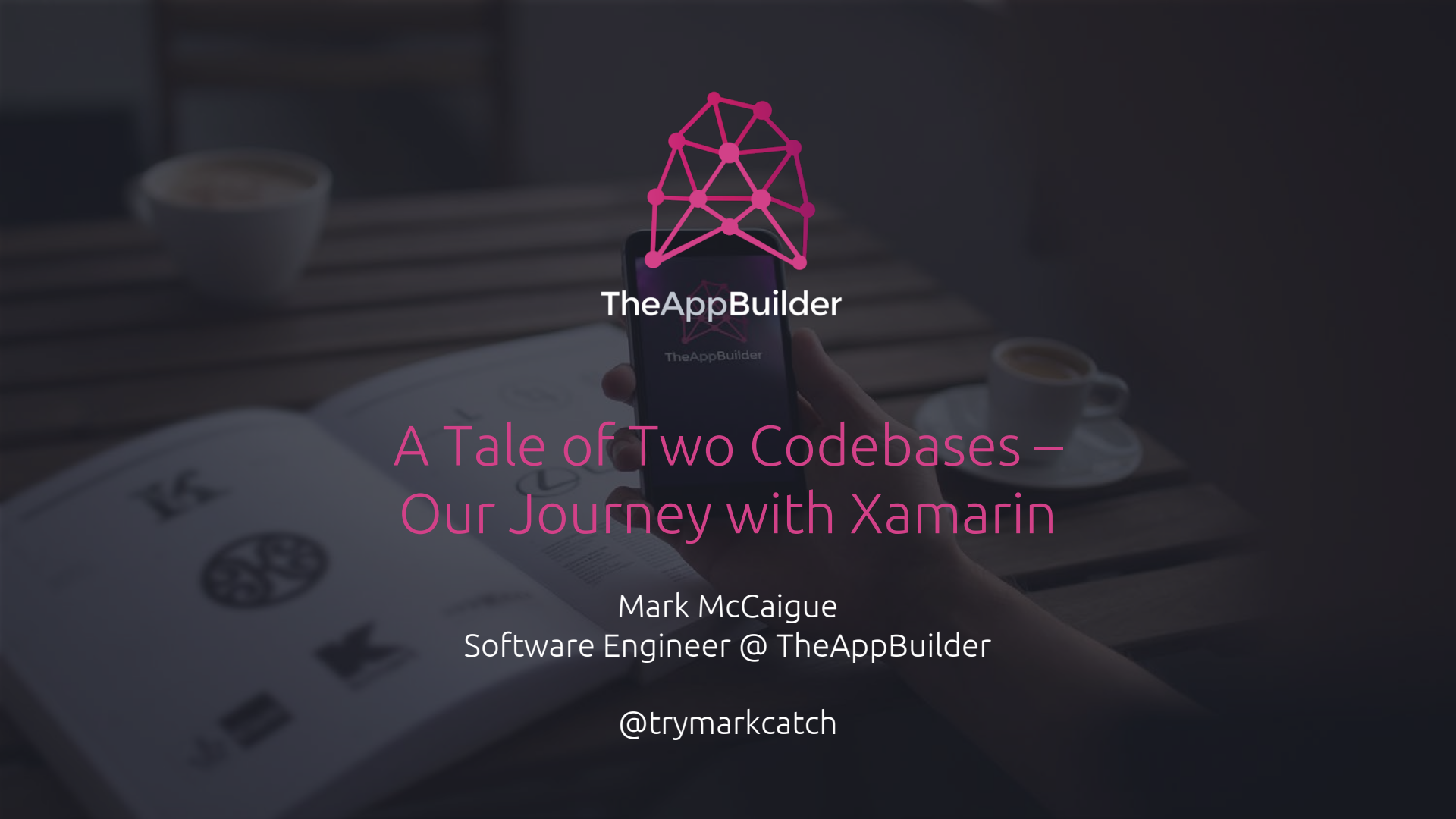


A Tale of Two Codebases – Our Journey with Xamarin

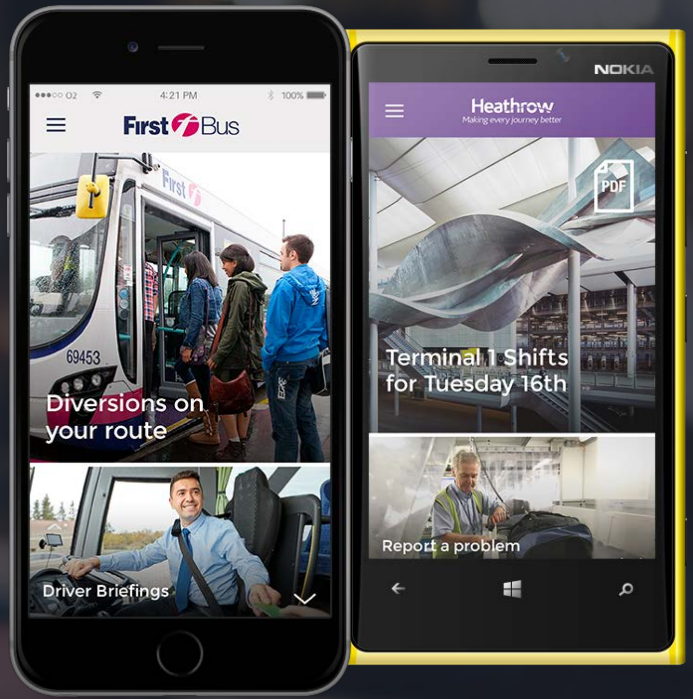
Mark McCaigue
Software Engineer @ TheAppBuilder

@trymarkcatch

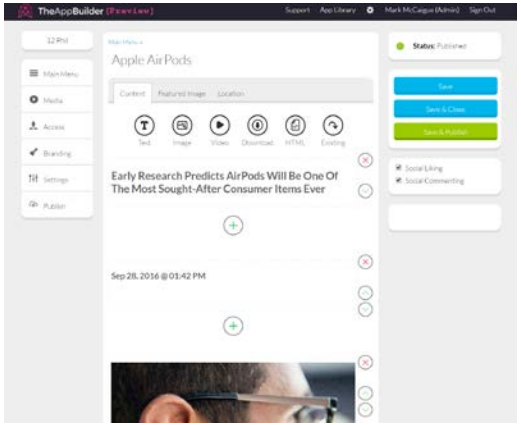
Who are TheAppBuilder?

TheAppBuilder: a Mobile Application Platform

- Publish your content into a native application
- Reach people on the move in real time
- 2-way engagement with comments and likes

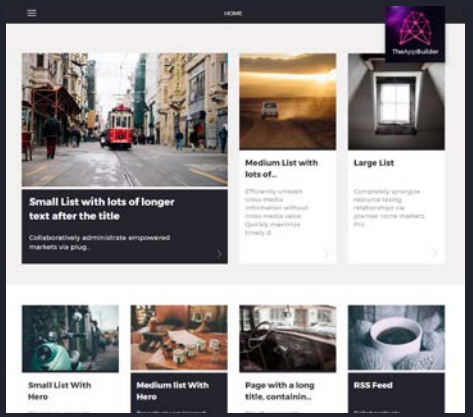


TheAppBuilder Products



Content Management System

Native iOS and Android clients



Web Client

Considering An Update

Native Development

Uses Native platforms and tools

Java on Android
Objective-C or Swift on iOS

One codebase per platform

Native UI

Hybrid Development

Uses Web Technologies and bridges to native functionality

HTML, JavaScript, and CSS to build UI and logic

Code Sharing

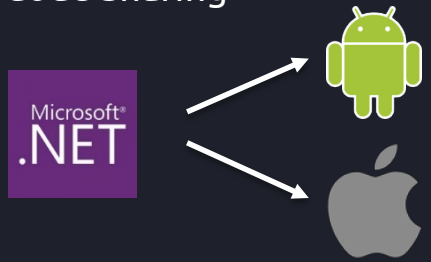
Xamarin

Written in C# with .NET APIs

Creates native apps from a shared C# codebase

Native UI

Code Sharing



Choosing an Architecture

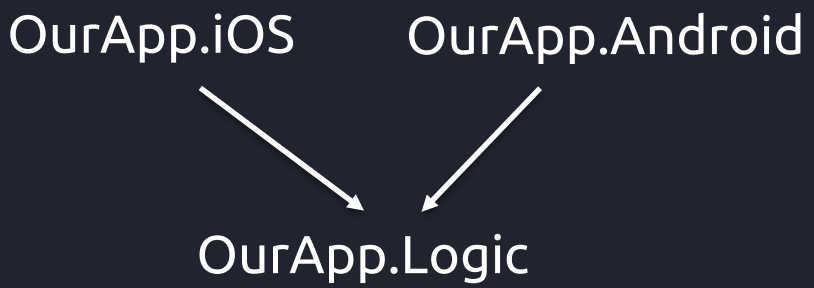
Code Sharing	Look and Feel	Performance
Team Skills	APIs & Features	Ecosystem

Code Sharing

- One shared code base
- Same development language and tools
- Write code for new features once
- Fix bugs once
- Common Architecture and Lexicon
  - Common Language leads to common patterns

Architecting Shared Code

- Shared code cannot depend on platform specific interfaces e.g. ViewControllers or Activities
- We can implement logic in our own C# objects and expose an interface we control
- Each platform builds against this shared interface



MVVM – Model View ViewModel

- Models represent your domain objects
- ViewModels implement application logic, and use standard .NET constructs for data and actions
  - No dependencies on UI layer
- Views ‘bind’ declaratively to ViewModels
- Models and ViewModels in shared code, Views can be in platform specific code, e.g. UIButton

Building Abstractions

or, how to call platform code from a shared location

Dependency Injection

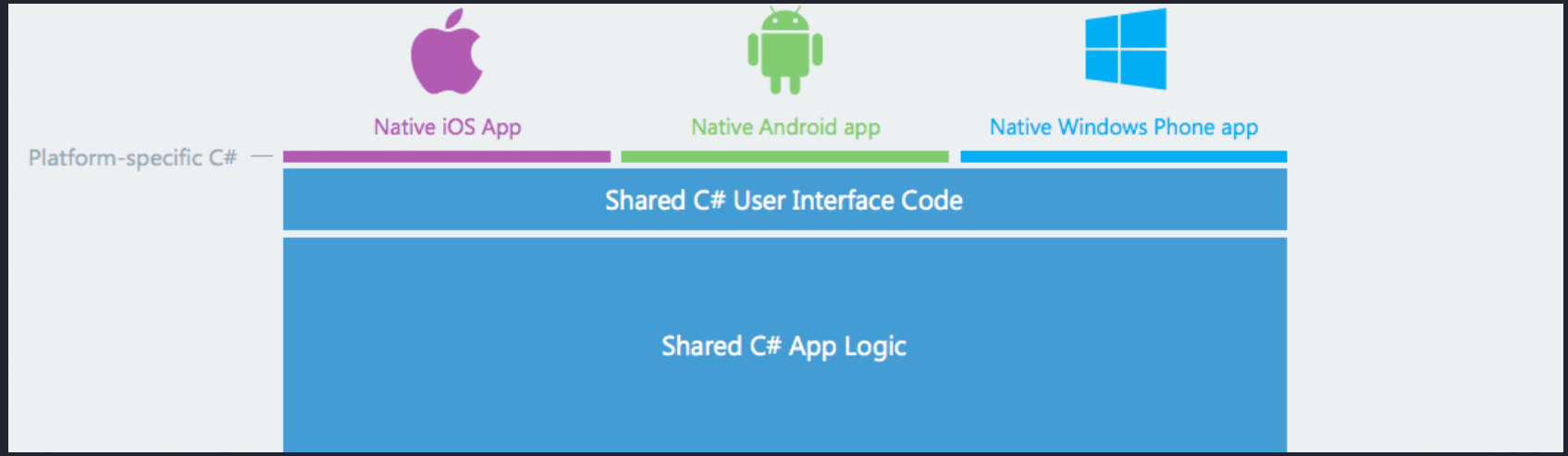
```
public interface IWrapPlatformSpecificCode
{
        void PerformAction();
}
```

```
public SharedType(IWrapPlatformSpecificCode service)
{
        service.PerformAction();
}
```

```
public class AndroidSpecific : IWrapPlatformSpecificCode
{
        public void PerformAction()
        {
                // Access specific Android API
        }
}
```

```
public class IosSpecific : IWrapPlatformSpecificCode
{
        public void PerformAction()
        {
                // Access specific iOS API
        }
}
```

The Xamarin Tech Stack




```
(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions;
public override bool FinishedLaunching (UIApplication app, NSDictionary options)

@Override
protected void onCreate(Bundle bundle)


protected override void OnCreate(Bundle bundle)
```

The Bits and Bytes

- iOS

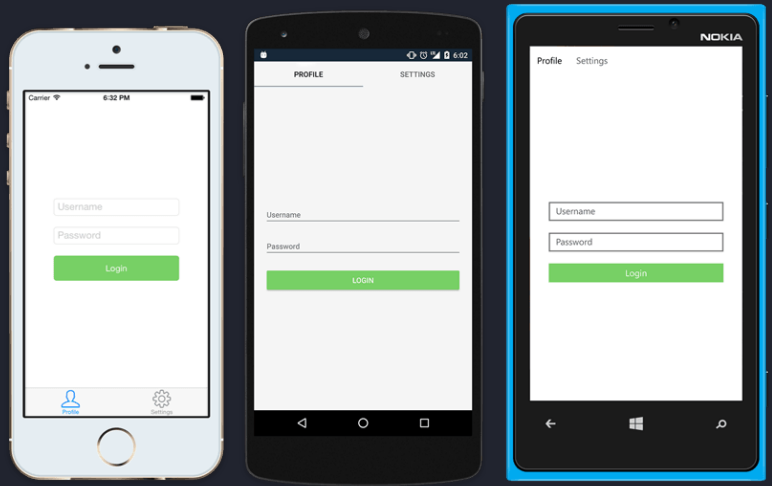
  C# is compiled Ahead-of Time to ARM assembly language
  Produces an IPA file, just like XCode

- Android

  C# is compiled to Intermediate Language and shipped with the Mono runtime, which performs JIT compilation
  Produces an APK file, just like Android Studio

Xamarin Forms

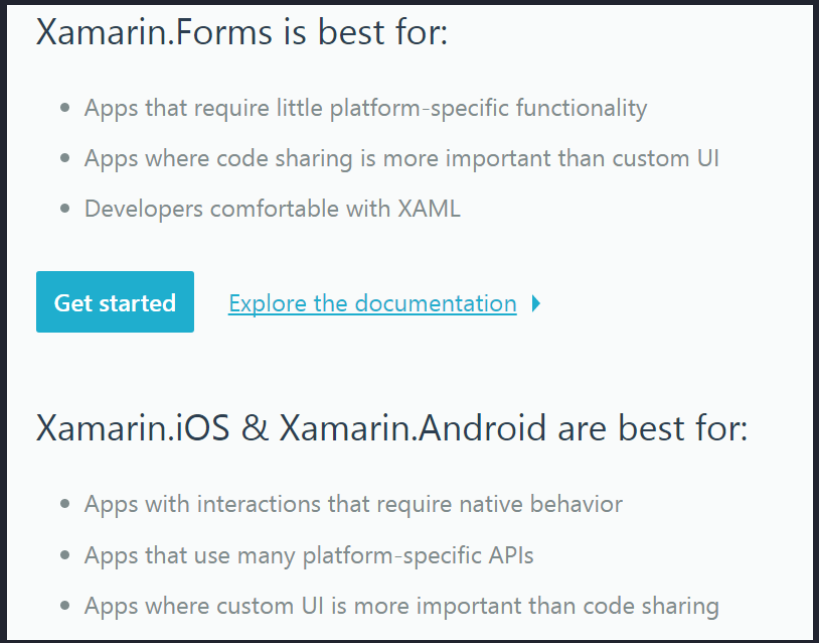
- A cross-platform UI framework
- Allows a UI to be built once against the Forms API and re-used cross platform
- At runtime, the views are translated to Native UI elements on each platform
- Implementation of Views, Layout Engine, and Navigation APIs
- Still possible to access Native UI elements and their APIs in platform specific code



The Reality Is

or, challenges, and trade-offs

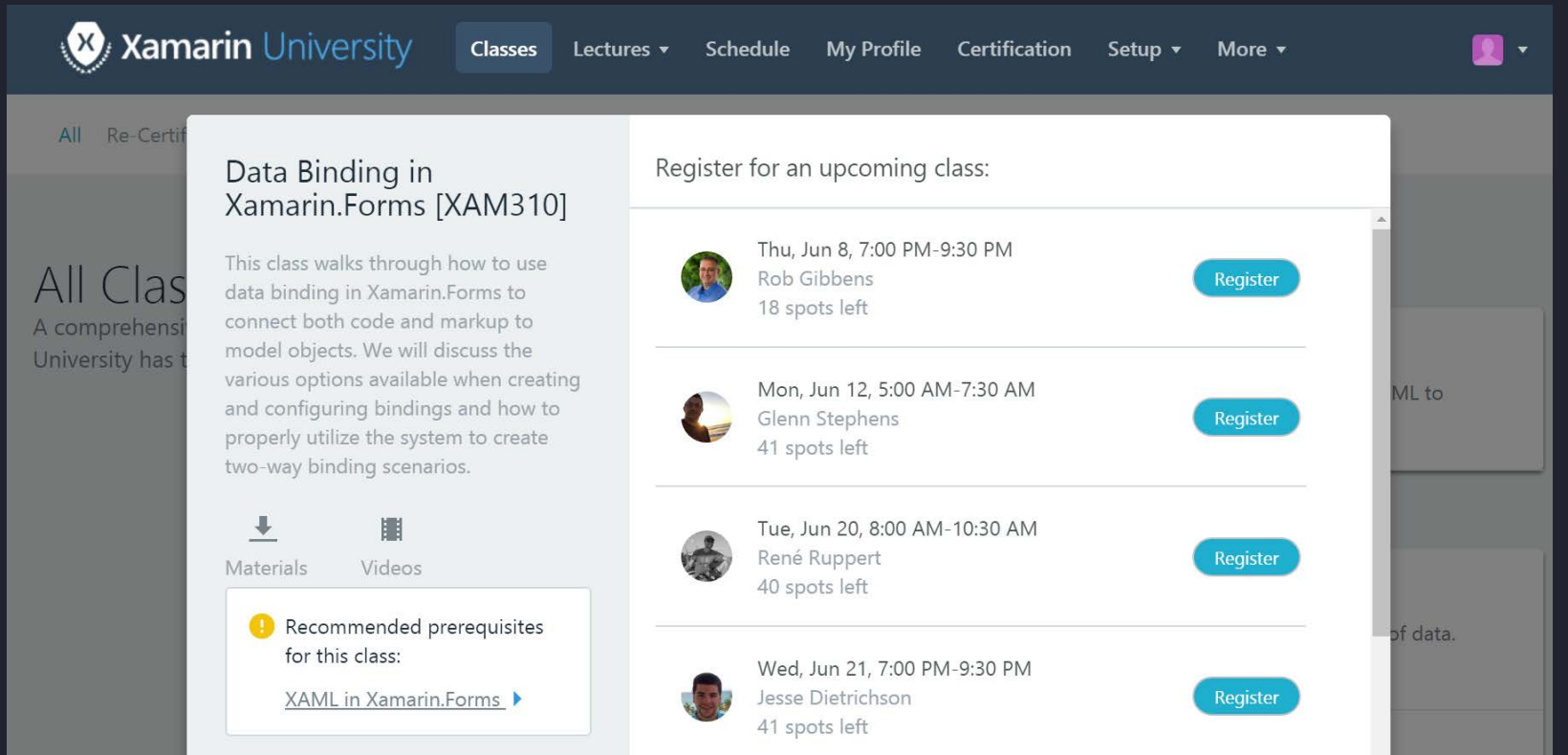
- Additional Dependencies
- Additional Build Tools
- Xamarin Forms
  - New APIs
  - Cross-platform Abstractions
  - Performance
  - Public Roadmap
- Balance against increased productivity and maintainability of shared code



Training and Skill Reuse

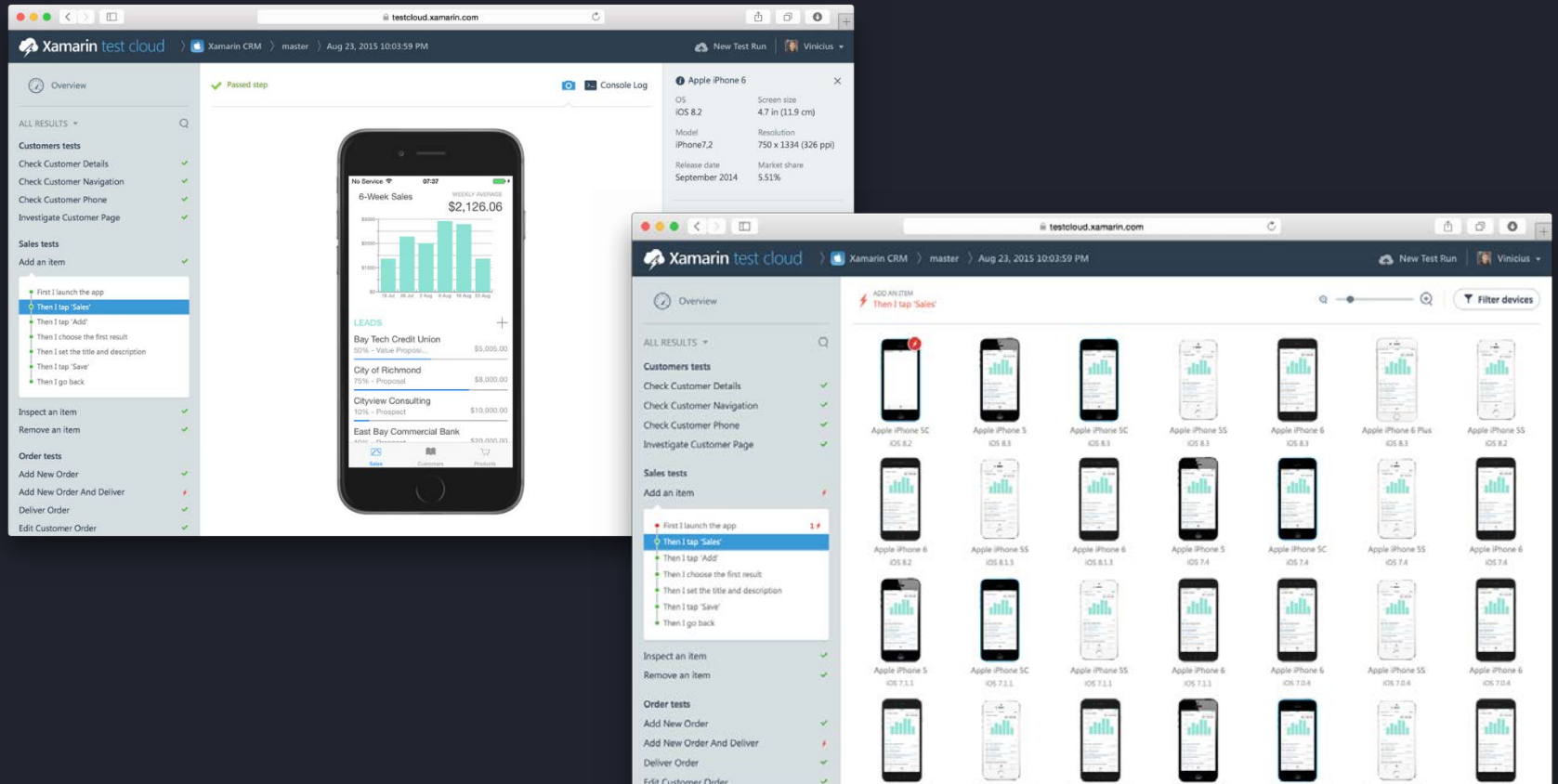
- The same C# language, the same .NET APIs
  - In our team, server side .NET developers
- MVVM and XAML in Xamarin.Forms are familiar for WPF or Silverlight developers
- Xamarin.iOS and Xamarin.Android use native APIs, can also be accessed via Xamarin.Forms
- Shared code architecture naturally encourages a front-end / back-end split for platform specific and reusable code respectively

Xamarin University



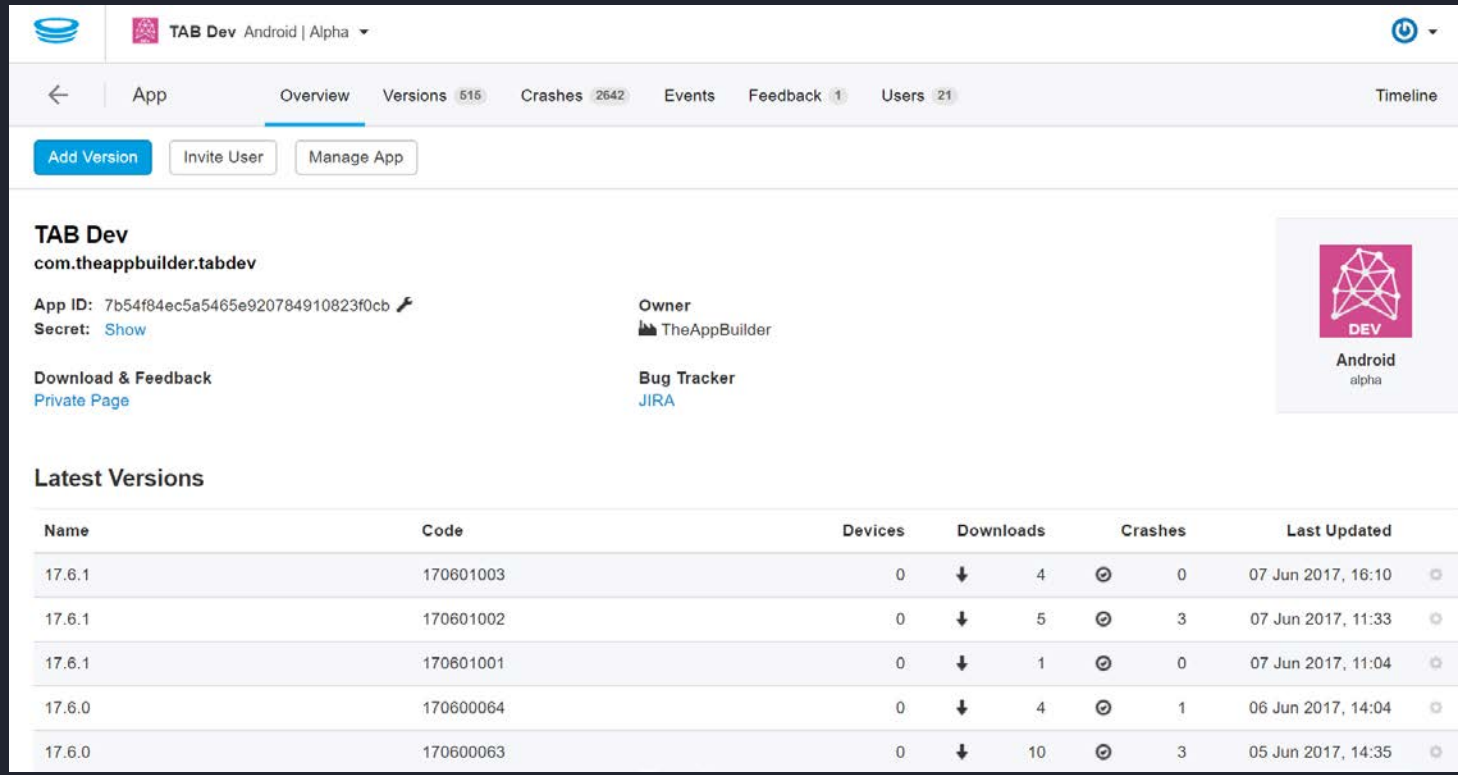
Testing – The Device Lab

Xamarin TestCloud



Distribution and Monitoring

- HockeyApp
- Google Play
- iOS App Store



Thanks for listening!

Any questions?

TheAppBuilder