2020

Backdrop: An Exploration of Flutter

Austin D. Latture
Grand Valley State University

Backdrop: An Exploration of Flutter

By
Austin D. Latture

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University
April, 2020

Dr. Jonathan Engelsma **April 11th, 2020**

Table of Contents

Abstract

Throughout the last decade, social media has become increasingly involved with daily life. Much of social media traffic is driven through multimedia content such as photos and videos. People enjoy documenting and sharing their lives on the internet. While there are many applications that enable the capturing, editing, and sharing of photos, there are few that enable the discovery of places in which photos can be taken. A fundamental requirement of a great photo is the background, so discovery of adequate backgrounds is important for interesting photography. Backdrop is an application that targets this space. Backdrop uses geographical data gleaned from both Google and users to show sample photos that are taken at various locations within a given radius of the user. Backdrop accomplishes this task by leveraging several Google APIs. Additionally, the development of Backdrop in the Dart language's framework called Flutter serves as an exploration into cross-platform mobile application development. This exploration details the costs, benefits, and lessons learned when utilizing a single code base for both major mobile application platforms, iOS and Android.

Introduction

Backdrop is a mobile application written in Google's language Dart, using the Flutter framework. The motivation behind developing Backdrop is two-fold. First, Backdrop works toward solving the problem of discovering interesting places to take photos. Many applications allow their users to share photos, but typically a user would need to become inspired by photos taken by other accounts. With Backdrop, a user simply opens the application and sees a map of interesting photos in their location. They can select a location and view the photos that were taken there, allowing for simple discovery of exciting backdrops. Users can also upload their own photos to Backdrop, allowing other people to view them. Finally, Backdrop is anonymous in that the photos taken are not attached or credited to a user's account.

Additionally, Backdrop serves as a means of exploring Flutter itself. Flutter was released by Google in 2017. The Flutter engine compiles the written Dart code into native code for either iOS or Android, depending on the operating system of the user. The advantage of Flutter is obvious in that it allows a development team to use a single code base for a mobile application while still servicing users on both the iOS and Android operating systems. This project serves as an exploration of not only the benefits, but also the costs of using a cross-platform methodology for mobile application development. Understanding these costs is important for the future of mobile application development because they can impact whether or not larger technology firms move toward cross-platform development in the future. If the costs of utilizing Flutter or another cross-platform mobile application framework can be minimized, large companies may be able to greatly reduce the costs of developing mobile apps.

Background and Related Work

There are many available applications that deal with photography and there are many available applications that deal with geography. However, there are very few that have the sole purpose of finding locations in which a user can take interesting photos. The most notable app that allows for this functionality is the search page on Instagram. This page is quite feature limited, however. Instagram allows users to select a category at the top of their search page which allows them find photos that are within that category. Some example categories are architecture, food, art, nature, and shopping. However, the map view is only available in certain locations and is not as feature rich as Backdrop. Alternatively, a user could simply use Google Maps to find places to take photos. Google Maps is obviously very feature rich, but it lacks the user-submitted photo element that Instagram has. The idea of Backdrop is to combine the strengths of each of these applications for this use case, thus eliminating their respective weaknesses.

Program Requirements

Backdrop is a mobile app like any other in that it resides on the home screen of either an iOS or Android device to be opened. Upon opening, the user sees a full screen map that shows all ‘Backdrops’ within a kilometer of them. This map is themed to fit a particular color scheme and provide a unique experience that allows it to stand out from other Google Maps implementations. On this map, the user will see custom markers that can be tapped, revealing a horizontally scrolling list of photos at the given location. These markers are color-coded to distinguish between user-submitted Backdrops and Backdrops that have been queried from Google. Additionally, the user can search for Backdrops anywhere in the United States with the search button in the top right corner. Next to the search icon button, there is a button that allows users to capture, crop, and upload photos to the Backdrop database, allowing them to be viewed by other users. Finally, there is a refresh button next to the add photo button that will refresh the map on the user’s screen. Refreshing updates the map with any new Backdrops that have been added since the user last opened the app and also centers the map on the user’s location. Furthermore, all of this functionality must be enabled for both iOS and Android and maintain at least 60 frames per second, however a constant framerate of 120 frames per second remains the long-term goal.

Implementation

As discussed previously, Backdrop is built in Flutter. One of the main advantages of Flutter is the broad library of ‘widgets’ made available to developers. There are hundreds of these widgets provided by default, and they provide all different kinds of functionality. For example, a Scaffold widget allows for the layout of Backdrop with the title bar at the top, the icons in the top right, and the map as the main focus point. Other widgets enable nearly every standard UI element that one would expect to find in a mobile app such as buttons, text boxes, drop down menus, search bars, list views, and much more. Another advantage

of Flutter is that widgets can be wrapped by other widgets and then encapsulated in their own files, allowing developers to design their own custom widgets. Flutter is also a framework of the Dart programming language which is quite easy to learn for developers who have experience in any of the C-based programming languages like Java, C++, or JavaScript because the syntax is very similar. Both Dart and Flutter were almost completely new technologies for me during the development of this app. My only experience in Flutter prior to building Backdrop was a simple tutorial app that was built by following a walkthrough article. My experience in Java and JavaScript was easy to leverage and made the learning curve quite flat.

Despite my lack of experience in Flutter, I did have some experience in the technology that was used for the backend of the app. For this, I utilized Firebase. I chose Firebase for two reasons: like Flutter, it is developed by Google and it provides a fairly robust NoSQL database for free. Even though I had some experience in Firebase, the development of the backend functionality and the asynchronous nature of the database calls was one of the most difficult parts of developing the app. Firebase does not come with an easy way to query anything based on radius, but is especially poorly equipped to handle querying photos based on radius. In order to solve this problem, I had to utilize a third-party library called GeoFlutterFire. This library allowed me to hash the latitude and longitude of a photo that is uploaded by a user, and then attach that hash to a unique ID. Then, when generating the map, I was able to iterate over the unique IDs in a given radius and fetch their geography information using this library. The photos themselves are stored separately and are only pulled from the database when the user taps on an icon. This technical design was chosen deliberately in order to optimize start-up time for the application and improve the framerate of the map itself.

GeoFlutterFire and Firebase are not the only third-party libraries that I utilized. In addition, I used Google Maps: Flutter to generate the map itself, Flutter Google Places to query for images, Flutter SpinKit to design a custom loader, ImagePicker and ImageCropper to allow the capturing and cropping of photos, and UUIDGen to create custom IDs for each image. These libraries were absolutely essential to my success in building the application. They allowed me to create very robust, interesting functionality with very few lines of code.

Results, Evaluation, and Reflection

The development of Backdrop was a fantastic learning experience. I started with almost no understanding of Dart or Flutter and finished with a solid foundation in both. I also had a lot of fun in building this application as there were constant puzzles to solve regarding the UI, the backend, and the logic interweaving the two. I also believe that I was able to adequately explore the costs and benefits of using a cross-platform mobile application development framework such as Flutter.

To begin, the benefits of Flutter are quite pronounced. The syntax is succinct, the learning curve is not steep whatsoever, and the built-in functionality is fantastic. Flutter's primary competition in the cross-platform space is the React-Native framework for JavaScript. This framework serves a similar purpose to

Flutter in that it offers a single code base alternative to developing an iOS app and Android app in tandem. Prior to exploring Flutter, I attempted to learn React-Native due to my industry experience in using React on the web. Flutter was much easier to install, setup, and get working than React-Native despite my experience in React. This is primarily due to the stark difference in built-in functionality between Flutter and React-Native. Flutter has hundreds of native widgets that are all imported with a single statement and come installed with the framework, while React-Native is built on a huge pile of open-source work from many places which necessitates excessive boilerplate. Even as a React developer, Flutter seems to be the clear choice when deciding on a cross-platform mobile application development framework. This could be because the Flutter developers have stated that the conceptual layering of widgets and the widget lifecycle is based on that of React, but all of the boilerplate of React-Native is unnecessary. Flutter also allowed me to test on both iOS and Android devices without having to write a single line of Swift, Kotlin or Java. I am familiar with all of these languages, but the sheer magnitude of code that would be necessary for this application to run on both iOS and Android would likely require twice the development time that building the application in Flutter did.

Like all software projects, building Backdrop was not an entirely positive experience. Most of the problems that I came across during the development were rooted in the simple fact that Flutter, and all of its third-party libraries, are still in the early stages of the development. The framework is not even three years old. At one point during development, I updated the version of Google Maps: Flutter that I was using, and my code no longer compiled as the developers of that library had changed some of the necessary syntax for utilizing it. Furthermore, at one point during development there were dependency issues that stagnated the process. Essentially, three different libraries were all dependent on different versions of iOS and Android and I was forced to contact the developer of one of these libraries and asked him to release an updated version of his work which he promptly did. If not for this developer updating his library's dependencies, I may have had to rebuild a significant portion of the application using a different library. Another issue with using a language other than Swift, Kotlin, or Java for mobile app development is the feature set of the libraries. For example, in native Android development there may be features in the Google Maps library that are not yet available for Google Maps: Flutter. Finally, performance and framerates were a brief issue at one point during development. Basically, my first iteration of the app was storing all photos that were nearby the user in memory, and then rendering them when the user chose to do so. This almost completely removed the need for a loader. However, the start-up time of the app was abysmally slow and because this is a mobile application, slow start-up times will only be exacerbated by a user having a poor network connection. Additionally, this solution resulted in extremely poor framerates when moving the map around, even on a new iPhone Xs. It remains unclear whether or not these performance issues were unique to Flutter or would have been an issue in React-Native, Swift, or Kotlin as well.

Conclusions and Future Work

As all applications are, Backdrop remains a work-in-progress. I have every intention of continuing to develop the application until it is suitable for deployment. There are very few known bugs at the moment, but the feature set is not quite complete yet. I intend to enable further customization for users of the application but doing so would require the implementation of user accounts. I want to allow the user to customize the radius around them that is queried during the initial rendering of the map. Eventually, if and when there are sufficient users of Backdrop, I would like to allow users to filter out certain 'types' of photos. If the user would only like to see photos taken in nature or in urban centers, I would like to build a way for them to set this preference. Finally, after a critical mass of users is reached, I would like to build a way to 'like' a photo, which would allow the most 'liked' photos to be recommended to users when they open the application in a new location for the first time. Within the realm of geographical data-driven photography, the number of potential features is quite large.

Bibliography

Flutter: Google (2017) source code (Version 1.12)

[Source code] https://github.com/flutter/flutter

Google Maps Flutter: Google (2018) source code (Version ^0.5.0)

[Source code]https://github.com/flutter/plugins/tree/master/packages/google_maps_flutter

Flutter Google Places: FlutterCommunity (2018)

[Source code] https://github.com/fluttercommunity/flutter_google_places

Flutter SpinKit: Jogboms (2019) source code (Version 3.1.0)

[Source code] https://github.com/jogboms/flutter_spinkit

ImagePicker: derTuca (2019) source code (Version 0.6.3)

[Source code] https://github.com/flutter/plugins/tree/master/packages/image_picker

ImageCropper: hnvn (2019) source code (Version 1.0.2)

[Source code] https://github.com/hnvn/flutter_image_cropper

FlutterFire: cbenhagen/franciscojma86 (2019) source code (Version 0.4.4)

[Source code] https://github.com/FirebaseExtended/flutterfire

GeoFlutterFire/Geohash: DarshanGowda0 (2019) source code (Version 2.0.3)

[Source code] https://github.com/DarshanGowda0/GeoFlutterFire

Appendices/Screenshots