C1989684

Dr Kathryn Jones

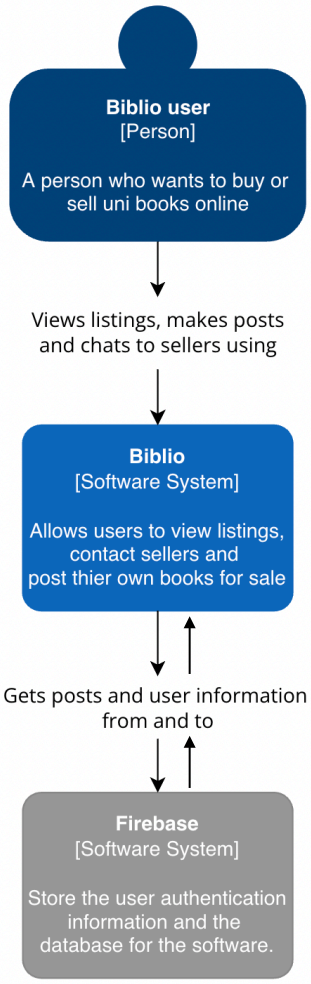
Commercial Frameworks, languages and tools

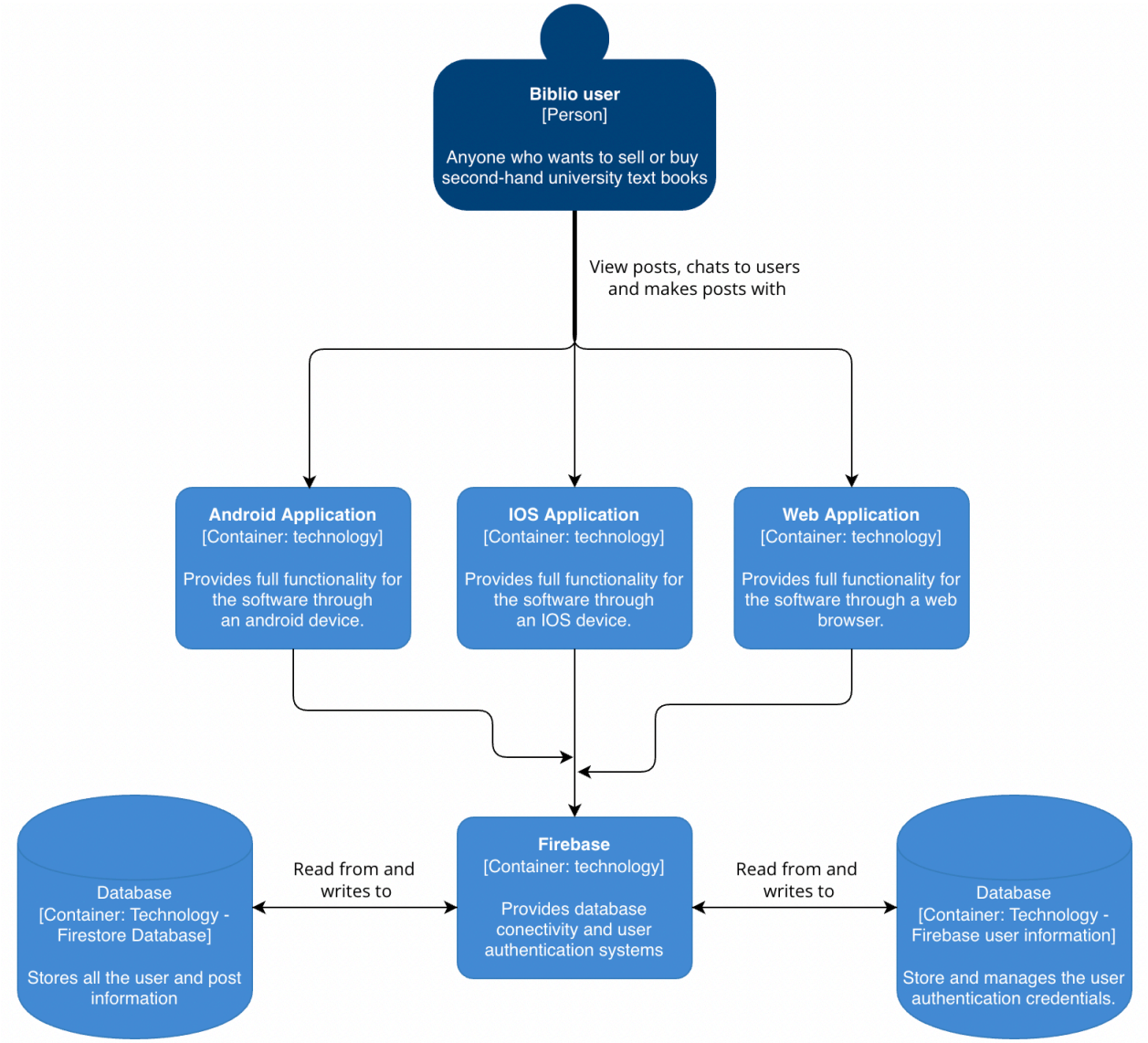
19 January 2022

Architecture, technology stack, frameworks, languages, tools and libraries in the context of the problem discussing what you would have kept the same or changed with evidence justifications and what this means for the future of commercial projects.

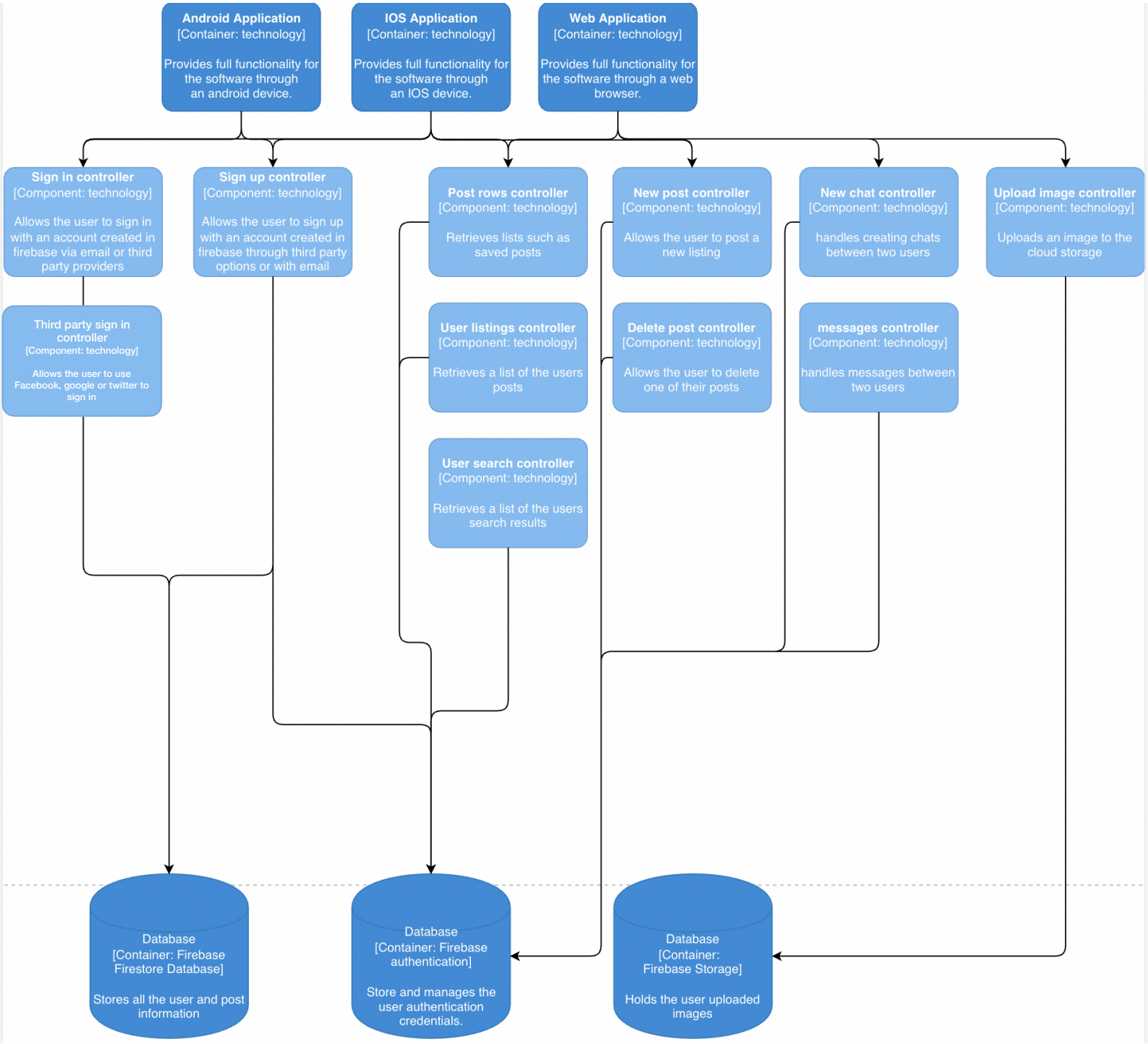
I developed my application, Biblio, using the Flutter framework which combines the java and dart languages to help develop IOS, Android and web applications.

My application followed the BLoC system architecture. This is very similar to the MVC architecture but the logic for controlling the back end is done alongside the front end logic rather than separately[1]. The system context consists of a person using the application, which then in turn talks to the back end of the system. And in return the backend updates the Biblio application view. I chose to do it this way as it is suited for use with Flutter and Firebase and I was already familiar with it from the MVC model. There are not really any other approaches to developing a Flutter application as it is how the framework is designed, however I found it to be intuitive to use and would be happy to use it again in a commercial environment.





The container diagram shows how a user can interact with the application through three different containers, a web application, android devices, and IOS devices. Each of these can then talk to Firebase which holds the databases needed for the application. Once again due to how Flutter is set up this is the only way to really do this, however this is the point of Flutter and was made very easy to do by the framework.



As my project grew the component diagram grew to become quite complex as a lot of my database interaction was done through each individual web page and widgets in accordance to the BLoC architecture. This meant it became quite hard to keep track

of each database query and in the future I would use the database methods file more to keep the requests more organised and make my code more readable and organised.

I used the Flutter framework in conjunction with Firebase to develop my application. Flutter “is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.[2]” I used Flutter to control the front end of the application and to make calls to Firebase, the other part of my stack, which handled the cloud database features for the program.

I found Flutter to be relatively intuitive to use and there was also a lot of detailed documentation on it which was well written and easy to understand. However it is still a relatively new framework (first unveiled in 2015)[3] and is still frequently updated. Because of this I found that some documentation and online advise is now outdated and there is less discussion online in my opinion compared to other languages or frameworks.

Firebase was very easy to use and had detailed instructions on setting up all aspects of the project with Firebase. Documentation also exists specifically for integration of Firebase into Flutter which was very helpful. Going forward I would continue to use Flutter as it has an active development team, the fundamentals are easy to grasp and it provides possibility for all aspects of app design that I needed. However the database within Firebase is a noSQL database and I did find it had some limitations with querying when compared to using an SQL database with a library such as https://pub.dev/packages/mysql1. Although I believe the ease of use with Firebase and extensive documentation makes up for this, and work arounds for the limitations exist. Another possible option would have been react native, which is also used to develop IOS and android applications, but it is not truly cross platform with a single

code base for web browser support and the documentation is generally seen as much poorer.[4] Although react native is better for certain use cases such as smaller sized apps, or You need a minimalistic UI, but rely on significant use of the phone hardware.[5]

Flutter uses the Dart language, which uses java for the logic of the application. I found this to be beneficial as I was already familiar with java from other projects. The development of the user interface was also very simple to understand and developing and implementing widgets across various pages was easy to do. For me I would continue to develop with dart as I am a lot less familiar with JavaScript which is what react native uses, however I would not let this dictate my decision if the app seemed better suited for one framework or the other.

The main tool I used was my IDE. I chose to use android studio as I have developed in it before and am familiar with the JetBrains suit of IDEs which all follow the same design making it easy to swap between. I could have also looked into using an IDE like visual Studio, which is very popular but this mostly comes down to personal choice as they offer almost identical features. Firebase is also a suit of tools from which I used the cloud Firestone database, storage, and user authentication tools. I personally found Firebase easy to use as discussed else where in the report. Finally for IOS development Xcode must be used for some of the set up of the project and to run an IOS simulator. I did find Xcode was slightly finicky to use and required some work to ensure IOS was fully working, but there are no other options for that and so in the future I would have to use Xcode again unless new options become available. Android studio can also open the IOS simulator on a Mac and so I continued to edit code there rather than in Xcode when working on IOS features.

I used a lot of libraries throughout the development of my project. The most important libraries I used were for Firebase connection allowing me to authenticate users and read and write on the database. However I used libraries throughout almost all pages in the app to help with small things like dismissing the keyboard when the user taps on the screen and for the money input when creating a post, and for vital features like google and Facebook sign in. Libraries were easy to use in Flutter and the pub.dev page has a pub score and ratings to help choose good libraries. There were often libraries to help when I got stuck with a problem during development and they all worked as I expected, for that reason I would continue to use Flutter libraries for development.

1. Dhanani, A. 2021. Which Pattern to Choose From MVVM and Bloc? Available at: https://flutteragency.com/which-pattern-to-choose-from-mvvm-and-bloc/ [Accessed: 22/01/22].

2. Flutter, 2021. Flutter homepage Available at: https://flutter.dev/ [Accessed: 22/01/22].

3. Google Developers Youtube, 2015. Sky: An Experiment Writing Dart for Mobile (Dart Developer Summit 2015) Available at: youtube.com/watch?v=PnIWl33YMwA [Accessed: 22/01/22].

4. Gawron, K. 2020. Pros And Cons of React Native Development in 2021 - for Developers and Business Owners. Available at https://www.monterail.com/blog/react-native-development-pros-cons [Accessed 22/01/22]

5. Skuza, B, Włodarczyk, D and Mroczkowska A. 2021. Flutter vs. React Native – What to Choose in 2022?. Available at https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021 [Accessed 22/01/22]