Tech Review: Team 13

Tarren Engberg CS361 - Fall 2018

Dec 2nd, 2018

1 Introduction

My role in our team is to lead front-end development client design from user interface GUI design to data handling and implementation. Our team is trying to build a mobile application that allows users to drop digital posts in physical locations. A secondary goal of ours is to learn modern techniques and technologies that will allow us to be more prepared for jobs we will likely go into after graduating next year. This technology review focuses on several front-end technologies and techniques that we are examining in order to make the best decision for our application's use case.

2 Outline

- Client Platform
  - React Native
  - iOS, Android Native
  - Progressive Web App (PWA)
- Data Handling / Structure
  - Flux/Redux
  - Model, View, Controller (MVC)
  - Higher Order Components & View Components vs. Container Components

3 Client Platform

There are three general options available today for building the client-side application for a mobile app. The decision we have to make is to weight the pros and cons between React Native, iOS/Android native and Progressive Web App (PWA). Each of these has strengths and weaknesses both short-term and long. We will examine each and run tests by making pre-development mini apps in order to make an informed decision on which will give the user of our application the best and smoothest possible experience.

3.1 React Native

React Native is a relatively new User Interface building framework that is based on component architecture rather than traditional, large MVC style structures. Developers will write in JavaScript in order to build the application and rely on the Facebook React Native renderer to output both iOS and Android versions of the application. There are obvious benefits to having once code base rather than two separate ones. App consistency across platforms is one important advantage of this model. Additionally, since the development of React Native apps is very similar to React's web development, the cross-over to building a web-app is fairly fast by comparison. Our client has mentioned that in the future it would be nice to have a web version of the app built for the larger screens of computers with more options possible for users. The article 'Let's Get Clear About React Native' details the pros and cons of React Native and let's readers know that companies often opt for it due to its time saving and resource saving nature. Building a native app takes a larger team and more time to complete, whereas building a React Native app can be done faster and includes the vast majority of the features of a fully Native app. One important note is that React Native Apps are NOT ionic. Ionic apps are essentially we browsers with no tool/utility bars. React Native apps actually translate to real Native API calls provided by the developer tools of Android and iOS. [**?**].

Pros include:

- Live & Hot reloading.
- Pre-built Components Increase Dev speed.
- Native Performance.
- Simplified UI.
- Growing Community.

- Immutability Tends Towards Stable, Testable and Predictable App

Cons include:

- User Navigation custom components need work. Wix has a good nav component.
- Few custom modules.
- May require Native code and knowledge.
- Not great for computation intensive clients

3.2 Native: iOS, Android

Native app development is ideal for a number of reasons. First, developers get access to all available developer tools and APIs provided by the operating system's maker. iOS and/or Android development would open up all possibilities and lend the deepest level of customization and raw efficiency. Since Native apps have every sensor and hardware resource readily available to it, the app is not limited in its functional abilities now or in the future as phones get more functionality. Native apps use UI kits provided by Apple and Google, making the user experience potentially more recognizable, if used properly. Some of the downsides of Native apps are that they must be downloaded and installed. Meaning the user must go through Apple's and Google App approval process in order to get the app published to the App Store and Google Play Store. Users only have a certain amount of storage and even a free app may not seem free if it takes up lots of space on a users phone. Developers must make distinct and different applications for iOS and Android platforms. Opening up the possibility of having different bugs and different performance between platforms. Development for iOS and Android tends to be more time costly. One of the largest downsides to native apps is that they must be upgraded regularly to fix bugs and resolve issues, which requires users who want to upgrade their apps. PWA solutions will load a new version on app launch and tend to take up much less space.

3.3 Progressive Web App (PWA)

Progressive Web Apps are defined by Google and offer the best of both the web and native applications. Web apps do not require download and installation, users may save them to their phone's home screen or app drawer and they work offline, unlike websites, through the use of service workers. PWAs also function on much less data when compared to Native applications, allowing users in underdeveloped countries to have access to them and also for users with small on-device storage to have more applications with less worry about storage space. Progressive web apps can also be discovered and crawled by search engines in order to make discover-ability possible on more than just the App Store or Google Play store. PWAs however, unless designed very carefully and executed well, often feel like a 'fake' app and since they don't use standard UI kits provided by an operating system, they can also feel cheap if not designed well. They are also not discover-able in the Google Play store or the Apple App Store.

4 Data Handling

4.1 Redux/Flux

Redux is a library based on the Flux method of data processing and rendering for React based User Interfaces. It is comprised of stores, actions, reducers, and subscriptions. Actions are things a component can do, reducers take a component through the process of doing an action, stores store the actions and subscriptions watch and handle state changes. Redux can be thought of as a predictable state container for JavaScript apps. Redux is not explicitly necessary for simple apps and many developers who don't handle lots of data will find that React on its own works fine without Redux.

4.2 Model View Controller (MVC)

MVC is a classical separation of concerns structural method that isolates views, which are render codes, from models which store data and the two are handled in the controller. MVC has faced recent criticism for not actually separating concerns, but rather separating technologies. Having the view code right next to the logic and implementation code can be extremely practical for everyday feature additions and bug fixes. Models are really the focus and elemental foundation of the MVC application. A model represents knowledge according to its author, and may be an object or set of objects arranged in the order that makes most sense for the application. The view is simply a filter of the model, highlighting and minimizing various aspects of the model for the user. This method is a software engineering model that has good merit for object oriented applications.

4.3 Higher Order Components & View Components vs. Container Components

Higher-order components take components in and return a transformed version of that component. This is advantageous for reusing code and abstracting repeated portions of similar components. Also, engineers may decide to split components into two broad categories. Firstly, visual components and secondarily, container components.

4.4 Conclusion

My recommendation is to choose the PWA option for initial simple client builds and attempt to build out the entire front-end application in PWA until we find that it is impossible or has a significant set back that we cannot foresee. The advantages of PWA technology are many, importantly, users don't have to download the app in order to use it, which is a major barrier to traditional native applications. Secondly, the client is highly interested in building out a full desktop class website in the future and by building the app as a PWA, it will be able to run well in desktop browsers with only style changes to help it adapt well to the full sized screen.

References

[1] Marshall Kirk McKusick, Keith Bostic, Michael J. Karels, John S. Quarterman. The Design and Implementation of the 4.4BSD Operating System `https://download.freebsd.org/ftp/doc/en/books/design-44bsd/book.pdf`

[2] Avishay Traeger *An Introduction to Linux Block I/O.* `https://researcher.watson.ibm.com/researcher/files/il-AVISHA`

[3] Mark Russinovich, David A. Solomon, Alex Lonescu. *Windows Internals Parts 2.* `http://ptgmedia.pearsoncmg.com/images/9780321968975/samplepages/9780321968975.pdf`

[4] Jian Huang *Introduction to System Calls (I/O System Calls).* `http://web.eecs.utk.edu/ huangj/cs360/360/notes/Syscal`

[5] Zhi Wang *The Linux System: Chapter 18.* `http://www.cs.fsu.edu/ zwang/files/cop4610/Fall2016/chapter18.pdf` Florida State University