Build Mobile Apps with Ionic 4 and Firebase

Hybrid Mobile App Development

Second Edition

Fu Cheng

Apress®

Build Mobile Apps with Ionic 4 and Firebase

Hybrid Mobile App Development

Second Edition

Fu Cheng

Apress®

Build Mobile Apps with Ionic 4 and Firebase: Hybrid Mobile App Development

Fu Cheng
Sandringham, Auckland, New Zealand

ISBN-13 (pbk): 978-1-4842-3774-8 ISBN-13 (electronic): 978-1-4842-3775-5
https://doi.org/10.1007/978-1-4842-3775-5

Library of Congress Control Number: 2018961558

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-3774-8. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To my wife Andrea and my daughters Olivia and Erica

Table of Contents

About the Author

Fu Cheng is a full-stack software developer living in Auckland, New Zealand, with rich experience in applying best practices in real product development and strong problem-solving skills. He is the author of the book *Exploring Java 9: Build Modularized Applications in Java* (Apress, 2018), which covers the new features of Java SE 9 and provides a deep dive of Java platform core features. He is also a regular contributor to IBM developerWorks China and InfoQ China, with more than 50 published technical articles covering various technical topics.

About the Technical Reviewer

Felipe Laso is a Senior Systems Engineer working at Lextech Global Services. He's also an aspiring game designer/programmer. You can follow him on Twitter as @iFeliLM or on his blog.

Acknowledgments

This book would not have been possible without the help and support of many others. Thank you to my editors, Aaron Black, Jessica Vakili, and James Markham; and the rest of the Apress team, for bringing this book into the world. Thank you to my technical reviewer Felipe Laso Marsetti for your time and insightful feedback.

Many thanks to my whole family for the support during the writing of this book.

Preface

Developing mobile apps is an interesting yet challenging task. Different mobile platforms have their own ecosystems. If you want to start building your own mobile apps, you have to learn new programming languages, frameworks, libraries, and tools. Building complicated mobile apps or games requires a lot of experience. These kinds of apps may not be suitable for beginners. But not all mobile apps are complicated. There are still many mobile apps that are content-centric. This kind of apps focuses on content presentations and doesn't use many native features. For these kinds of apps, PhoneGap and its successor Apache Cordova offer a different way to build them.

Mobile platforms usually have a component called WebView to render web pages. Cordova uses this component to create a wrapper for running web pages inside of native apps. Cordova provides different wrappers for different platforms. The web pages become the mobile apps to develop. After using Cordova, developers can use front-end skills to create cross-platform mobile apps. This empowers front-end developers to create good enough content-centric mobile apps. Many other frameworks build on top of Cordova to provide out-of-box components to make building mobile apps much easier.

This book focuses on the latest version 4 of the popular Ionic framework. The best way to learn a new framework is using it in real-world product development. This book is not a manual for Ionic 4, but a field guide of how to use it. We'll build a Hacker News client app using Ionic 4 and use this as the example to demonstrate different aspects of Ionic 4. This book not only covers the implementation of the Hacker News client app, but also the whole development life cycle, including development, unit tests, end-to-end tests,

continuous integration, and app publish. After reading this book, you should get a whole picture of building mobile apps using Ionic 4.

Most of the nontrivial mobile apps need back-end services to work with them. Using mobile apps back-end services is a new trend that eliminates the heavy burden to write extra code and maintain the back-end infrastructure. Google Firebase is a popular choice of mobile apps back-end services. The Hacker News client app uses Firebase to handle user authentication and storage of user favorites data. After reading this book, you should be able to integrate Firebase in your own apps.

Included in This Book

This book is organized around building the Hacker News client app. Starting from the local development setup, each chapter focuses on implementing a user story. Here are some important topics:

- Understanding Web Components and Stencil
- Ionic 4 list component with infinite scrolling and pull-to-refresh
- State management with NgRx
- Routing with Angular Router
- User management and third-party service logins with Firebase
- Data persistence with Firebase Cloud Firestore
- Unit testing with Karma and end-to-end testing with Protractor
- App publishing
- Features provided by Ionic Pro, including Ionic Deploy and Monitor

Below are the versions of main frameworks, libraries, and tools used in the app.

- Angular - 6.1.10
- RxJS - 6.3.3
- NgRx - 6.1.0
- Firebase - 5.5.4
- AngularFire2 - 5.0.2
- Jasmine - 3.2.0
- Karma - 3.0.0
- Protractor - 5.4.1

Not Included in This Book

Before you start reading this book, you should be clear about what is not included in this book.

- This book is NOT an Ionic 4 manual. Although this book covers many Ionic components, it doesn't include details about these components. If you want to know more about Ionic components, you should refer to the official documentation.
- This book is NOT a Firebase manual. The official Firebase documentation should be the place to look for more information about Firebase.
- This book is NOT a guide of building Angular web applications. Even though this book uses Angular as the framework binding for Ionic 4, it focuses on building mobile apps, not web applications. However, some practice used in the sample app can be applied to other Angular applications.

Who Should Read This Book?

This book is a field guide of building mobile apps with Ionic 4 and Firebase. Everyone who is interested in mobile apps development with Ionic 4 should read this book. This book is especially suitable for the following readers.

- Individual developers. This book is a perfect match for individual developers. No matter if you want to build mobile apps for a living or as side projects, as an individual developer, you're responsible for the whole development life cycle. This book covers all the necessary phases in the development life cycle.
- Team members in a small team. If you are on a small team, it's likely that you may need to touch different aspects of product development. This book can help you to understand the development life cycle to easily shift between different roles.
- Development team members in a large team. If you are on a large team, you may only be responsible for some pages or components. This book covers some advanced topics related to web development, including routing, state management with NgRx, and advanced usage of RxJS, which can be useful in development.

How This Book Is Organized

This book is organized around building the example Hacker News client app with Ionic 4 and Firebase. Below is a brief description of all the chapters in this book.

- Chapters 1–4: Introduction of necessary background knowledge and local development environment setup.
- Chapters 5–11: Implementation of all user stories of the example app with Ionic 4 and Firebase.
- Chapters 12–15: Additional topics related to Ionic.

Depending on your skill level, you can choose to skip some of these chapters. If you are an experienced Angular developer, you can skip the first four chapters.

Prerequisites

Ionic 4 is framework agnostic, and this book uses Ionic Angular as the framework binding. Basic knowledge of Angular and TypeScript is required to understand the code in this book. This book provides the basic introduction to Angular and TypeScript, but it's still recommended to refer to other materials for more details.

To build Ionic 4 apps running on iOS platform, macOS is required to run the emulator and Xcode. You may also need real physical iOS or Android devices to test the apps.

CHAPTER 1

Getting Started

Mobile apps development is a hot topic for both companies and individual developers. You can use various kinds of frameworks and tools to build mobile apps for different platforms. In this book, we use Ionic 4 to build so-called hybrid mobile apps. As the first chapter, this chapter provides the basic introduction of hybrid mobile apps and helps you to set up the local environment for development, debugging, and testing.

After reading this chapter, you should have a basic understanding of how hybrid mobile apps are built and get a local development environment ready to use.

Mobile Apps Refresher

With the prevalence of mobile devices, more and more mobile apps have been created to meet all kinds of requirements. Each mobile platform has its own ecosystem. Developers use SDKs provided by the mobile platform to create mobile apps and sell them on the app store. Revenue is shared between the developers and the platform. Table 1-1 shows the statistics of major app stores at the time of writing.

F. Cheng, *Build Mobile Apps with Ionic 4 and Firebase*,
https://doi.org/10.1007/978-1-4842-3775-5_1

Table 1-1. *Statistics of Major App Stores*

App Store	Number of available apps	Downloads to date
App Store (iOS)	2.2 million	140 billion
Google Play	2.8 million	82 billion
Windows Store	669,000+	--
BlackBerry World	245,000+	4 billion

The prevalence of mobile apps also creates a great opportunity for application developers and software companies. A lot of individuals and companies make big money on the mobile apps markets. A classic example is the phenomenal mobile game Flappy Bird. Flappy Bird was developed by Vietnam-based developer Dong Nguyen. The developer claimed that Flappy Bird was earning $50,000 a day from in-app advertisements as well as sales. Those successful stories encourage developers to create more high-quality mobile apps.

Let's now take a look at some key components of mobile app development.

Hybrid Mobile Apps

Developing mobile apps is not an easy task. If you only want to target a single mobile platform, then the effort may be relatively smaller. However, most of the time we want to distribute apps on many app stores to maximize the revenue. To build the kind of apps that can be distributed to various app stores, developers need to use different programming languages, SDKs, and tools: for example, Objective-C/Swift for iOS and Java for Android. We also need to manage different code bases with similar functionalities but implemented using different programming languages. It's hard to maximize the code reusability and reduce code duplications

across different code bases, even for the biggest players in the market. That's why cross-platform mobile apps solutions, like Xamarin (https://www.xamarin.com/), React Native (https://facebook.github.io/react-native/), RubyMotion (http://www.rubymotion.com/), and Flutter (https://flutter.io/) also receive a lot of attention. All these solutions have a high learning curve for their programming languages and SDKs, which creates a burden for ordinary developers.

Compared to Objective-C/Swift, Java, C#, or Ruby, web development skills – for example, HTML, JavaScript, and CSS are much easier to learn. Building mobile apps with web development skills is made possible by HTML5. This new type of mobile apps is called hybrid mobile apps. In hybrid mobile apps, HTML, JavaScript, and CSS code run in an internal browser (WebView) that is wrapped in a native app. JavaScript code can access native APIs through the wrapper. Apache Cordova (https://cordova.apache.org/) is the most popular open source library to develop hybrid mobile apps.

Compared to native apps, hybrid apps have both their benefits and drawbacks. The major benefit is that developers can use existing web development skills to create hybrid apps and use only one code base for different platforms. By leveraging responsive web design techniques, hybrid apps can easily adapt to different screen resolutions. The major drawback is the performance issues with hybrid apps. As the hybrid app is running inside of an internal browser, the performance of hybrid apps cannot compete with native apps. Certain types of apps, such as games or apps that rely on complicated native functionalities, cannot be built as hybrid apps. But many other apps can be built as hybrid apps.

Before making the decision of whether to go with native apps or hybrid apps, the development team needs to understand the nature of the apps to build. Hybrid apps are suitable for content-centric apps, such as news readers, online forums, or showcasing products. These content-centric apps act like traditional web apps with limited user interactions. Another important factor to consider is the development team's skill sets. Most

apps companies may need to hire both iOS and Android developers to support these two major platforms for native apps. But for hybrid apps, only front-end developers are enough. It's generally easier to hire front-end developers rather than Java or Swift/Objective-C developers.

Apache Cordova

Apache Cordova is a popular open source framework to develop hybrid mobile apps. It originates from PhoneGap (`http://phonegap.com/`) created by Nitobi. Adobe acquired Nitobi in 2011 and started to provide commercial services for it. The PhoneGap source code contributed to the Apache Software Foundation and the new project Apache Cordova was started from its code base.

An Apache Cordova application is implemented as a web page. This web page can reference JavaScript files, CSS files, images, and other resources. The key component of understanding how Cordova works is the `WebView`. `WebView` is the component provided by native platforms to load and run web pages. Cordova applications run inside the `WebViews`. A powerful feature of Cordova is its plugin interface, which allows JavaScript code running in a web page to communicate with native components. With the help of plugins, Cordova apps can access a device's accelerometer, camera, compass, contacts, and more. There are already many plugins available in Cordova's plugin registry (`http://cordova.apache.org/plugins/`). We can easily find plugins used for common scenarios.

Apache Cordova is just a runtime environment for web apps on native platforms. It can support any kinds of web pages. To create mobile apps that look like native apps, we need other UI frameworks to develop hybrid mobile apps. Popular choices of hybrid mobile apps UI frameworks include Ionic framework (`http://ionicframework.com/`), Sencha Touch (`https://www.sencha.com/products/touch/`), Kendo UI (`http://www.telerik.com/kendo-ui`), and Framework7 (`http://framework7.io/`). Ionic framework is the one we are going to cover in this book.

Ionic Framework

Ionic framework is a powerful tool to build hybrid mobile apps. It's open source (https://github.com/ionic-team/ionic) and has over 35,000 stars on GitHub, the popular social coding platform. Ionic framework is not the only player in hybrid mobile apps development, but it's the one that draws a lot of attention and is recommended as the first choice by many developers. Ionic is popular for the following reasons:

- Based on Web Components standards and is framework agnostic. Web Components are W3C specifications of components for the web platform. Ionic components are built as custom elements using its own open source tool, Stencil. Being framework agnostic makes Ionic components work with any framework. Developers are free to choose the framework to use, including Angular, React, and Vue.
- Provides beautifully designed out-of-box UI components that work across different platforms. Common components include lists, cards, modals, menus, and pop-ups. These components are designed to have a similar look and feel as native apps. With these built-in components, developers can quickly create prototypes with good enough user interfaces and continue to improve them.
- Leverages Apache Cordova as the runtime to communicate with native platforms. Ionic apps can use all the Cordova plugins to interact with the native platform. Ionic Native further simplifies the use of Cordova plugins in Ionic apps.
- Performs great on mobile devices. The Ionic team devotes great effort to make it perform well on different platforms.

The current release version of Ionic framework is 4.0. Ionic 4 is the first version of Ionic to be framework agnostic. Ionic Core is the set of components based on Web Components. Ionic Angular is the framework binding of Ionic Core with Angular. This book focuses on Ionic Angular with Angular 6.

Apart from the open source Ionic framework, Ionic also provides a complete solution Ionic Pro for mobile app development, which includes the following products:

- **Ionic Creator** – Ionic Creator is a desktop app to create Ionic apps using drag-and-drop. It helps nontechnical users to quickly create simple apps and prototypes.
- **Ionic View** – Ionic View allows viewing Ionic apps shared by others directly on the phones. It's a great tool for app testing and demonstration.
- **Ionic Deploy** – Ionic Deploy performs hot updates to apps after they are published to app stores.
- **Ionic Package** – Ionic Package builds Ionic apps and generates bundles ready for publishing to app stores. With Ionic Package, we don't to manage local build systems and can use the cloud service instead.
- **Ionic Monitor** – Ionic Monitor can monitor apps and report runtime errors.

This book also covers usage of these products in Ionic Pro. Ionic Pro offers a free starter plan to try out features provided by Ionic Deploy and Ionic Monitor. With a $29 per month plan, you can access all features in Ionic Pro. Check out the pricing (https://ionicframework.com/pro/pricing) of Ionic Pro if you want to know more about it. Another important project to mention is Ionic Capacitor (https://capacitor.ionicframework.com/), which will replace Apache Cordova to build

native progressive web apps. Even though Capacitor is out of the scope of this book, it's still worthwhile to check out this project to see the future of developing hybrid mobile apps with Ionic.

Firebase

Mobile apps usually need back-end services to work with the front-end UI. This means that there should be back-end code and servers to work with mobile apps. Firebase (https://firebase.google.com/) is a cloud service to power apps' back ends. Firebase can provide support for data storage and user authentication. After integrating mobile apps with Firebase, we don't need to write back-end code or manage the infrastructure.

Firebase works very well with Ionic to eliminate the pain of maintaining back-end code. This is especially helpful for hybrid mobile apps developers with only front-end development skills. Front-end developers can use JavaScript code to interact with Firebase.

How to Build Mobile Apps

Even with the frameworks and services mentioned above, it's still not an easy task to build mobile apps. There are multiple stages in the whole development life cycle from ideas to published apps. A typical process may include the following major steps:

- Ideas brainstorming. This is when we identify what kind of mobile apps to build. It usually starts from vague ideas and expands to more concrete solutions.
- Wire-framing and prototyping. This is when we draw on the whiteboard to identify main usage scenarios. Prototypes may be created to demonstrate core usage scenarios for better communications with stakeholders.

- User experiences design. This is when all pages and navigation flows are finalized, and we are now clear what exactly needs to be built.
- Implementation. This is when the development team implements the pages to fulfill requirements.
- Testing. Unit testing should be part of implementation of pages and components. End-to-end testing is also required to verify all usage scenarios. All these tests should be executed automatically.
- Continuous integration. Continuous integration is essential for code quality. If every code commit can be tested automatically, then the development team will be more confident about the product's quality.
- Publishing. This is when the app is published to app stores.
- Operations. After the app is published, we still need to continuously monitor its running status. We need to capture errors and crash logs occurred on users' devices.

This book is a guide to build mobile apps that focuses on implementation and testing. Topics including continuous integration, app publishing, and monitoring are also covered.

Prepare Your Local Development Environment

Before we can build Ionic apps, we first need to set up the local development environment. We'll need various tools to develop, test, and debug Ionic apps.

Node.js

Node.js is the runtime platform for Ionic CLI. To use Ionic CLI, we first need to install Node.js (https://nodejs.org/) on the local machine. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It provides a way to run JavaScript on the desktop machines and servers. Ionic CLI itself is written in JavaScript and executed using Node.js. There are two types of release versions of Node.js – the stable LTS versions and current versions with the latest features. It's recommended to use Node.js version 6 or greater, especially the latest LTS version (8.12.0 at the time of writing).

Installing Node.js also installs the package management tool npm. npm is used to manage Node.js packages used in projects. Thousands of open source packages can be found in the npmjs registry (https://www.npmjs.com/). If you have background with other programming languages, you may find npm is similar to Apache Maven (https://maven.apache.org/) for Java libraries or Bundler (http://bundler.io/) for Ruby gems.

Ionic CLI

After Node.js is installed, we can use npm to install the Ionic command-line tool and Apache Cordova.

```
$ npm i -g cordova ionic
```

Note You may need to have system administrator privileges to install these two packages. For Linux and macOS, you can use `sudo`. For Windows, you can start a command-line window as the administrator. However, it's recommended to avoid using `sudo` when possible, as it may cause permission errors when installing native packages. Treat this as the last resort. The permission errors usually can be resolved by updating the file permissions of the Node.js installation directory.
