Technological Alternatives for Developing Our Jodel

In this document, we consider four technological alternatives for developing our Jodel:

- Native languages
- Ionic
- Flutter
- React native

Native Languages

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.

Xcode is an integrated development environment (IDE) for macOS containing a suite of software development tools developed by Apple for developing software for macOS, iOS, iPadOS, watchOS, and tvOS. First released in 2003, the latest stable release is version 11.0 and is available via the Mac App Store free of charge for macOS Mojave users.

Advantages of Native Languages

Rich Pre-styled Components Library

As you'd expect, both languages/ frameworks ship with a rich suite of native UI elements which you can use to build your user interfaces. Obviously, they're non-adaptive since you build your UI for only one platform at a time. But the components provide the default platform styles and can of course also be customized.

Plentiful Ecosystem / Third-party Libraries

These languages/frameworks are extremely popular and therefore Stackoverflow is exploding with threads on them. You find plenty of libraries you may use and you therefore probably won't find an issue that can't be solved.

Popularity / Coverage

These two options are the most popular. Have a look at the books, courses, articles, threads and media coverage that exists on these technologies. If someone wants to build an app, he's most likely looking into the native languages first.

Best Performance

Well, you probably can't beat the native languages. Well-written native code should always be more performant than compiled native code.

Access to All Native Device Features

Needless to say, that you got ALL APIs for a given platform available here. You are directly accessing the native APIs after all.

Real World Usage

The vast majority of apps available is written in these native languages. You find thousands of big apps and games that use these languages.

Disadvantages of Native Languages

Highest Development Cost

Obviously you can't use Android Studio for iOS development (or the other way around). Therefore, code re-usability across platforms is inexistent here.

Steepest Learning Curve

You will have to learn different languages for each platform, that is different from the languages used for full-stack Web and Enterprise application development.

Ionic

With Ionic, you still create a real native app but you do this by creating a web app (with HTML, JS and CSS) which will be wrapped by a real native app that hosts a webview (basically a hidden browser).

Advantages of Ionic

Write Once, Use Everywhere

Awesome re-usability! The "wrapped web app" concept ensures that you can easily re-use your code - you're just building a wrapped web app in the end. The great component library of adaptive components (i.e. automatically styled for the platform the app runs on) also helps.

With Ionic, you will get a super-fast development experience. We can use our web development know-how and build mobile apps with one tech stack in little time.

Learn Once, Write Everywhere

There is very little to learn: it's the same TypeScript, React, HTML and CSS used for full-stack Web application development.

Disadvantages of Ionic

Lowest Performance

Ionic offers the lowest performance as its a wrapped web app in the end. But this is often misunderstood! "Lowest" sounds extremely bad but "lowest" doesn't actually mean "bad" or even "horrible". Instead, you got worse performance than with the other approaches, but on the devices we got these days, your app will probably run more than smooth! If you only (imaginary!) got 100fps instead of 105fps - would you notice a difference? You wouldn't.

Limited Access to Advanced Native Devices Features

Ionic uses Cordova or its own solution, Capacitor, to give you access to native device features. It provides a very decent set of packages to access common native device functionalities like the camera. You can also write your own wrappers around native functionalities and then include them in your code of course.

Flutter

Flutter is both a SDK (software development kit) and a framework for Dart - a programming language developed by Google. Flutter itself is also developed by a Google team.

The idea behind Flutter is that you write Dart code which can be compiled to native code that runs on the target device. You use Dart + Flutter framework to build user interfaces composed of so-called widgets. Flutter ships with a bunch of pre-configured widgets (buttons, tabs etc.) and you typically use these to then also build your own, more complex widgets.

Features of Flutter Compared

Reuse of Code Across Platforms

Also really great to reuse. The widgets it ships with often don't adapt to the underlying platform, instead you use Material Design on both platforms by default. The Flutter team is providing more and more iOS-styled components though. You can find out on which platform you're running and manually swap widgets but that's of course a bit more work than required by Ionic.

Component Library

Flutter also ships with a comprehensive suite of built-in widgets. These mostly use the Material Design, some Cupertino-style (iOS style) widgets exist, too. More and more iOS-styled components are getting added. With all these widgets, you can quickly create nice-looking UIs without doing too much manual styling. Only if you need different looks for different platforms, some effort can be required since the widgets don't adapt automatically.

Gentle Learning Curve

Single language for all platforms, but different from those used in full-stack Web development.

Good Performance

This offers you real native apps (compiled from code) and therefore, it provides a better performance than Ionic does.

Access to Native Device Features

Over the last year, the Flutter team put a lot of effort into providing official packages for some of the most common native device features you need access to. There also is a very vibrant ecosystem and hence you find a package for pretty much any native feature you might want to access. You can also write and connect real native code if you need to.

React Native

React Native is a technology developed by Facebook.

It uses JavaScript and the React library to allow you to build user interfaces composed of React components.

Unlike in "normal" React apps built for the browser, you'll NOT use HTML tags. Instead, you'll use a set of pre-built components which will be compiled to native code by the React Native toolchain.

Features of React Native Compared

Reuse of Code Across Platforms

Also compiles to native defaults but only provides a basic set of components to start with. You have to style most of them on your own, hence more work is required to achieve appropriate styles on both platforms. Generally, code can be re-used though (since you still only use one language and libraries like Redux need no adjustment).

Component Library

A decent set of built-in components is provided but a lot of them need to be styled by you. And they're non-adaptive, instead - just as with Flutter - you get alternatives for both operating systems. That requires adjustments in your code, where you have to choose, use and style widgets conditionally.

Gentle Learning Curve

Single language for all platforms, but different from those used in full-stack Web development.

Good Performance

This offers you real native apps (compiled from code) and therefore, it provides a better performance than Ionic does.

Access to Native Device Features

Being the most popular solution, you find a rich set of third-party packages as well as some built-in APIs for accessing native platform functionalities. Relying on third-party packages of course has the disadvantage that the core maintainers of that package might quit, hence the support is not on the same level as it is with Ionic.

Conclusion

Our main concern here is the development schedule. In this regard, Ionic has the advantage. Both in its write once and use everywhere approach and by leveraging our full-stack development expertise.

On the other hand, with current devices, performance is not such a concern, except perhaps for some low-end Android devices.

Also, our application does not require access to any advanced native feature of the device.

Ionic still has no ground-breaking killer app but it has a nice showcase of apps that are using it. Some bigger companies are using it and the speed of development you can achieve with Ionic is probably especially appealing to highly agile and small teams like us. Not meaning that no bigger companies are using it, as you can see in the showcase.

Therefore, these considerations lead us to the conclusion that Ionic is probably the best solution. This conclusion should be validated in a first two-week spike in early November.