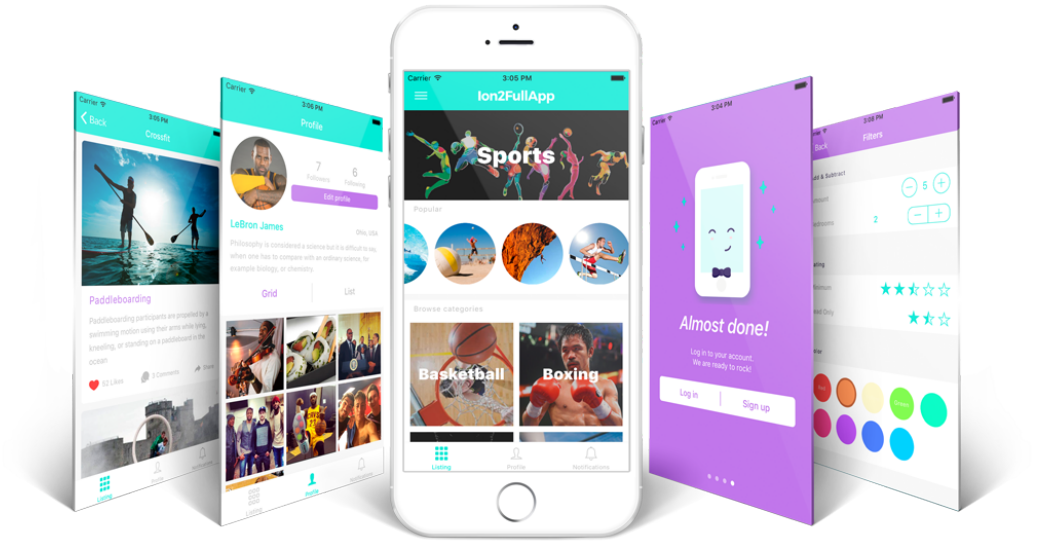
Ion2FullApp - Full Starter Mobile App for ionic 2

A BEAUTIFUL MULTI PURPOSE MOBILE STARTER APP WITH TONS OF FUNCTIONALITIES AND EXAMPLES



Ion2FullApp is a mobile app both for Android and iOS made with Ionic framework and Cordova.

Ionic framework is an open source front-end SDK for developing awesome hybrid mobile apps with HTML5, CSS and JavaScript. Ionic is focused mainly on the look and feel, and UI interaction of your app.

Cordova is a platform to build Native Mobile Applications using HTML5, CSS and JavaScript.

You can buy this product here:
https://ionicthemes.com/product/ion2fullapp-full-ionic2-app-template.

TRY BEFORE YOU BUY

See a **video preview** **here.**

Download the **APK** from here.

Try how this component looks and feels in your device using **Ionic View**

1. Download Ionic View app.
2. Open the app on your phone.
3. Enter the following APP_ID: **9c8fa63b**

BENEFITS

FOR DEVELOPERS ...
This is the time to save your expensive time and write less code for your new app. You can build a rich app with Ionic in a very straightforward way. Your knowledge about using CSS, HTML, and JavaScript will help serve as the building blocks of your app. Ionic also has lots of tutorials you can use.

FOR DESIGNERS ...
Customization of layout has no limit with Ionic. You don't even have to modify a complicated widget. It is pretty clean and simple to start your own customization and create a satisfactory design.

FEATURES

CUTTING EDGE TECHNOLOGY **ionic 2** **+** **Angular 2** **+** **Sass**
We went all in and became experts on the new **ionic 2** and **Angular 2** to get you this awesome starter app that goes beyond the basics.
Ionic offers a free and open source framework, with components of CSS, JS, mobile optimized HTML, as well as gestures and tools needed to build high end apps with amazing user friendliness.
ionic 2 is the latest version of the framework and it's built using **Sass** and optimized for handling **Angular 2** (that means it inherits all the benefits of **Angular 2**!) to construct a functional application structure.
This gives you a mix of power and beauty. Using Ionic to develop hybrid apps gives you the advantage of accessing the native API's of devices, that's the difference between mobile web and hybrid apps.

BUILT WITH SASS

This component uses SASS, which basically is CSS with superpowers. Each component has its dedicated sass files and partials well structured with independent variables so you can have maximum modularity, flexibility and customizability.

MORE THAN JUST LAYOUT

This project uses AngularJS to construct a functional application structure including UX interactions such as form validations, navigation logic between views, etc which makes a coherent app not just html and css. Although it comes with the frontend part and many interactions, it's worth mentioning that this project does not come with the backend part of the app, it's meant to be as generic as possible so it can work with any backend solution.

As we mentioned before in addition to AngularJS interactions this project enables you to access the native API's of devices for example through the social share integration. If you need more interactions you can check ngCordova

INDEX

SET UP

First you will need to have Ionic installed in your environment. Please refer to the Ionic official installation documentation in order to **install Ionic and cordova**

After you have Ionic and Cordova installed, you can finish the setup of this starter theme by running these commands inside your ionic app folder.

1. Run **npm install** in order to install all the node modules required by the app.

TEST YOUR APP

Now we have everything installed, we can test the app.

Run **ionic serve** to start a local dev server and watch/compile Sass to CSS. This will start a live-reload server for your project. When changes are made to any HTML, CSS, or JavaScript files, the browser will automatically reload when the files are saved.

Go here to learn how to test your app like a Native app on iOS and Android devices or emulators.

HOW TO

Ionic 2 is very different from Ionic 1. We have much experience building apps and themes with both of them and let me tell you that in my opinion Ionic 2 is much more mature, stable and understandable than Ionic 1. Many hard things that require workarounds with Ionic 1 are now trivial to implement with Ionic 2. It's clearly a step forward!

In this section I'll do my best to explain you the basics of an Ionic 2 app so you have a better understanding on where to start and where to focus.

APP STRUCTURE

There are three main folder structures in an Ionic 2 app.

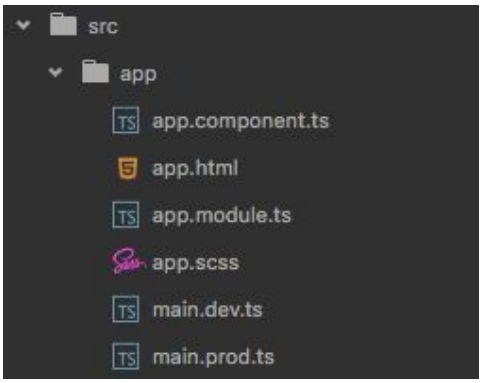
- **App**: has all the files needed to bootstrap the app and the main structure we will build the app upon.
- **Pages**: as you imagined, here goes all the app pages html, js and css. Including models, services to access backend data, etc.
- **Theme**: this folder contains all the superpowers of Sass (variables, mixins, shared styles, etc) that makes super easy to customize and extend the app.

And there are other secondary but also important folders

- **Components**: here we included custom components that enhance the user experience of our full starter template app.
- **Assets**: in this folder you will find images, sample data json's, and any other asset you may require in your app.

Note: *if you want more information, please refer to Ionic Getting Started pages.*

app



app.component.ts

It's the main entry point of our app. It defines the basic structure and initial navigation of the app.
In our case we have a combination of tabbed navigation and side menu navigation. In this file we define which page would be the first one and the options and navigations of the side menu.
It's also where we get notified when the platform is ready and our plugins (cordova, native stuff) are available. That enables you to do any higher level native things you might need.

app.html

Here we define the navigation and it’s root. Also, as we have a side menu, here is where we should place it’s layout.

app.module.ts

This file includes the main angular 2 module (NgModule) of our app. It’s also the place where we should declare the vast majority of our dependencies, such as pages, custom components, services, etc.

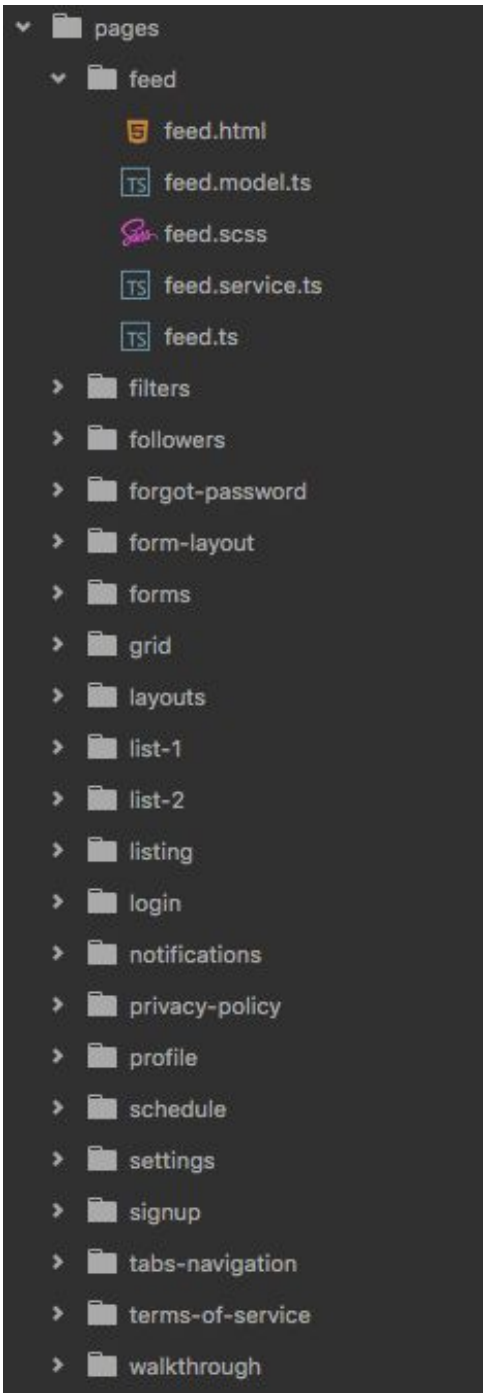
app.scss

This is the main entry point for our app sass files/styles. Here is the place you should include your app shared imports and global sass you may use and apply globally. Additionally, this file can be also used as an entry point to import other Sass files to be included in the output CSS.
This is **NOT** the place to include shared Sass variables. You have to define, adjust, add those in "theme/variables.scss".

main.dev.ts - main.prod.ts

These are ionic auto-generated files, and they take care of the bootstrapping of app.

pages



Each page has it's folder. And within that folder you will find every related file for that page. This includes the html for the layout, sass for the styles, the data modeling for the content that will fill the layout, any additional service to access the data that will be presented in the page and the main page component.

Let's have a closer look at what I'm talking about deglossing the feed page.

feed.html

All the layout for the page. Everything we do is crafted using cutting edge techniques and technologies and we always code towards customizability and ease of use. We also make use of the awesome ionic 2 components.

feed.model.ts

As you may know Angular 2 relies heavily on typescript and object oriented programming. It's a good and recommended approach to follow these principles along the way, That's why we create models to represent the data that's going to be presented in each layout. That's what you will find in these files.

feed.scss

Centralized styles for this page. This app structure makes it easy for you to know and centralize in one place where you should change stuff.

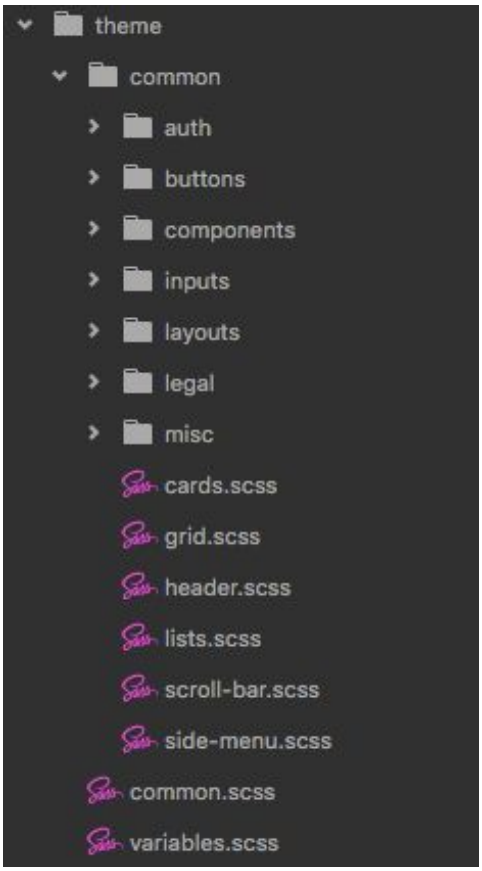
feed.service.ts

In this file you will find methods to access and pull the data that will be presented in the page. This is where you should call your backend API. In our case we are using sample data pulled from a series of json files.

feed.ts

Here are all the functionality and interactions of the page and where the view logic should go.

theme



Here you will find all the variables, mixins, shared styles, etc that makes this app template so customizable and extendable.
Maybe you don't know Sass, in short is a superset of css that will ease and speed your development cycles incredibly.

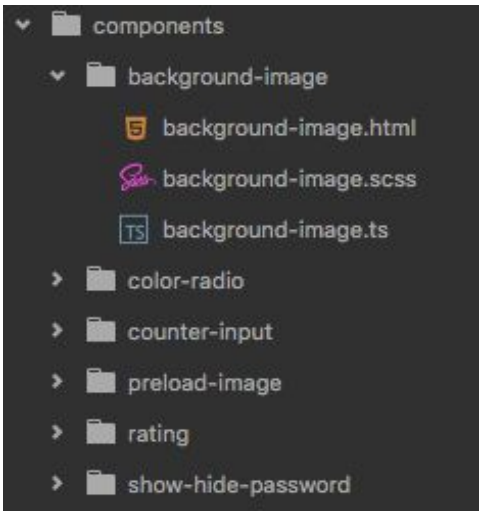
common

Under the theme/common folder you will find (classified by component/functionality) all the shared styles, this way we encourage code reuse and prevent DRY.

variables.scss

This is the predefined ionic file where you should include all the variables you may use in your app. In our case we defined all the colors used within the app in easy to change variables so you can play around and try different color schemes.

components



These are custom components we created to improve the user experience of our app template.

background-image

This component adds a background image to the desired html element and clips itself to fill the area of that html element.

preload-image

This component prevents your content to jump around while images load. We do this by specifying an aspect ratio for the image you want to load and applying the technique described in this article.

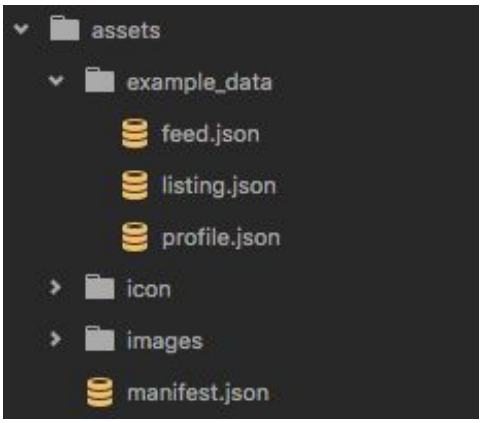
color-radio, counter-input, rating

These custom inputs/components are meant to improve the user experience in forms, filter pages,

etc. You may find them handy depending on your use cases.

show-hide-password
In mobile devices sometimes is hard to type and is very common you miss some character when typing passwords. This directive let you show/hide the unmasked password so you improve user experience by avoiding typos on passwords.

assets



Here go all the images you may use in your app as well as other assets.
In our case we simplified the data layer of the app by creating a series of jsons with the sample data of each page. We followed this path because there are so many options and flavours when it comes to backend implementations that it turns impossible to provide examples for every option.

CUSTOMIZE IT

New pages?

In Ionic 2 there's some more boilerplate compared to ionic 1, but don't panic. The new ionic CLI also has more tools to help you out with this.
Take for example the new generator functions, they provide an easy way to create pages and services for your app. This makes going from a basic app to a full featured app with navigation much easier.
To create a page you can use the following command:

```
# ionic g page <PageName>
ionic g page myPage

√ Create app/pages/my-page/my-page.html
√ Create app/pages/my-page/my-page.ts
√ Create app/pages/my-page/my-page.scss
```

Note: *Please refer to ionic documentation for more information about adding pages to your app.*

Custom styles?

We have talked about this above. Believe me, Sass will be your best friend when it comes to styling, it gives superpowers to your css.
The way we structured and build the styling system for this app template will enable you to quickly modify, add and create new styles to match the look and feel and the use cases of your app. There's much to learn about Sass to get the most out of it. We believe learning by example is the best way, and you will enjoy that benefit with our app template. Also you will find useful documentation about Sass here and here.

New layouts?

If you want to implement new layouts I would highly recommend you to first search within ionic vast list of components to check if there's one that fit your needs.
If you find none, then try to think the way you do when developing for the web, after all one of the beautiful things and advantages about ionic is that you build stuff using web technologies (html, css, js), and such the web is an open platform you will find tons of solutions for already solved issues.
In particular when I don't find the component I need, the approach I follow is first think the structure and elements I may need to represent and build the shell of what I want. I constantly rely on ionic grid system. And since I discovered it, I've become an expert on Flexbox, and let me tell you, that makes the difference!

Splash screen and app icon

An app icon and splash screen (launch image) are important parts of any app, yet making them used to be incredibly tedious. You needed numerous icons for iOS and Android, and then you had to deal with splash screens and all their different sizes.
To save you the stress of dealing with all that, ionic enables you to generate app icons and splash screens via the Ionic CLI.

With the ionic CLI, all you need is a resource directory and two images. These images can be .png files, Photoshop .psd files, or Illustrator .ai files, named `icon.png` and `splash.png`. With the images in a resources directory, `./resources`, the `ionic resources` command will generate the icons and splash screen images for each platform setup in the project, sending them to Ionic's image resizing and cropping server, so you don't have to install extra dependencies.

If you only need to update one of the resources, or you only want to generate icons and not both, the ionic resources command has two flags that allow you to target each asset, instead of generating both.

```
ionic resources --icon
ionic resources --splash
```

If you need any further assistance with this, please go to http://ionicframework.com/blog/automating-icons-and-splash-screens/ for more information.

DATA INTEGRATION

The key to an evolving app is to create reusable services to manage all the data calls to your backend.
In this section we won't discuss any backend implementation in particular because as we explained above, there are so many options and flavours when it comes to backend implementations that it turns impossible to provide examples for every option.
We will focus instead on the app's side of the problem, how to handle data calls, as this works the same and is independent on the way you implement the backend. We will talk about models and services and how they work together to achieve this.

We encourage the usage of models in combination with services for handling data all the way from the backend to the presentation flow.

Models

Domain models are important for defining and enforcing business logic in applications and are especially relevant as apps become larger and more people work on them.
At the same time, it is important that we keep our applications DRY and maintainable by moving logic out of components themselves and into separate classes (models) that can be called upon. A modular approach such as this makes our app's business logic reusable.
To learn more about this, please visit this great post about angular 2 domain models.

Services

As you may know, ionic 2 is implemented on top of angular 2 and it borrows it's best parts. Angular 2 enables you to create multiple reusable data services and inject them in the components that need them.
Refactoring data access to a separate service keeps the component lean and focused on supporting the view. It also makes it easier to unit test the component with a mock service.
To learn more about this, please visit angular 2 documentation about services.

NIFTY STUFF

There are several things that made us believe ionic 2 is much better, easier and mature than ionic 1 to build high quality apps. Some of them are explained below.

Object oriented everything!

As we mentioned when we talk about models, they are key for defining and enforcing business logic in applications. Angular 2 embraces this and that's why you will see object oriented stuff all along the way. Depending on the style guide you followed for developing angular 1 applications, this may sound akin to you or not as angular 1 didn't embraced OOP the way angular 2 does.

Another key ingredients to my previous argument are Typescript and ES6. Angular 2 is written in TypeScript, and TypeScript is a superset of JavaScript, in particular ES6.

ES6

One of the new features of ES6 are Classes. They are a simple sugar over the prototype-based OO pattern. Having a single convenient declarative form makes class patterns easier to use, and encourages interoperability. You can learn more in this article about ES6 features.

Typescript

TypeScript makes abstractions explicit, and a good design is all about well-defined interfaces. It is much easier to express the idea of an interface in a language that supports them. I won't go in detail here because much it's said in this post about typescript and angular 2. But as a conclusion let me tell you that TypeScript takes 95% of the usefulness of a good statically-typed language and brings it to the JavaScript ecosystem. You still feel like you write ES6: you keep using the same standard library, same third-party libraries, same idioms, and many of the same tools (e.g., Chrome dev tools). It gives you a lot without forcing you out of the JavaScript ecosystem.

New Ionic 2 Navigation

This may be the key point that differs more an angular 2 app. Ionic 2 doesn't use angular 2 routing system, instead they created their own navigation to better serve the use cases of mobile apps. They inspired themselves in the way ios handles navigation between pages, pulling a pushing views instead of a tree structure most commonly seen on the web.
At first I was reticent about this approach as I was used to the way ionic 1 worked with tree structures for the routing. After some months playing around with ionic 2 I must admit that this new navigation is much more easy and understandable than the one in ionic 1.
In ionic 1 you were reaching the limits of the framework everytime you wanted to implement a sophisticated navigation. I have created complex navigation structures in this app template and I don't get that feeling, in the contrary, complex navigations can be easily implemented with ionic 2 navigation system.

As a final note, in ionic 2, don't think about your pages as specific route paths, but rather views you can display anywhere at anytime in your app, for example, a ContactDetail page could be displayed through any number of user navigation flows. It could even link to itself infinitely.
In Ionic 1 you had to say "this page lives at this route" and new flows required new routes. Ionic 2 liberates you from strict paths -> pages.

Ahead of Time

Using this angular 2 technique you can radically improve performance by compiling Ahead of Time (AoT) during a build process.
An ionic application consist largely of components and their HTML templates. Before the browser can render the application, the components and templates must be converted to executable JavaScript by the Angular compiler.

When developing the app you can compile the app in the browser, at runtime, as the application loads, using the Just-in-Time (JiT) compiler. This is the standard development approach, it's great .. but it has shortcomings.
JiT compilation incurs a runtime performance penalty. Views take longer to render because of the in-browser compilation step. The application is bigger because it includes the Angular compiler and a lot of library code that the application won't actually need.
The Ahead-of-Time (AoT) compiler can catch template errors early and improve performance by compiling at build time.
If you want to learn more about AoT I would suggest you reading these posts here and here.

Further reading
Tutorials here!

KNOWN ISSUES

Solve 404 errors with the whitelist plugin
If you're using a newer version of Cordova (or the latest Ionic CLI) to develop your app, you may be experiencing http 404 errors when your app tries to make network requests.
This can be solved quickly with the Cordova whitelist plugin! Please read more here:
http://docs.ionic.io/docs/cordova-whitelist

Using Android version <4.4
Older versions of Android devices use Android's default browser, which has significantly less performance and standards compliance than modern Chrome. Using Crosswalk gives you a specific and more performant version of Chrome to use on all Android devices, in order to reduce fluctuations and fragmentation among devices.
Crosswalk is an open source project that allows you to specify a version of Chrome to use as your web browser in Android. The compiled app will have your code hosted inside of this Chrome webview.
Please read more here http://blog.ionic.io/crosswalk-comes-to-ionic/

Errors when running npm install on Windows
You should solve your errors by reading this links:

- http://stackoverflow.com/questions/21365714/nodejs-error-installing-with-npm/21366601#21366601
- http://forum.ionicframework.com/t/ionic-setup-sass-error/17843

USEFUL LINKS

Ionic Getting started guide
ionicframework.com/getting-started

Ionic Documentation
ionicframework.com/docs

Visit the Ionic Community Forum
forum.ionicframework.com

SUPPORT

If you are facing an issue related to Ionic please refer to: forum.ionicframework.com

If you found an error or a bug in this component please contact us. We are also happy to walk customers through the template structure and answer any support queries in that regard.

FAQS

After doing `$ionic serve` I'm getting the following error:

```
watch failed: "main.dev.ts" and "main.prod.ts" have been deprecated. Please
create a new file "main.ts" containing the content of "main.dev.ts", and
then delete the deprecated files. For more information, please see the
default Ionic project main.ts file here:
https://github.com/driftyco/ionic2-app-base/tree/master/src/app/main.ts
ionic-app-script task: "watch"
Error: "main.dev.ts" and "main.prod.ts" have been deprecated. Please create
a new file "main.ts" containing the content of "main.dev.ts", and then
delete the deprecated files. For more information, please see the default
Ionic project main.ts file here:
https://github.com/driftyco/ionic2-app-base/tree/master/src/app/main.ts
```

On 12/12/2016 Ionic pushed a new version of **ionic-app-scripts** and they introduced some breaking changes, but don't worry, it's very easy to fix this, it will take 1 min maximum.

To fix this do the following:
Delete src/app/main.dev.ts and src/app/main.prod.ts and create a file src/app/main.ts with the content from src/app/main.dev.ts

You can learn more here: https://github.com/driftyco/ionic-app-scripts/blob/master/CHANGELOG.md

What program do you use to edit the code?

Our preferred editors are:

1. Atom https://atom.io/
2. Sublime http://www.sublimetext.com/

Can I use Phonegap Build?

Yes, you can use PhoneGap Build in order to build this app. Ionic is a front-end framework built on top of Cordova so you can use PhoneGap Build in the same way you would use it with other phonegap/cordova application.

Adobe PhoneGap provides a way for users to create mobile applications using technologies such as HTML, CSS, and Javascript. https://build.phonegap.com

You can find the whole documentation for the config.xml here:
http://docs.phonegap.com/en/3.5.0/config_ref_index.md.html

RTL support

The HTML5 attribute dir=auto sets the directionality of the element according to the first characters with strong directionality.

The only way to make an element's directionality depend on its own content in this sense is to set the `dir` attribute to the value `auto` on the element itself. You cannot make this attribute inherited.

Remote Debugging on Android with Chrome

https://developer.chrome.com/devtools/docs/remote-debugging

Remote Debugging on iOS with Safari

http://www.jasenlew.com/2014/07/28/web-inspect-an-ionic-app-on-your-iphone-ios-6/

Older versions of Android devices (4.0-4.3)

Older versions of Android devices (4.0-4.3) use Android's default browser, which has significantly less performance and standards compliance than modern Chrome.
Using Crosswalk gives you a specific and more performant version of Chrome to use on all Android devices, in order to reduce fluctuations and fragmentation among devices.

Crosswalk is an open source project that allows you to specify a version of Chrome to use as your web browser in Android.
The compiled app will have your code hosted inside of this Chrome webview.

Please read more here http://blog.ionic.io/crosswalk-comes-to-ionic/

iOS 9 APP TRANSPORT SECURITY

iOS 9 introduces a new security feature that blocks non-HTTPS traffic in your app. However, this is a new feature that is only enabled for apps building with XCode 7 and iOS 9 SDK today.
To fix, edit the MyApp-Info.plist file and add the contents of this patch.
You can also fix this by adding this plugin into your app cordova-plugin-transport-security.

CHANGELOG

Version 1.0 - released 21 November 2016

First release