.NET MAUI

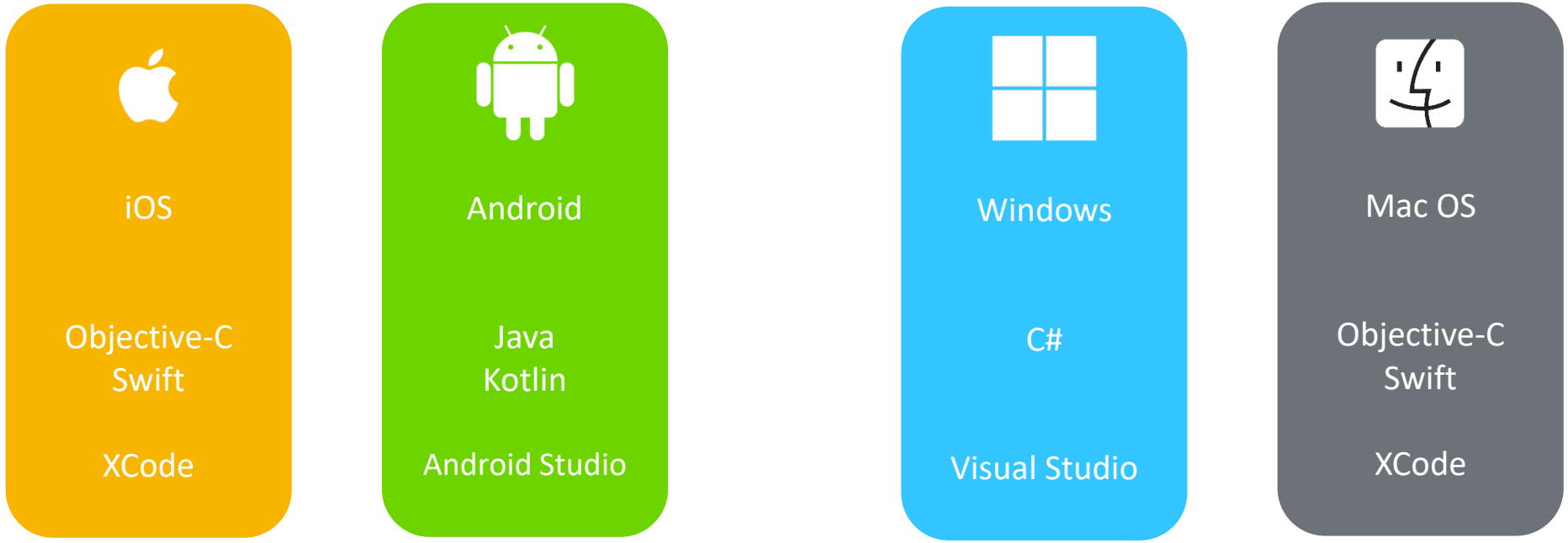
Roman Jašek

Software Architect, Riganti s.r.o.
Microsoft Most Valuable Professional (MVP)

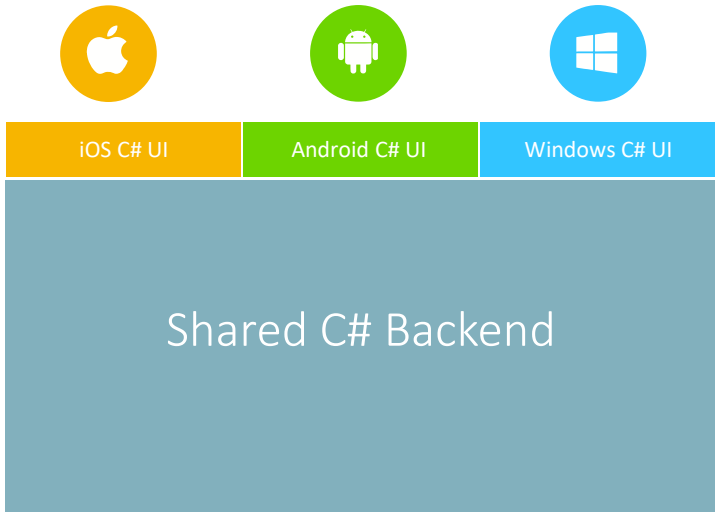
roman.jasek@riganti.cz

Who knows?

„Standard“ Application Development



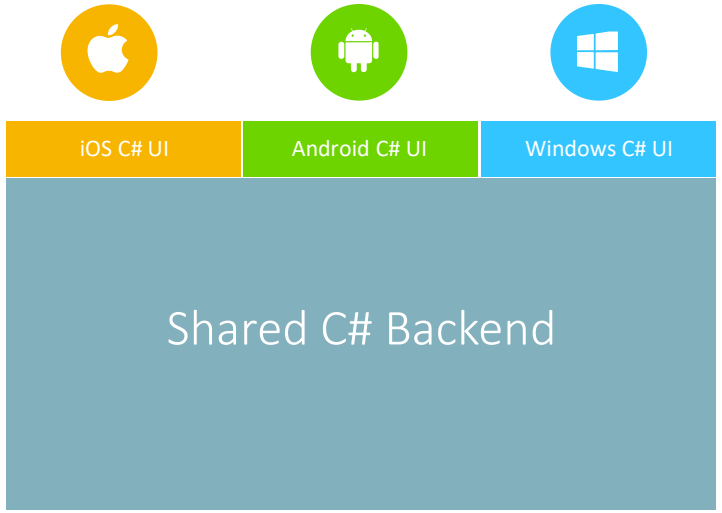
Xamarin



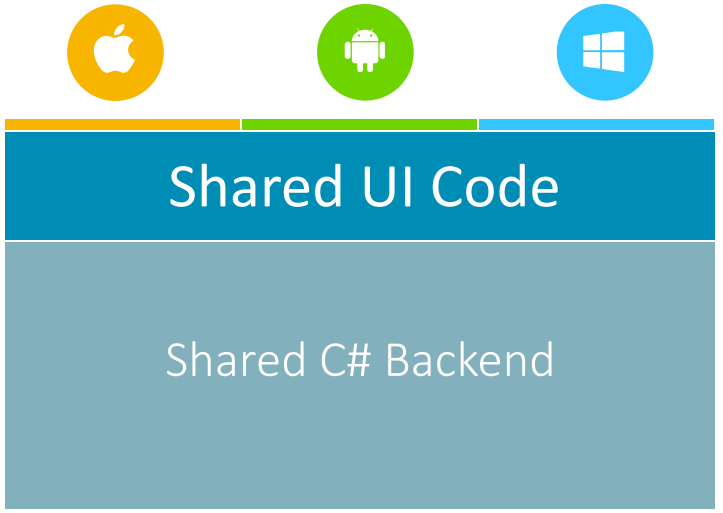
Xamarin

- **Common logic** for all platforms
- **UI** written in **C#**
- Use development practices analogous to the **specific target platforms**

Xamarin + Xamarin.Forms



Xamarin



Xamarin.Forms

How It Works - Structure

- Platform specific frameworks
  - .NET for Android
  - .NET for iOS
  - .NET for MacOS
  - Windows UI (WinUI) library
- Common BCL - .NET 6
- .NET Runtimes
  - Mono – Android, iOS, MacOS
  - WinRT – Windows

How It Works - UI

- Platform specific UI
  - Different platforms - different ways of defining UI
  - Can be defined separately using platform specific APIs
  - .NET for Android, .NET for iOS, .NET for MacOs, WinUI

- Common UI
  - Single framework for defining UI – mobile & desktop
  - XAML

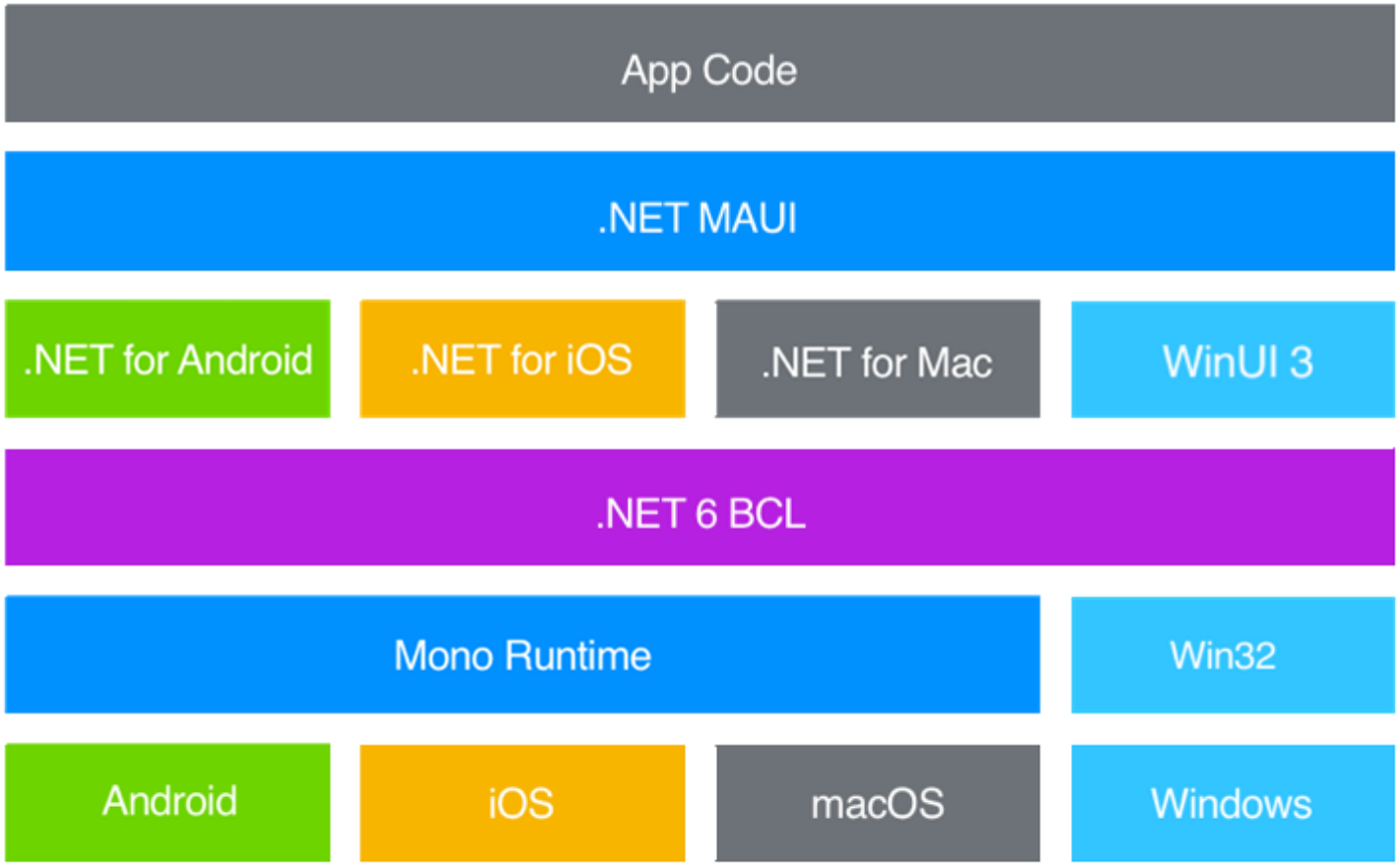
How It Works - compilation

- Android
  - C# compiles to intermediate language (IL)
  - JIT compilation to native assembly on app launch
- iOS
  - Fully ahead-of-time (AOT) compiled to native ARM assembly code
- MacOS
  - Using Mac Catalyst
    - Apple’s solution to bring iOS Apps to desktop
    - Provides access to Mac OS APIs
- Windows
  - WinUI 3 library
  - Native apps and UWP

How It Works – Application Startup

- .NET Generic Host – DI, logging, configuration…
- Static **MauiProgram** class
  - Create builder, register dependencies..
  - Create a **MauiApp** instance
- **App** class
  - Derives from Application
  - Initializes application, sets initial page
- Separate startup points for each platform (Platforms folder)
  - **MainApplication.cs** (Android), **Program.cs** (iOS)
  - **App.xaml.cs** (Windows), **Program.cs** (Mac Os)

How .NET MAUI Works



.NET MAUI

- Collection of Controls
- Layout engine for pages
- Navigation – pages, drawers
- Customizable handlers – enable platform specific controls
- APIs for native device features – GPS, accelerometer…
- Graphics library for 2D drawing code
- Single project, multi-targeting system
- .NET hot reload

.NET MAUI Essentials

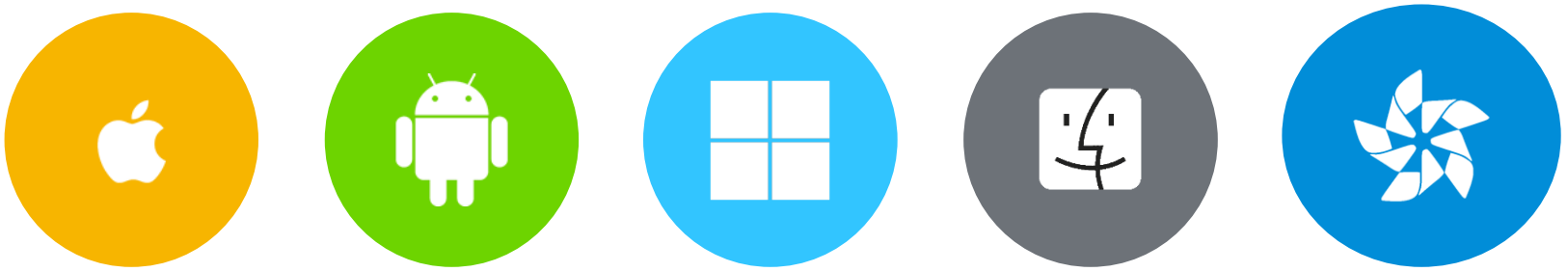
- Extension library, continuation of Xamarin Essentials
- Cross-platform APIs for native device features
- Included directly in MAUI, no need for extra Nuget packages

.NET MAUI Essentials

Accelerometer	App Actions	App Information
App Theme	Barometer	Battery
Clipboard	Color Converters	Compass
Connectivity	Contacts	Detect Shake
Display Info	Device Info	Email
File Picker	File System Helpers	Flashlight
Geocoding	Geolocation	Gyroscope
Haptic Feedback	Launcher	Magnetometer
MainThread	Maps	Media Picker
Open Browser	Orientation Sensor	Permissions
Phone Dialer	Platform Extensions	Preferences
Screenshot	Secure Storage	Share
SMS	Text-to-Speech	Unit Converters
Version Tracking	Vibrate	Web Authenticator

Platforms

Official support *



Community

* **Tizen .NET** supported by Samsung

XAML

- Format for serialization of hierarchy of objects
- Mostly used for UI

- Possible connection to a code-behind class

Where is XAML used?

- Windows Presentation Foundation (WPF)
- Silverlight
- Windows Phone
- Universal Windows Platform (UWP)
- Windows Workflow Foundation (WF)
  - Not used for UI
- Xamarin.Forms
- .NET MAUI

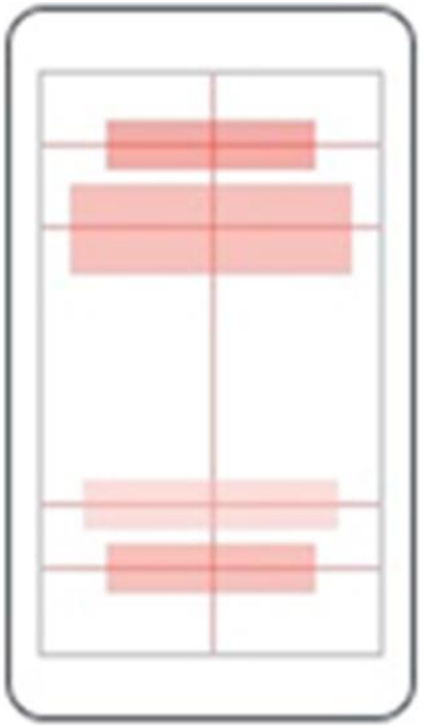
XAML

- **x:Class** … name of generated class
- **UserControl** … inheritance
- **xmlns:x** … special namespace for XAMLu (mandatory)
- **xmlns** … namespace with built-in framework controls

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="CookBook.Mobile.Views.Page1">
</ContentPage>
```

Layouts – multiple components

AbsoluteLayout

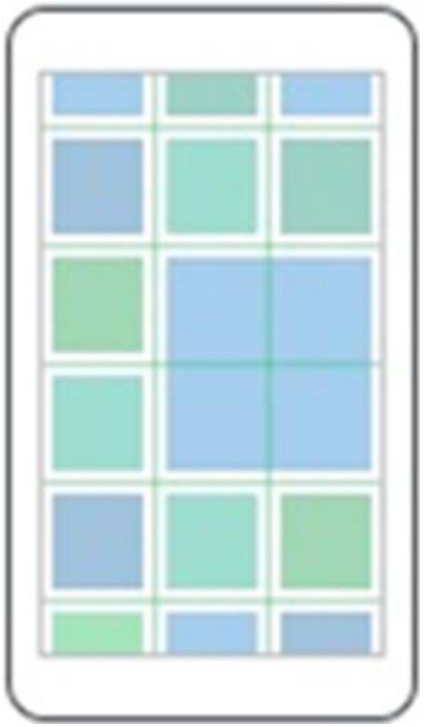


RelativeLayout

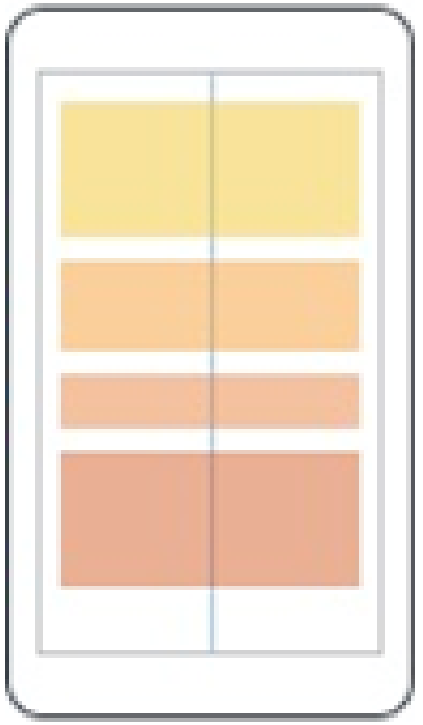
FlexLayout

Layouts – multiple components

Grid

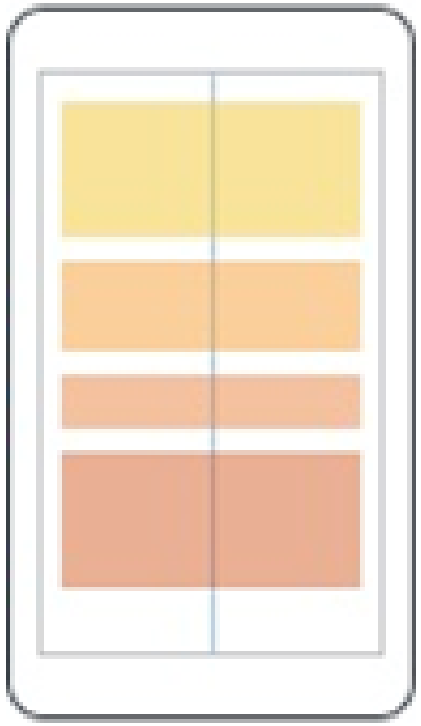


StackLayout

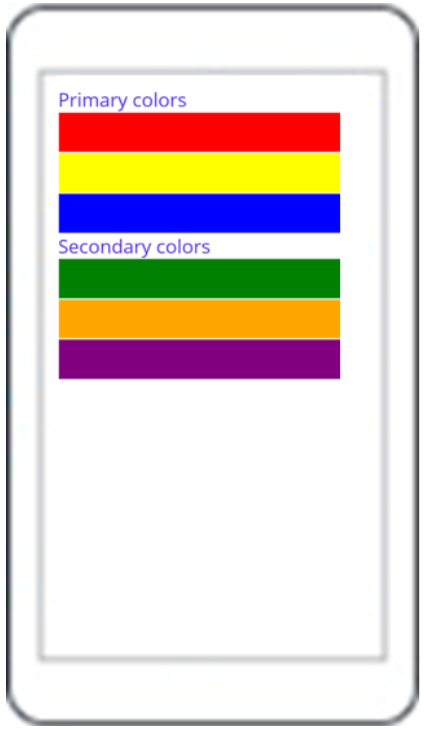


Layouts – StackLayout

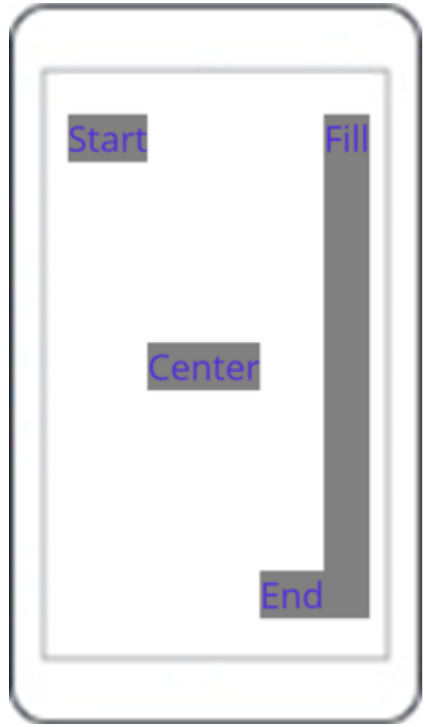
StackLayout



VerticalStackLayout



HorizontalStackLayout

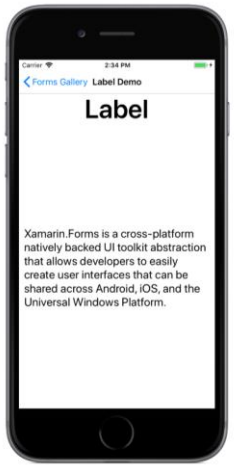


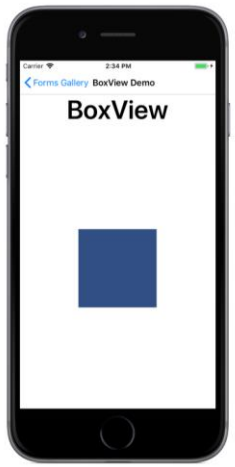
StackLayout…

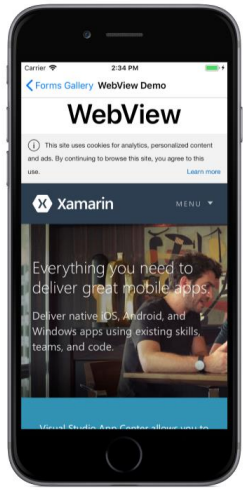
- HorizontalLayout, VerticalLayout
  - Individual layouts for single direction
  - Separate LayoutManagers with Measure methods
  - Recommended

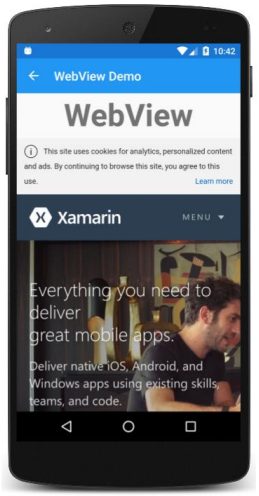
- StackLayout
  - Wraps **HorizontalStackLayout** and **VerticalStackLayout**
  - Has Orientation
  - Useful for adaptive layouts

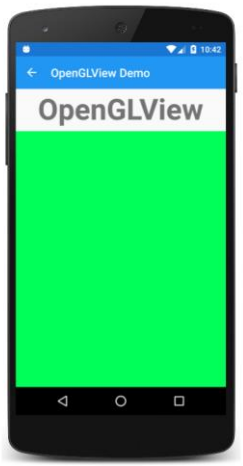
Content Presentation

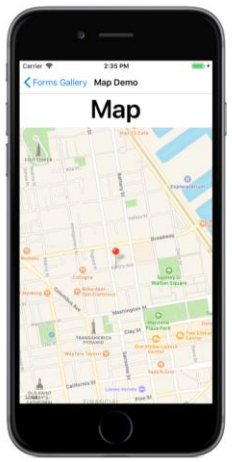


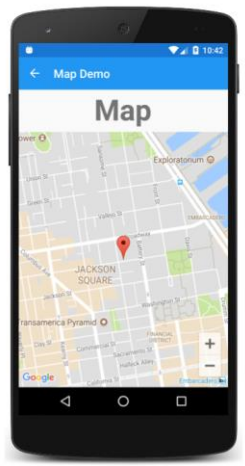




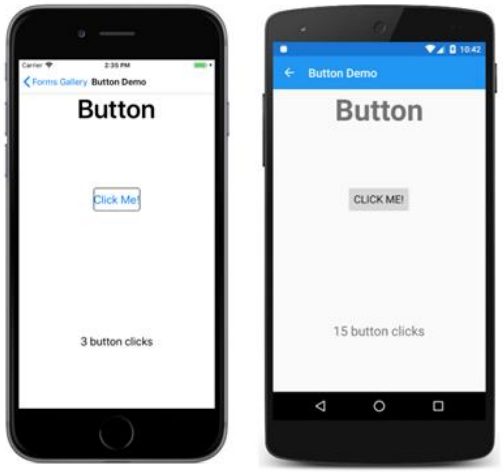


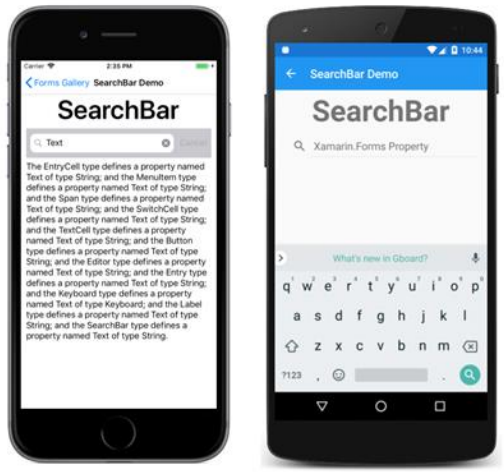


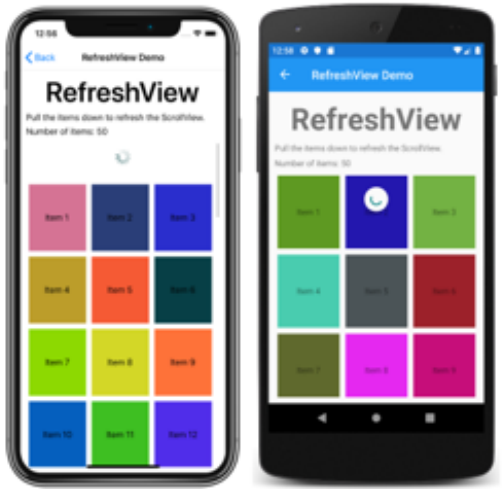


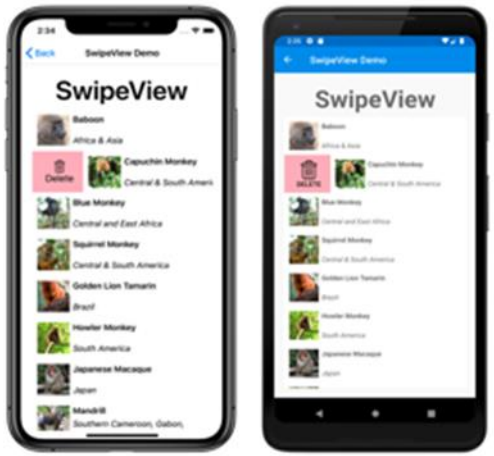


Actionable Controls

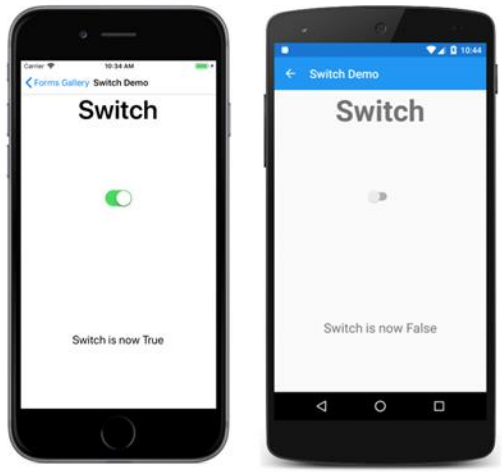


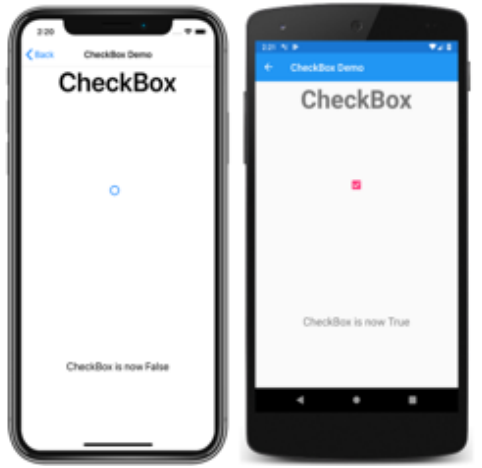




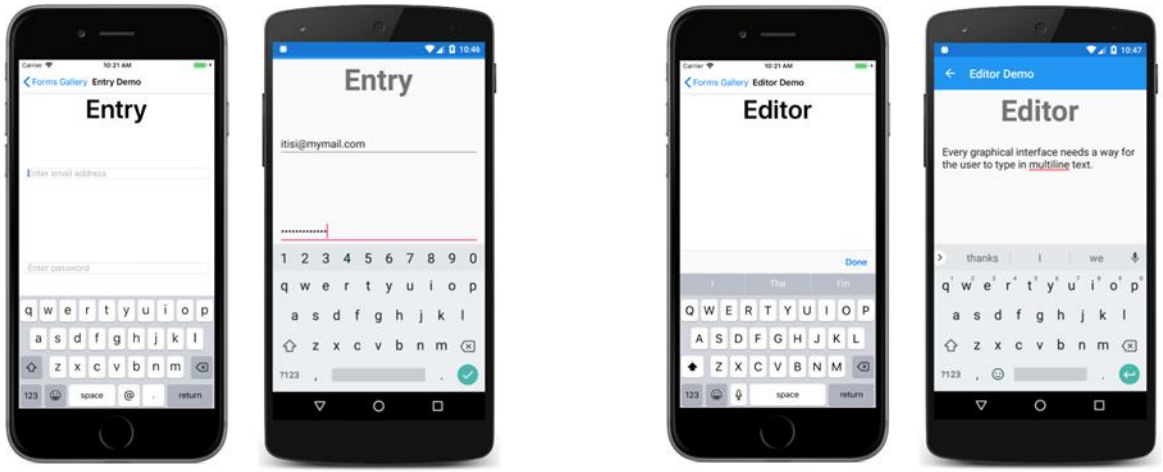


Setting Values

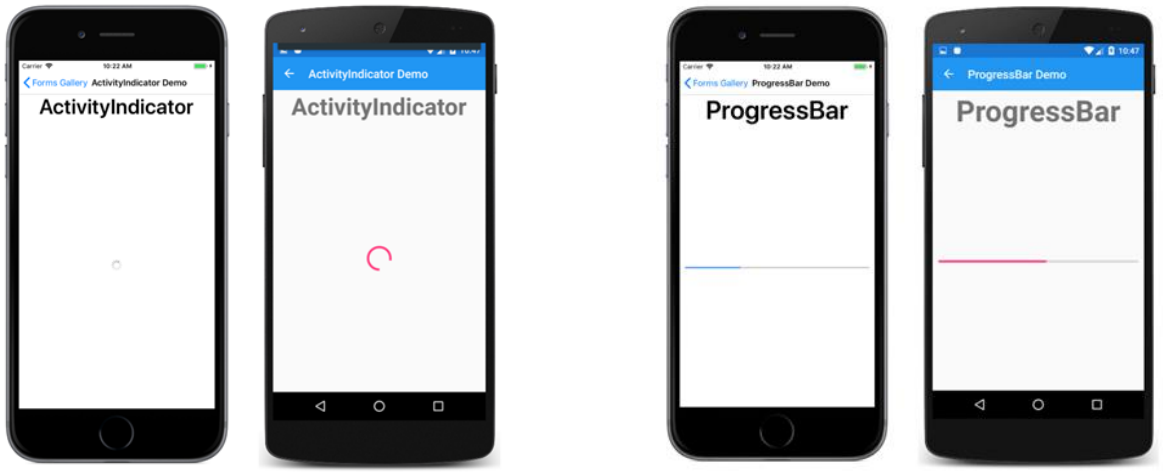




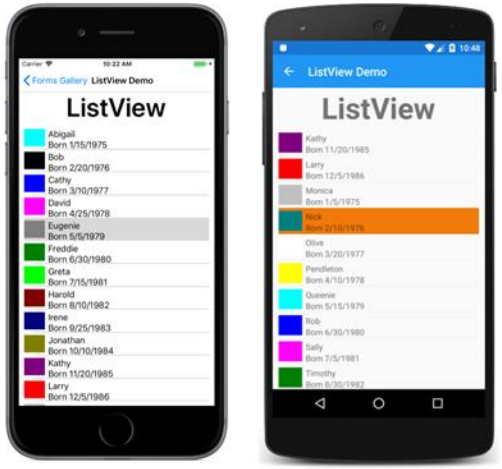
Editing Text

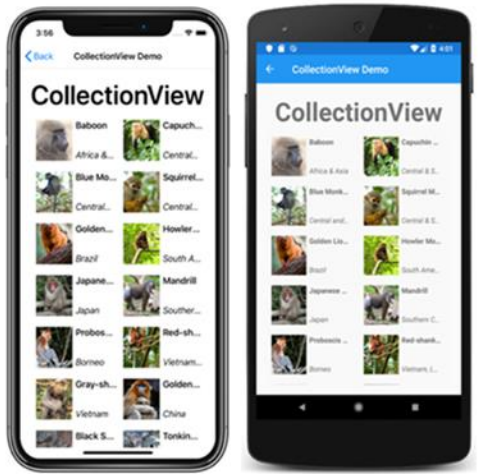


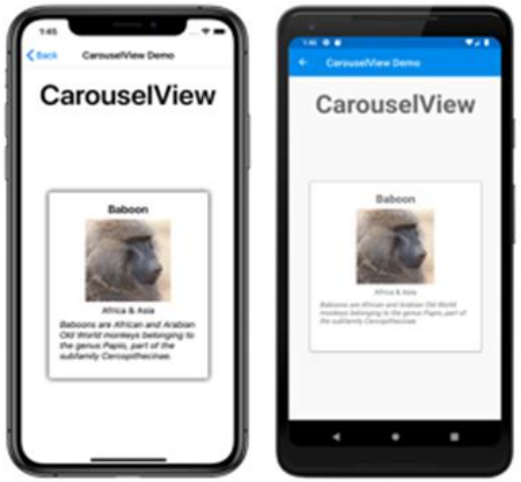
Activity Indication

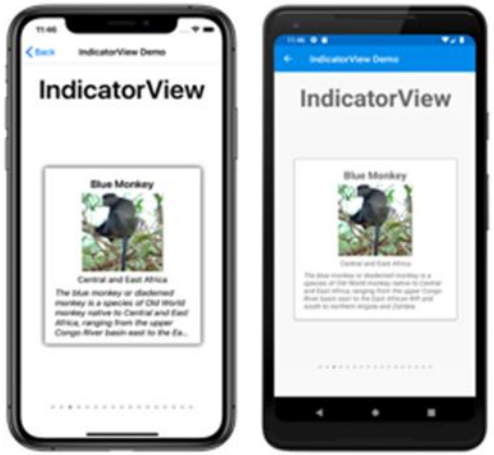


Collections

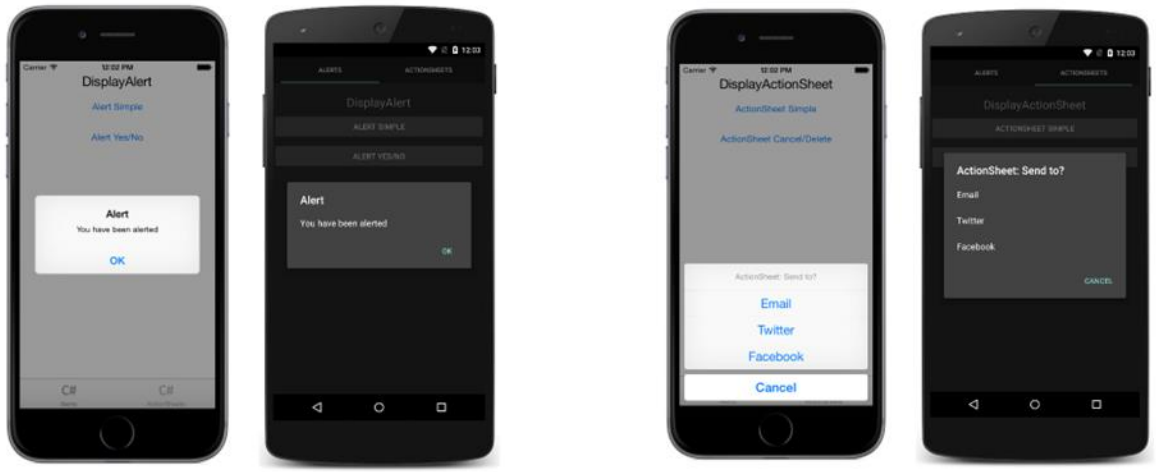








Pop-ups


DisplayAlert
Alert Simple
Alert Yes/No
Alert
You have been alerted
OK
DisplayAlert
ALERT SIMPLE
ALERT YES/NO
Alert
You have been alerted
OK
DisplayActionSheet
ActionSheet Simple
ActionSheet Cancel/Delete
Email
Twitter
Facebook
Cancel
DisplayActionSheet
ACTIONSHEET SIMPLE
ActionSheet: Send to?
Email
Twitter
Facebook
CANCEL

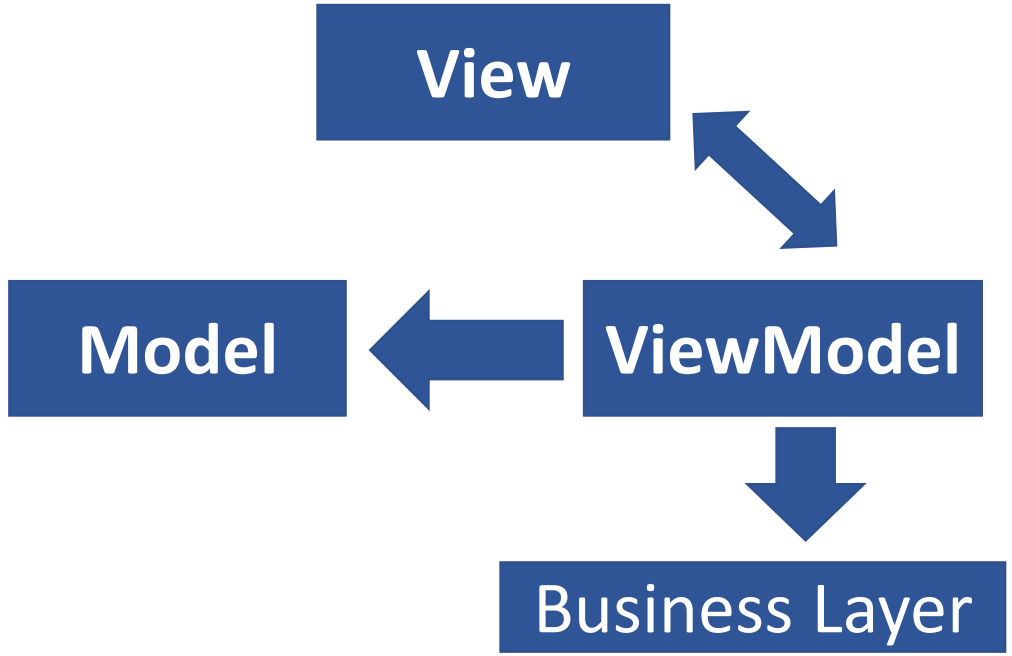
Demo time!

Model View ViewModel

Model – Reprezentuje data.

View – Zobrazuje data uživateli a dává mu možnost ovládání programu a zadávání nových dat.

ViewModel– drží si kontext aktuální (části) obrazovky.



Data binding

- Connection of code and XAML
- **Between ViewModel and View**
- `Text="{Binding Path=Operand1}"`
- **Between controls in View**
- `Text="{Binding Source={x:Reference Name=DisplayAlertButton}, Path=Text}"`

MVVM Frameworks

- MVVM Cross
- Simple MVVM
- MVVM Light
- Catel
- ReactiveUI

.NET MAUI Blazor

- Use Blazor in applications
- Razor, standard Blazor code
- Hosted in WebView
- Runs natively
  - No SignalR, no WebAssembly
  - Access to system APIs (file access, network access…)
- But why?
  - Blazor/web developer using razor – develop mobile applications
  - Use Blazor components in any .NET applications (MAUI, WPF, UWP…)
  - The cross-platform UI stuff is already solved – it’s in a browser!

MAUI vs. Xamarin

- .NET 6, C# 10
- Single project
  - Unified working with images, fonts, splash screen and other resources
  - Custom Renderers -> Handlers
- IoC/DI
- Unified working with WinRT
- Official support for Mac OS

Xamarin Migration

- Current solution - manual process
  - Create new project
  - Copy code files
  - Bulk rename Xamarin namespaces to MAUI namespaces
  - Update dependencies to target .NET 6 and MAUI
- Expected solution - .NET upgrade assistant ()
  - Automatically migrate to single project and swap namespaces
  - Update dependencies (manual)
  - Register compatibility services or renderers (manual)

Current state

- Implementation
  - All platforms supported as single project
  - Most controls are available, still working on some attributes
- VS integration
  - Project templates available
  - Compilation, errors, warnings, syntax highlighting, IntelliSense
- Documentation
  - Installation, Getting started, Sample application, Basics...
  - Release notes for Preview versions
  - Areas specific to MAUI (control handlers…)
  - New controls documentation added every month
  - Deeper knowledge - linking to Xamarin documentation

Current state

- Current release: Preview 14 (03/2022)
- Planned GA release: Q2/2022
- Next GA release – synced with .NET 7 release

Advantages of .NET MAUI

- XAML
- .Net
- Good documentation (some for Xamarin)
- Sharing code – 90%+

Materials

- Slides – IS
- Current state of code – IS
- GitHub: https://github.com/jasho/cookbook-maui