

TheAppBuilder

A Tale of Two Codebases – Our Journey with Xamarin

Mark McCaigue
Software Engineer @ TheAppBuilder

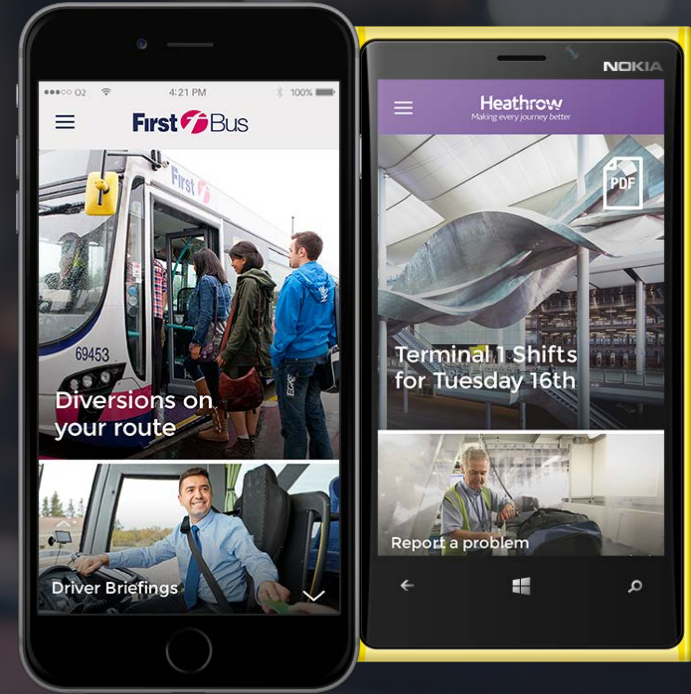
@trymarkcatch

Who are TheAppBuilder?

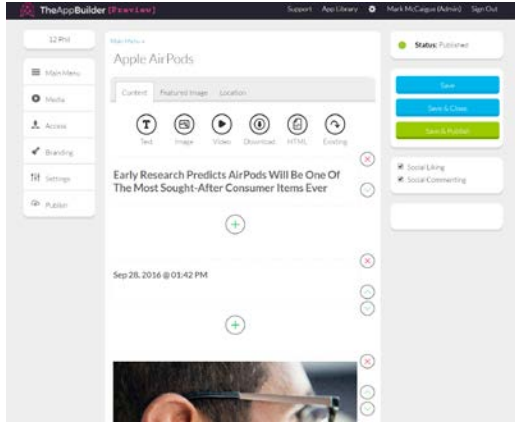


TheAppBuilder: a Mobile Application Platform

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



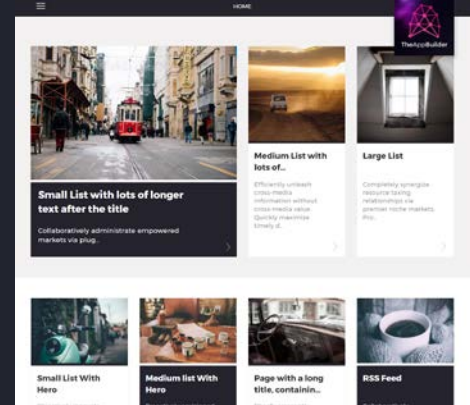
TheAppBuilder Products



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

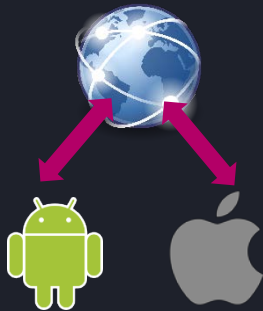


Hybrid Development

Uses Web Technologies and bridges to native functionality

HTML, JavaScript, and CSS to build UI and logic

Code Sharing



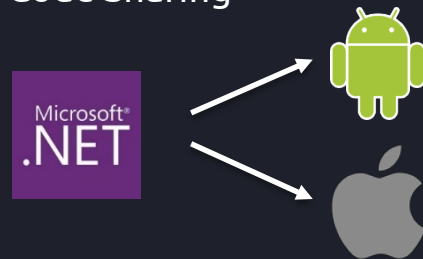
Xamarin

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

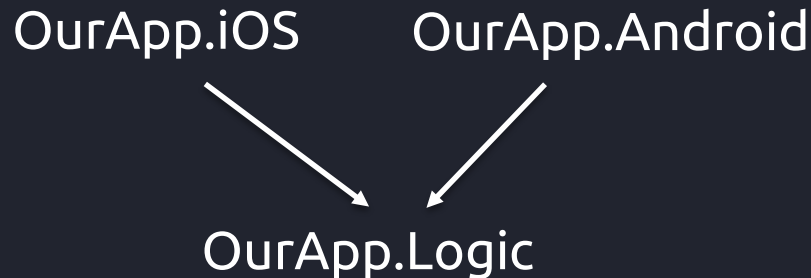
Ecosystem

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

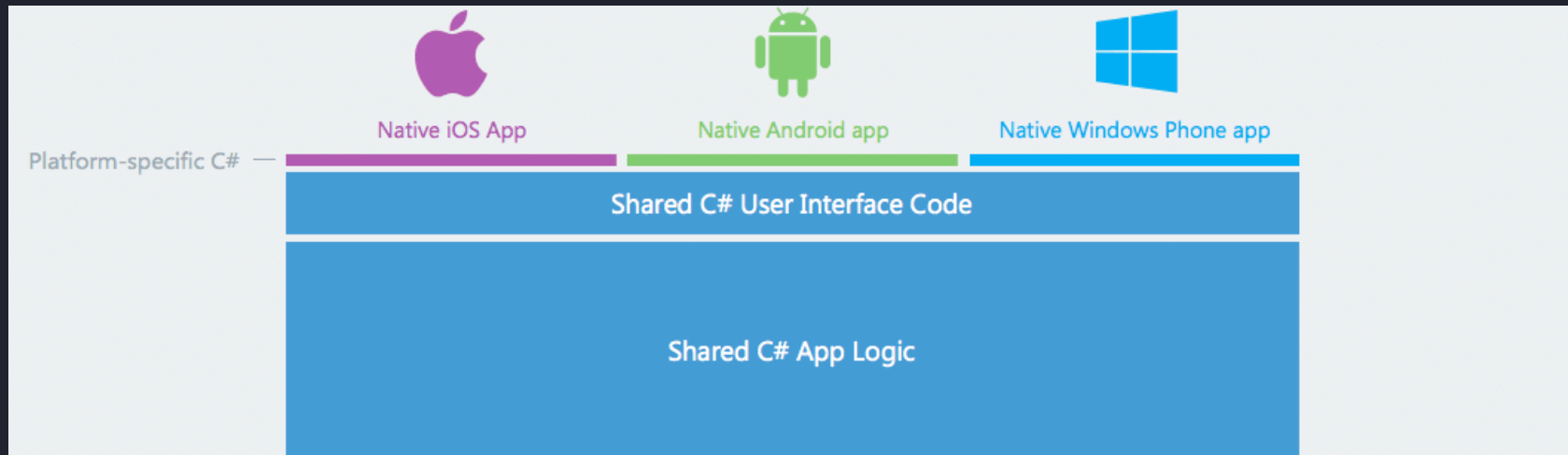
```
public interface IWrapPlatformSpecificCode
{
    void PerformAction();
}
```

```
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

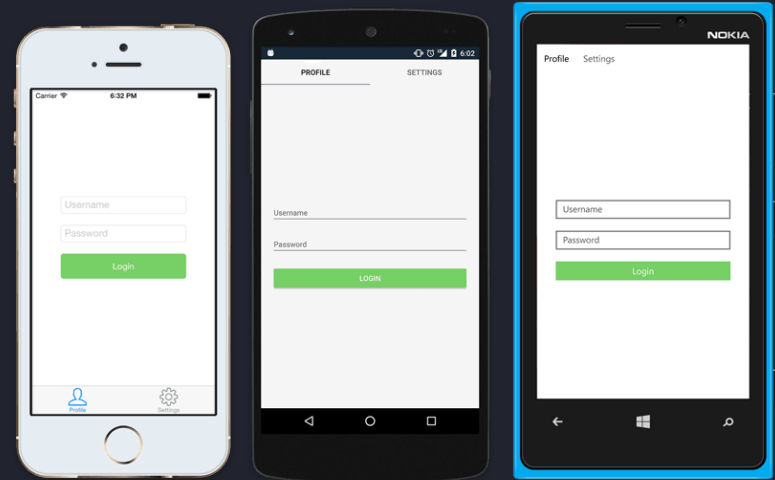
C# is compiled Ahead-of Time to ARM assembly language
Produces an IPA file, just like XCode

- 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

Xamarin.Forms is best for:

- Apps that require little platform-specific functionality
- Apps where code sharing is more important than custom UI
- Developers comfortable with XAML

[Get started](#)

[Explore the documentation](#) ▶

Xamarin.iOS & Xamarin.Android are best for:

- Apps with interactions that require native behavior
- Apps that use many platform-specific APIs
- Apps where custom UI is more important than code sharing

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



Xamarin University

Classes

Lectures ▾

Schedule

My Profile

Certification

Setup ▾

More ▾



All Re-Certif

Data Binding in Xamarin.Forms [XAM310]

This class walks through how to use data binding in Xamarin.Forms to connect both code and markup to model objects. We will discuss the various options available when creating and configuring bindings and how to properly utilize the system to create two-way binding scenarios.



Materials



Videos



Recommended prerequisites for this class:

[XAML in Xamarin.Forms](#) ▶

Register for an upcoming class:



Thu, Jun 8, 7:00 PM-9:30 PM
Rob Gibbens
18 spots left

Register



Mon, Jun 12, 5:00 AM-7:30 AM
Glenn Stephens
41 spots left

Register



Tue, Jun 20, 8:00 AM-10:30 AM
René Ruppert
40 spots left

Register



Wed, Jun 21, 7:00 PM-9:30 PM
Jesse Dietrichson
41 spots left

Register

Testing – The Device Lab



Xamarin TestCloud

Xamarin test cloud | Xamarin CRM | master | Aug 23, 2015 10:03:59 PM

Overview | **Passed step** | Console Log

Apple iPhone 6

- OS: iOS 8.2
- Screen size: 4.7 in (11.9 cm)
- Model: iPhone7,2
- Resolution: 750 x 1334 (326 ppi)
- Release date: September 2014
- Market share: 5.51%

Test Steps:

- First I launch the app
- Then I tap 'Sales'**
- Then I tap 'Add'
- Then I choose the first result
- Then I set the title and description
- Then I tap 'Save'
- Then I go back

App Interface:

6-Week Sales: \$2,126.06

LEADS:

- Bay Tech Credit Union: 50% - Value Proposition - \$5,005.00
- City of Richmond: 75% - Proposal - \$8,000.00
- Cityview Consulting: 10% - Prospect - \$10,000.00
- East Bay Commercial Bank: 40% - Prospect - \$20,000.00

Xamarin test cloud | Xamarin CRM | master | Aug 23, 2015 10:03:59 PM

Overview | **ADD AN ITEM** | Then I tap 'Sales'

Filter devices

Available Devices:

- Apple iPhone 5C (iOS 8.2)
- Apple iPhone 5 (iOS 8.1.3)
- Apple iPhone 5C (iOS 8.1)
- Apple iPhone 5S (iOS 8.1)
- Apple iPhone 6 (iOS 8.1)
- Apple iPhone 6 Plus (iOS 8.1)
- Apple iPhone 5S (iOS 8.2)
- Apple iPhone 6 (iOS 8.2)
- Apple iPhone 5S (iOS 8.1.3)
- Apple iPhone 6 (iOS 8.1.3)
- Apple iPhone 5 (iOS 7.1)
- Apple iPhone 5C (iOS 7.1)
- Apple iPhone 5S (iOS 7.1.1)
- Apple iPhone 6 (iOS 7.1.1)
- Apple iPhone 6 (iOS 7.0.4)
- Apple iPhone 5S (iOS 7.0.4)
- Apple iPhone 6 (iOS 7.0.4)

Distribution and Monitoring

- HockeyApp
- Google Play
- iOS App Store

The screenshot shows the HockeyApp dashboard for an application named 'TAB Dev Android | Alpha'. The interface includes a top navigation bar with tabs for Overview, Versions (515), Crashes (2642), Events, Feedback (1), and Users (21). Below the navigation bar are buttons for 'Add Version', 'Invite User', and 'Manage App'. The main content area displays the app's name, bundle ID, and owner information. A 'Latest Versions' table lists the most recent builds with their names, codes, device counts, download statistics, crash reports, and update times.

TAB Dev
com.theappbuilder.tabdev

App ID: 7b54f84ec5a5465e920784910823f0cb
Secret: [Show](#)

Owner
TheAppBuilder

Bug Tracker
[JIRA](#)

Download & Feedback
[Private Page](#)

Latest Versions

Name	Code	Devices	Downloads	Crashes	Last Updated
17.6.1	170601003	0	↓ 4	0	07 Jun 2017, 16:10
17.6.1	170601002	0	↓ 5	3	07 Jun 2017, 11:33
17.6.1	170601001	0	↓ 1	0	07 Jun 2017, 11:04
17.6.0	170600064	0	↓ 4	1	06 Jun 2017, 14:04
17.6.0	170600063	0	↓ 10	3	05 Jun 2017, 14:35

Thanks for listening!

Any questions?



TheAppBuilder