IPHONE, ANDROID & WP7

Is .NET the platform to rule them all?

If you ask me 2010 was the year of the mobile app. Everyone was talking about them, from consumers around the lunch table, to CEOs in the boardrooms. It was like 1999 all over again; only this time it was not about building websites, but to have some kind of offering in one of the major mobile app stores.

By Jonas Follesø

According to the analysts at IDG the huge interest in mobile apps is not just a fad. They expect download numbers to grow from 10.9 billion in 2010 to 76.9 billion in 2014. As a result app revenue will surpass $35 billion the same year. In short, mobile is going to play an important role for software developers over the next few years.

As a .NET developer you are faced with some tough decisions. Which platform do you bet on? How do you get your skills up-to-date quickly enough? Do you suck it up and learn CocoaTouch and Objective-C to target the popular Apple iOS devices? Should you learn Java and go for the Android? Or should you skip two of the most popular mobile platforms and stick to C# and develop for the newly released Windows Phone 7?

Businesses face some of the same decisions, as they need to decide where to focus their resources and efforts. Should they target all the major platforms, or focus only on iOS devices? What will be the cost of maintaining these applications over time? A quick search for "DnB NOR" (Norway's largest financial services group) in the iTunes App Store reveals that they already have five apps available. This is for iOS devices alone. In addition, two of the applications are also available in Android Market. With seven apps published already one can start to imagine what the cost and technical challenges of maintaining them over time will be.

When choosing which technology to use for implementing mobile apps, the popular choices have traditionally been either web-based apps

written in HTML, CSS & JavaScript, or alternatively fully native apps tailored to a specific platform. Both alternatives have their pros and cons. Web-based applications might be quicker to build and easier to make cross-platform, but the drawback is that you do not have full access to the phone specific APIs. And it is harder to provide the same high fidelity user experience as a native app offers. With native apps, the challenge might be a steep learning curve and having to re-implement the same app across different platforms.

With the release of MonoTouch at the end of 2009 Novell introduced a new way to build apps for iOS devices. Against all odds the programmers at Novell had found a way to bring Mono, the open source implementation of .NET, to the iPhone without requiring unlocking or jail-breaking the device. The MonoTouch framework enables developers to build native iOS apps using the C# programming language using many of the core .NET libraries. MonoTouch is designed to conform to the strict requirements set by Apple for App Store eligibility. MonoTouch allows .NET developers to take advantage of their existing skills, reuse existing code and libraries, making the transition to the iOS platform easier.

Instead of porting Windows Forms or WPF to the iPhone, MonoTouch provides a thin .NET layer on top of the native CocoaTouch APIs giving your applications a fully native look and feel, making them indistinguishable from the traditional iPhone applications. And where it makes sense, MonoTouch will provide standard .NET APIs instead of the CocoaTouch counterparts. For example, you use standard ADO.NET to talk to SQLite databases on the device, and XML can be processed using the familiar System.Xml.Linq API. MonoTouch apps are developed using the MonoDevelop IDE, an experience very similar to that of Visual Studio.

Managed runtimes like the CLR (Common Language Runtime) and JVM (Java Virtual Machine) typically use just-in-time (JIT) compilation, where byte code is translated to machine code at runtime. Apple forbids the inclusion of mechanisms that enable execution of arbitrary third party code, effectively blocking the use of non-standard runtimes, interpreters and virtual machines. The developers at Novell were able to overcome this obstacle by using an alternative ahead-of-time (AOT) compilation model where .NET Common Intermediate Language (CIL) is compiled directly to native code. AOT compiles the MonoTouch applications to native executables that do not rely on a virtual machine to run.

With the success of MonoTouch, Novell decided to take a similar approach to bring .NET to the Android platform. MonoDroid provides .NET bindings against the Android API, as well as a .NET-to-Java interop mechanism. Android is a more open platform than iOS and allows for JIT compilation of code. The Mono runtime will run side-by-side with the Dalvik runtime, providing direct access to many of the core Linux OS APIs through the standard .NET libraries like System, System.IO and System.Net. Just as with MonoTouch, a MonoDroid app will use the native UI toolkit, giving the user the same great user experience as with any other native app. MonoDroid integrates with Visual Studio making the experience of writing apps for Android even more familiar.

Both MonoTouch and MonoDroid expose the same subset of the .NET

core API, which is a subset based on the Silverlight API. This is the same subset Microsoft chose to use for their newly released Windows Phone 7. This means that there now exists a core set of APIs that are the same across the three major smartphone platforms. By carefully crafting an application in such a way as to separate business logic from user interface code, it is possible to reuse code and libraries across all platforms. I expect patterns to emerge over time that describe how to best structure your code for maximum reuse across the different mobile platforms.

For many developers mobile apps have made it fun to program again. Within weeks you can go from an idea to an app that can be downloaded by millions of users through one of these app stores. Who knows, something that starts of as a hobby project might turn out to be the next Angry Birds. No matter if you do app development as a hobby or professionally there is nothing more gratifying as seeing your first app being available in one of the app stores.

Happy app-ing everybody!

In this article I will introduce C# and .NET as an alternative for writing cross platform mobile applications.

Jonas Follesø is a Scientist and Senior Consultant at Bekk Consulting. Jonas has been writing software for the .NET platform since the v1.0 beta days. He is currently spending much of his time exploring different approaches to mobile development.