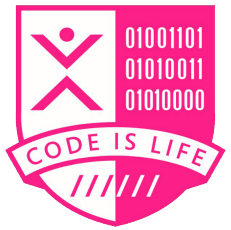
Nick Kozlov, Spiridonov Maximilian

Telegram: @EnormousRage, @maxspT



What is Xamarin now?

- **Xamarin Platform** - the main product, which allows you to make mobile apps with Visual Studio
- **Xamarin Test Cloud** – with this service you can test your apps on more than 1000 different mobile phones.
- **Xamarin Insights** – this service allows you to collect statistics on mobile applications: the number of application starts, user actions, application failures and more
- **Xamarin University** – online courses for learning Xamarin

Mobile development

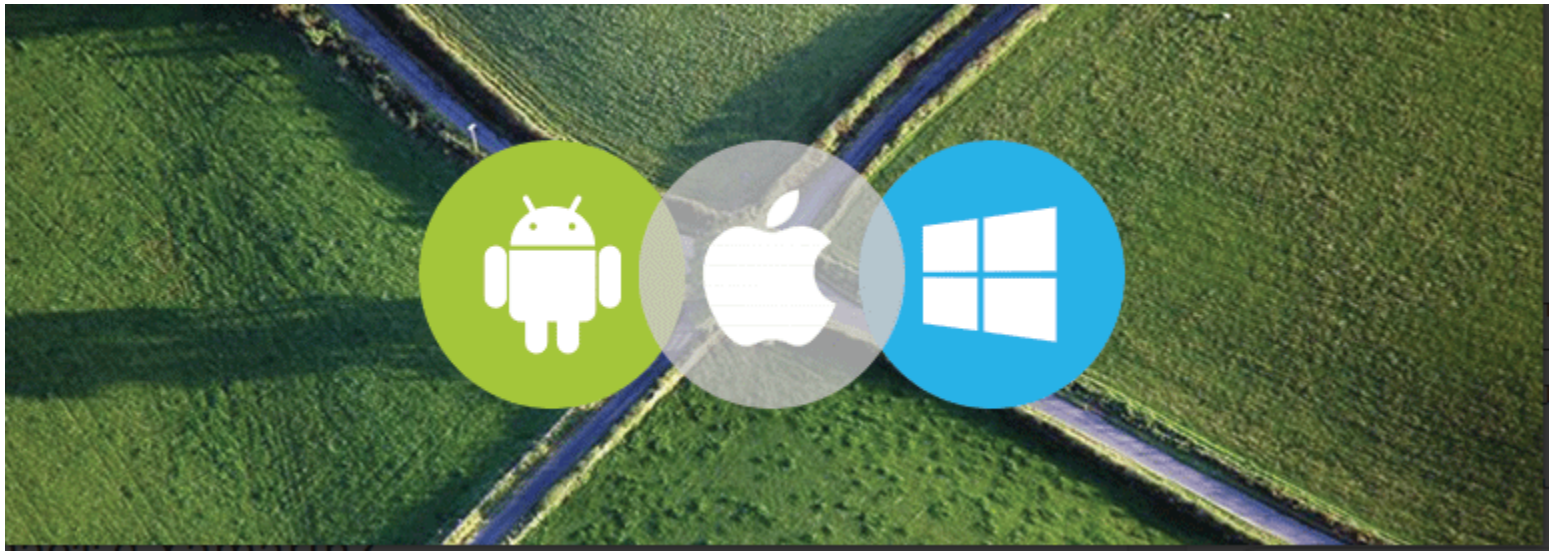
Mobile apps are developed for more than one platform (IOS & Android) according to statistics. So, developers have some difficulties:

1. **Different approaches** – to design GUI. Developers should make their apps and **adjust** them to different interfaces.
2. **Different API** – difference in program interfaces and implementations of various functions also requires the developer's **attention**. **Every platform has it's specific features** there
3. **Different platforms:**

Xcode – Swift and Objective-C

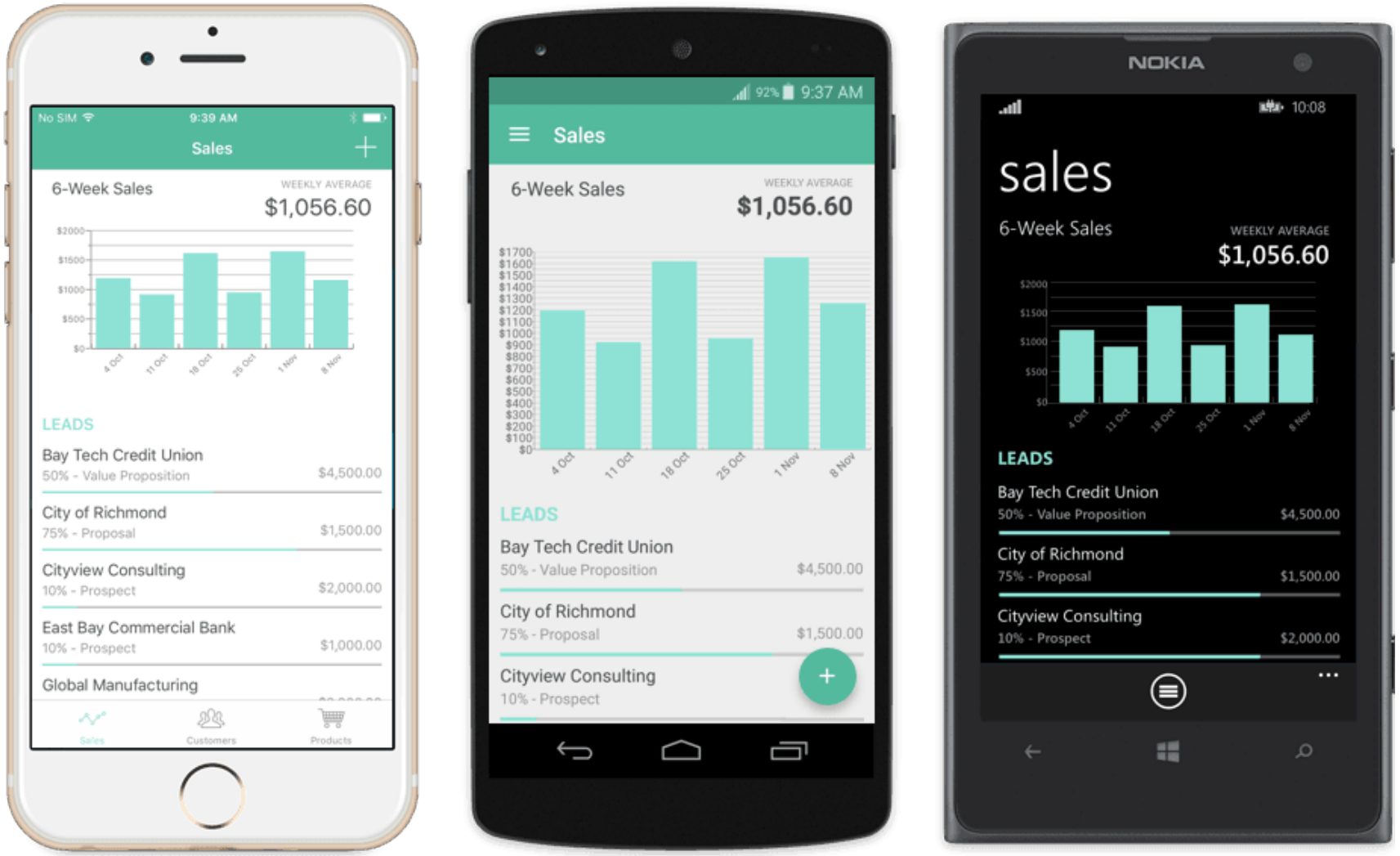
Android Studio, Eclipse – Java

Key feature of Xamarin



Code and app elements are reusable in cross platform app development. It saves a lot of resources

Xamarin is the best solution for this projects



One code – million of possibilities

Why we need Xamarin ?

- **Business-logic** of the app is not changed when you developed it one time. For example, if we are talking about internet-banking than there are the same (not 100%) backend for desktop, web and mobile solution. With Xamarin you can easily build apps with business-logic
- Algorithms developed on C# for a single platform can be used for all other platforms. For example, more than 75% of code is reusable **(Xamarin.Android and Xamarin.iOS)**

± Xamarin

+

1. Economy of resources
2. Reusable code
3. Easy to add a new platform
4. A single language – C#
5. Cross platform

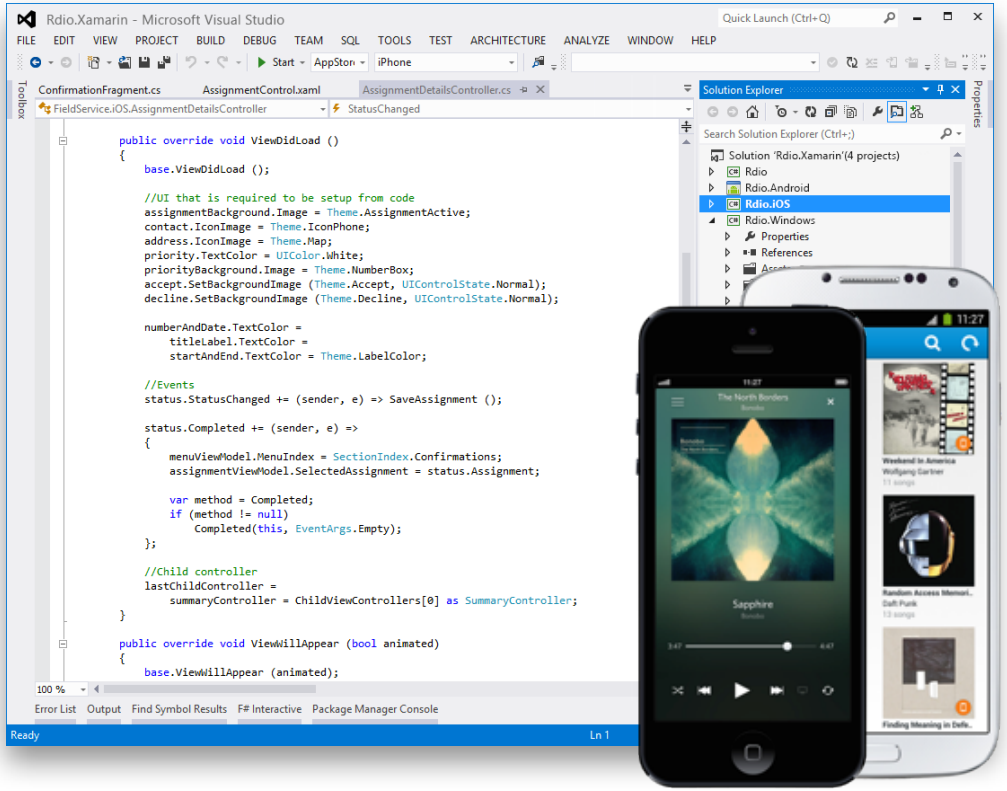
-

1. Without experience in mobile development there is no point to use Xamarin
2. There aren’t a lot of basic controls
3. Small community
4. If you need to develop IOS apps you need a Mac

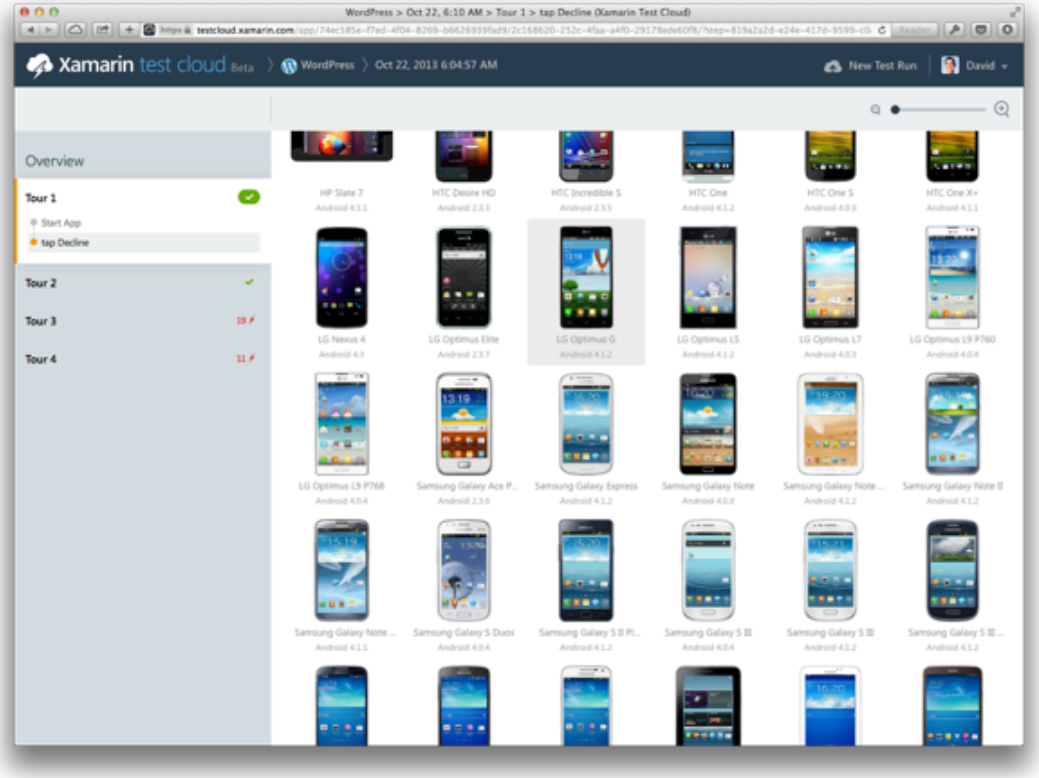
Cross platform

- **C#** – allows you to use all power of .Net Framwork like: Generics, LINQ, Parallel Task Library
- **Mono .NET framework** – allows you to develop Android apps with .Net
- **Compiler** – compiles the executable file for IOS or use Mono.Net for Android. Also, the compiler produces various optimizations specific to the development of mobile applications, which reduce the use of memory and the size of executables.
- **IDE** – Visual Studio 2017 for Windows and Visual Studio for Mac

Xamarin tools and services

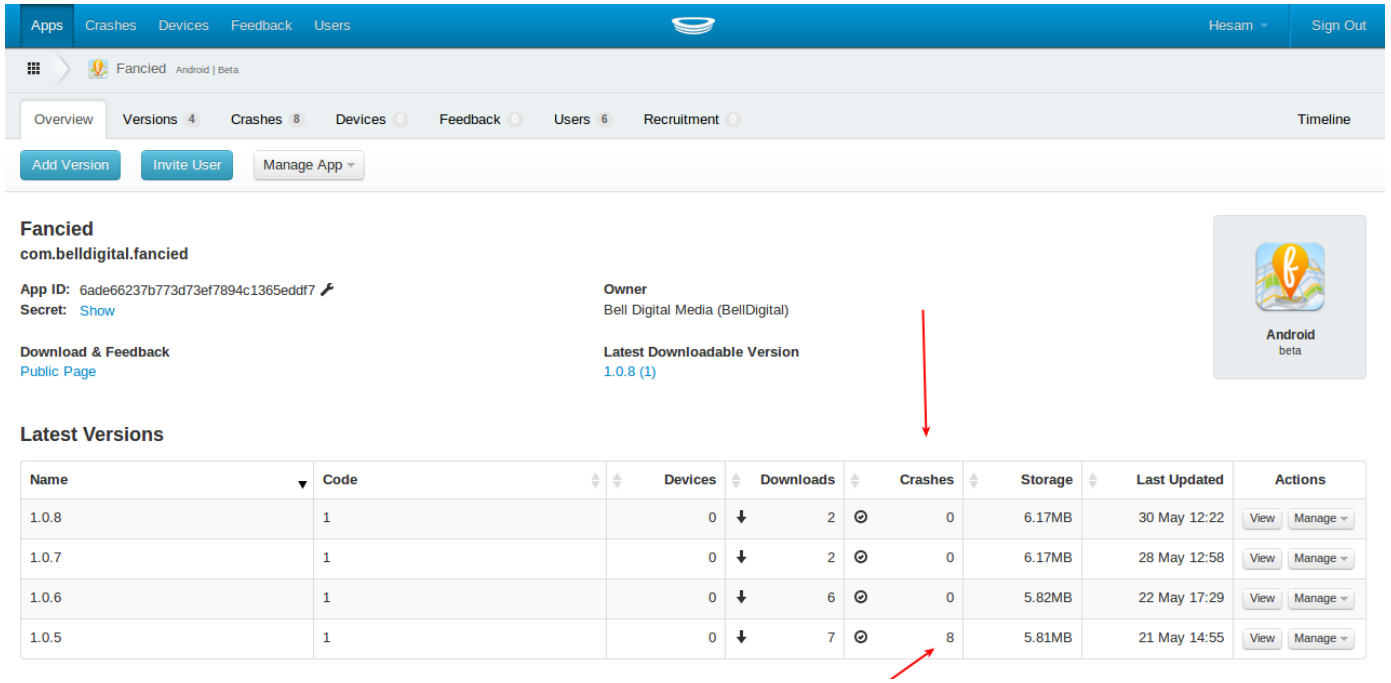


Native mobile apps
IOS, Android, Windows mobile



Automatically test your app
With Xamarin Test Cloud

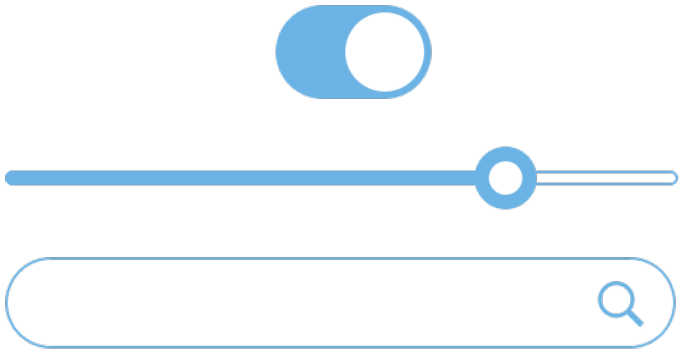
Xamarin tools and services



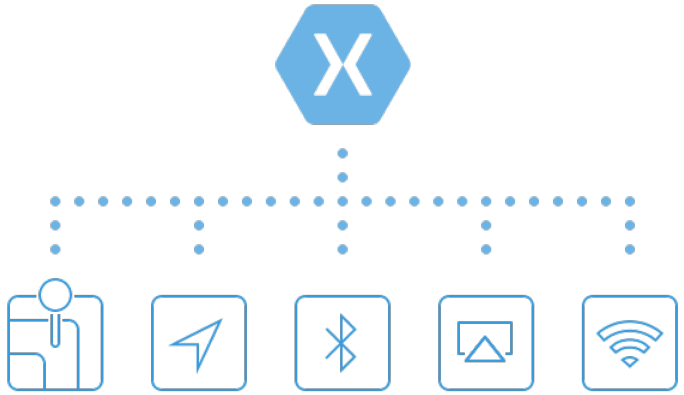
All statistics
crash reporting, user metrics, feedback

Make Xamarin apps
with Azure backend

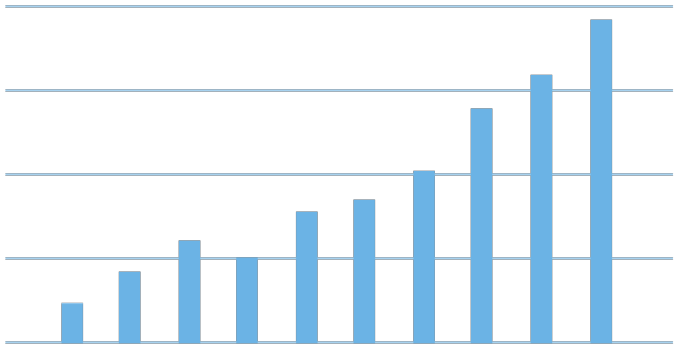
Native?



Native UI



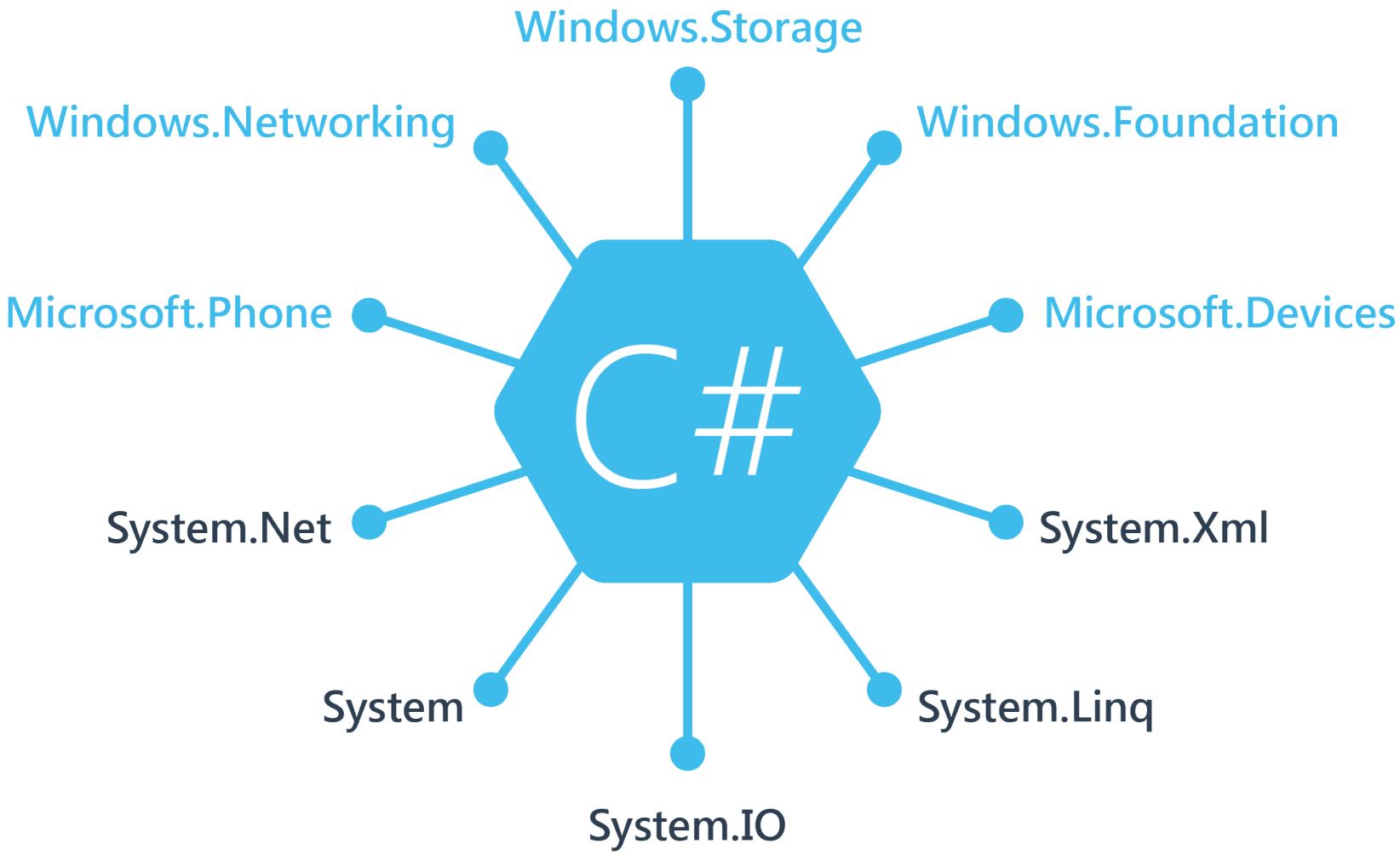
Native API Access



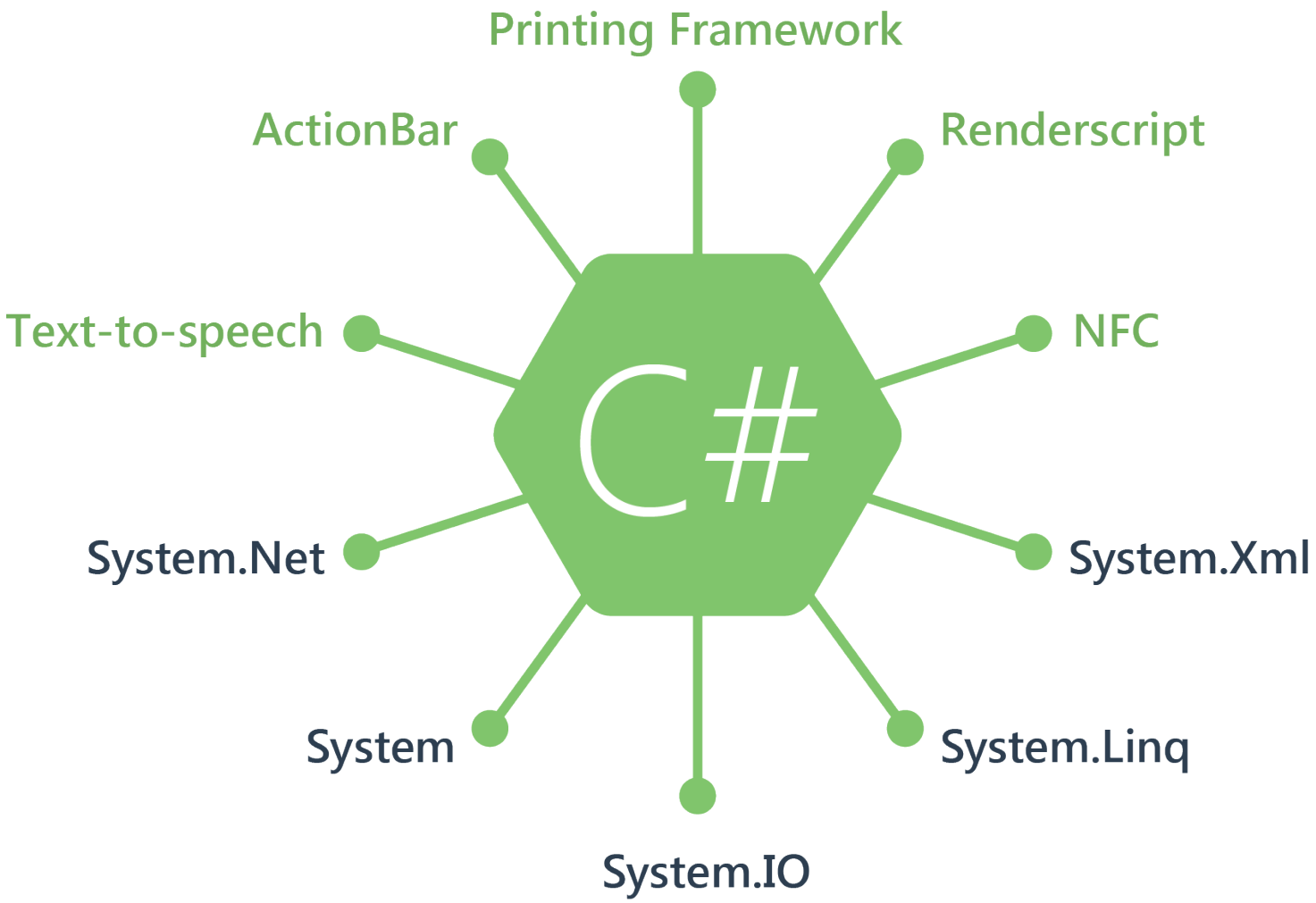
Native Performance

Xamarin apps look and feel native, because they are native

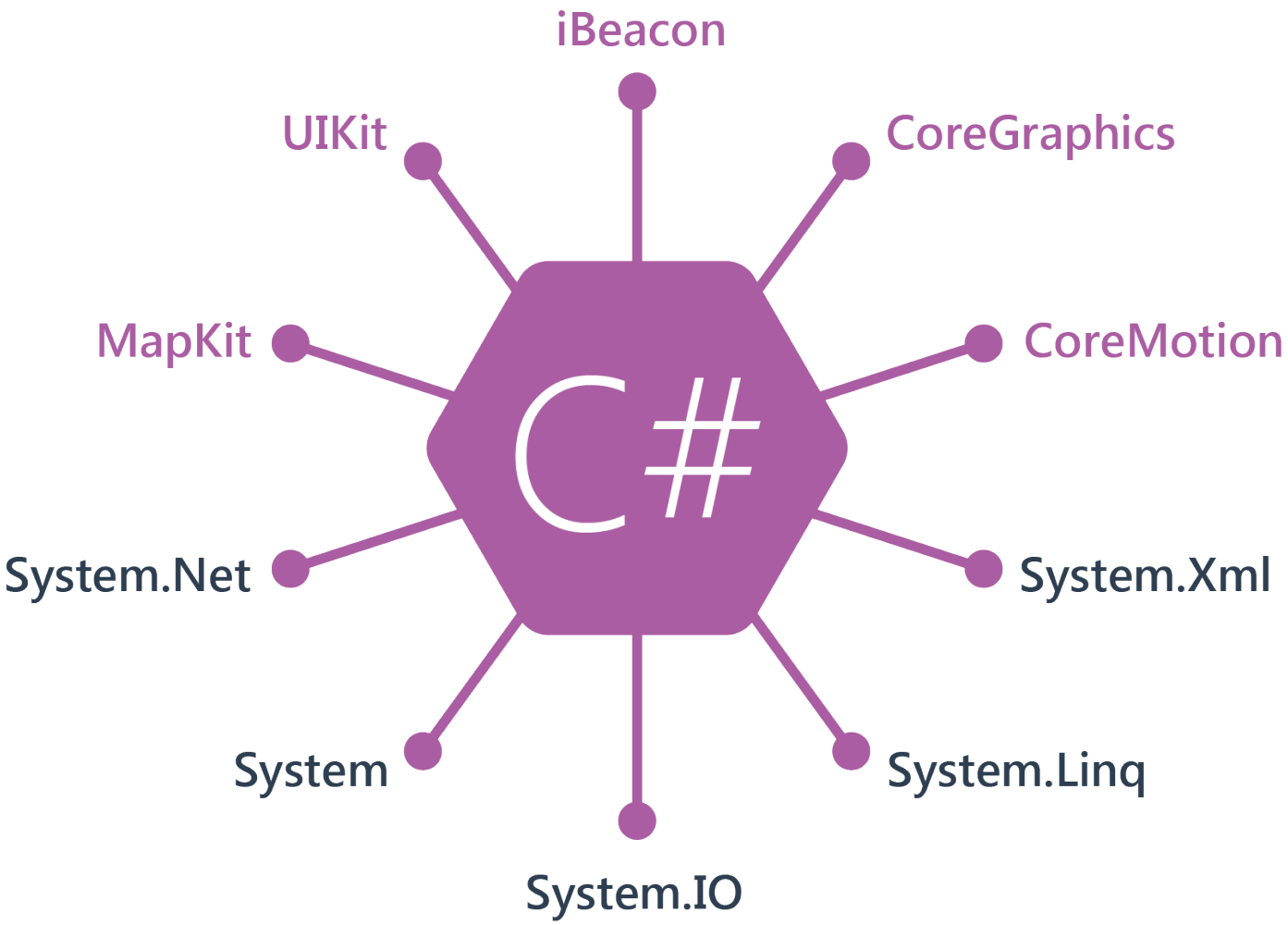
API : Windows coverage



API : Android coverage



API : IOS coverage



How to install Xamarin

Requires 3-4 hours first time

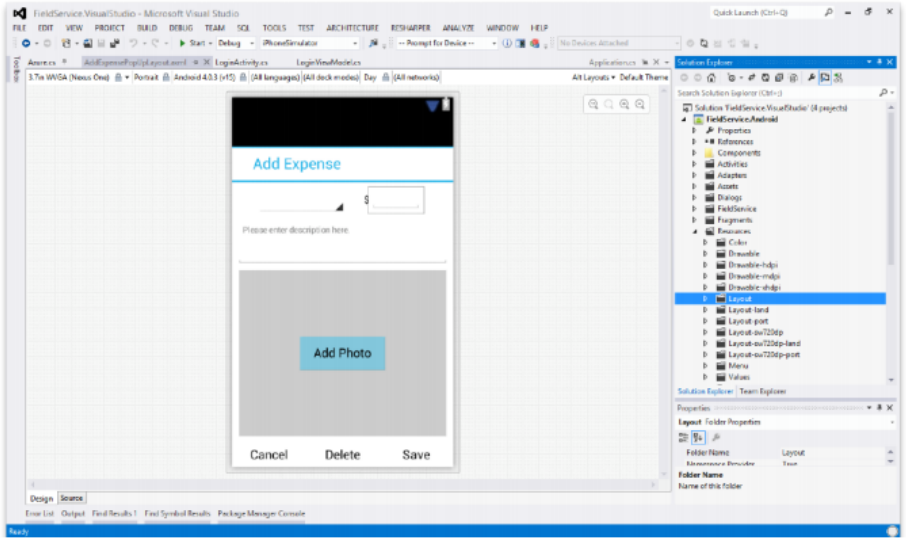
Official links

System requirements:

https://aka.ms/xamreq

Tools and instruments:

xamarin.com/download



How to install Xamarin

Hardware:

Minimum:

- CPU with Virtualization (Hyper-V) or Intel HAXM
- 6 GB of RAM
- High-speed HDD

Optimal:

- Intel Core i7 with 4 cores / 8 threads or more
- 16 GB of RAM or more
- SSD 256GB (SATA III or m.2)

Software:

- Windows 10 Pro / Enterprise
- Visual Studio 2017 Community / Enterprise (for IOS)
- Java SDK and JRE (8 and 7)
- Android SDK (21-26 api)
- Android AVD
- Intel HAXM
- Xamarin package for Visual Studio

Emulator | Simulator

Android

Visual studio emulator

- Hyper-V support
- 6 GB RAM
- Windows 8.1 or 10 Pro

aka.ms/droid-emu

Android emulator with HAXM

- Intel HAXM support

aka.ms/droid-haxm

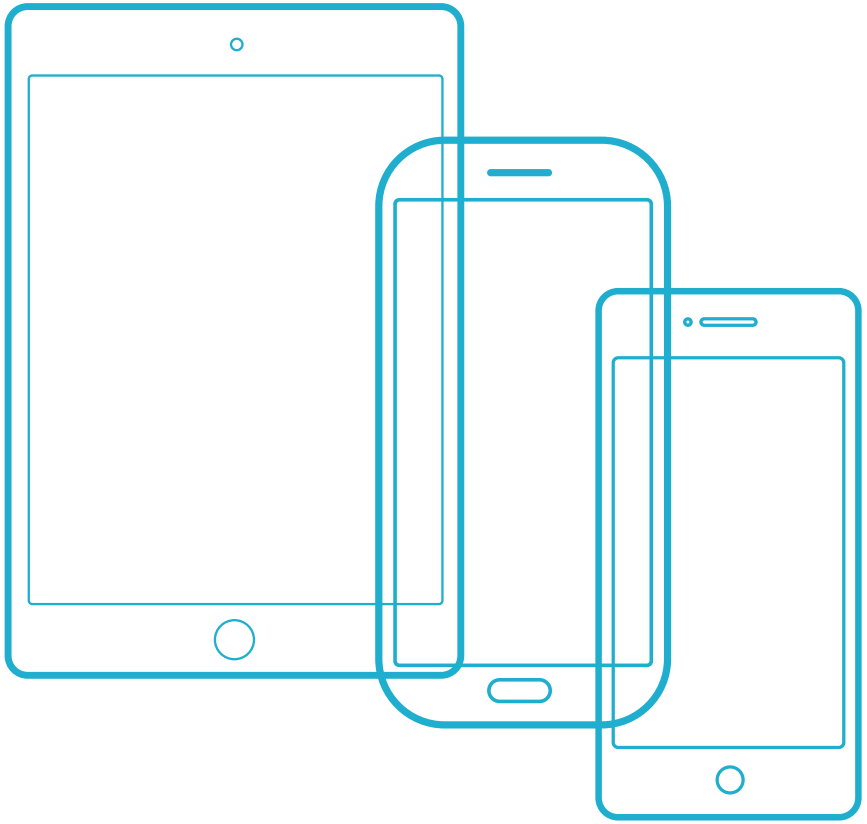
IOS

Xamarin Mac Agent for Visual Studio

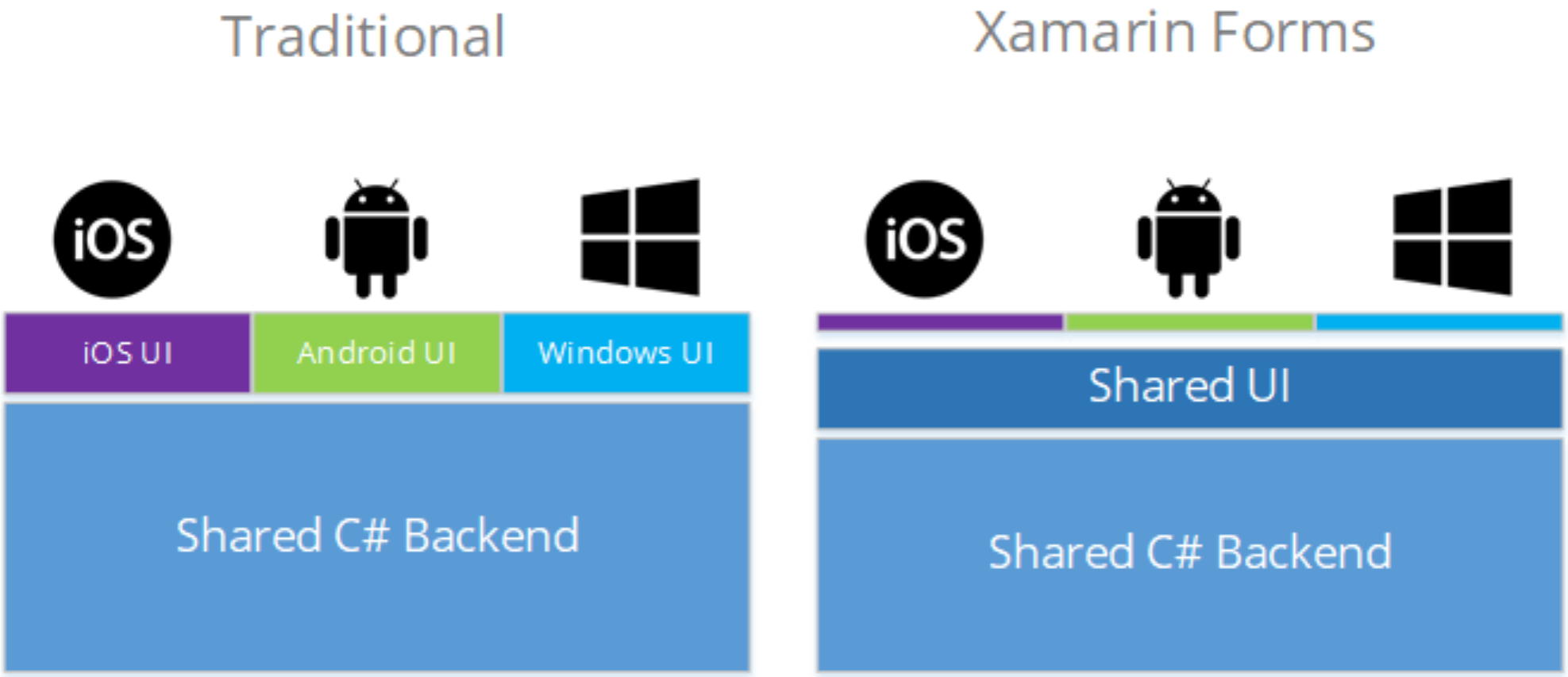
- You need a Mac
- Latest Mac OS X
- Latest Xcode

aka.ms/ios-sim

Let’s run our first Xamarin App



Traditional Xamarin and Xamarin Forms

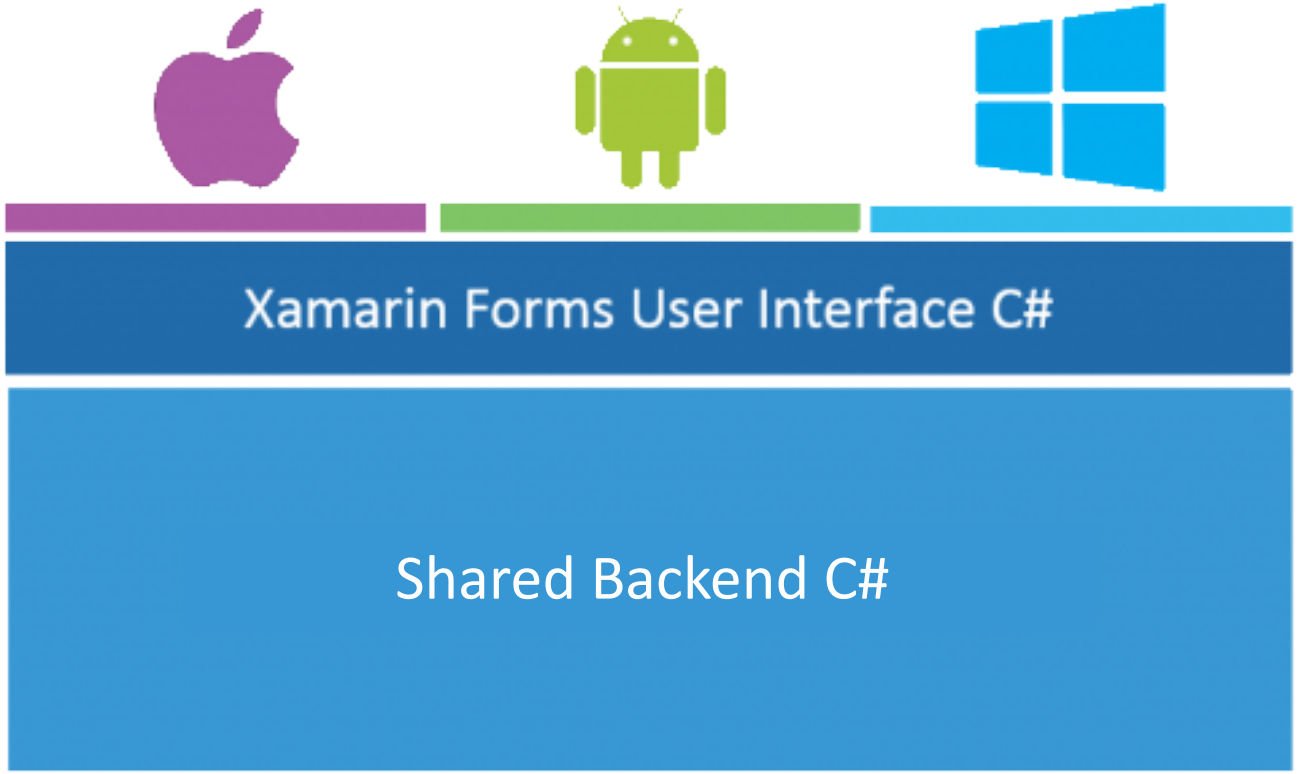


Xamarin.Forms

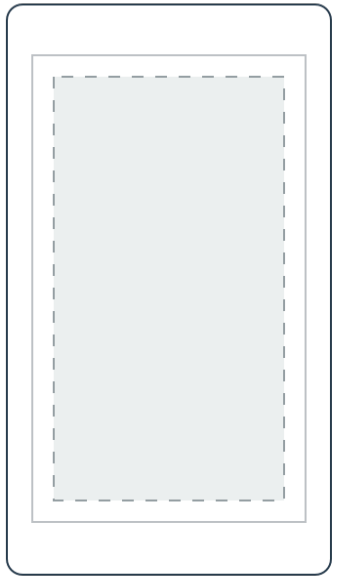
Quickly and easily build native user interfaces using shared code

Xamarin.Forms elements map to native controls and behaviors

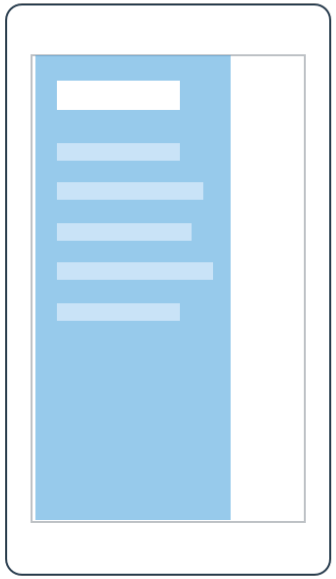
Mix-and-match Xamarin.Forms with native APIs



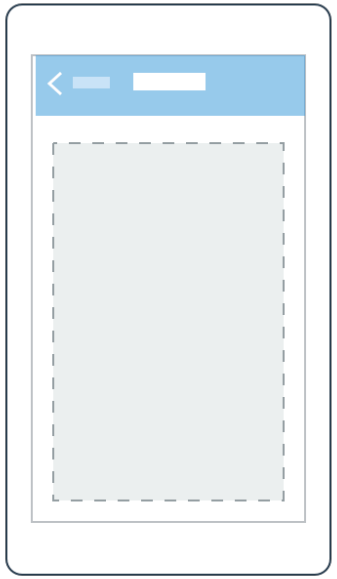
Xamarin.Forms : Pages



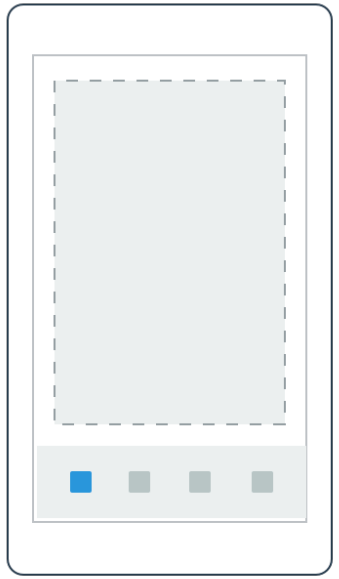
Content



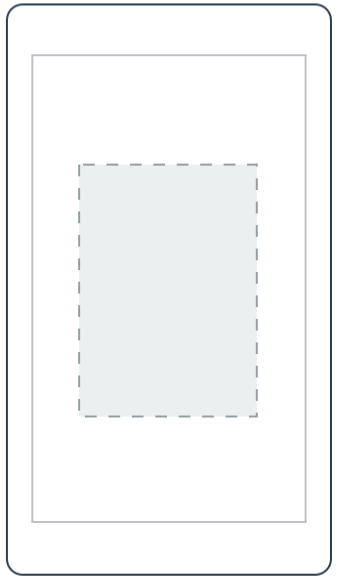
Master Detail



Navigation

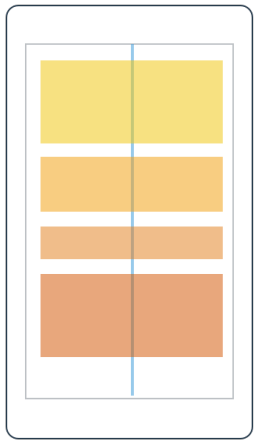


Tabbed

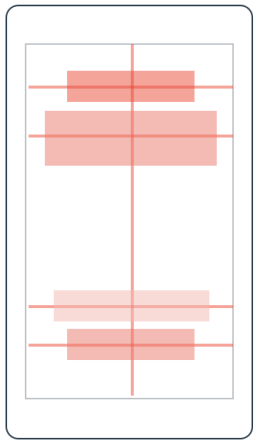


Carousel

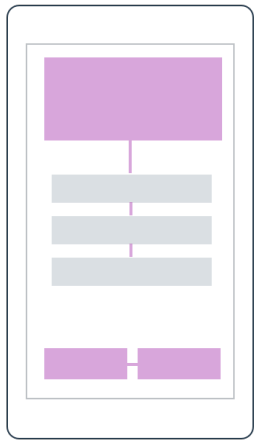
Xamarin.Forms : Layouts



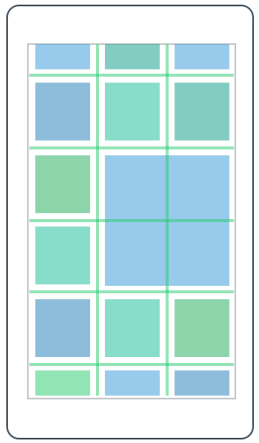
Stack



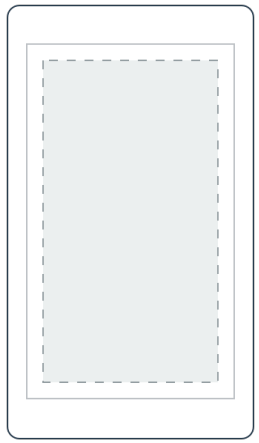
Absolute



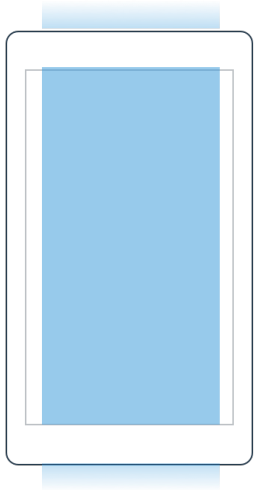
Relative



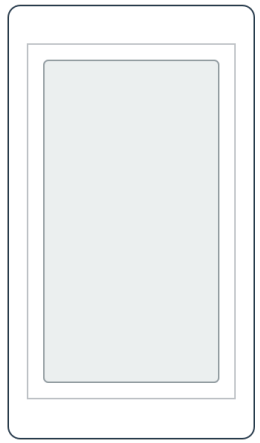
Grid



Content view



Scroll view

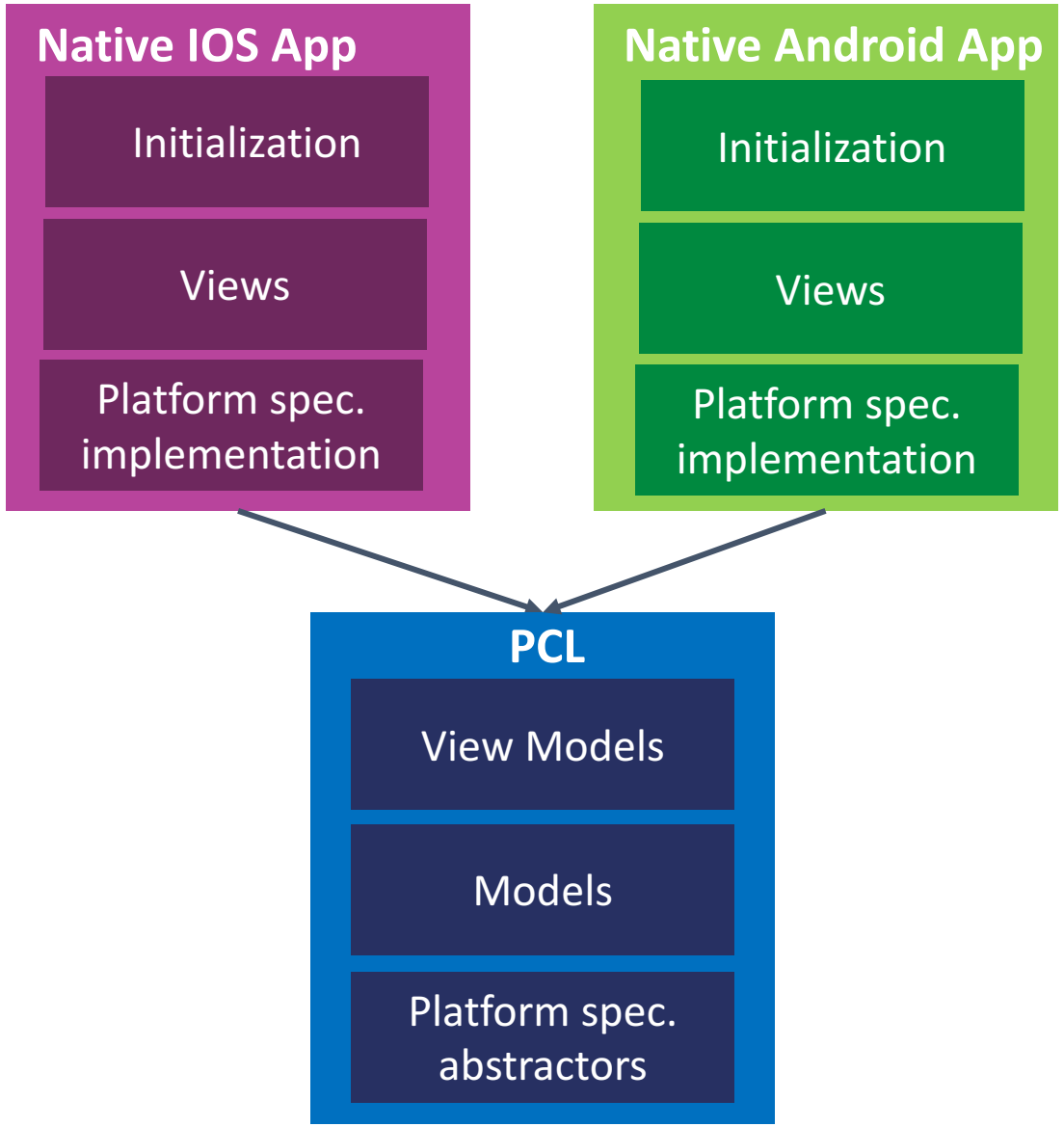


Frame

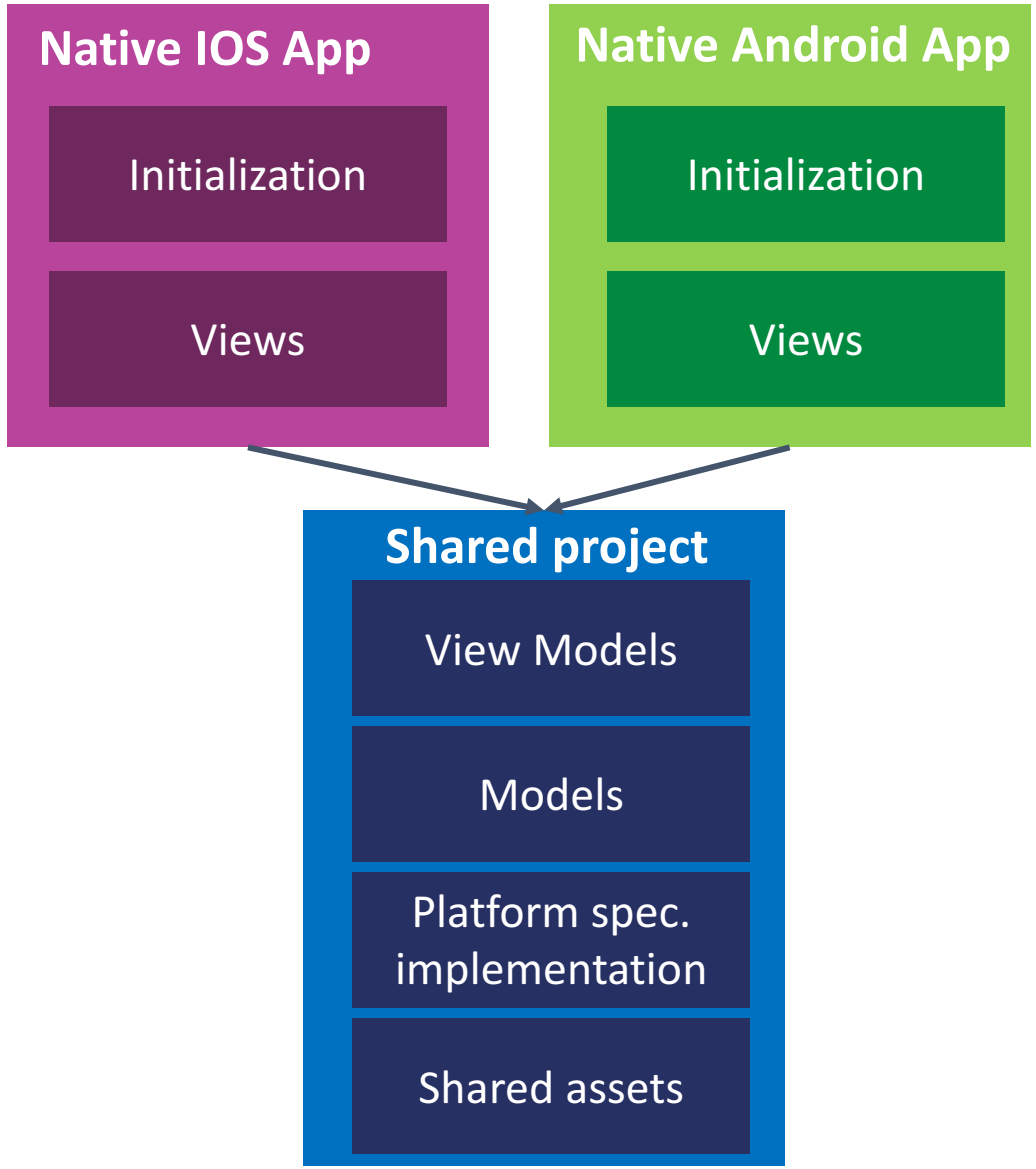
Xamarin.Forms : Controls

ActivityIndicator	BoxView	Button	DatePicker	Editor
Entry	Image	Label	ListView	Map
OpenGLView	Picker	ProgressBar	SearchBar	Slider
Stepper	TableView	TimePicker	WebView	EntryCell
ImageCell	SwitchCell	TextCell	ViewCell	

Xamarin.Forms: PCL vs Shared



Portable cross app and distributable libraries



Easiest way to share code in the same app

Xamarin.Forms: PCL vs Shared

PCL

Benefits:

- Allows you to share code across multiple projects.
- Refactoring operations always update all affected references.

Disadvantages:

- Cannot use compiler directives.
- Only a subset of the .NET framework is available to use, determined by the profile selected

Shared project

Benefits:

- Allows you to share code across multiple projects.
- Shared code can be branched based on the platform using compiler directives (eg. using #if __ANDROID__)
- Application projects can include platform-specific references that the shared code can utilize (such as *usingCommunity.CsharpSqlite.WP7* in the Tasky sample for Windows Phone).

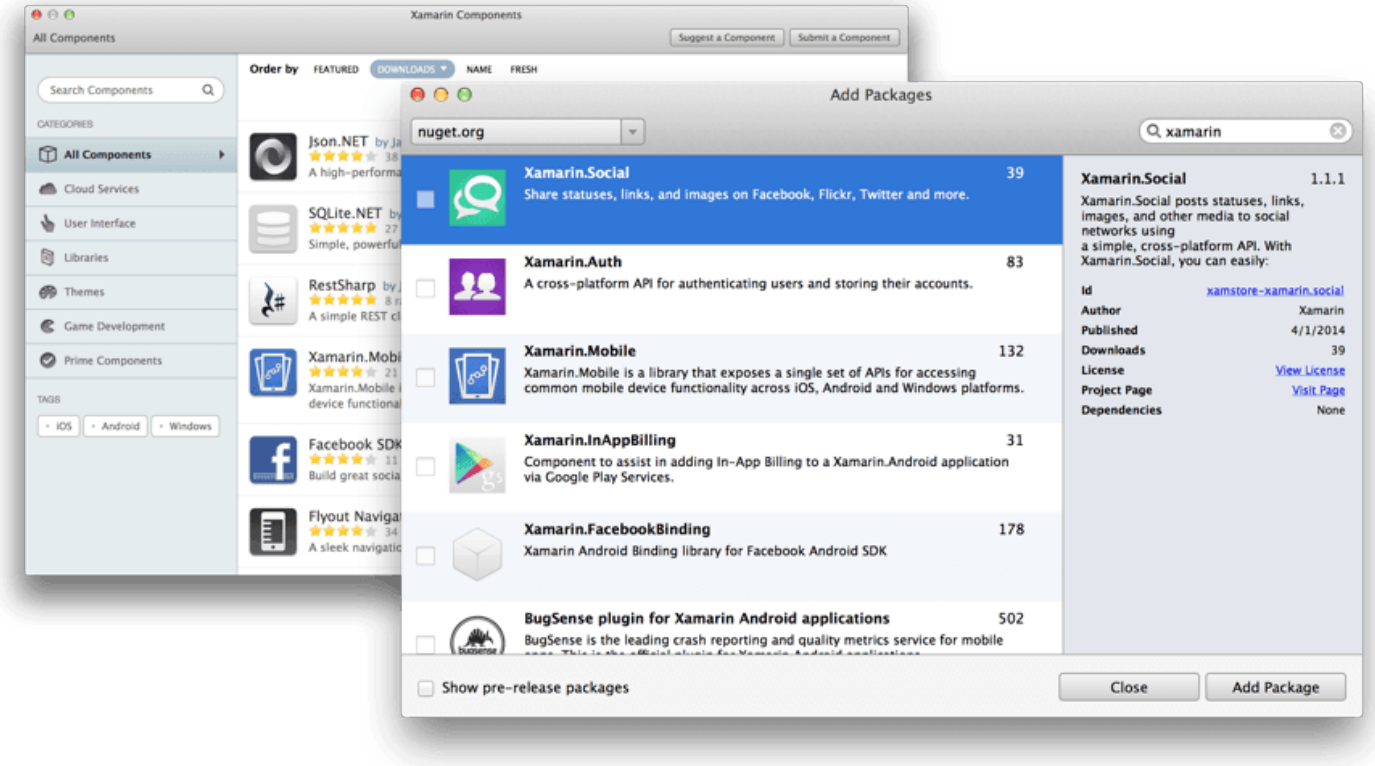
Disadvantages:

- Has no 'output' assembly. If you wish to share your code as a assembly then Portable Class Libraries or .NET Standard are a better solution.
- Refactorings that affect code inside 'inactive' compiler directives will not update the code.

Xamarin.Forms : Components source

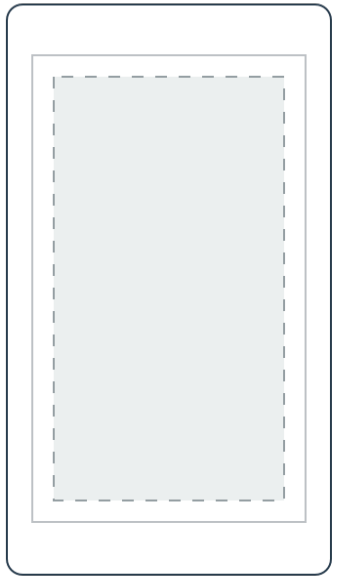
Components.xamarin.com

- Cloud services
- Libraries (SDK)
- UI
- Plugins
- Themes
- Game Dev
- Prime Components

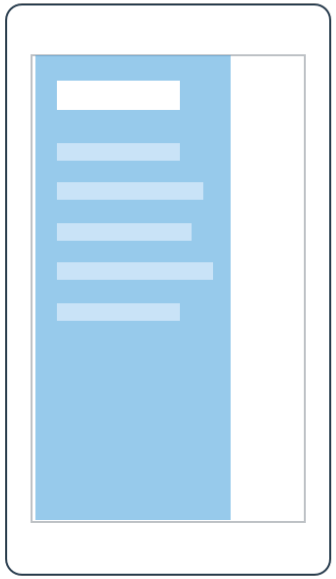


github.com/xamarin/Xamarin.Forms

Xamarin.Forms : Practice



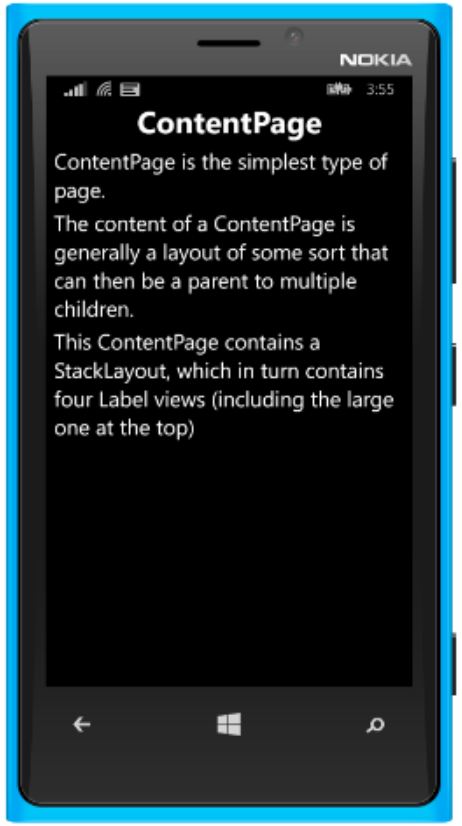
Content



Master Detail

Xamarin.Forms : Practice

ContentPage example

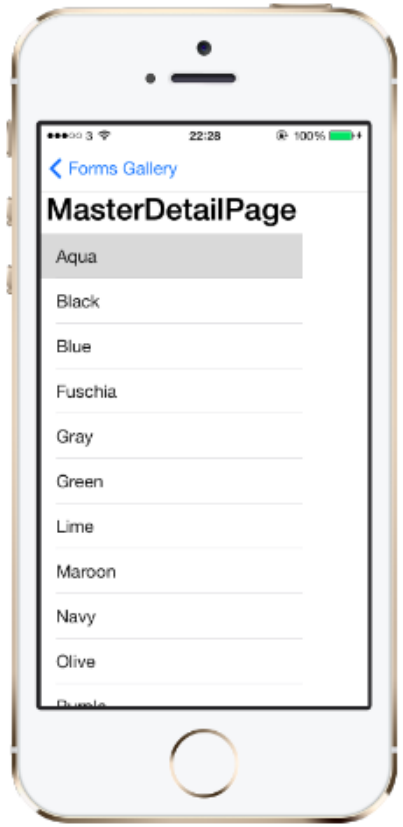


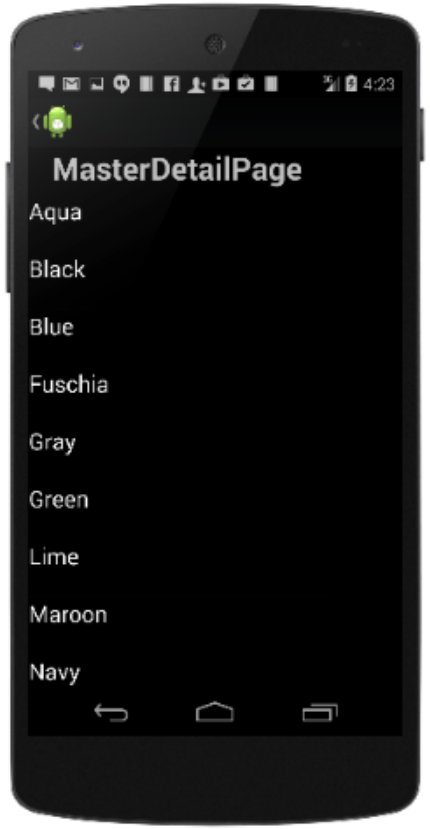
Xamarin.Forms : Practice

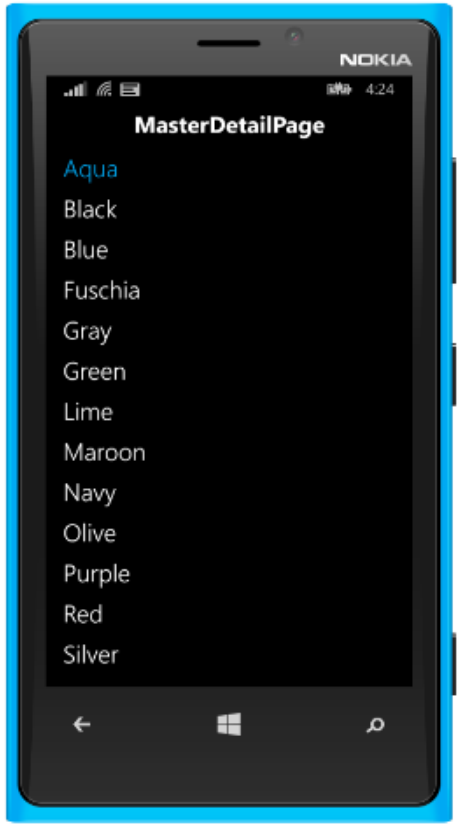
1. ContentPage: stack layout with Controls : Button, Label
2. ContentPage: absolute layout with Controls : BoxView, Entry
3. ContentPage: relative layout with Controls : Image, Editor
4. ContentPage: grid layout with Controls : ActivityIndicator, TableView

Xamarin.Forms : Practice

MasterDetailPage example







Xamarin.Forms : Practice

Add ContentPages to our MasterDetailPage

Useful links

- System requirements for PC: https://aka.ms/xamreq
- Tools and instruments for Xamarin: https://xamarin.com/download
- VS emulator for Android on Windows: https://aka.ms/droid-emu
- System requirements for PC to work with emulator: https://aka.ms/droid-emu-req
- Android-emulator with Intel HAXM: https://aka.ms/droid-haxm
- IOS-simulator instruction : https://aka.ms/ios-sim