.NET Multi-Platform App UI (MAUI)

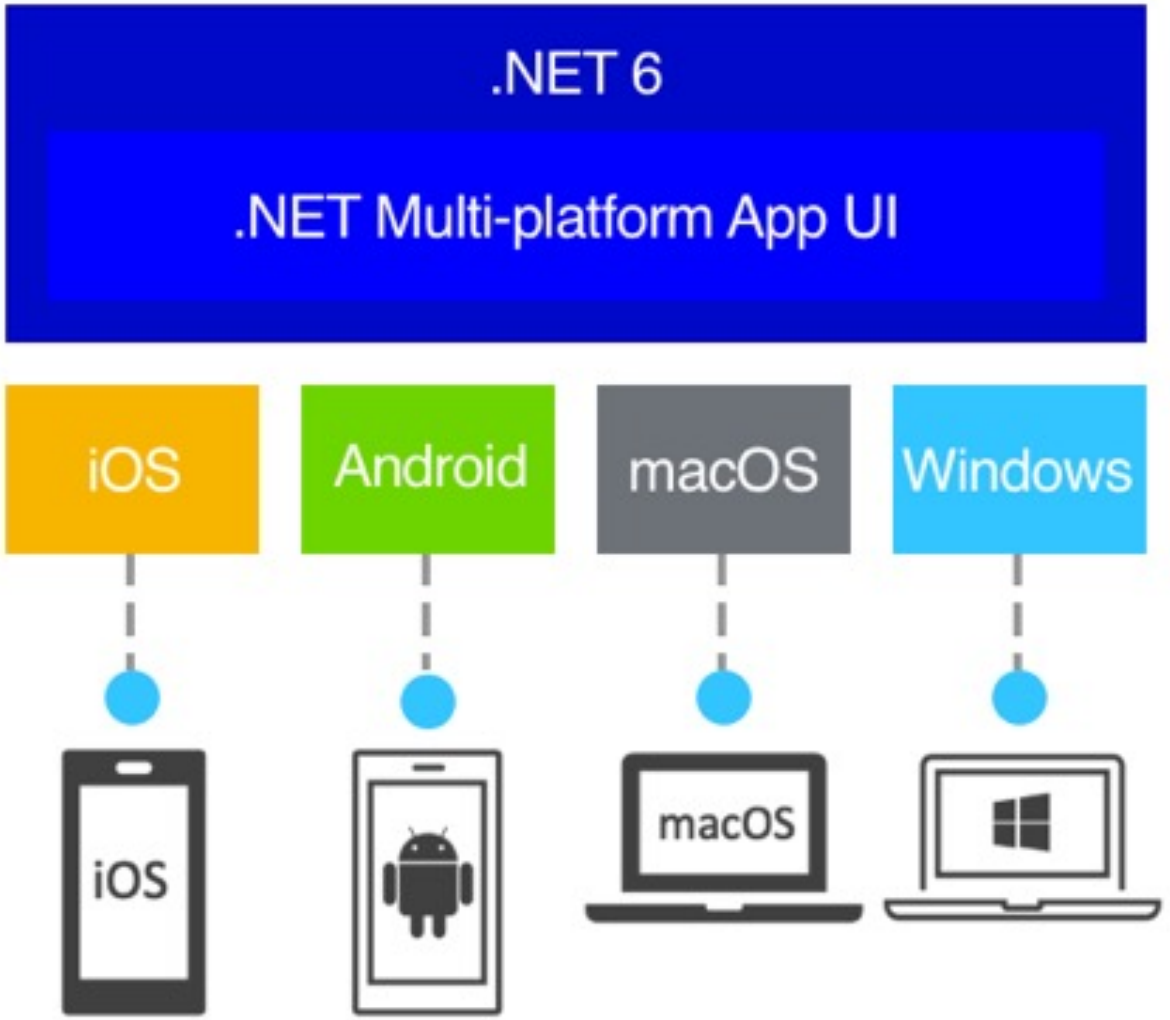
Dan Ciprian ARDELEAN

@danardelean

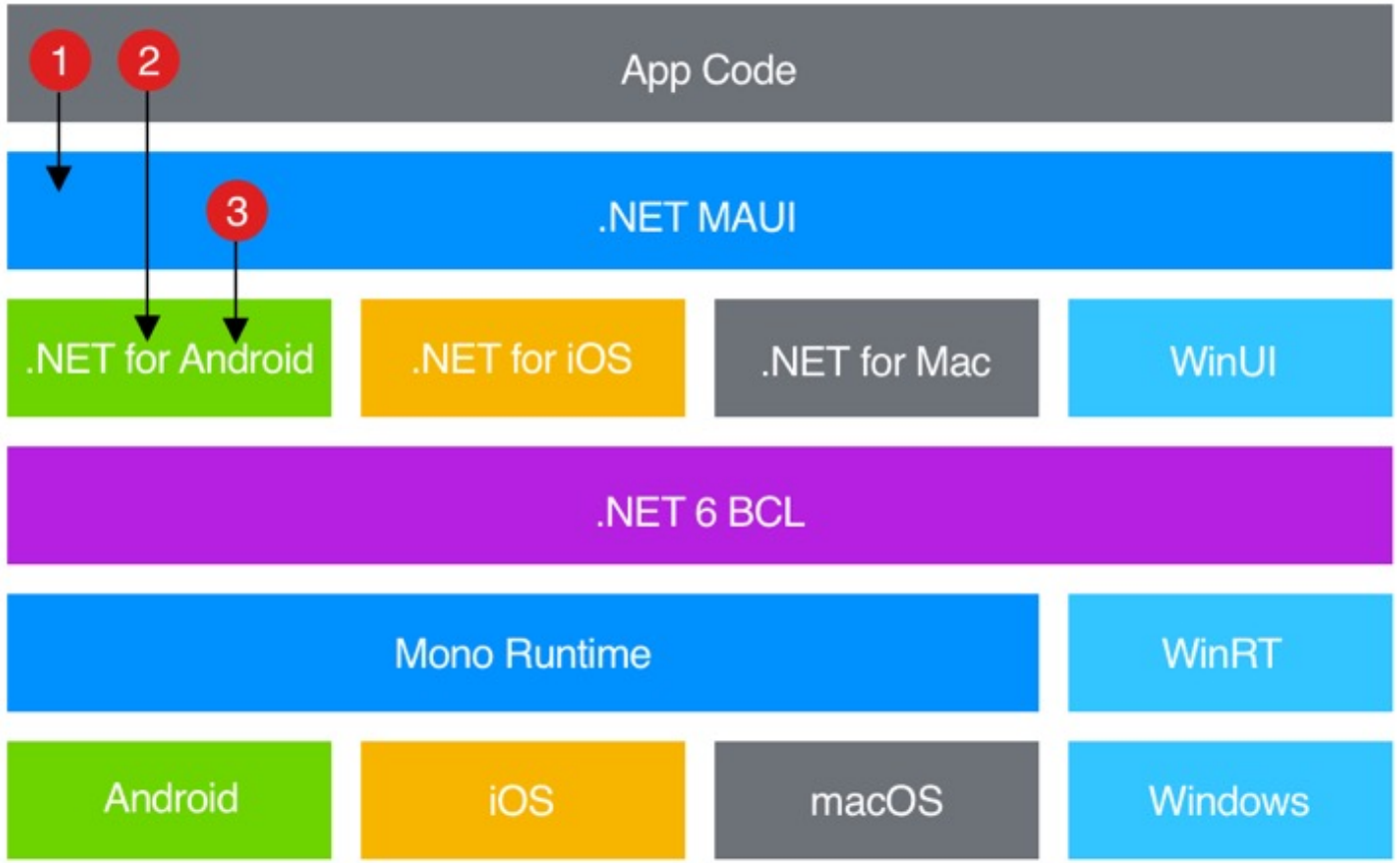
CTO Mahiz Srl

CommunityDays.it

Cos’è MAUI?


.NET 6
.NET Multi-platform App UI
iOS
Android
macOS
Windows
iOS
macOS

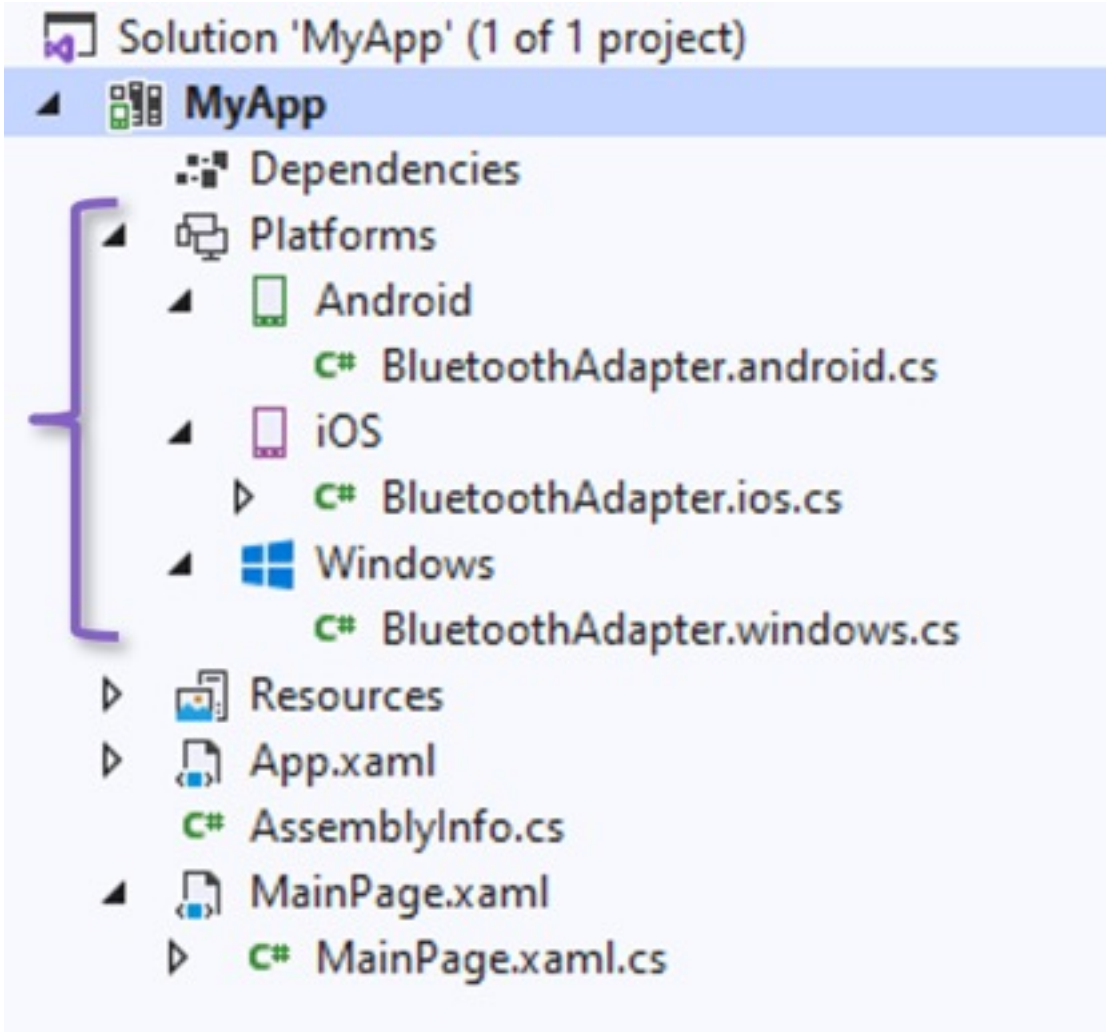
Architettura delle applicazioni .NET MAUI

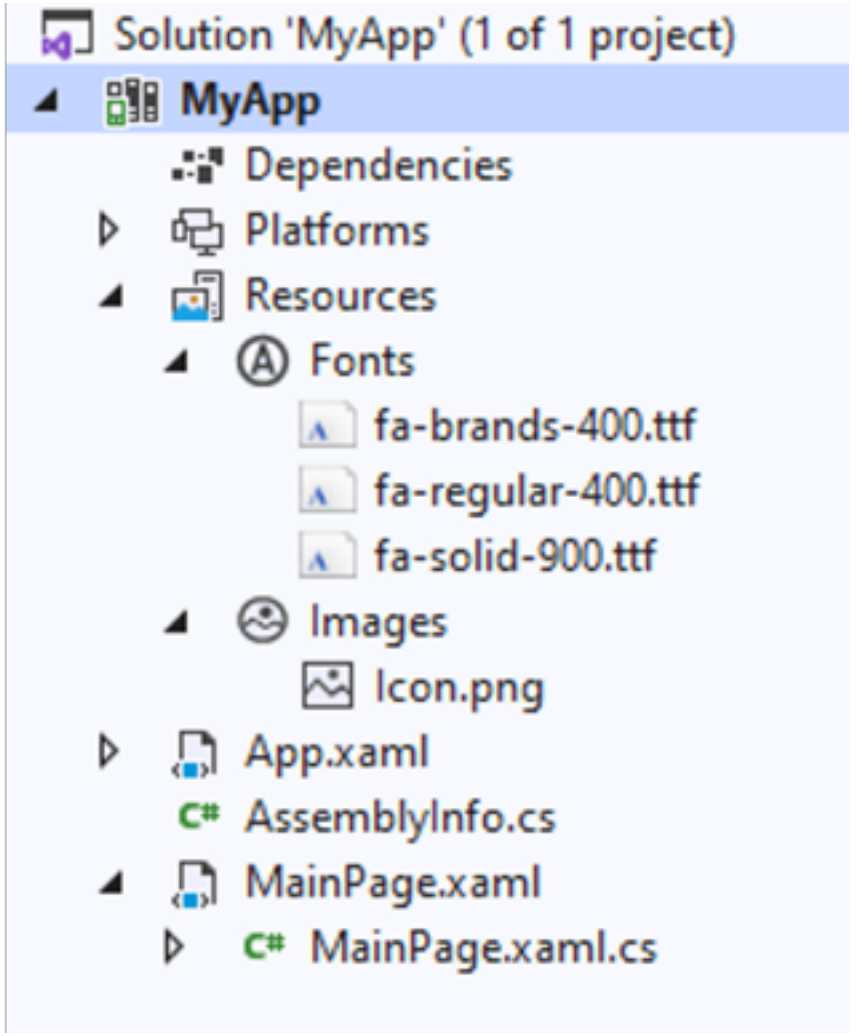


Piattaforme supportate

- **Android 5.0 (API 21)** or higher
- **iOS 10** or higher.
- **macOS 11** (Big Sur) or higher.
- **Windows desktop** and the **Universal Windows Platform (UWP),** using Windows UI Library (WinUI) 3

Single project = Multi targeting





MAUI Startup

```
using Microsoft.Maui;
using Microsoft.Maui.Hosting;

public class Startup : IStartup
{
    public void Configure(IAppHostBuilder appBuilder)
    {
        appBuilder
            .UseMauiApp<App>();
    }
}

using Microsoft.Maui;
using Microsoft.Maui.Controls;

public partial class App : Application
{
    protected override IWindow CreateWindow(IActivationState activationState)
    {
        return new Window(new MainPage());
    }
}
```

MAUI Startup

```
public class Startup : IStartup
{
    public void Configure(IAppHostBuilder appBuilder)
    {
        appBuilder
        .UseMauiApp<App>()
        .ConfigureServices(services => {
#if WINDOWS
        services.AddSingleton<ITrayService, WinUI.TrayService>();
        services.AddSingleton<INotificationService, WinUI.NotificationService>();
#elif MACCATALYST
        services.AddSingleton<ITrayService, MacCatalyst.TrayService>();
        services.AddSingleton<INotificationService, MacCatalyst.NotificationService>();
#endif
        })
        .ConfigureFonts(fonts => {
        fonts.AddFont("fa-solid-900.ttf", "FontAwesome");
        fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
        fonts.AddFont("OpenSans-SemiBold.ttf", "OpenSansSemiBold");
        });
    }
}
```

MAUI Startup

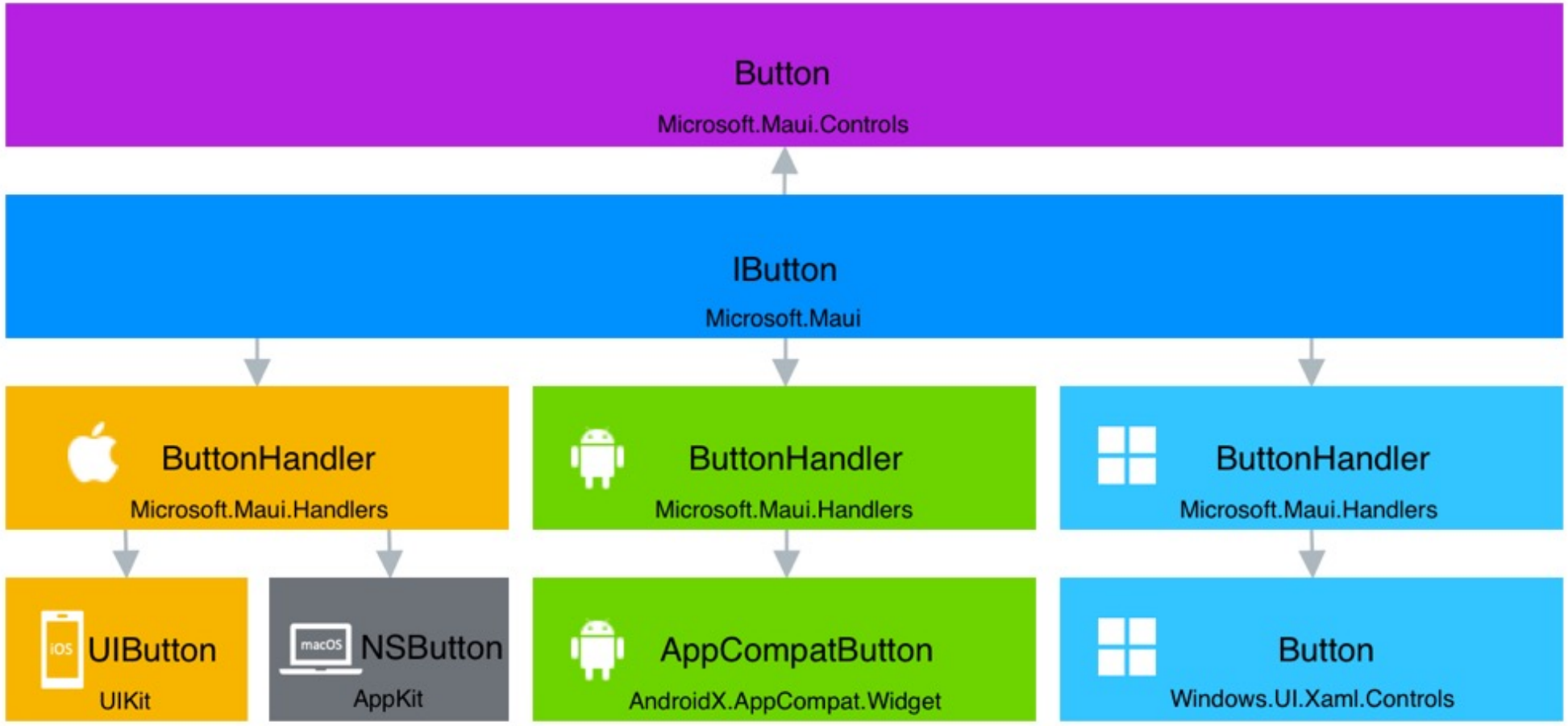
```
public class Startup : IStartup
{
    public void Configure(IAppHostBuilder appBuilder)
    {
        appBuilder
            .UseMauiApp<App>()
            .ConfigureFonts(fonts =>
            {
                fonts.AddFont("Lobster-Regular.ttf", "Lobster");
            });
    }
}
```

```
<ItemGroup>
  <MauiFont Include="Resources\Fonts\*" />
</ItemGroup>
```

```
<!-- Use font name -->
<Label Text="Hello .NET MAUI"
       FontFamily="Lobster-Regular" />
```

```
<!-- Use font alias -->
<Label Text="Hello .NET MAUI"
       FontFamily="Lobster" />
```

MAUI Handlers



IButton.cs

```
namespace Microsoft.Maui
{
    /// <summary>
    /// Represents a View that reacts to touch events.
    /// </summary>
    public interface IButton : IView, IText, IPadding
    {
        /// <summary>
        /// Occurs when the Button is pressed.
        /// </summary>
        void Pressed();

        /// <summary>
        /// Occurs when the Button is released.
        /// </summary>
        void Released();

        /// <summary>
        /// Occurs when the Button is clicked.
        /// </summary>
        void Clicked();
    }
}
```

ButtonHandler - shared

```
namespace Microsoft.Maui.Handlers
{
    public partial class ButtonHandler
    {
        public static PropertyMapper<IButton, ButtonHandler> ButtonMapper = new PropertyMapper<IButton, ButtonHandler>(ViewHandler.ViewMapper)
        {
#if WINDOWS || __ANDROID__
            [nameof(IButton.Background)] = MapBackground,
#endif
            [nameof(IButton.CharacterSpacing)] = MapCharacterSpacing,
            [nameof(IButton.Font)] = MapFont,
            [nameof(IButton.Padding)] = MapPadding,
            [nameof(IButton.Text)] = MapText,
            [nameof(IButton.TextColor)] = MapTextColor,
        };

        public ButtonHandler() : base(ButtonMapper)
        {

        }

        public ButtonHandler(PropertyMapper? mapper = null) : base(mapper ?? ButtonMapper)
        {
        }
```


ButtonHandler.iOS.cs

```
public partial class ButtonHandler : ViewHandler<IButton, UIButton>
{
        protected override UIButton CreateNativeView()
        {
            var button = new UIButton(UIButtonType.System);
            SetControlPropertiesFromProxy(button);
            return button;
        }

         protected override void ConnectHandler(UIButton nativeView)
         {
             nativeView.TouchUpInside += OnButtonTouchUpInside;
             nativeView.TouchUpOutside += OnButtonTouchUpOutside;
             nativeView.TouchDown += OnButtonTouchDown;

             base.ConnectHandler(nativeView);
         }

         protected override void DisconnectHandler(UIButton nativeView)
         {
             nativeView.TouchUpInside -= OnButtonTouchUpInside;
             nativeView.TouchUpOutside -= OnButtonTouchUpOutside;
             nativeView.TouchDown -= OnButtonTouchDown;
```

ButtonHandler.iOS.cs

```
public partial class ButtonHandler : ViewHandler<IButton, UIButton>
{
      protected override void SetupDefaults(UIButton nativeView)
      {
          ButtonTextColorDefaultNormal = nativeView.TitleColor(UIControlState.Normal);
          ButtonTextColorDefaultHighlighted = nativeView.TitleColor(UIControlState.Highlighted);
          ButtonTextColorDefaultDisabled = nativeView.TitleColor(UIControlState.Disabled);

          base.SetupDefaults(nativeView);
      }

       public static void MapText(ButtonHandler handler, IButton button)
       {
           handler.NativeView?.UpdateText(button);

           // Any text update requires that we update any attributed string formatting
           MapFormatting(handler, button);
       }

       public static void MapTextColor(ButtonHandler handler, IButton button)
       {
           handler.NativeView?.UpdateTextColor(button, ButtonTextColorDefaultNormal, ButtonTextColorDefaultHighlighted, ButtonTextColorDefaultDisabled);
       }

       public static void MapCharacterSpacing(ButtonHandler handler, IButton button)
       {
           handler.NativeView?.UpdateCharacterSpacing(button);
       }
```

ButtonHandler.iOS.cs

```
public partial class ButtonHandler : ViewHandler<IButton, UIButton>
{
      void OnButtonTouchUpInside(object? sender, EventArgs e)
      {
          VirtualView?.Released();
          VirtualView?.Clicked();
      }

      void OnButtonTouchUpOutside(object? sender, EventArgs e)
      {
          VirtualView?.Released();
      }

      void OnButtonTouchDown(object? sender, EventArgs e)
      {
          VirtualView?.Pressed();
      }
```

Personalizzare Handlers

- Cambiare il colore di background su tutti controlli su Android

```
#if ANDROID
Handlers.ViewHandler
    .ViewMapper[nameof(IView.BackgroundColor)] = (h, v) =>
    {
        (h.NativeView as global::Android.Views.View).BackgroundColor
                = RandomColor();
    };
#endif
```

- Non sottolineare il testo su Android

```
#if ANDROID
Handlers.EntryHandler
    .EntryMapper[nameof(IEntry.BackgroundColor)] = (h, v) =>
    {
        (h.NativeView as global::Android.Views.Entry).UnderlineVisible = false;
    };
#endif
```

Fallback to renderers

```
appBuilder
      .ConfigureMauiHandlers(handlers =>
      {
#if WINDOWS10_0_17763_0_OR_GREATER
            handlers.AddCompatibilityRenderer<ImageButton,Microsoft.Maui.Controls.Compatibility.Platform.UWP.ImageButtonRenderer>();
#elif ANDROID
            handlers.AddCompatibilityRenderer<ImageButton,Microsoft.Maui.Controls.Compatibility.Platform.Android.ImageButtonRenderer>()
#elif IOS || MACCATALYST
            handlers.AddCompatibilityRenderer<ImageButton,Microsoft.Maui.Controls.Compatibility.Platform.iOS.ImageButtonRenderer>();
#endif
      })
      .ConfigureFonts((hostingContext, fonts) =>
      {
            fonts.AddFont("fa_solid.ttf", "FontAwesome");
            fonts.AddFont("opensans_regular.ttf", "OpenSansRegular");
            fonts.AddFont("opensans_semibold.ttf", "OpenSansSemiBold");
      })
      .UseMauiApp<App>();
```

Migrazione da Xamarin.Forms

Old namespace	New namespace
`xmlns="http://xamarin.com/schemas/2014/forms"`	`xmlns="http://schemas.microsoft.com/dotnet/2021/maui"`
`using Xamarin.Forms`	`using Microsoft.Maui` AND `using Microsoft.Maui.Controls`
`using Xamarin.Forms.Xaml`	`using Microsoft.Maui.Controls.Xaml`

.NET MAUI Workload

https://github.com/dotnet/maui/tree/main/src/Workload

- The idea, is a project to be able to set $(UseMaui):

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFrameworks>net6.0-android;net6.0-ios</TargetFrameworks>
    <OutputType>Exe</OutputType>
    <UseMaui>true</UseMaui>
  </PropertyGroup>
</Project>
```

`$(UseMaui)` automatically brings in the following workload packs:

- `Microsoft.NET.Sdk.Maui`
- `Microsoft.Maui.Controls.Sdk`
- `Microsoft.Maui.Resizetizer.Sdk`
- `Microsoft.Maui.Core.Ref.[platform]`
- `Microsoft.Maui.Core.Runtime.[platform]`
- `Microsoft.Maui.Controls.Ref.[platform]`
- `Microsoft.Maui.Controls.Runtime.[platform]`
- `Microsoft.Maui.Dependencies`
- `Microsoft.Maui.Essentials.Ref.[platform]`
- `Microsoft.Maui.Essentials.Runtime.[platform]`
- `Microsoft.Maui.Extensions`
- `Microsoft.Maui.Templates`

- BlazorWebView is an addition to MAUI, project can currently opt into it by adding .Razor to the Sdk attribute. `<Project Sdk="Microsoft.NET.Sdk.Razor">` sets `$(UsingMicrosoftNETSdkRazor)`, which triggers the MAUI workload to include:
  - `Microsoft.AspNetCore.Components.WebView.Maui`

Da dove inziamo?

- **.NET 6 Preview Installers**

  - Windows: dotnet-sdk-6.0.100-preview.5.21302.13-win-x64.exe
  - macOS: dotnet-sdk-6.0.100-preview.5.21302.13-osx-x64.pkg

- **Installazione workload**

dotnet workload install microsoft-android-sdk-full

- microsoft-android-sdk-full
- microsoft-ios-sdk-full
- microsoft-maccatalyst-sdk-full
- microsoft-macos-sdk-full
- microsoft-tvos-sdk-full

Da dove inziamo?

dotnet tool install -g **redth.net.maui.check**

maui-check

DEMO

Roadmap

- https://github.com/dotnet/maui/wiki/Roadmap

Icon	Description
⚠️	Pending
⏳	Underway
✅	Done
💔	Never implemented in Xamarin.Forms for this platform

Overview

To track ongoing progress, filter on the handlers label.

Pages

Control	Android	iOS / Mac Catalyst	Windows
ContentPage	⏳	⏳	⏳
FlyoutPage	⚠️	⚠️	⚠️
NavigationPage	⏳	⏳	⏳
TabbedPage	⚠️	⚠️	⚠️

⚠️ Button

API	Android	iOS / Mac Catalyst	Windows
BackgroundColor	✅	✅	✅
BorderColor	⚠️	⚠️	⚠️
BorderWidth	⚠️	⚠️	⚠️
CharacterSpacing	✅	✅	✅
Clicked	✅	✅	✅
Command	✅	✅	✅
CommandParameter	✅	✅	✅
ContentLayout	⚠️	⚠️	⚠️
CornerRadius	⚠️	⚠️	⚠️
FontAttributes	✅	✅	✅
FontFamily	✅	✅	✅
FontSize	✅	✅	✅
ImageSource	⚠️	⚠️	⚠️
Padding	✅	✅	✅
Pressed	✅	✅	✅
Released	✅	✅	✅
Text	✅	✅	✅
TextColor	✅	✅	✅

⚠️ CollectionView

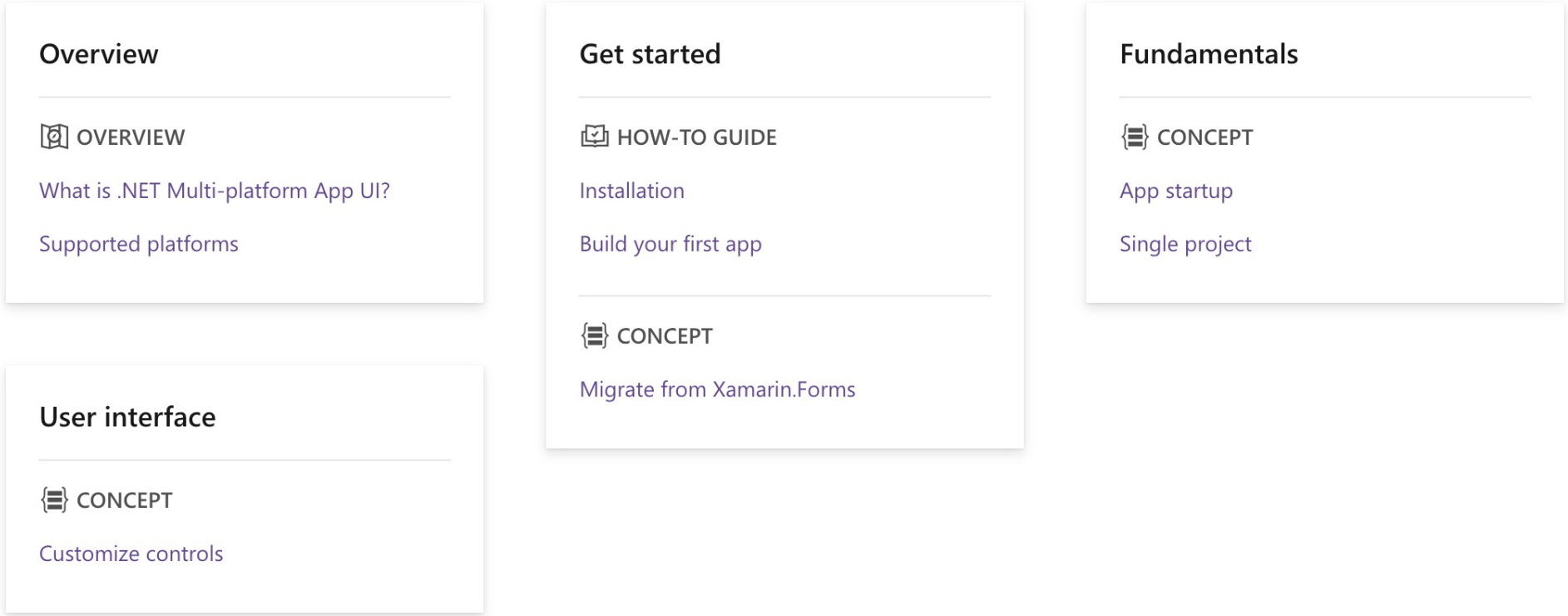
API	Android	iOS / Mac Catalyst	Windows
ItemsSource	⚠️	⚠️	⚠️
ItemTemplate	⚠️	⚠️	⚠️
ItemsPanel	⚠️	⚠️	⚠️
ItemSizingStrategy	⚠️	⚠️	⚠️
SelectionMode	⚠️	⚠️	⚠️
SelectedItem	⚠️	⚠️	⚠️
SelectedItems	⚠️	⚠️	⚠️
SelectionChangedCommand	⚠️	⚠️	⚠️
SelectionChangedCommandParameter	⚠️	⚠️	⚠️
EmptyView	⚠️	⚠️	⚠️
Scrolled	⚠️	⚠️	⚠️
ScrollTo	⚠️	⚠️	⚠️
Header	⚠️	⚠️	⚠️
HeaderTemplate	⚠️	⚠️	⚠️
Footer	⚠️	⚠️	⚠️
FooterTemplate	⚠️	⚠️	⚠️
IsGrouped	⚠️	⚠️	⚠️
GroupHeaderTemplate	⚠️	⚠️	⚠️
GroupFooterTemplate	⚠️	⚠️	⚠️

- https://github.com/dotnet/maui/wiki/Status

Documentazione

- **.NET Multi-platform App UI documentation**

  https://docs.microsoft.com/en-us/dotnet/maui/



Repository con esempi

- **.NET 6.0.0 Client App Samples for Android, iOS, macOS, and Windows**
  - https://github.com/dotnet/maui-samples

- **Control Gallery**
  - https://github.com/davidortinau/ControlGallery/ControlGallery (github.com)

- **Weather '21**
  - https://github.com/davidortinau/WeatherTwentyOne

DOMANDE

Grazie!

- Il materiale sarà online nei prossimi giorni su http://www.communitydays.it