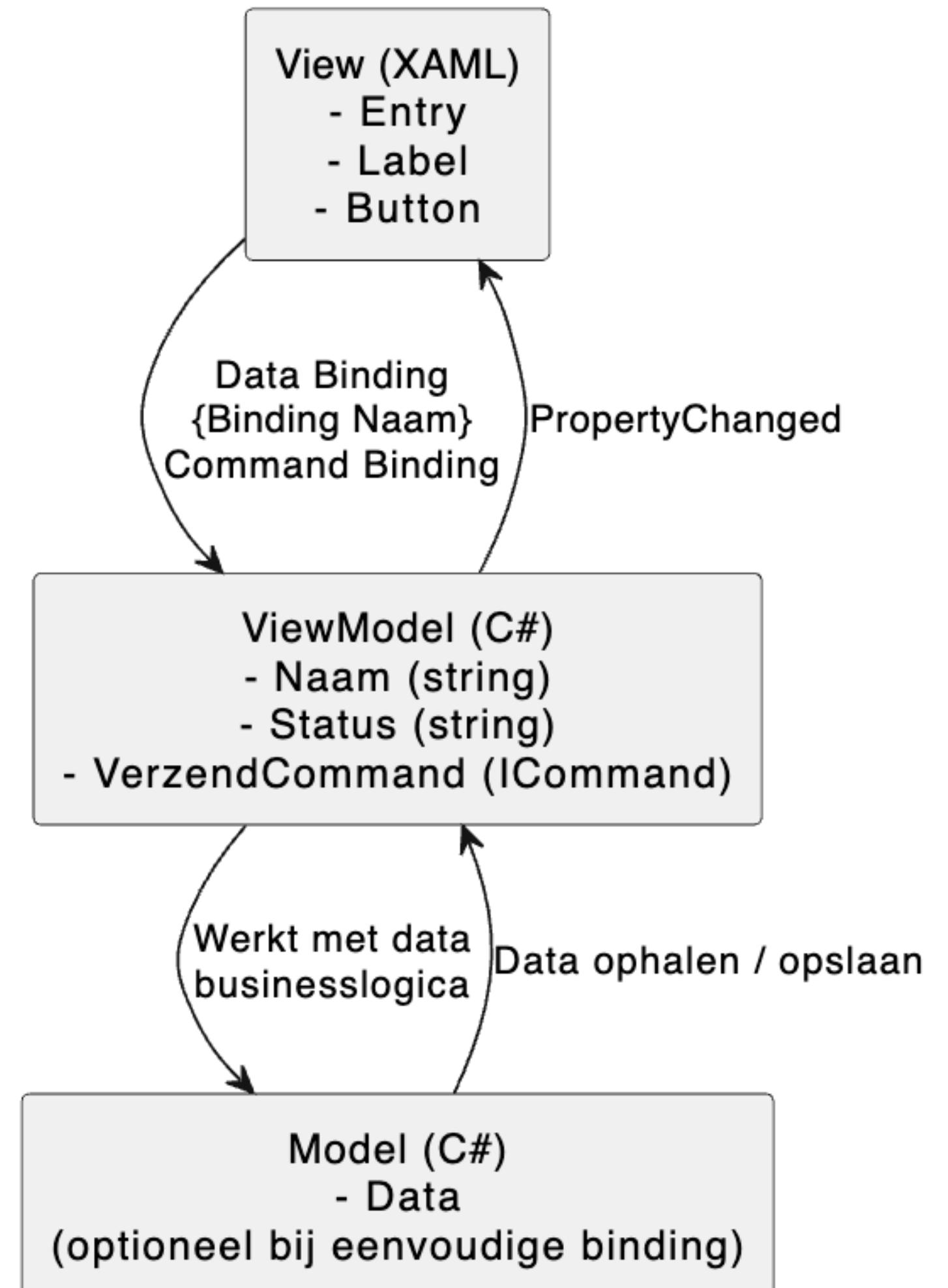
```
private void OnSendClicked(object sender, EventArgs
e)
{
    string naam = NaamEntry.Text;

    if (string.IsNullOrWhiteSpace(naam))
    {
        StatusLabel.Text = "Status: Vul een naam
in.");
    }
    else
    {
        StatusLabel.Text = $"Status: Hallo, {naam}!";
    }
}
```

MVVM

Je view krijgt slechts de data die het nodig heeft van het ViewModel, inclusief uitgevoerde business logica.

MVVM in .NET MAUI

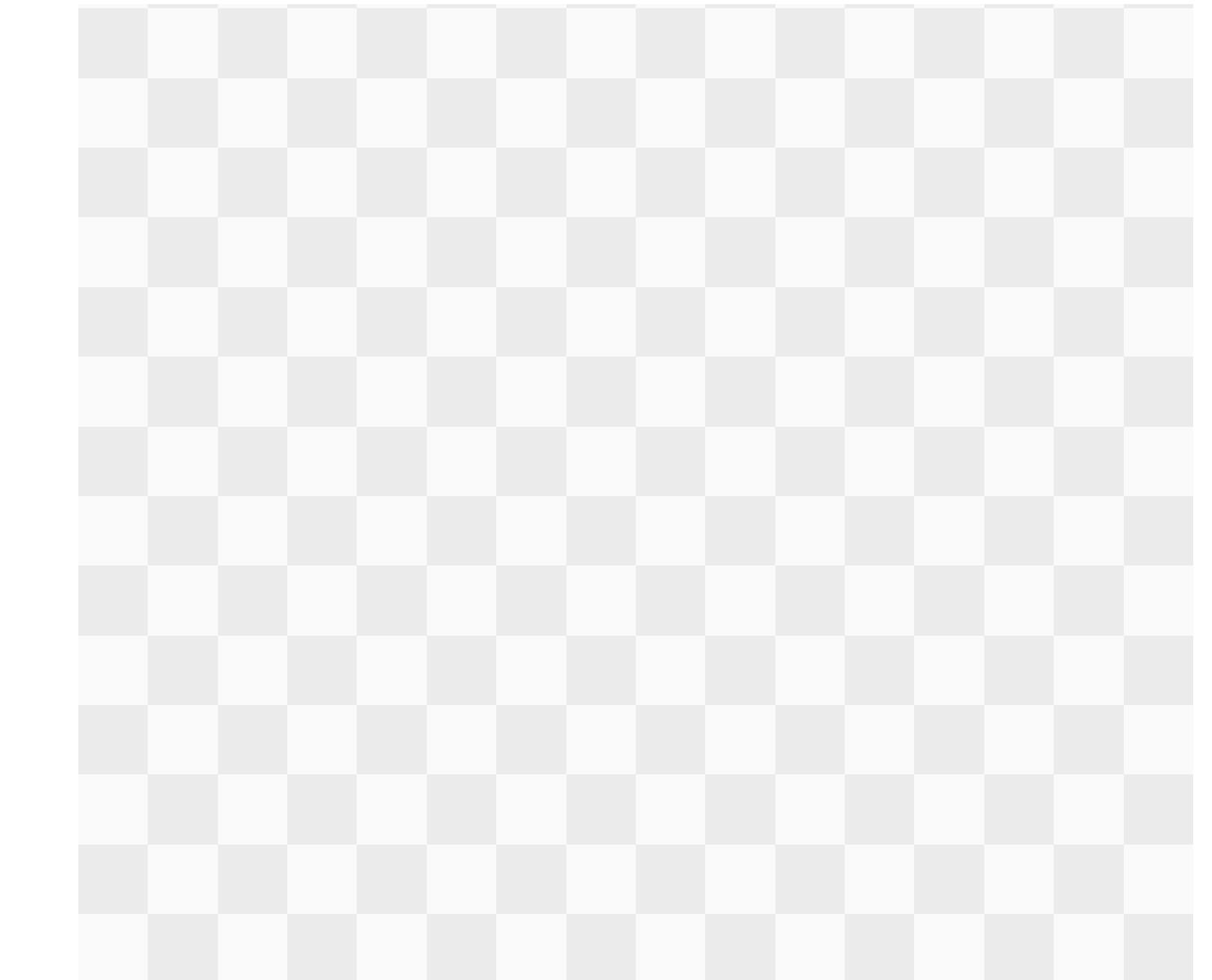
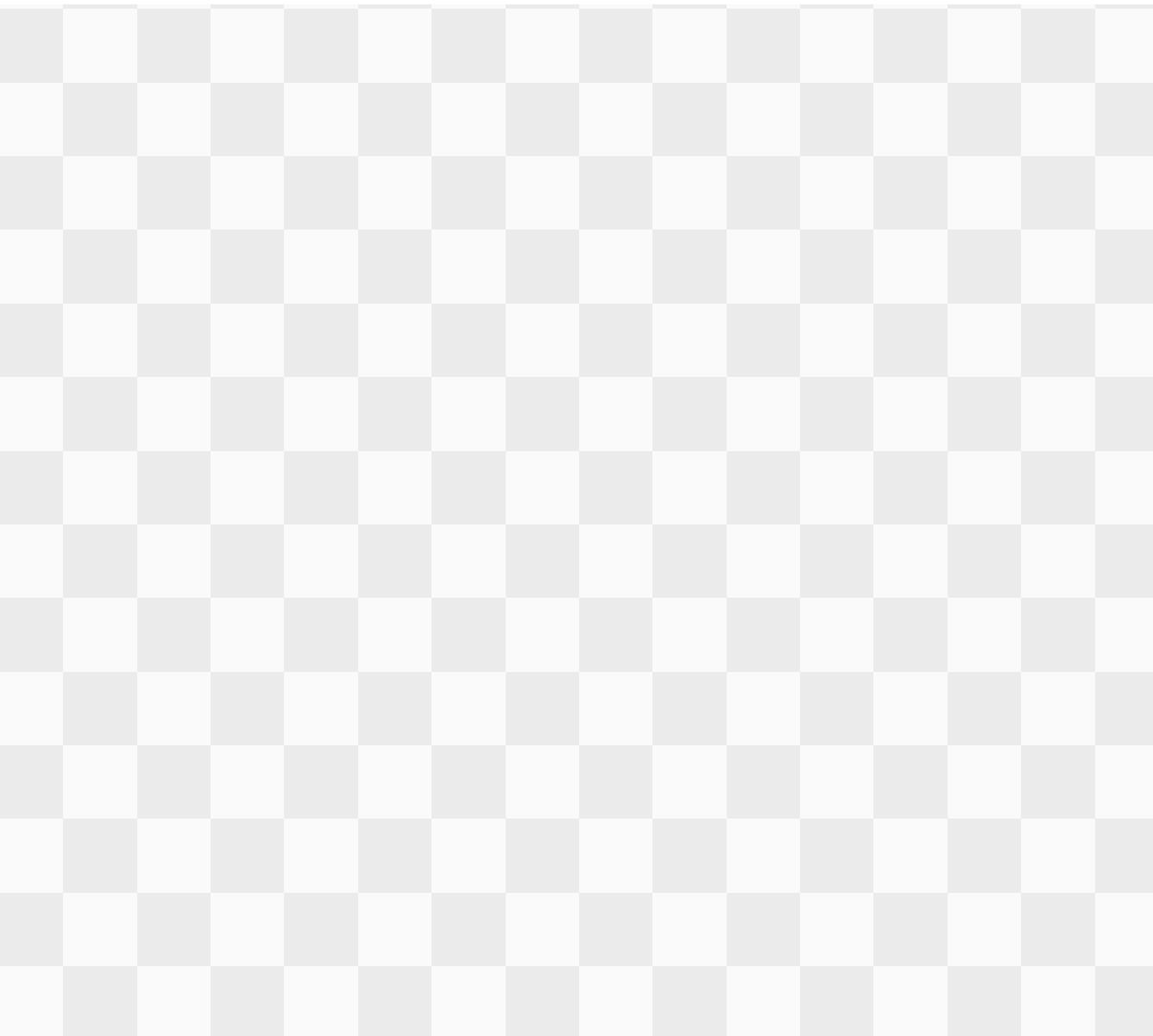


MVVM implementatie



```
<StackLayout Spacing="15" Padding="20">
    <Entry Placeholder="Voer je naam in"
        Text="{Binding Naam}" />

    <Button Text="Verzenden"
        Command="{Binding VerzendCommand}"
    />
    <Label Text="{Binding Status}" />
</StackLayout>
```



First thing

Add a quick description of each thing, with enough context to understand what's up.

Second thing

Keep 'em short and sweet, so they're easy to scan and remember.

Third thing

If you've got a bunch, add another row, or use multiple copies of this slide.