Mindtree
A Larsen & Toubro Group Company

Emerging Trends in **Multiplatform Application Development**

Welcome to possible

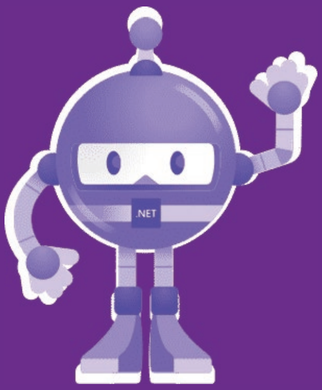
A Mindtree Whitepaper

Emerging trends in Multi-Platform App development are gaining popularity for two reasons. One is single code base and wider audience reach, and secondly, less development time and effort. They are also cost-effective and provide native look across different platforms. These trends are driving the entire software market, and it is the optimal time to consider investing in these frameworks. There are few frameworks which serve this purpose such as Flutter, React Native, .NET MAUI and Ionic. This article will cover the aspects of React Native and .NET MAUI.

React Native

In 2012, Facebook acquired Instagram and wanted to use React for their timelines, which was followed by open sourcing React project. Facebook started to develop mobile platform technology and introduced React Native for native look and feel. In order to extend support for Windows, Microsoft created an extension and generated UWP template, which uses WinRT inside React Native code.

.NET MAUI



Microsoft developed VS Code with Electronjs. They observed that electronjs requires a lot of space, since it bundles Chromium and nodejs (for communication with host OS). In the objective of reducing the space and increasing the efficiency, they created a small package called “WebWindow” which uses WebView2 (web control for binding HTMl, js, css) and ultimately resulted in better efficiency and lesser space.

The extension of this project along with migration of Xamarin Forms into a separate framework is.NET MAUI which is the Microsoft’s greatest focus area for the current year. It allows developers to run application on any platform. It supports C# and XAML for UI components.

Few developers feel difficult to code in XAML. As a solution to this problem, Microsoft has also provided MAUI with Blazor, where developers can write code using HTML, Javascript and CSS and render them as native application.

Key factors

1. Installation & Setup:

A React Native application is created with just two commands. Using the “expo-cli” npm package, we can scan the QR-code and deploy the build into any physical device. This makes development fairly faster. We can also use AVD (Android Virtual Device) configuration and run it as an Android emulator.

The setup for .NET MAUI needs the developer to carefully follow the documentation. Microsoft has developed a new package called “maui-check” which does all necessary installations like Android SDK, ios SDK , Android emulator (AVD configuration) and the latest dotnet version. We can also install the workloads separately. However, in some cases, where the workload doesn’t install properly, manually installation is advised. Microsoft has promised to add dependencies in the Visual Studio installation. The Visual Studio directly runs on Android emulator connected through AVD with just one click. There is an option to change target platform and run accordingly.

2. Documentation & Community Support:

React Native’s documentation is pretty well written which helps a developer build applications faster. Since the Javascript world is huge, the community support for React Native is also relatively high.

On the other hand, MAUI is still in a development stage and lacks important concepts. We need to refer Xamarin Forms for further development guidelines. Xamarin Forms documentation is well explained, with proper links, description for all APIs, and controls for developers to use.

3. Language & Learning Curve:

The React Native app can be developed using the most widely used programming language – Javascript and Typescript (extended version of javascript). Since it uses JSX Javascript Syntax Extension, which combines HTML elements with Javascript, it becomes difficult to understand the core concepts, which makes the learning curve steep. However, the learning curve of React Native is almost flat if the developer has hands-on experience.

.Net MAUI can be built in C# with a dotnet environment. UI elements are built using XAML (XML-based Application Markup Language). During compilation, this XAML code is converted to native code. The learning curve of XAML is steep. On the other hand, MAUI with BLAZOR rescues developers by providing the facility of running the BLAZOR application in MAUI. MAUI has BlazorWebView (A WebView control to run Blazor application) which hosts the BLAZOR application and provides native look.

4. IDE & Debugging tools

MAUI can be developed using Visual Studio IDE. It has debugging, hot reload and various diagnostic tools to support developers in debugging and error fixing scenarios. However, MAUI is supported only for Windows users, and there is a cost associated with the enterprise version of Visual Studio. Non-Windows users will not be able to explore .NET MAUI.

On the other hand, VS code becomes the first choice for React Native development. It has minimum in-built debugging feature, and there are a few custom debugging libraries, which can be used as primary UI tools.

5. Generated Project Structure & Memory

.Net MAUI code structure has only two folders;

1. Main App – All the common code resides here. It has a platform folder which contains platform-specific configuration files.

2. MainApp.WinUI – code specific to Windows resides here.

React Native also has a simple project structure. Platform-specific configuration and build requirements reside in separate folders. However, the size of the project is ultimately large than MAUI.

6. Architecture:

React Native – It has three core building blocks and follows MVVM pattern.

1. JS Interpreter – entire Javascript code is compiled and the bundle starts when user runs the application.

2. Bridge (Javascript Interface) – It is based on concept of "Shared Ownership" where communication between native module and UI components happen. Any user event that triggers native API or UI changes is given to the bridge to handle.

3. Yoga – Facebook's open source layout engine used to design flexible layout across different platforms. It calculates the layouts and their dimension for UI.

.NET MAUI – It contains two different folders; one is the main app where all the code resides and other folder is created using WinUI for Windows application development. WinUI uses WinRT Windows Runtime API to access all native controls. It uses Single project MSIX packaging extension, which creates support for debugging and working with WinUI project without using separate packaging code.

.NET MAUI with Blazor uses Microsoft.Edge.WebView2 embedded web control for binding web technologies like HTML, CSS and JS into native apps. During compilation, the XAML code is converted into Intermediate Language (native platform-specific code is generated) using XAMLC.

7. Third party tool integration & Handy Solutions

React Native is an open source with lot of third-party libraries. Integration & customization to any custom library is extremely easy. It has readily available solutions for most UI features. .Net MAUI is relatively new, and hence, third-party library support is very less.

8. Configuration to support Windows

Since React Native is an open source, Microsoft has created an extension to support the Windows platform. It requires a "reactive-native-Windows" NPM package, which creates a UWP project template inside the React project. UWP (Universal Windows Platforms – Framework to build any application on Windows) uses WinRT API for native control. We need to download the build tools required to run Windows app.

In .Net MAUI, during installation, "maui-check" checks for platform specific SDKs and workloads. There is no additional installation required. The project comes with Windows support with a separate folder for Windows specific codes.

9. Security

We have to jump through a few more hoops with React Native to achieve the levels of security that come with .NET as default. When we build a bank or financial app, due to data confidentiality, additional security layers are required.

.Net MAUI follows Microsoft security policies which adds basic security features. However, it is advised to configure additional security features based on the requirements.

10. Compilation

Net MAUI uses AOT Ahead of Time Complication .It compiles XAML into native platform-specific code, which reduces load time. XAML is converted into intermediate language (IL) using XAMLC. Blazor Application also supports AOT compilation. Hence, compile time errors can be rectified.

React Native uses a Javascript Interpreter, where instead of compiling the source code into native code, it takes the code to host the Javascript engine. It uses an abstraction layer called "react-native bridge" that invokes actual rendering. Hence, the application loads slower.

11. Performance

The performance of React Native is good with few drawbacks, which can be solved by performing small workarounds. Application load time is long due to the JS-Interpreter compilation process. There are issues with navigations, and support for 64-bit mode on Android is not available.

In case of .Net MAUI, it is too early to discuss the performance. MAUI can be used to build fast loading applications with a 64-bit mode support on Android. It also provides AppShell, which provides for boilerplate navigation components that align perfectly with Windows and the mobile environment.

12. Code reusability across different platforms

Code reusability features are similar in both technologies. We can create any common code and use them across your entire application. In React Native, we can create services and inject them in any component. MAUI has a dependency injection support at the start up level.

13. Ease of writing platform specific codes

In React Native,

1. First, we can import Platform Module and use its Platform.OS to identify the platform and render the logic/style based accordingly.

2. Second, we can use Platform.Select to render component specific files.

In .Net MAUI, we have

1. OnPlatform to execute actions based on target platform.

2. Idiom to change the layout and functionality based on target platform.

Apart from these, few other methods such as Device.Style, Device.BackgroundColor, Device.FlowDirections are available.

14. Deployment

Visual Studio has made the build process simple for developers. We can create deployable bundles such as ".exe" files for Windows, ".apk" files for Android apps with one click. It is possible to directly deploy the Windows app to local machine by using the "deploy" option on WinUI project. We can test Android and iOS by running on emulator devices connected through AVD.

With React Native, to build the application and package into a deployable module is a struggle. In case of iOS, we need to install "pod" then open project in Xcode and change a few settings, and finally choose build configuration option as "Release". For Android, we need to navigate to the Android folder and run ". /gradlew assembleRelease" to get the deployable .apk package. For Windows, it is straight forward that the build process packages the application as "exe".

15. Native Control

React Native has fewer native controls available. In order to access hardware like the camera, we need third party libraries and customization, which requires developer skillset on Javascript.

On the other hand, MAUI has full hardware support using MAUI.Essentials. It provides various APIs to work with camera, app actions, file picker, email, maps, permission, geo location and many more.

16. User experience

React Native requires additional styling libraries like Material UI to make app features more attractive. Though libraries provide sufficient templates, on customization, they lack a native look of the application. It also needs lot of effort to customize available templates

.Net MAUI itself provides lot of Layouts like AppShell, StackLayout, FlexLayout, ScrollView, Frame and Grid.
It has several view tags and cells, which makes the UI feel native on any device.

Evaluation

Features	React Native	.Net MAUI
Performance	🟢✓	🟢✓
Code reusability across different platforms	🟢✓	🟢✓
Ease of writing platform specific codes	🟢✓	🟢✓
Deployment	🔴✗	🟢✓
Native control	🟠✓	🟢✓
User experience	🟠✓	🟢✓
Installation & setup	🟢✓	🟠✓
Documentation & community support	🟢✓	🔴✗
Language & learning curve	🟠✓	🟠✓
IDE & debugging tools	🟠✓	🟢✓
Generated project structure & memory	🟢✓	🟢✓
Third party tool integration & handy solutions	🟢✓	🔴✗
Configuration to support Windows	🟠✓	🟢✓
Security	🔴✗	🟢✓
Compilation	🟠✓	🟢✓

Conclusion

React Native is the best choice if you have very small & delicate UI with lot of third-party library integrations and customization. On the other hand, .Net MAUI looks promising for a native look and many platform-specific functionalities. It is still immature, but has the power to affect the market.