Native cross platform

Development for Windows, Android and iOS with Xamarin

History

- **Built by the engineers who brought Mono, Monotouch, Mono for Android (OSS)**
- **Ximian (2001) → Novell (2003) → Xamarin (2011)**
- **2013 → V2**
- **2014 → V3**

Xamarin tools

- **Xamarin Studio**
  - → For Mac OS
  - → For Windows
- **Visual Studio extensions**
  - → Android designer
  - → iOS designer
  - → Connection to Mac OS for build/debug

Xamarin

Xamarin is an excellent product which has brought revolution in the field of software development.

It is the one which brings the .net and c# to both Android as well as ios.

It is good to know that in spite being fully .net; it is capable in producing true Android as well as ios apps at the same point of time.

This simply means that it holds the capability to fulfill the distribution requirements of the Google's and ios own stores.

A brief intro into the working of Xamarin

Basically Xamarin is based on top of mono touch. When we say that, we mean that with it, it gets possible to develop both ios and Android apps along c#. The ios of Xamarin does full ahead of time compilation which leads to this interoperability.

As the compilation happens, there is produced an ARM binary which makes it compatible with Apple's app store. "Xamarin. Android" takes the benefit of Just in time compilation on the Android devices.

Reasons supporting the use of Xamarin for ios and Android cross-platform development:

Lesser to understand and learn: Any experienced .net/c# developer would feel working at home along Xamarin.

It makes one implement c# and does full implementation of the .net class libraries.

New users of Xamarin are not forced to learn much and they need to learn only the C# language and one core set of classes that could be effective on both the platforms.

It leads to faster time to market as Xamarin reduces the development time commendably. If you want a faster way of development, then it can be trusted with ease.

There are fewer bugs- As it involves less writing down of the codes, there are chances of lesser bugs along Xamarin.

Dev Environment

Setup with Visual Studio (PC)

Build

Build

Setup with Xamarin Studio (Windows)

Build

Setup with Xamarin Studio (MacOS)

Build

Build

Setup with Visual Studio (Mac)

Visual Studio

Build

Build

iOS

WHO CARES ABOUT CROSS PLATFORM?

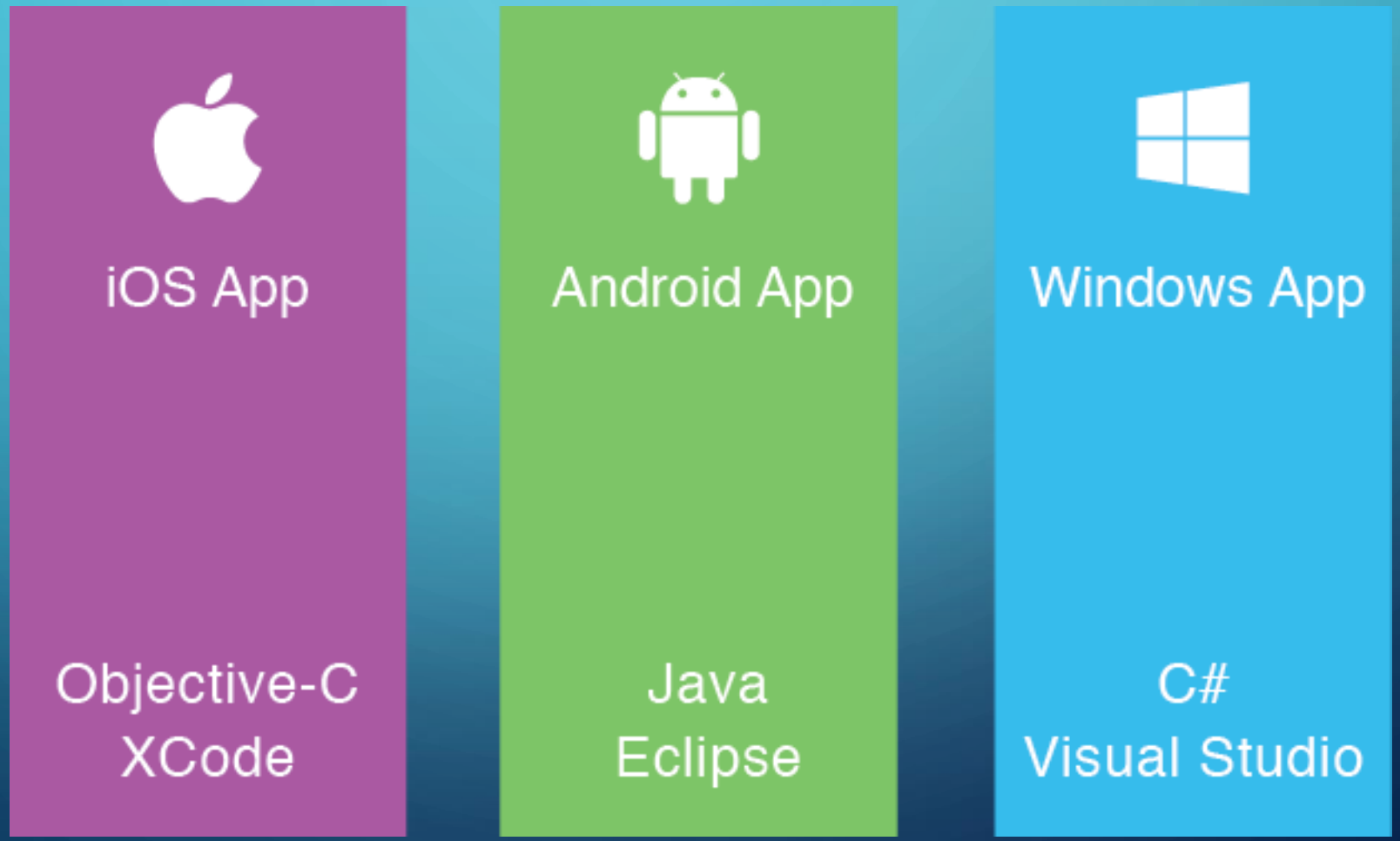
- 2013 App Economy was 68 billion USD according to DeveloperEconomics.com or roughly 10 USD per person
- 2016 estimated App Economy will be 143 billion USD according to DeveloperEconomics.com or roughly 20 USD per person
- Problem is that there is no OS monoply. What is a developer to do?

NO YOU CAN'T JUST TARGET ANDROID!

- Android dominates the phone market 81% shipping in Q3 2013 vs 13% iOS
- iOS dominates the tablet market 52% primary target is iPad only 28% primary target is Android

CROSS PLATFORM STRATEGIES

SILO APPROACH



SILO – WRITE APP ON EVERY TARGET

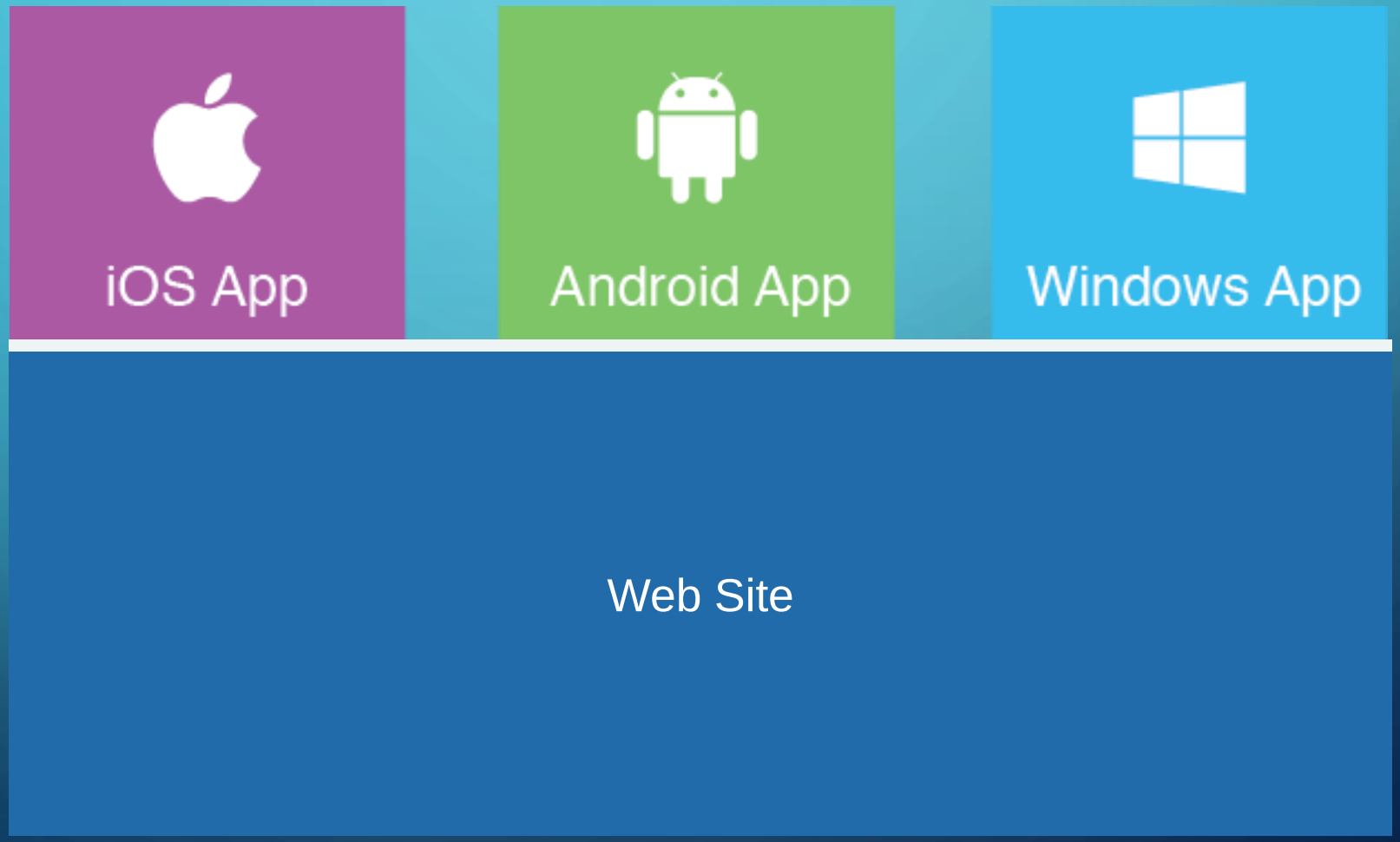
BENEFITS

- Full native experience
- Total access to the device as provided by SDK
- Share Web API

NEGATIVES

- Minimal re-use mostly on back end Web API
- Higher development cost from multiple teams (silo teams) or expensive multi-device developers
- Multiple codebases to maintain and extend
- One platform rules the others are subservient

TARGET BROWSER NOT OS



HTML – WRITE APP USING MOBILE WEB

BENEFITS

- Provide consist experience regardless of target
- Cheap as it is just HTML
- Single codebase to maintain and extend
- No need for revenue sharing as no need to be in app stores

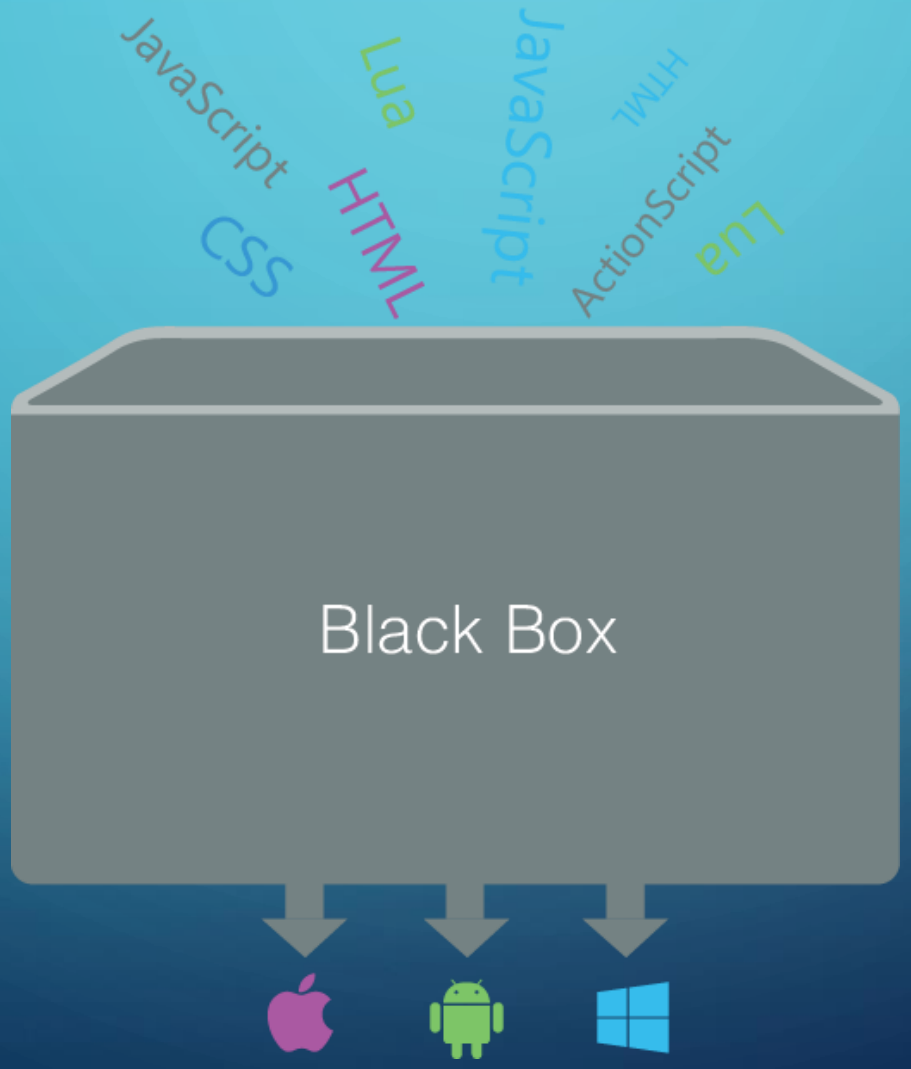
NEGATIVES

- User experience tends to be webish and not native
- Need to still test and debug multiple targets
- Features tend to be a subset common to all targets

HTML – WRITE APP USING MOBILE WEB

- Tools
  - HTML5
  - jQuery Mobile
  - Sencha Touch
  - ASP.NET
  - J2EE

TARGET DEVELOPER PLATFORM



MEAP – WRITE APP USING MOBILE ENTERPRISE APPLICATION PLATFORM

BENEFITS

- Provide consist experience regardless of target
- Cheaper as App is developed once for all targets
- Single codebase to maintain and extend
- Apps can be in app store if needed

NEGATIVES

- User experience tends to be webish and not native
- Need to still test and debug multiple targets even when MEAP only thing updated
- Features tend to be a subset common to all targets
- Vendor risk and lock in

MEAP – WRITE APP USING MOBILE ENTERPRISE APPLICATION PLATFORM

- Tools
  - Phone Gap
  - Dojo
  - Sencha

MDAP – WRITE APP USING MOBILE DEVELOPMENT APPLICATION PLATFORM

- Strategies
  - Tool generating target app
  - Write app in single language and compile multiple targets

MDAP – WRITE APP USING MOBILE DEVELOPMENT APPLICATION PLATFORM

BENEFITS

- Provide consist experience regardless of target
- Single codebase to maintain and extend
- Hit a lot of targets at once

NEGATIVES

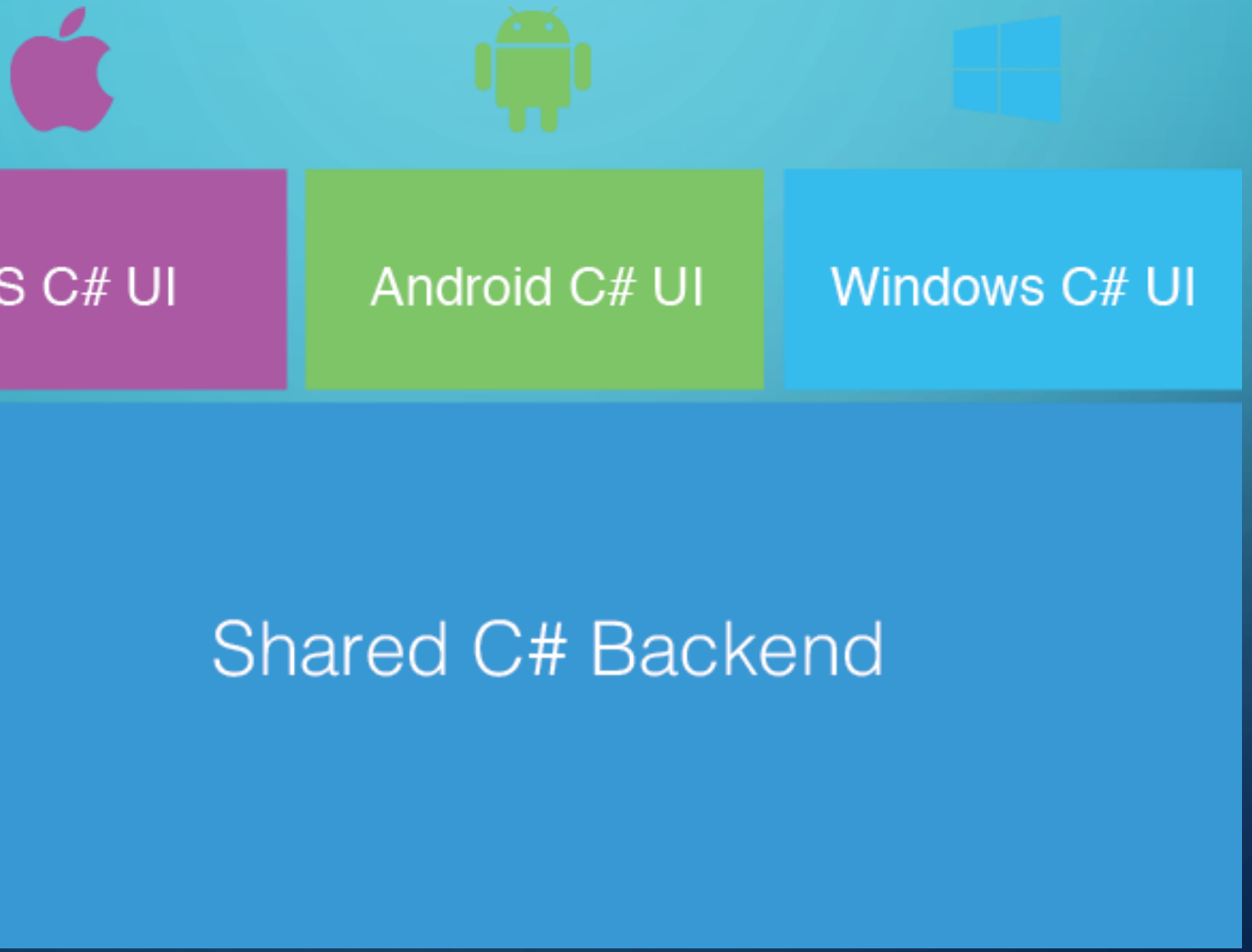
- Need to still test and debug multiple targets even when MEAP only thing updated
- Features tend to be a subset common to all targets
- Vendor risk and lock in
- May have to wait on new targets

MDAP – WRITE APP USING MOBILE DEVELOPMENT APPLICATION PLATFORM

- Tools
  - Appcelerator
  - Embarcadero
  - Rhomobile
  - RubyMotion
  - Unity
  - **Xamarin**

GOING CROSS PLATFORM USING .NET

XAMARIN APPROACH



xamarin forms

Xamarin principles

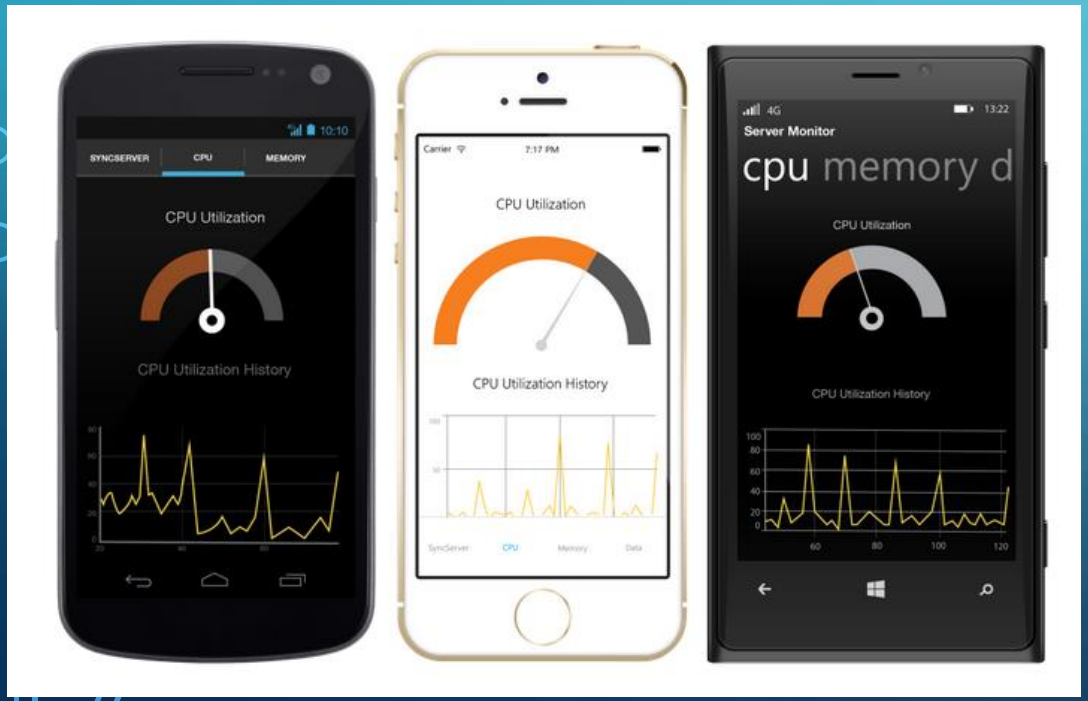
- Coding in C# (or F#)
- Taking advantage of the huge .NET ecosystem
  - → Tools
  - → .NET framework
  - → Libraries and components (community, OSS, NuGet, etc)
- Adding native iOS / Android components
- Access to native iOS / Android code

Share more

- **Sharing the User Interface**
  - → Define once
  - → Run on supported platforms
- **Quick prototyping**
  - → Try quickly how the UI works
- **Evolve your application**
  - → Start in 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	



Syncfusion
Deliver innovation with ease

Telerik
Develop experiences

steema

ComponentOne

DevExpress

INFRAGISTICS

Ahead of Time Compilation

What is full ahead of time compilation?

Full Ahead of time compilation or AOT is the feature of the mono runtime code generator. Mono time code generator works in two modes.

The first one is just-in-time compilation (also called JIT) and the second one is ahead of time compilation (also called AOT).

AOT itself can be broken in two stages and the initial step calls for precompiling the assemblies.

The second step is automatic and in it, the Mono runtime loads any precompiled code that has been generated.

DEMOS

XAMARIN

BENEFITS

- Re-use .NET skills
- Leverage existing .NET technology
  - JSON.NET
  - OAUTH.NET
  - SignalR
- High code re-use 80+%
- Tailor UI/UX to target

NEGATIVES

- Need to still test and debug multiple targets even when MEAP only thing updated
- Multiple codebase for UI
- No sharing of UI
- Vendor risk and lock in although Xamarin is a strategic partner for MS
- May have to wait on new targets like Android

XAMARIN FORMS APPROACH



Shared UI Code

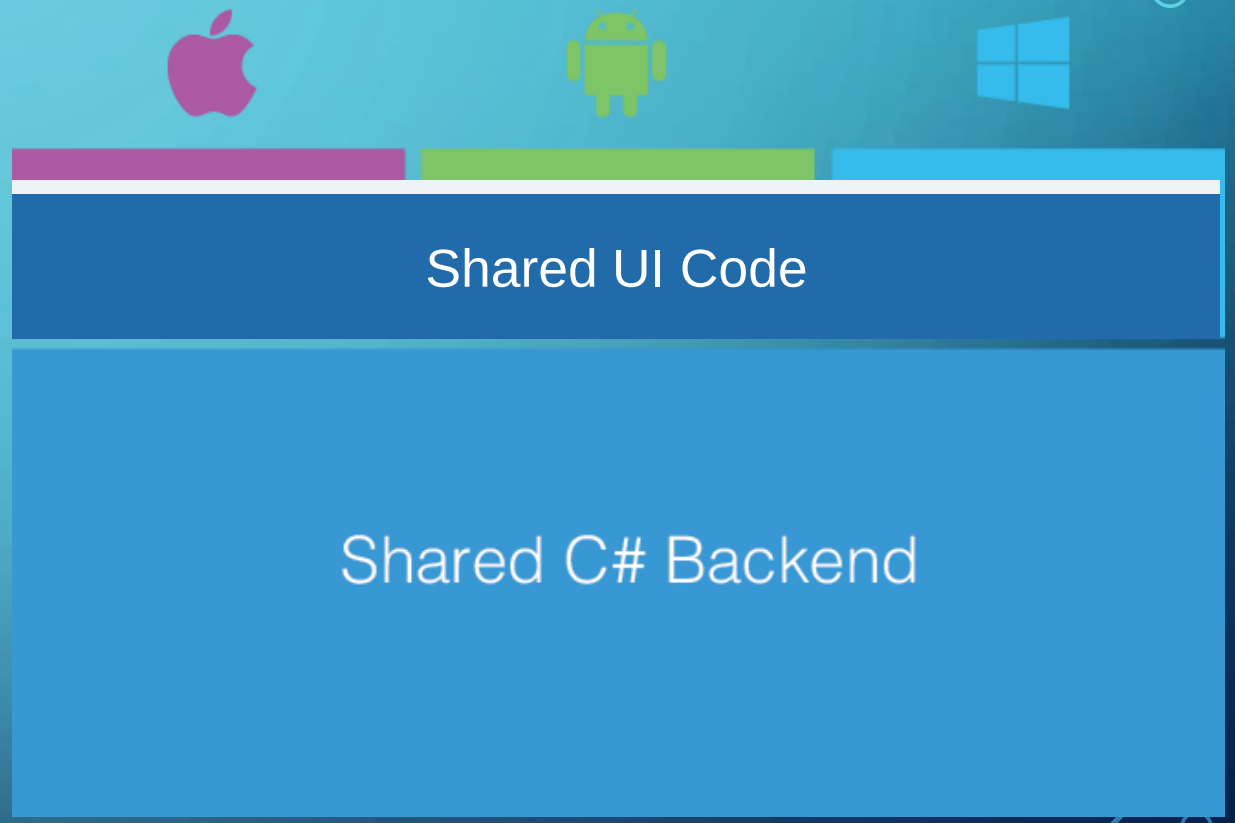
Shared C# Backend

XAMARIN + XAMARIN.FORMS

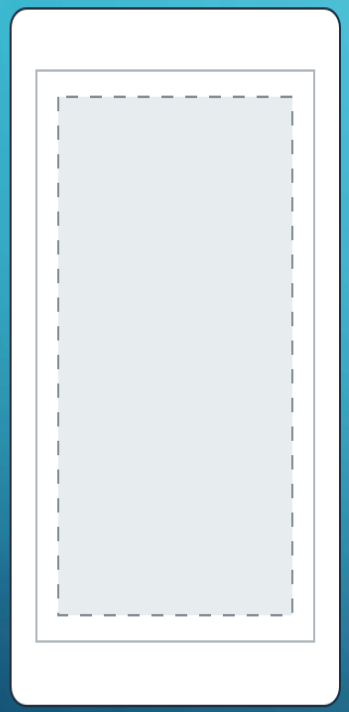
Quickly and easily build native user interfaces using shared code

Xamarin.Forms elements map to native controls and behaviors

Mix-and-match Xamarin.Forms with native APIs

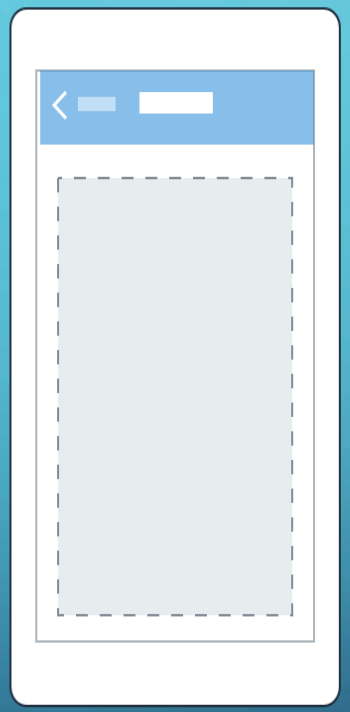


PAGES

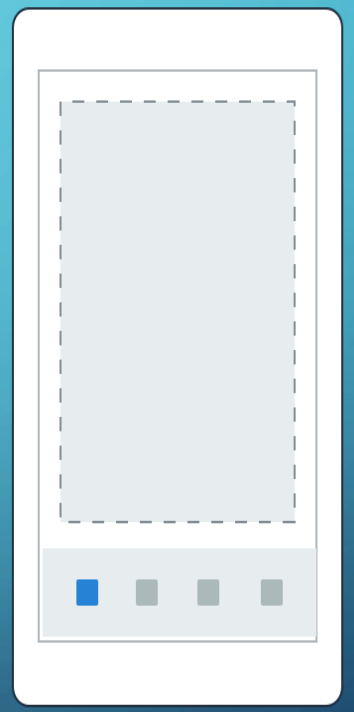


Content

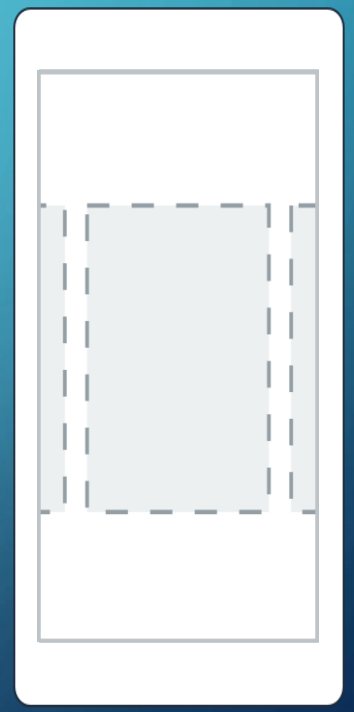
MasterDetail



Navigation

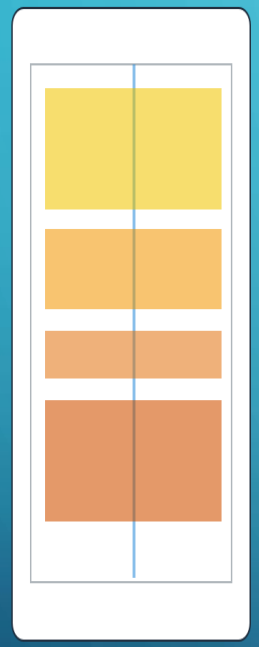


Tabbed

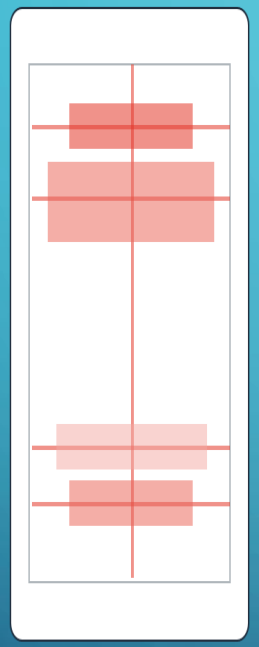


Carousel

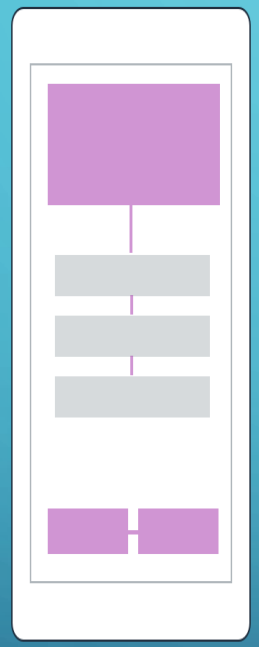
LAYOUTS



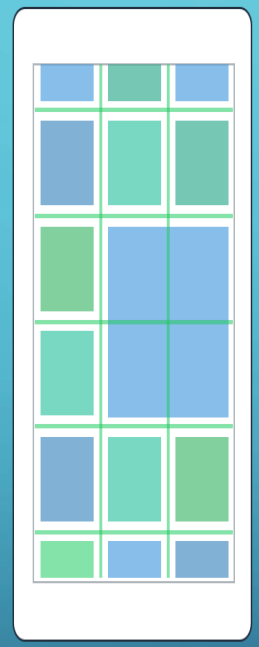
Stack



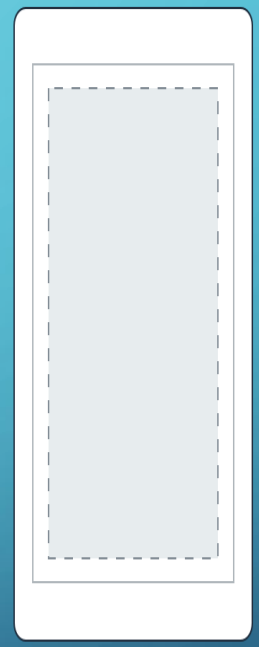
Absolute



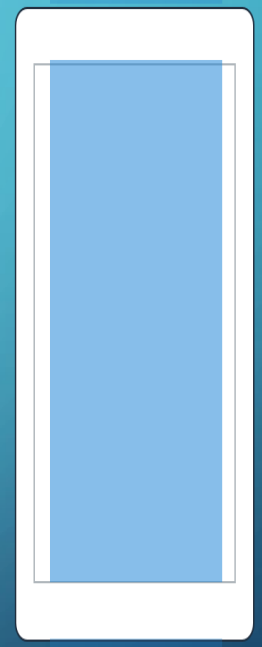
Relative



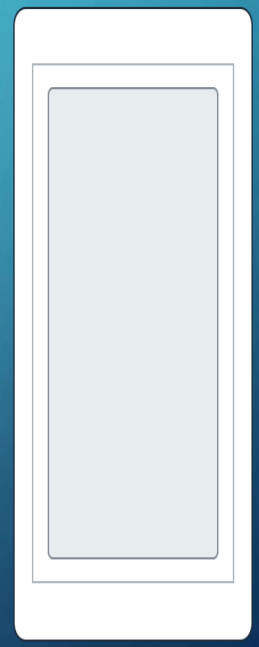
Grid



ContentView



ScrollView



Frame

.NET AS A MDAP

- Visual Studio as development environment
- Xamarin to reach non-MS platforms
- Xamarin.Forms to share UI on iOS, Android, and Windows Phone
- Azure Mobile Services
  - Security
  - Data Services
  - Notification
- TFS
- Xamarin Test Cloud

KEY TAKE AWAYS

- Strategies
  - Silo – Pure Native
  - Mobile Web – Web Apps
  - MDAP – Target a Dev Tool
- Cross Platform is not theory or option. It is the new reality.
- .NET is a viable platform using MS on MS tech and Xamarin to reach non-MS tech (iOS, Android, Mac, Linux, Google Glass, etc.)

More power

- **Custom controls and renderers**

→ **Customize and Modify the rendering**

- **Use XAML / Data Binding**

→ **Decoupled patterns**

→ **Maximize code reuse**

→ **Maximize unit testing**

- **Animations**

what else?

Xamarin Test Cloud

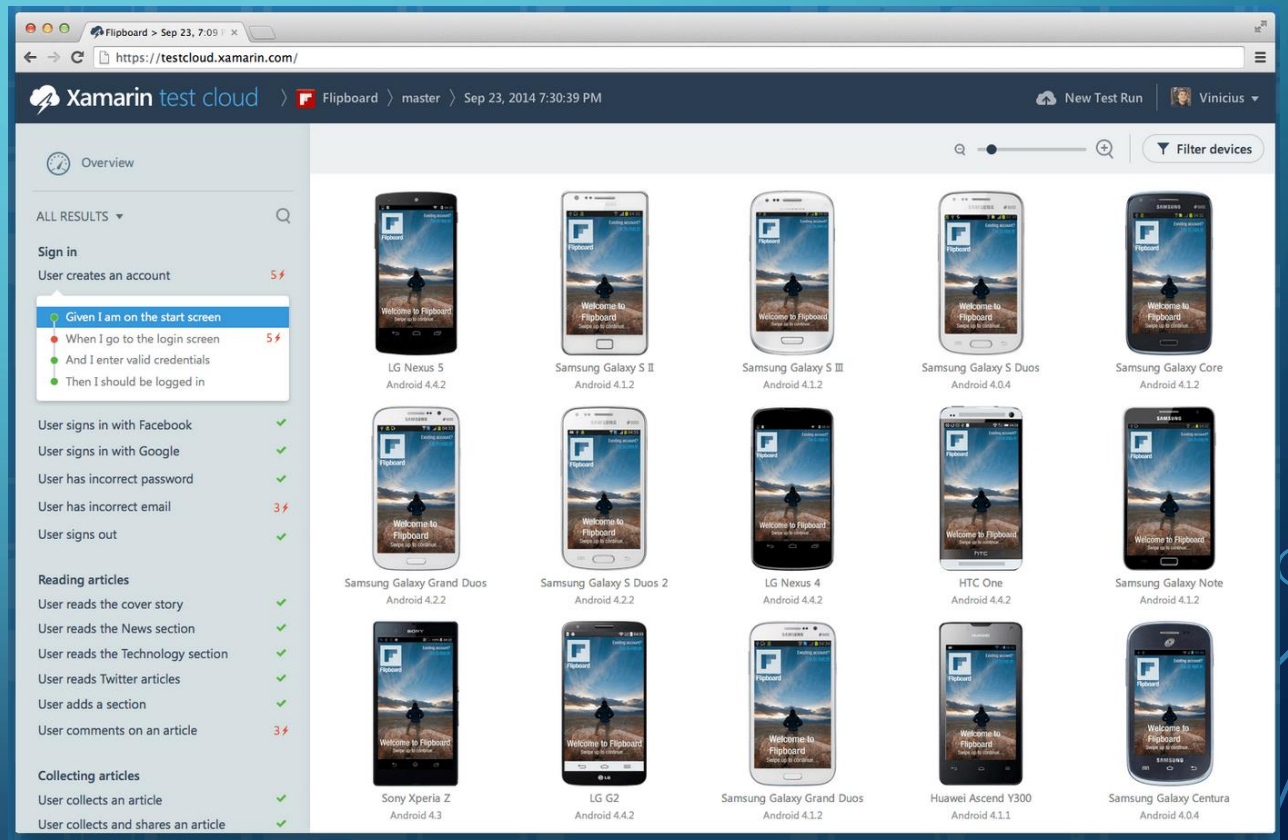
- Testing on hundreds of real devices
- Scripting the tests
- Getting feedback

→ Screenshots (scripted)

→ Crash information

→ Call stack

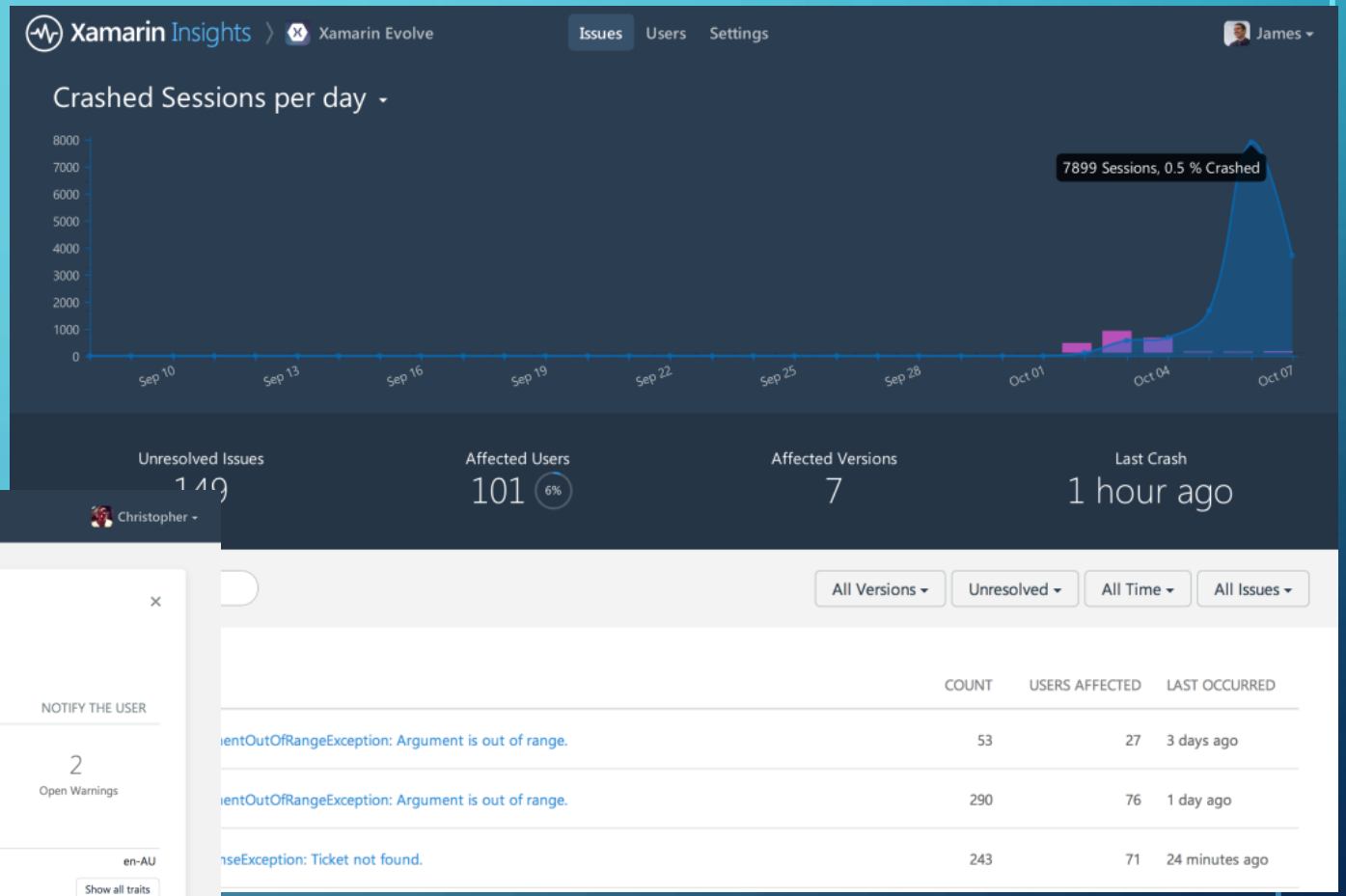
→ …

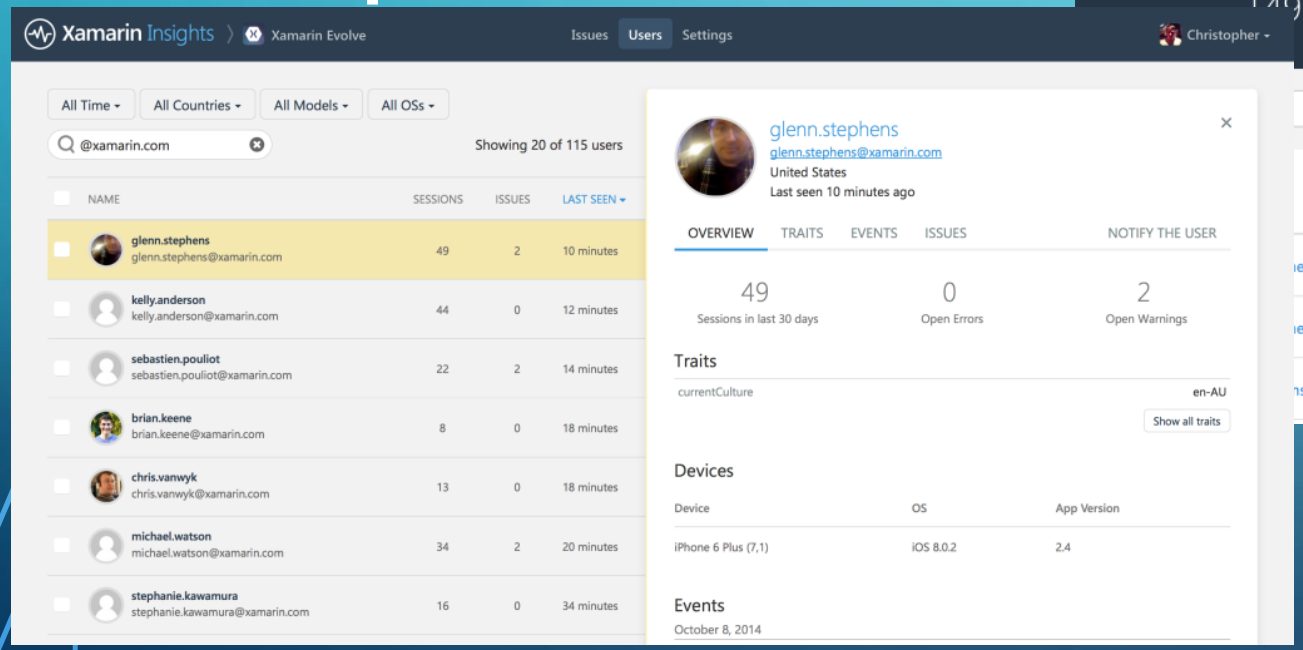


http://blog.xamarin.com/new-xamarin-test-cloud-features-2/

Xamarin Insights

- Analytics platform
- Cross platform





http://blog.xamarin.com/monitoring-your-apps-with-xamarin-insights/

Conclusion

- **Sharing code (“classic” Xamarin)**

→ **Full native UI experience**

- **Sharing UI (Xamarin Forms)**

→ **Some compromises**

→ **Full native controls**

→ **Ideal for prototyping / Iterative development**

- http://xamarin.com/

APPENDIX

Worldwide Smartphone Sales to End Users by Operating System in 2Q16 (Thousands of Units)

Operating System	2Q16 Units	2Q16 Market Share (%)	2Q15 Units	2Q15 Market Share (%)
Android	296,912.8	86.2	271,647.0	82.2
iOS	44,395.0	12.9	48,085.5	14.6
Windows	1,971.0	0.6	8,198.2	2.5
Blackberry	400.4	0.1	1,153.2	0.3
Others	680.6	0.2	1,229.0	0.4
Total	**344,359.7**	**100.0**	**330,312.9**	**100.0**

Source: Gartner (August 2016)

- http://www.gartner.com/newsroom/id/3415117