

Microsoft

Demo - SmartHotel 360 mobile apps

Last updated: 12/11/2017

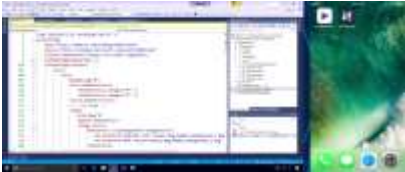

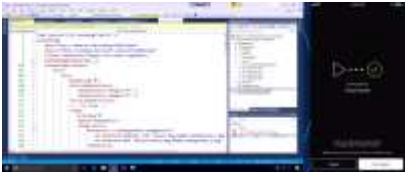
System requirements (setup)


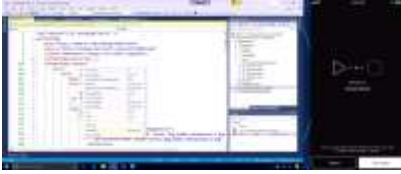
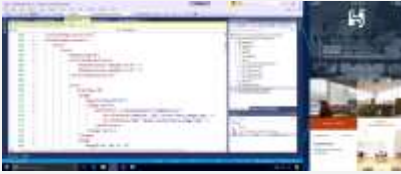
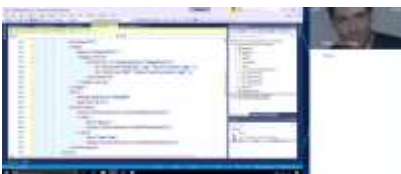
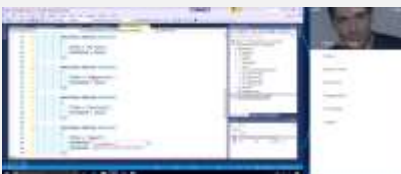
- PC running Windows 10 Fall Creators Update (16299)
 - **Visual Studio 2017 version 15.5** with the **Mobile Development with .NET** workload installed
 - Enable **Xamarin Live Player** (Tools → Options → Xamarin → Other → Enable Xamarin Live Player)
- Mac running macOS High Sierra (10.13)
 - **Visual Studio for Mac version 7.3** with **Android + Xamarin.Forms** and **iOS + Xamarin.Forms** installed
 - **Xcode 9.1** and command line tools
- Android phone
- iPhone

Demo Pre-Setup

- Keep the solution folders to show the code handy
- Keep number of source files and solutions open to show the integration
- Install Xamarin Live Player -> Test Flight version for iOS: <http://aka.ms/liveplayeralpha>


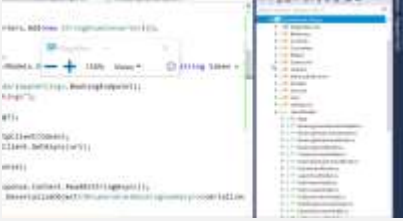
Demo 1 – Get started fast using Xamarin Live Player

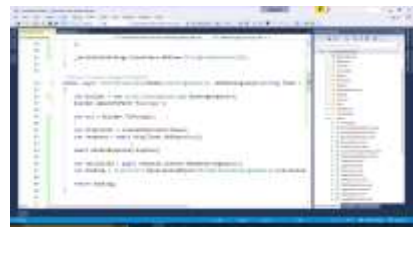
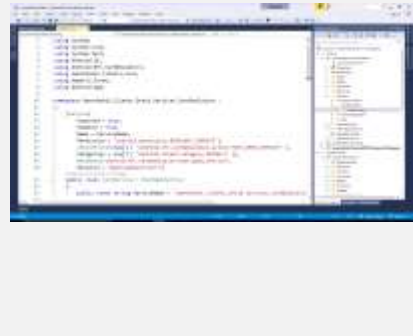
Screen	Narrative	Notes
	We now have hundreds of thousands of developers using Visual Studio and Xamarin to build their mobile apps and one of the themes of feedback we hear is that the initial setup process to get started can be quite daunting. This is why we improved the experience so you can get started within 5 minutes.	<p>Set up this demo by opening the SmartHotel.Clients.Live.sln solution and the following files in the IDE:</p> <ul style="list-style-type: none"> - MenuViewModel.cs - MenuView.xaml - HomeView.xaml - LoginViewModel.cs <ul style="list-style-type: none"> o Ensure a breakpoint is set on line 50 of the class (if (isValid)) - LoginView.xaml
	Let's jump into the SmartHotel360 app that I have set up here in Visual Studio 2017. You can see my iPhone on the right side and I'm going to be using the new Xamarin Live Player to directly execute and debug my app on the iPhone.	Run the Xamarin Live Player app on the iPhone.
	To connect Xamarin Live Player with Visual Studio, we'll go to Tools → Xamarin Live Player → Manage Devices... to pair it using the QR code.	Pair the Xamarin Live Player app with Visual Studio by hitting Pair Live Player in the app and scanning the QR code.
	Now all I have to do is select my Player in the deployment targets and hit F5 to start debugging.	Make sure the Live Player is selected as the deployment target and start debugging.

	<p>The app's code is now being interpreted in the Xamarin Live Player. The beauty of this is that nothing is compiled, but I can see the app experience and start interacting with it.</p> <p>Once the breakpoint hits, I have access to everything you would expect: all the locals, the Call Stack, etc.</p>	<p>Log into the app using any username and password, tap sign in, and the breakpoint hits.</p>
	<p>I could use the Xamarin Live Player this way, debugging with nothing more than Visual Studio and my device, but having to stop debugging to make changes every time limits my productivity.</p> <p>Let's go back to the LoginView and use the Live Run feature.</p>	<p>Select the LoginView.xaml tab, right-click anywhere in the code editor and select Live Run.</p>
	<p>Immediately you will see this single screen of the app in the Xamarin Live Player on my iPhone. This page looks good, so let's switch over to the HomeView. As you can see, the Xamarin Live Player updates the view instantly, without having to do anything.</p>	<p>Select the HomeView.xaml tab and notice the Xamarin Live Player switches the view.</p>
	<p>Let's take a look at the MenuView and show the real power of Live Run. Not only can I see the UI, but I can make adjustments and have the Xamarin Live Player immediately show me the result. For example, if I go to line 52 and change the Height to 15* instead of 25*, you immediately see the change on my iPhone. I didn't even have to save the file!</p>	<p>Select the MenuView.xaml tab and notice the Xamarin Live Player switches the view.</p> <p>Go to line 52 and change the <code><RowDefinition Height="25*" /></code> line of code to say <code><RowDefinition Height="15*" /></code>.</p>
	<p>What's great about Live Run, is that it's not only for the XAML user interface, but also the code-behind. Let's jump to the MenuViewModel and do just that. In here, let's jump to line 40 and change the text to "Book a hotel room". As you see, the app updates again, even when I make a change in my data-bound ViewModel.</p>	<p>Select the MenuViewModel.cs tab.</p> <p>Go to line 40 and change the <code>Title = "Book a room",</code> line of code to say <code>Title = "Book a hotel room",</code>.</p>

	<p>With Live Run I never have to stop my development, it just automatically redeploys the code whenever I make changes.</p> <p>Xamarin Live Player is also available in Visual Studio for Mac, and we're working to bring all the Live Run features you saw here to Visual Studio for Mac soon.</p>	
--	---	--

Demo 2 – Native app experiences with Visual Studio and Xamarin

Screen	Narrative	Notes
	Now I want to show you what the code for this app looks like, when fully built out into a solution.	Set up this demo by opening the SmartHotel.Clients.sln solution. Open the SmartHotel.Clients/Services/Booking/BookingService.cs class in the IDE.
	As you can see, I have my Android, iOS, and UWP client apps in this solution. I have shared code as a .NET Standard library, that goes across multiple projects. I have a Smart Door Android tablet app, that can detect NFC connections and open a physical door. There's also ASP.NET Core back-end services, and of course Unit Test.	Show each of the projects as you're talking through them.
	Jumping into the shared code, you can see how much efficiency we get here. Everything from Models, ViewModels, Service calls, Helper classes, to Views, it's all shared across. The UI is all written in XAML, and even that's shared across multiple apps with Xamarin.Forms.	<p>Expand the Shared Code project (SmartHotel.Clients) and show all that's in there.</p> <p>Expand the Views folder when talking about XAML.</p>

	<p>With Visual Studio and Xamarin, writing apps across all these platforms doesn't mean you have to work in multiple languages. As you can see here in the BookingService, everything is written in the C# you know and love. I use a HttpClient, async/await, and even Json.NET to deserialize my data.</p>	<p>Show the GetBookingsAsync() method on line 34 of BookingService.cs.1</p>
	<p>However, if I need to, I can always come into the specific platform projects to access 100% of the native APIs available on that platform. For example, opening the CardService, we can see calling native Android APIs to talk to the NFC chip on the device.</p> <p>As you can see, Visual Studio and Xamarin give you the power and flexibility to create fully native app experiences, all in C#.</p>	<p>Open the SmartHotel.Clients.Android/Services/CardEmulation/CardService.cs class in the IDE.</p> <p>Depending on time, this is where you can also show the apps running, highlighting the native look & feel, and performance.</p>