Technologies and Considerations

In this report, we will be comparing different frontend, backend, CI/CD, and database options to create a mobile app that runs on Android. For frontend, we will be looking at React Native, Ionic, and Flutter. For our backend, we will look at Java with Spring, Python with Django, and Node with Express. For CI/CD, we will compare Travis, Jenkins, and CircleCI, and for the database, we will look at MySQL, SQLite, and MongoDB.

Since each component services quite a different purpose, we will be using different categories for the 6th consideration

Summary

Of the frontend technologies we looked at, Ionic was deemed least viable due to its relatively low popularity and poor performance, having to rely on a web view to essentially transform a webpage into a native mobile app. Flutter takes the crown for performance and design, and while it's popularity is on the rise, it is written in Dart which makes getting started more difficult. Given the maturity of React Native's community, the fact that it's written in JS, we opted to use it for our frontend.

The three CI/CD tools we looked at are at similar levels of maturity and have been at the top of the industry for most of the last decade. They all cover the other technologies we use and while Jenkins takes the crown for performance and scale, for the purposes of our project, the easier, more lightweight setup option offered by CircleCI and TravisCI are more valuable. Since TravisCI has no free plan for closed source projects, we believe it's best to use CircleCI.

We decided to go with Node with Express.js for our backend. Python is great for ease of development, but it hinders performance. We found Node to be a great in-between in terms of performance and the difficulty of the language. The immense popularity of javascript as well as the many packages available via npm should allow for easier development. Additionally, since we have decided to use React Native for our frontend, using Node for our backend allows us to stick to the same development language. This allows for some coherency between the backend and frontend, and requires less time to learn overall.

Although we aren't using a database, if we were to choose one for this project, we'd have to go with MongoDB. Although SQLite is serverless, performance degrades as the volume of data increases. MySQL is great for performance, but requires much more overhead when compared to SQLite. MongoDB requires a server to run, but maintains great performance while handling high volume requests with non-complex data. Our app doesn't require complicated data, but it does require a high volume of requests. The flexibility that MongoDB allows with schemas is enticing as well, since we aren't bound to the limitations of relational databases.

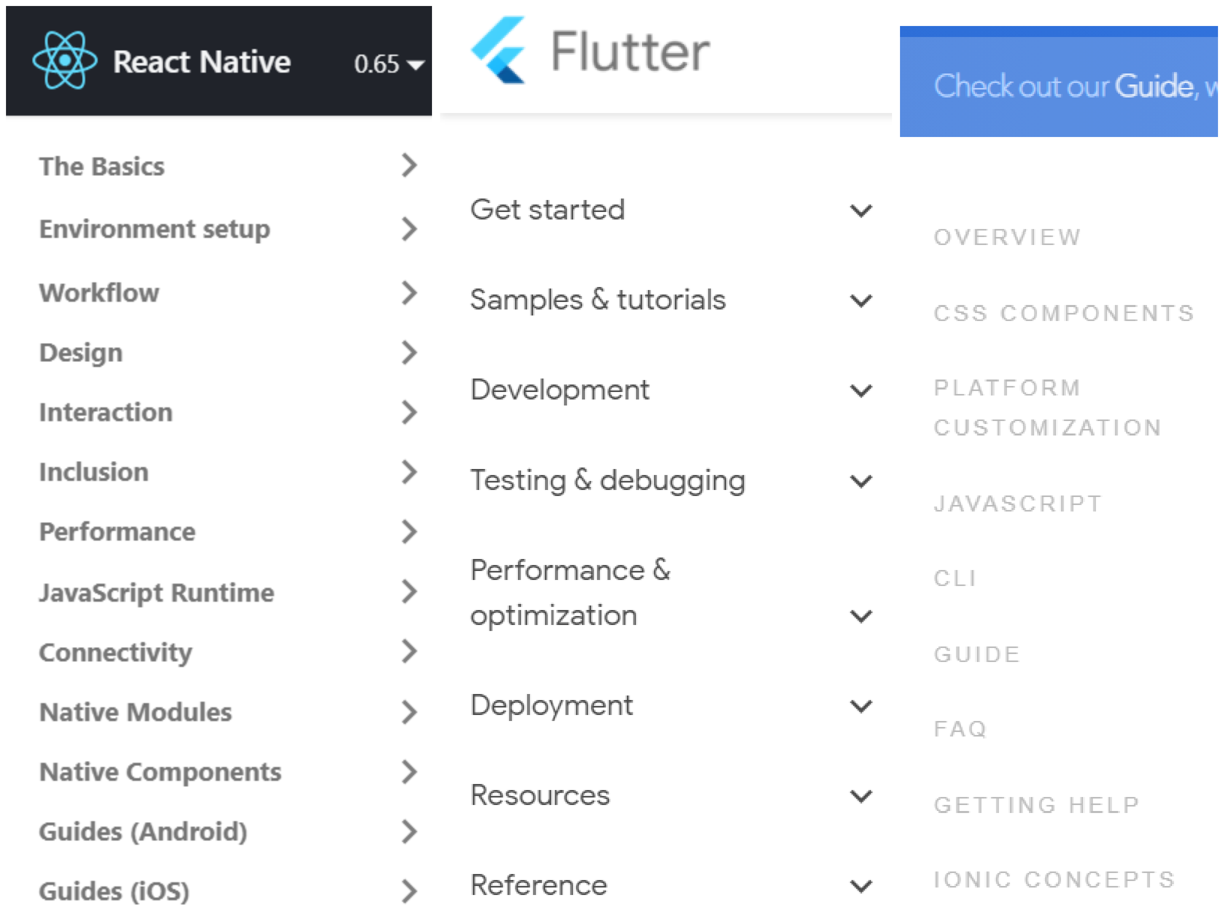
Frontend technology

Ease of development

At the core of it's philosophy, Ionic aims to create a framework that embraces the open web. This means using HTML,CSS, and Javascript which is familiar to many programmers and makes getting started with the framework easier. In contrast, Flutter's philosophy is to create it's own self-contained ecosystem based on Dart- a lesser known language that enables the framework to do things its own way. While this makes getting started harder, the language is still OOP and the framework is the youngest of it's peers- this recency means its custom functionalities are better designed to meet modern development needs. React Native takes a more balanced approach, it lets users program in JSX which makes getting started much easier if one already knows JavaScript and while it does not use CSS, it has a styling system that is very similar.

Maturity

Having extensive, clear, and detailed documentation with relevant examples can be a lifesaver when working with any framework.

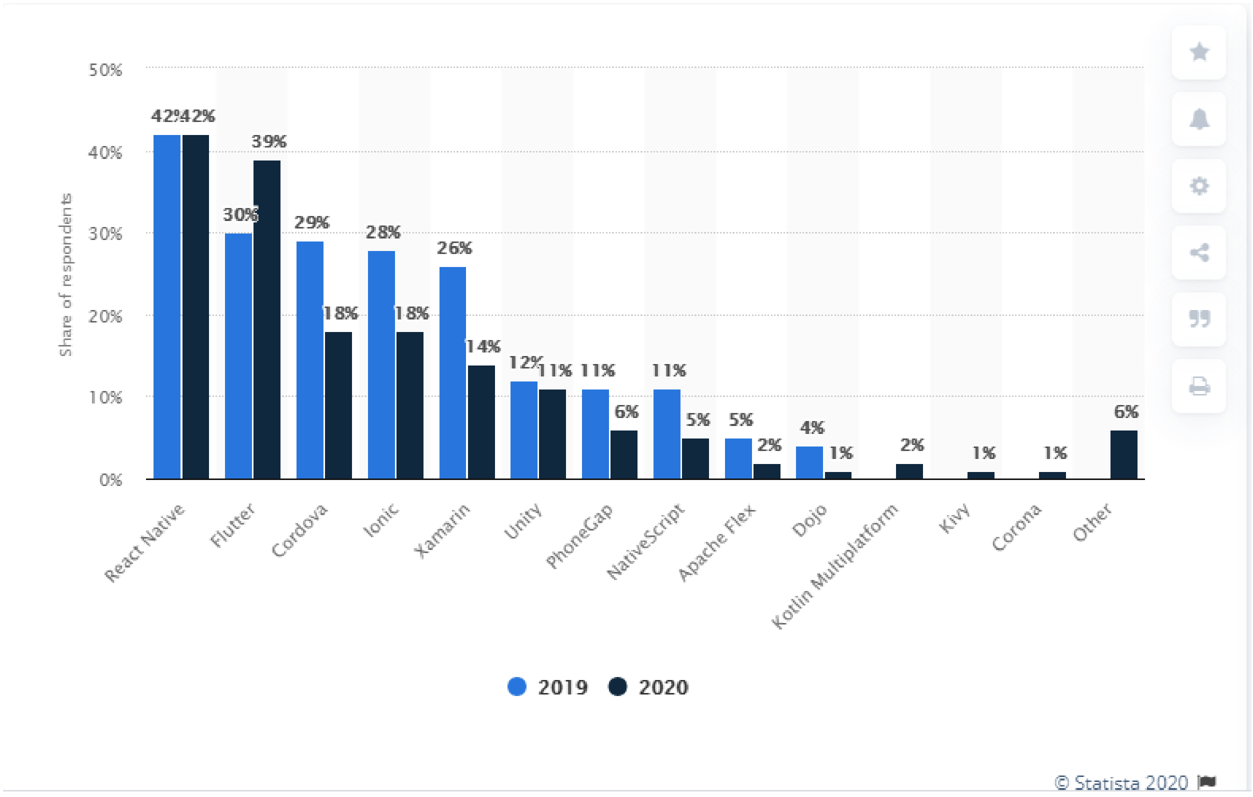


A comparison and quick dive into each framework's official documentation suggests that React Native and Flutter have relatively more expansive and better maintained docs compared to

Ionic. This is expected given the maturity of React Native and the sudden rise of popularity from Flutter. Each website also has its own getting started section with beginner friendly tutorials on environment setup. React Native's documentation goes more in depth on advanced features such as networking and security while Flutter offers more guides on switching over from different frameworks. Again, these differences are likely a result of React's maturity and Flutter's focus on expanding and getting users to switch over.

Domains covered/Cross platform compatibility
Both React Native, Flutter, and Ionic are all suited for cross platform applications with each boasting a "one code base, any platform" design. React Native has arguably the best code reusability having a philosophy centered around reusable components and a community has made many custom libraries for development needs. In cases where there is an incompatibility, both React Native and Flutter offer ways to modify sections of code. Since React Native and Ionic are both JS based, they would pair well with a Node.js backend and allow for an application built on a common language. This will make switching between frontend and backend easier and reduce the chances of bugs that arise from different languages evaluating expressions differently.

Popularity:
From Stack Overflow to reddit to blog posts, having a large community is crucial for resolving roadblocks when it comes to developing under any framework. We will begin our report by comparing the popularity between different websites. Of a 2020 Statista survey, React Native ranked highest among the three frontend technologies.



However, a deeper look revealed that Flutter has been gaining popularity faster on websites such as Stack Overflow and is set to pass React Native in 2021. While Ionic has been around longer than any of the others, it's discussion rate peaked in 2017 and is currency on the decline. React Native is currently at its peak and boasts a mature community for support. Flutter is a runner up and looks to overtake R.N. in the future based on current growth trends. A Linkedin search also showed significantly more results for Flutter, followed by React Native and then Ionic.

Performance
On a busy day, when stores are swamped and customers are eager to checkout and take their groceries home, we want our app to be fast and smooth. As the name suggests, React Native uses JS to wrap around Native components making it almost just like a native framework. Ionic on the other hand, displays content through a Webview which it does through a plugin, this extra step translates to slower performance on large apps. Flutter interacts with native components directly, without the need for a web viewer or JS bridge like Ionic and React. This makes it respond much faster than the others.

Design
A smooth design that makes users feel as if they are using a native application- not just a web page transformed into a mobile app- makes for a better experience. This category is arguably Ionic's biggest weakness, having to depend on a wrapper that essentially transforms a web view into a native view. On top of the performance hit this creates, the design doesn't quite feel like a true native app. React Native delivers a much better view by running a bridge that communicates between JS events and native ones. This gives it a much more native feel compared to Ionic's web view rendering. Flutter provides the most native like experience by having two sets of components, one conforming to iOS standards and the other to Android. This means Flutter compiles directly to the native system and does not require a medium that acts as a translationer.

CI/CD technology

Ease of development
CircleCI is a relatively lightweight tool that does not take long to set up and can automatically detect github repositories once connected to the account. When integrating special frameworks like Cypress tests, CircleCI offers community plugins called orbs that do the required installations so we don't have to do so manually. TravisCI is similar with it also being relatively lightweight and easy to set up. However, a common problem users report is that there is sometimes no notification when a build fails. Jenkins, being a more elaborate system, offers more flexibility and customization but in exchange has a more complicated setup process.

Maturity
CircleCI and Jenkins began in 2011 while Travis was released to the public slightly later in 2013. Jenkins, being the only open source tool of the three, has had the most updates and community support, which is the reason behind why it's a framework rich with features and customization. The problem with that however is that it can be difficult for beginners working on small projects to use. CircleCI and TravisCI were released around the same time and after almost ten years, it's team has had time to iron out the major issues.

The domains covered by the technology
CircleCI can support almost any language and has demo repositories on Github for each of the languages it supports. Travis supports all three of the backend languages we chose for our report and has an in depth guide for each of them. Jenkins also supports our chosen technologies but lacks the breadth of tutorial videos for different languages compared to the other two.

The popularity of the technology
While there is no official number, google search results consistently place Jenkins at the forefront of CI/CD tools. While CircleCI has a free option and is used by famous companies such as Lyft, Spotify, and Coinbase, it's reasonable to believe Jenkins is still the most popular given it is totally free and open source. TravisCI is likely the least popular of the big three because it does not have a free option for close source projects and does not hold the same level of prevalence in search results. A Linkedin job search in Canada containing "Jenkins" also yielded the most results, followed by CircleCI and TravisCI, although most posts simply mention CI/CD without specifying the technology.

Performance, scale and speed of the solutions built with these technologies.
Circle and Travis CI are both more lightweight options compared to Jenkis. This is expected given Jenkins extensive customization options which in exchange, allows it to be more optimized for large software systems. Jenkins takes the crown for performance and scale but for the purposes of our checkout tool, this speed is less important compared to the faster setup time offered by CircleCI and Travis. Each technology also supports caching which greatly reduces project build times after the first try, unless the dependencies are completely changed.

Pricing
While all three are free for open source projects, Jenkins is a completely open source tool and free for anyone to use. CircleCI offers a free plan for smaller closed source projects with the option to upgrade to be able to run more tests in parallel. TravisCI operates under a similar business model except it has no free plan for closed source projects.

Backend Technology

Ease of Development
Of the available languages for backend, Python is likely the most understood one between my partner and I. Python alongside the Django framework is well-suited for fast prototyping without having to read through too much documentation. Javascript is the most unfamiliar language of the chosen options, it would require my partner and I to read up on the language's syntax. Fortunately, Javascript is a versatile language that can be used for both the frontend and backend, meaning coherency between frontend and backend development, and only requiring to learn a single language. My partner and I have worked on a Java project together before. Therefore, using Java, like Python, would allow us to get started on our project without having to read too much documentation. All three of these languages support object oriented programming, which is one of the more understood methods of programming between my partner and I.

Maturity
Python is the oldest of the languages, its first version being released in 1994. Java saw its 1.0 release in 1996, and Node is the most recent of the three choices, being released in 2009. Because of this, Python and Java are much more mature than Node. However, despite Node being more recent, it has received tons of support via packages and documentation as it becomes one of the more popular languages for backend services.

Popularity
We can look towards the Stack Overflow 2021 developer survey to determine which of our possible backend web frameworks is the most popular, as well as the languages they use. Express reigns supreme, sharing 23.82% of the total popularity, while Django and Spring lag behind sharing 14.99% and 14.56% respectively. In terms of the languages that each of these backend frameworks use, Javascript also remains the most popular with Python and Java approaching behind once again. All in all, it seems like the demand for Node.js developers is increasing and will continue to do so.

Performance
Backend performance is extremely important when dealing with many requests at a time. Python trades off performance for ease of development. Since Python is interpreted, we lose out on the performance that natively compiled programs offer. Node and Java are relatively similar in terms of speed, their differences mostly depend on use case. Node, however, does not support multithreading like Java does, which could hinder performance on server side applications that are designed to take many requests at the same time.

Domains Covered
Each of these backend technologies have their respective frameworks for developing REST APIs, which is their primary intention. Python uses Django alongside Django-Rest, Node uses Express JS, and Java uses Spring Boot for REST API development. Additionally, NPM, Node's package manager, is currently the largest language-specific package manager, meaning if there

is a specific use case we need, there is most likely a package out there that supports it. Python sees similar support with its PIP package manager, and finally, Java has also received support with a variety of packages.

Cross Platform Compatibility

All three of these backend languages are known to have excellent cross-platform support. Since none of these languages are compiled (such as C and C++ for example), they only require that the machine hosting the application have support for the engine, virtual machine, or interpreter that the language needs to run on. Nowadays, a majority of hosting solutions support these languages without having to write any machine-specific code.

Database

Relevant Features

SQLite has its name for a reason, it differs from MySQL in the fact that it doesn't require the installation of a database management system and has a very light footprint (only requiring 250Kb-300Kb of memory versus MySQL's 600Mb requirement). MongoDB is a document-oriented database which uses JSON documents to store information. Because of its use of JSON-like documents for storage, it can store a much higher variety of data types when compared to relational database management systems.

Performance

Performance amongst these databases depends on usage. For example, MySQL is much more suited for dealing with large volumes of data efficiently and is considered highly scalable. SQLite is intended for smaller volumes of data, as its performance degrades as volume increases. MongoDB is more suited to handle many requests at once, but with low-complexity data. Additionally, MongoDB is horizontally scalable, making it better than SQLite with its limited vertical scaling capabilities.

Database Type

All of the considered databases differ somewhat from each other. MySQL and MongoDB, for example, require a database server to function, while SQLite is serverless. MongoDB differs more from the other two choices since it is commonly classified as a NoSQL database program. MongoDB is document oriented and instead uses JSON-like documents to store information.

Support in Backend Technology of Choice

Since we have decided to work with Node as our backend, it's important to know how well that specific database is supported within that backend technology and whether it can even be connected. Luckily, all three of the databases we considered are supported through Node's various packages. MongoDB can be connected through the *mongodb* package, MySQL through the *mysql* package, and SQLite through the *sqlite3* package.

Popularity

After comparing the popularities of these databases using the Stack Overflow 2021 developer survey, we can see that MySQL is the database of choice among developers, with more than half of respondents preferring it. Meanwhile, SQLite and MongoDB lag behind with 32.18% and 27.7% of the responses respectively. It should be noted, however, that MongoDB has started to see a small but sure increase in popularity over the years. Especially being a part of the MEAN stack, which is the technology stack of choice for many big companies. Despite this, MySQL has remained the most popular database by a large margin for a significant amount of time.

Decided Solutions

We opted to use React Native for the frontend because of its stronger performance, large community, and greater popularity as opposed to Ionic. Flutter was a strong second option but we decided against it because of the extra time needed to learn Dart. We went with CircleCI for CI/CD because it has a free plan and without the complexities of Jenkins. Node was our backend technology of choice due to the fact that it uses the same language as our frontend and boasts good performance while maintaining ease of use. Finally, although we opted not to use a database for this project, MongoDB would have been our database of choice for its document-based database system which allows for a high number of requests and versatility in allowed data structures.