**1) Setting up your web app/website/HTML files**

**Option A) Use a URL for accessing your web application (= *store files online*)**

Create a mobile-optimized website, for example, using [jQuery Mobile](https://jquerymobile.com/) and upload your site to your webserver/webspace. WebViewGold supports all kinds of web apps/websites (including HTML, PHP, WordPress, Progressive Web Apps, HTML 5 Games, WiX, apprat.io, bubble.io, …). Please make sure to optimize your website to feel like an app. Please consider a professional User Experience/UX Review on [UXreviewer.io](https://www.uxreviewer.io/) or similar services for best Play Store approval results. Such a UX review also helps you to achieve better customer retention in general for your product with suggestions for improvement.

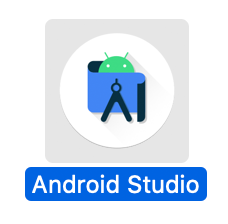
**Option B) Use a local HTML folder for accessing your web application (= *store files within the app*)**

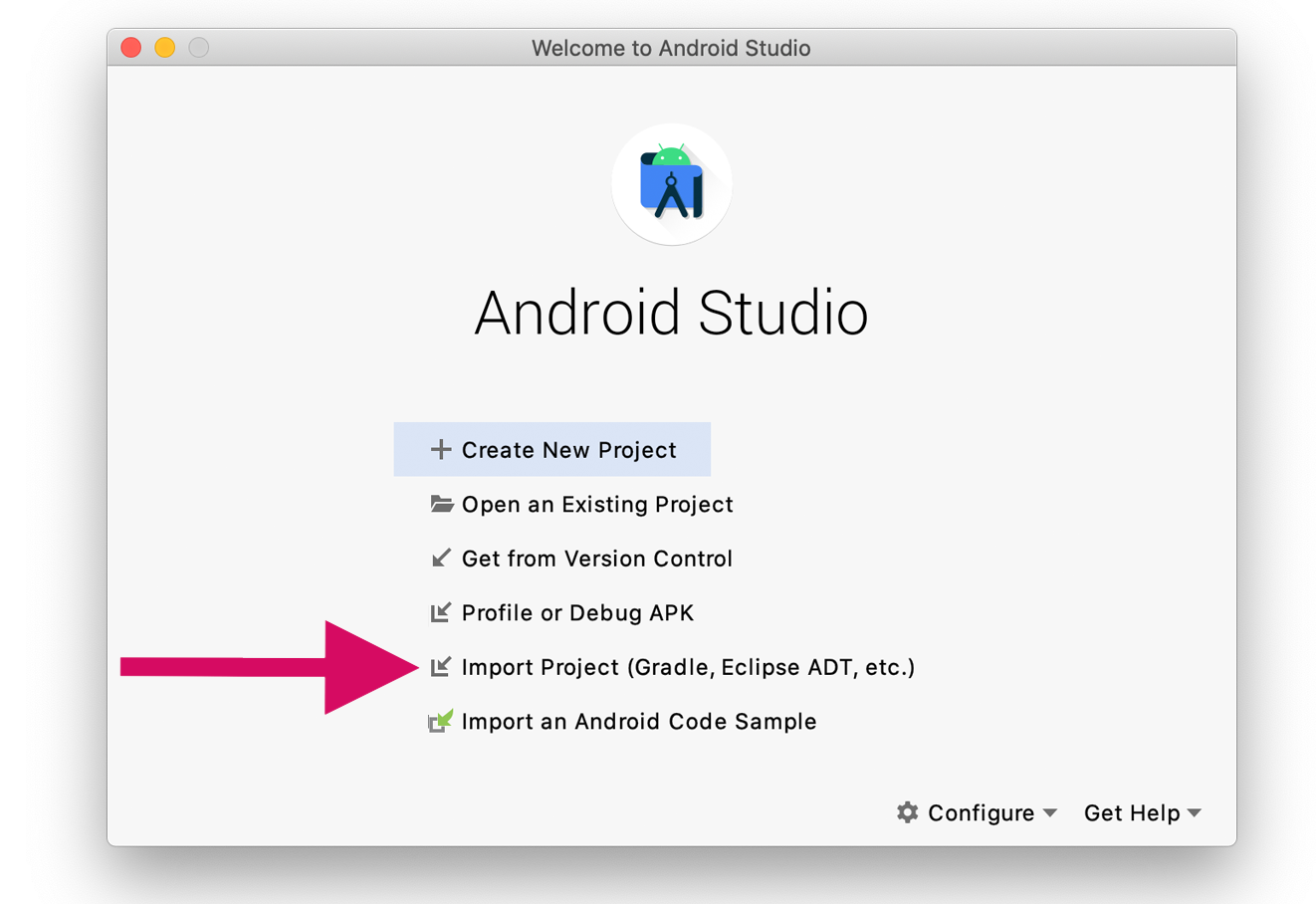
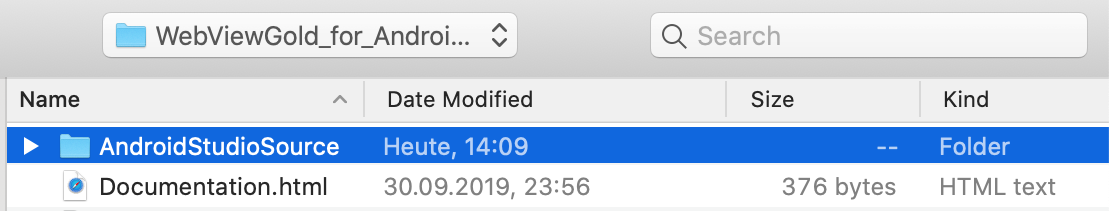
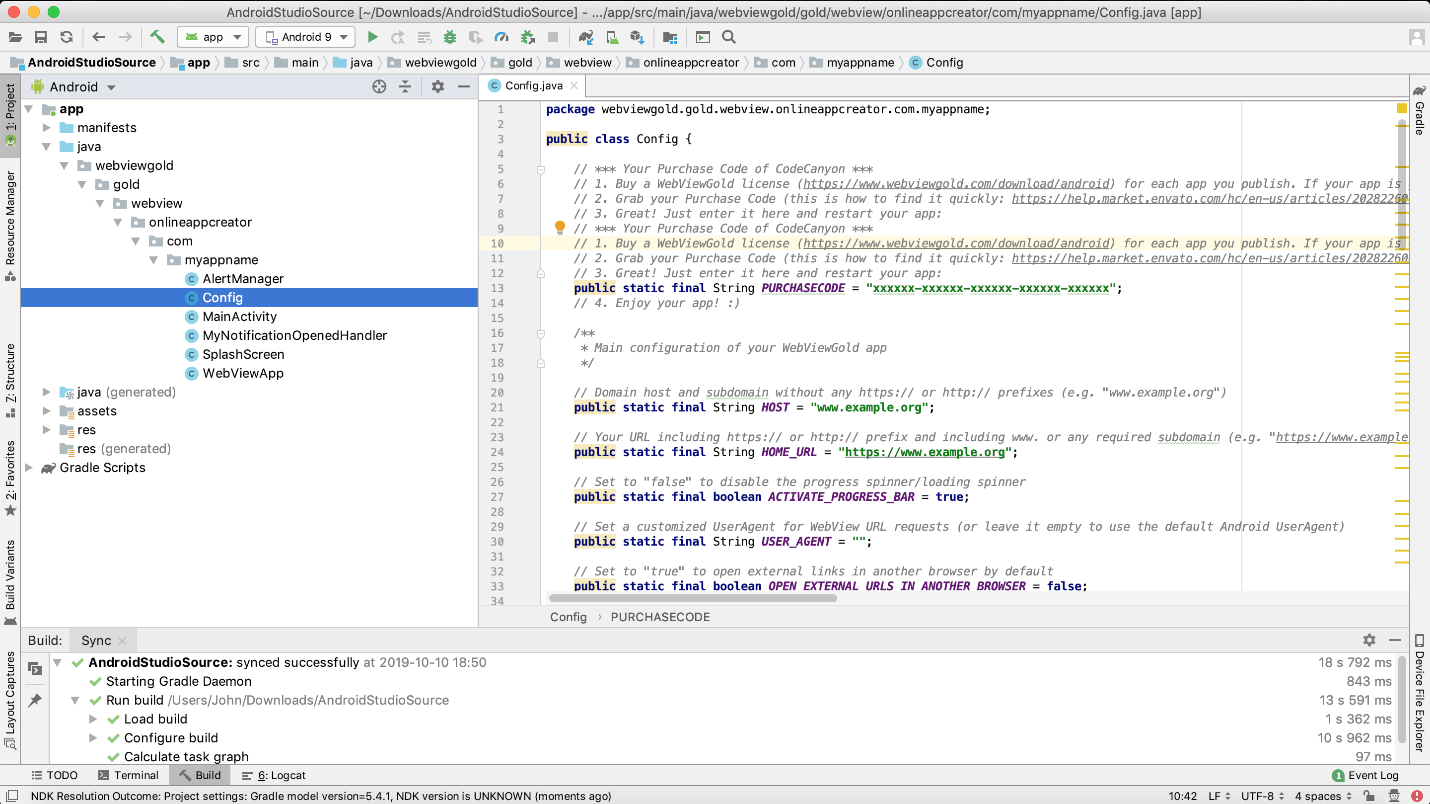
WebViewGold supports web apps bases on local HTML folders, too: Copy your HTML/CSS/JavaScript files to the Android Studio project and set **USE\_LOCAL\_HTML\_FOLDER** to **true** (see [next step](https://www.webviewgold.com/docs/android/?downloaded=true&update=no#setup)). Please consider a professional User Experience/UX Review on [UXreviewer.io](https://www.uxreviewer.io/) or similar services for best Play Store approval results. Such a UX review also helps you to achieve better customer retention in general for your product with suggestions for improvement.

**Option C) Use an HTML folder if the user is offline, and use a remote URL if the user is online (= *store files online + store backup/fallback files within the app*)**

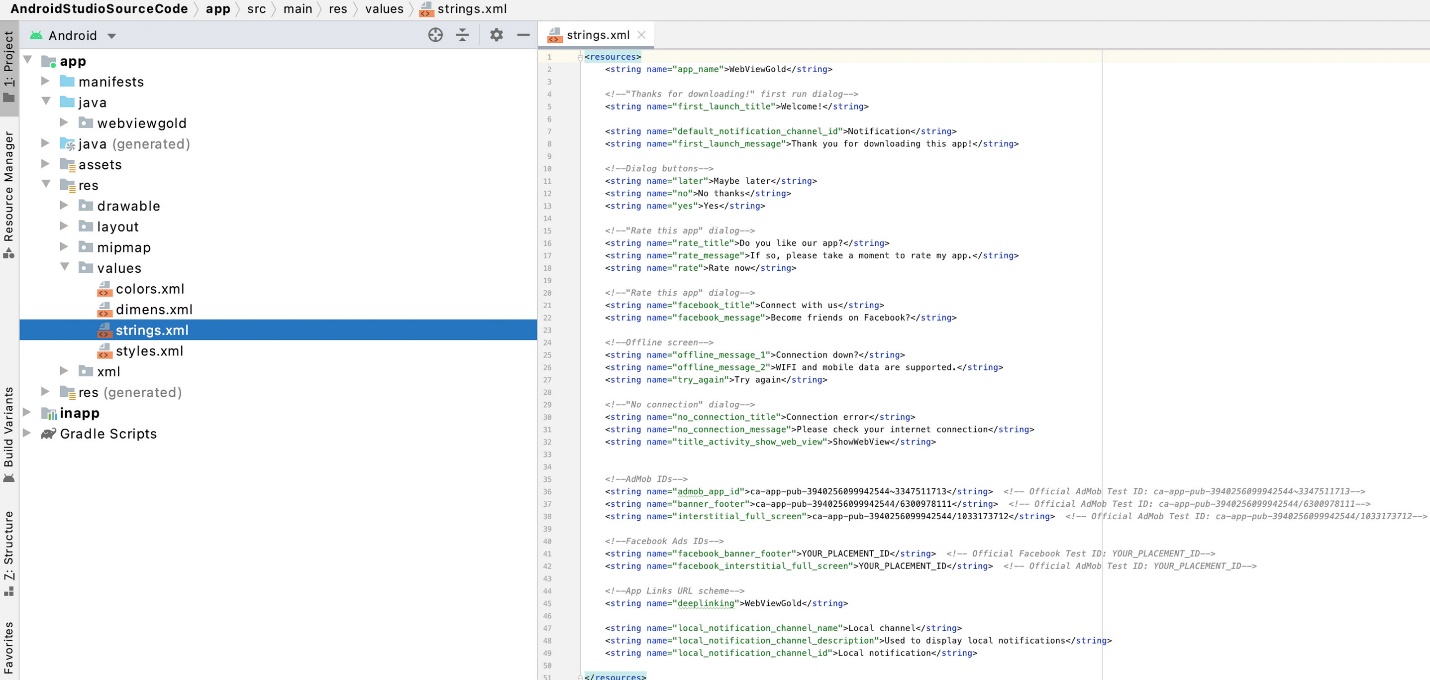
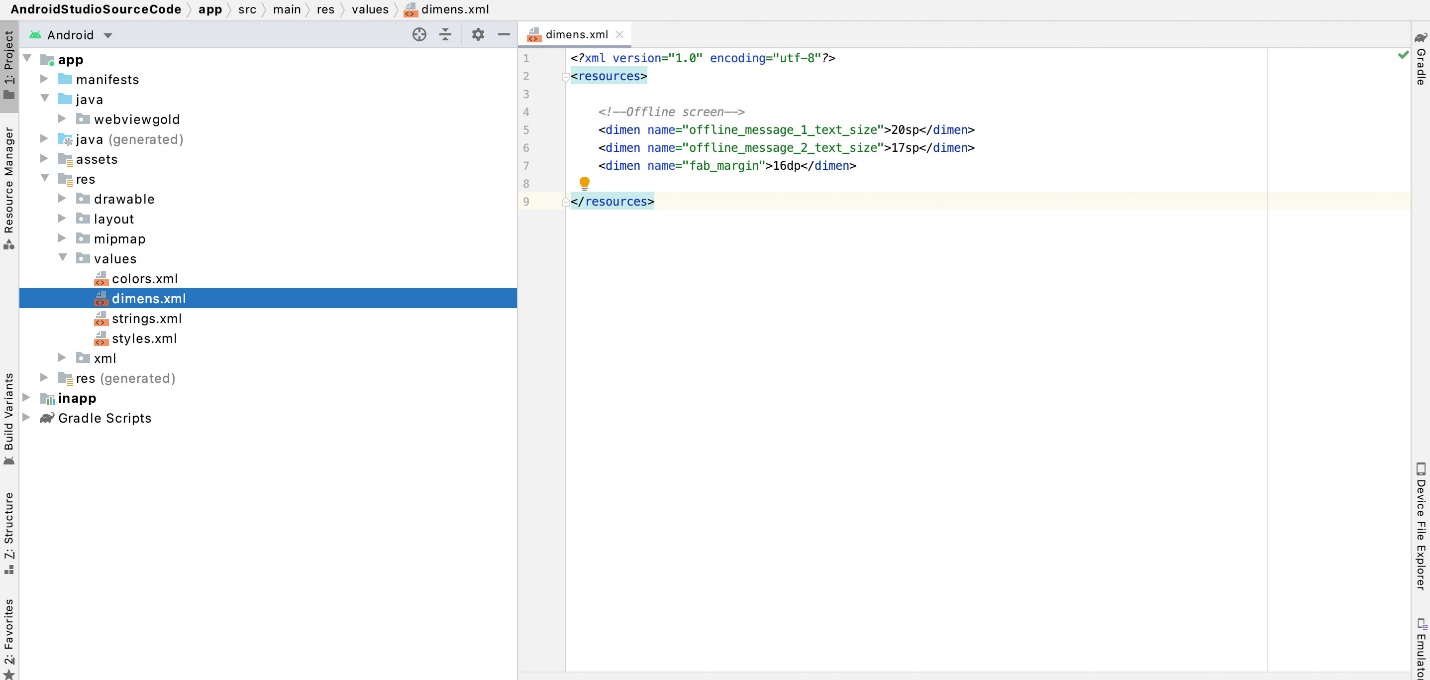
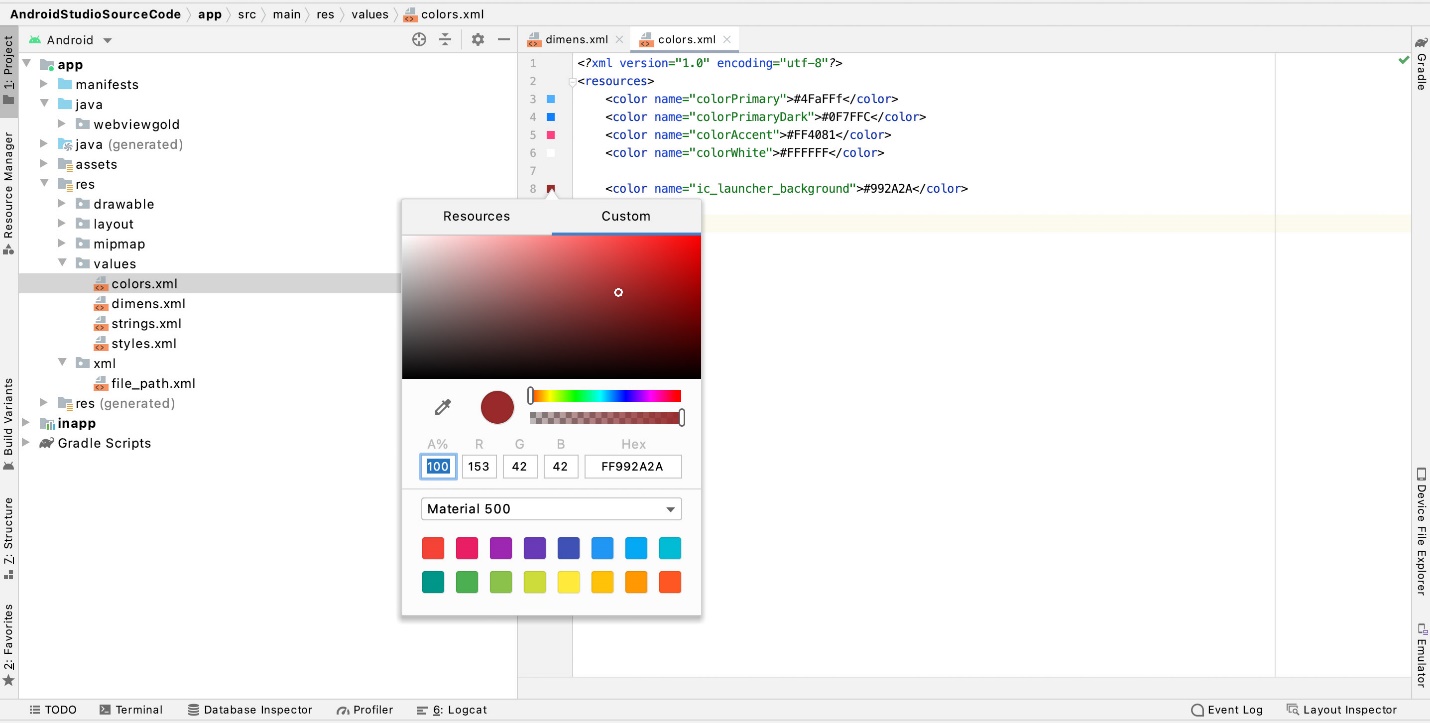
Copy your HTML/CSS/JavaScript files to the Android Studio project, set **USE\_LOCAL\_HTML\_FOLDER** to **false** (see [next step](https://www.webviewgold.com/docs/android/?downloaded=true&update=no#setup)) and set **FALLBACK\_USE\_LOCAL\_HTML\_FOLDER\_IF\_OFFLINE** to **true** (see [next step](https://www.webviewgold.com/docs/android/?downloaded=true&update=no#setup)). Please consider a professional User Experience/UX Review on [UXreviewer.io](https://www.uxreviewer.io/) or similar services for best Play Store approval results. Such a UX review also helps you to achieve better customer retention in general for your product with suggestions for improvement.

**2) Setting up WebViewGold for Android**

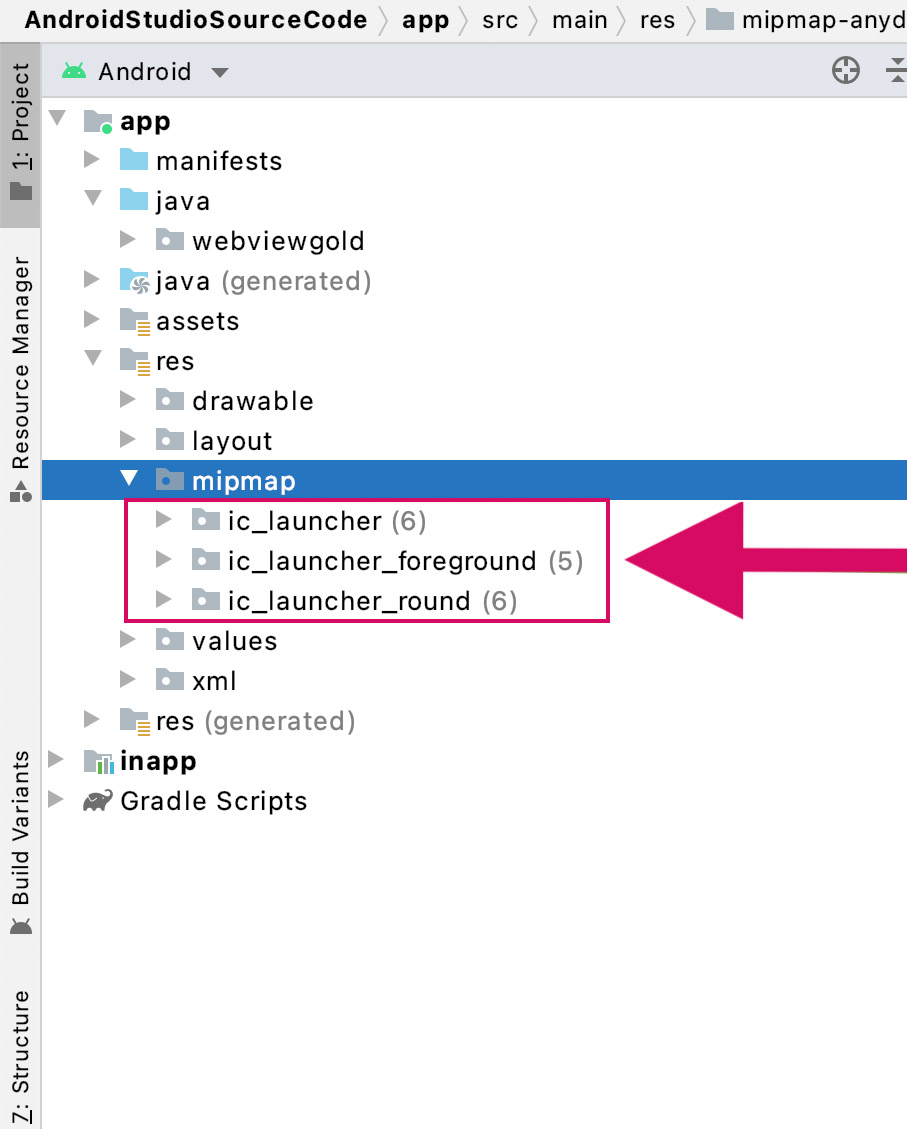
**Download** the WebViewGold app template for Android [**here**](https://www.webviewgold.com/download/android).  
  
Afterward, open **Android Studio**:  
  
[](https://www.webviewgold.com/docs/android/assets/img/androidstudio2020.png)

Choose **Import project**:  
  
[](https://www.webviewgold.com/docs/android/assets/img/androidstudio_welcome2020.png)  
  
Select your **AndroidStudioSource** folder and wait until the project loaded completely:  
  
[](https://www.webviewgold.com/docs/android/assets/img/filepicker.png)  
  
Afterward, open **Config.java** file to edit main configuration details (URL, URL Handling, UserAgent, dialogs). Please take a look at our [YouTube channel](https://www.youtube.com/channel/UChuOoCMcXP_8SQPbZyXTPjA?sub_confirmation=1) for specific setup tutorials. Please don't forget to add your [Envato/CodeCanyon Purchase Code](https://help.market.envato.com/hc/en-us/articles/202822600-Where-Is-My-Purchase-Code-) in the **PURCHASECODE** field for each app published by you. One license per each customized end product is required. Please take a look for license info of your CodeCanyon purchase: [https://codecanyon.net/licenses/standard](https://codecanyon.net/licenses/standard?ref=onlineappcreator)  
The main configuration document **Config.java** looks like this:   
  
[](https://www.webviewgold.com/docs/android/assets/img/config.png)

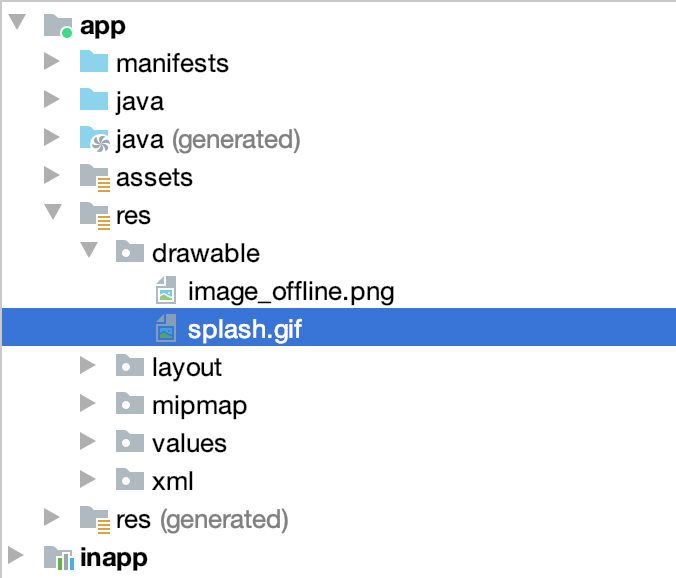
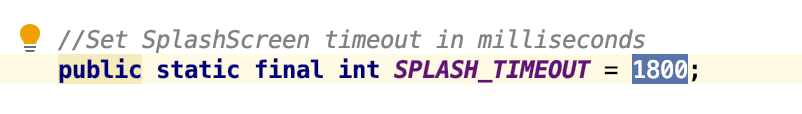
Setting up apps with WebViewGold is **very easy** and does not require any coding knowledge. Please take a look at this **overview** of the built-in configuration options in **Config.java** file:

Open **strings.xml** file to edit app name & text strings:  
  
[](https://www.webviewgold.com/docs/android/assets/img/screenshot_Strings.jpg)  
  
Open **dimens.xml** file to modify text formatting:  
  
[](https://www.webviewgold.com/docs/android/assets/img/screenshot_Dimens.jpg)  
  
  
Open **colors.xml** file to edit colors of the status bar, splash screen, and loading sign/loading indicator:  
  
[](https://www.webviewgold.com/docs/android/assets/img/screenshot_colors.jpg)

**3) Change the App Icon**

[](https://www.webviewgold.com/docs/android/assets/img/screenshot_Icon.jpg)  
A beautiful app icon is a perfect first impression for your customers. It's easy to add your app icon to your WebViewGold project: Make a right-click on each ic\_launcher.png size (from hdpi to xxxhdpi) and click "Reveal in Finder" (Mac) or "Reveal in Explorer" (Windows). Replace the file with your icon. Alternatively, use this helpful tool: [**Launcher Icon Generator**](https://jgilfelt.github.io/AndroidAssetStudio/icons-launcher.html) (we are not affiliated with the developer company).  
  
[](https://www.webviewgold.com/docs/android/assets/img/roundvsboxicon.png)  
On devices where square icons are used, the icon of "ic\_launcher.png" is displayed; on devices with round icons, either "ic\_launcher\_round.png" or "ic\_launcher\_foreground.png". These three icon files must be placed in the folders "mipmap-hdpi", "mipmap-xhdpi", "mipmap-xxhdpi" and "mipmap-xxxhdpi".

**4) Change Splash Screen**

Would you like to make your app more attractive to your customers and display your branding as a welcome image? Sure, add your individual splash screen graphics. Just replace the included **splash.gif** files with your own image file. Make sure to use GIF format (animated or static) and 128x128 Pixels size.   
[](https://www.webviewgold.com/docs/android/assets/img/splashgif.png)  
You can change the timeout in **Config.java** file:   
[](https://www.webviewgold.com/docs/android/assets/img/splash_timeout.png)

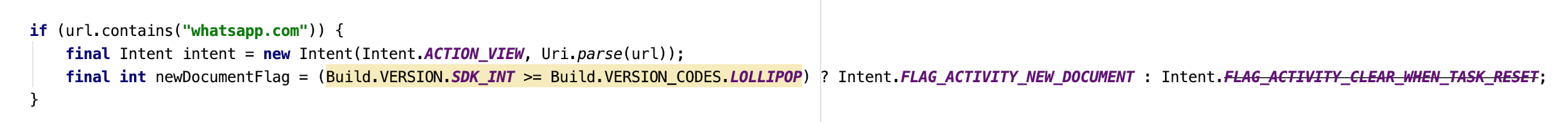
**URL Handling – Open links in System Browser**

It often makes sense in WebView based apps to treat different web addresses differently. If you place a social media button in your web app, a click on it should probably not open the social media page in your app, but in the right app (e.g., the system browser). It's good that WebViewGold has all the features to enable powerful configuration and treat links the way you want it.

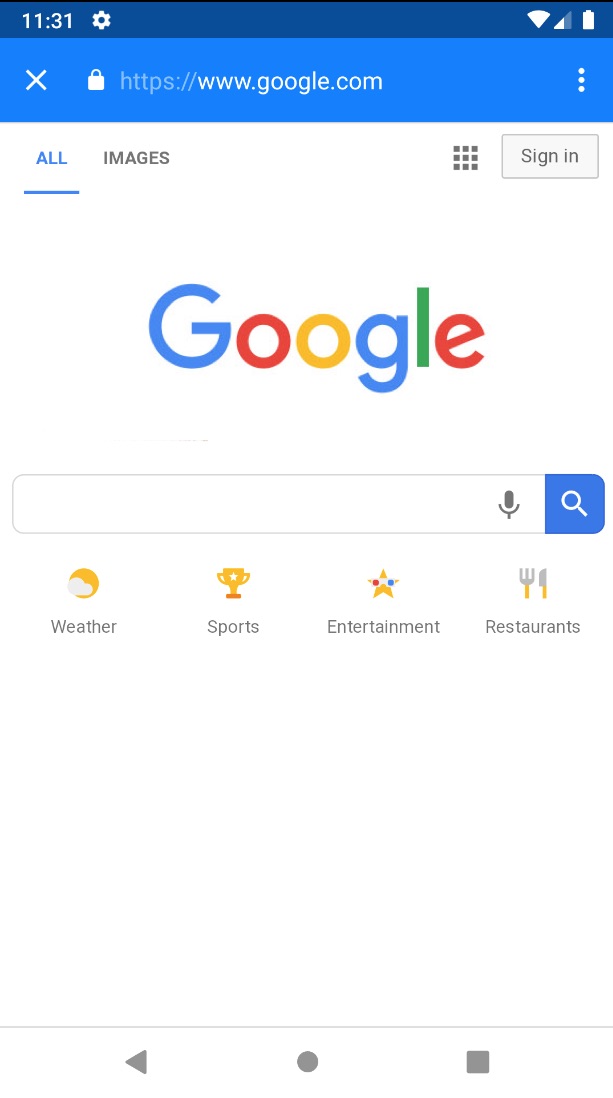
Option 1: Open all external hosts in System Browser

If you want to **open every URL, which is not on your host** in System Browser, please set **OPEN\_EXTERNAL\_URLS\_IN\_ANOTHER\_BROWSER** (in Config.java file) to **true**.

Option 2: Open only specific hosts in System Browser

Please search for this snippet (CMD+F) in MainActivity.java:  
  
[](https://www.webviewgold.com/docs/android/assets/img/externalhost.png)  
  
To add another domain, just copy this snippet and insert another host, which should be opened in System Browser. Please enter the host precisely the way you link to it.

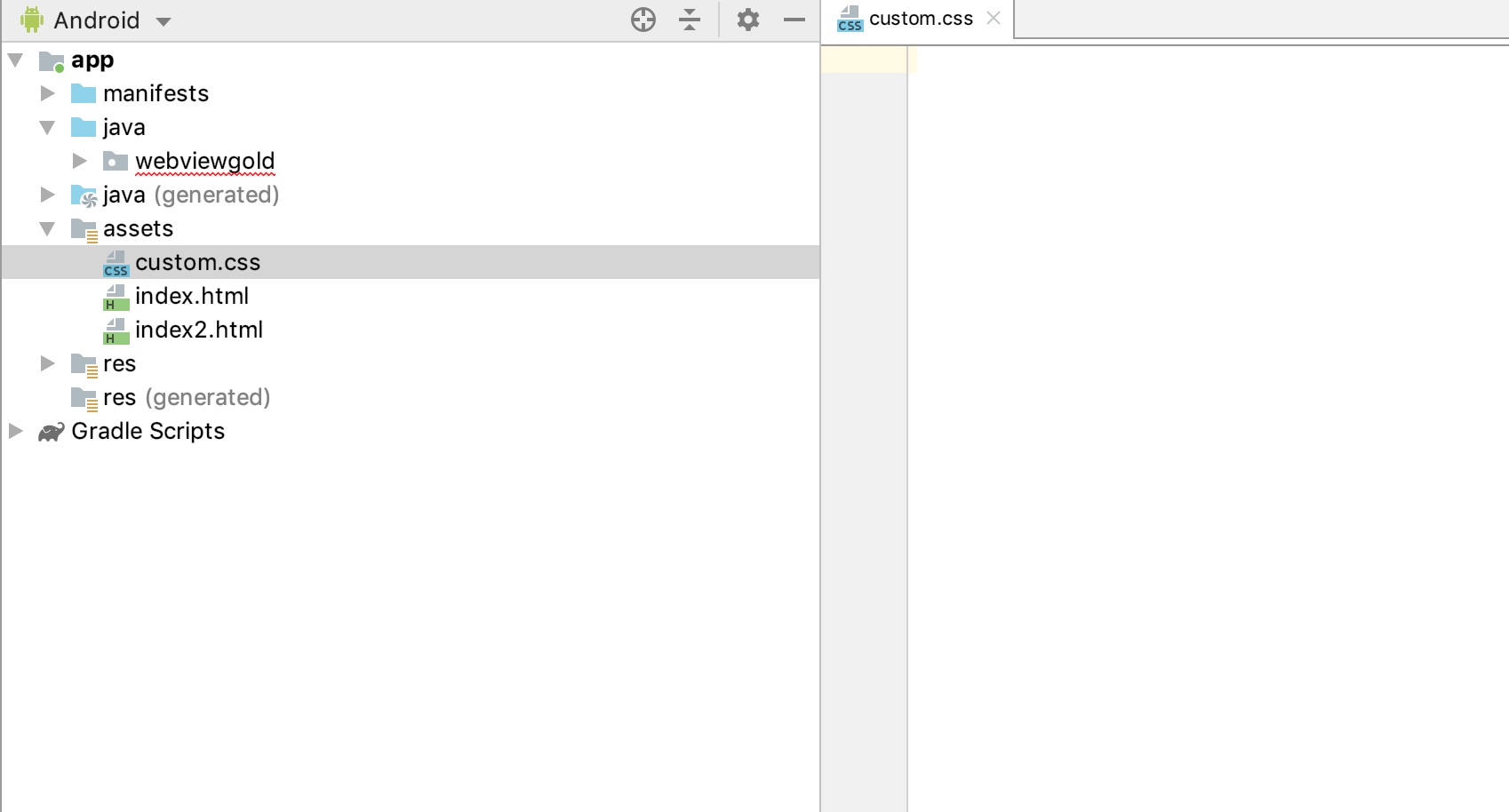
**URL Handling – Open links in a tab/additional In-App-Browser**

There are some cases for links that do not feel to be "important enough" to open them in the System browser externally but "too specific" for opening them in the app WebView itself (e.g., your Terms and Conditions page). For this particular kind of links, we added a creative idea of URL Handling: a tab that opens an additional in-app-browser over the main WebView content itself. It will look like this:  
  
[](https://www.webviewgold.com/docs/android/assets/img/tabs.jpg)  
  
As you see this in-app-browser includes an address bar and navigation buttons. That's a considerable difference to the primary, full-screen WebView content. So this kind of combination of app and browser experience is great for tutorials, knowledge databases, and all type of content that requires any type of navigation. To use this kind of in-app-browser, link to an URL in this format:

<a href='#' onClick="top.location.href='https://www.google.com';">Open Google</a>

Please insert <https://www.webviewgold.com/demo/testlinks.php> as WebView URL to test how your app will handle different types of links. You can open the same domain <https://www.webviewgold.com/demo/testlinks.php> in your desktop browser and see the source code to copy & paste the way of link.

**Custom CSS API**

The Custom CSS feature of WebViewGold allows you to customize the appearance of your web application without the need to customize the CSS style sheets of your web application itself. The custom CSS classes will be applied to all website loads requested by your application built with WebViewGold. Just insert the custom CSS in the file **custom.css** and save the changes:  
  
[](https://www.webviewgold.com/docs/android/assets/img/custom_css.jpg)

**HTML5 Geolocation API**

The HTML5 Geolocation feature enables users to communicate their own position to a web application. Check out the [Mozilla guide](https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API) on how to implement geolocation into your website or web app. In WebViewGold, the HTML5 Geolocation API will work by default after you integrated it into your web content. Demo: <https://www.w3schools.com/html/html5_geolocation.asp>

**Push Notifications API via OneSignal, Firebase, and JavaScript**

Push notifications are notifications that appear on your smartphone without opening the app. These types of messages allow fantastic interaction with customers and users of the app. Your app does not have to be open. This way, your customers do not miss any important news or messages. Push Notifications are suitable for many cases, e.g., for updates to orders, live tickers, or social media community. There must be an internet connection to allow app users to receive push messages. WebViewGold supports multiple ways of reaching your users via push notifications.

**Option 1:** Use Local Push Notification JavaScript API

You can insert a piece of code (JavaScript) in your website, to define a push notification, which can be shown **after** your app was closed (while scheduling it during the app is open). This works without OneSignal configuration at all.

<script>

var seconds = 120; //seconds from now on

var message = "It is Happy Hour now! Check it out!"; //the message

var button = "Open App"; //the text of the button

window.location.href = "sendlocalpushmsg://push.send?s="+ seconds +"=msg!"+ message +"&!#"+ button +"";

</script>

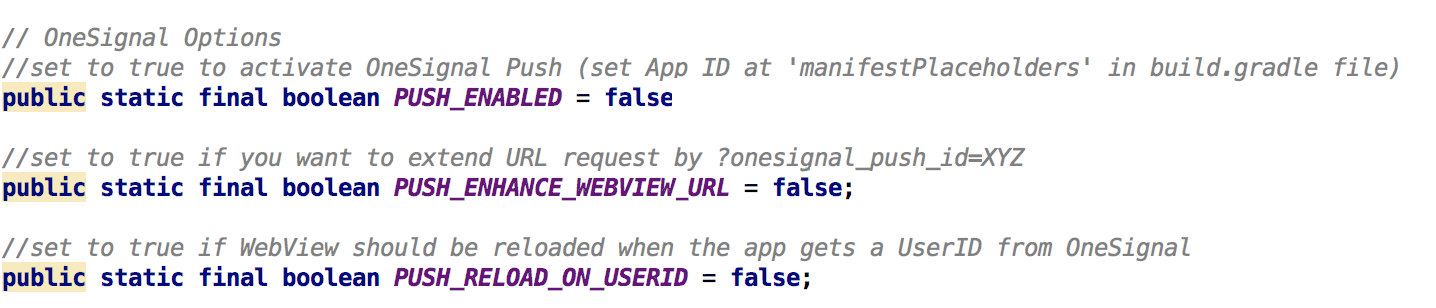
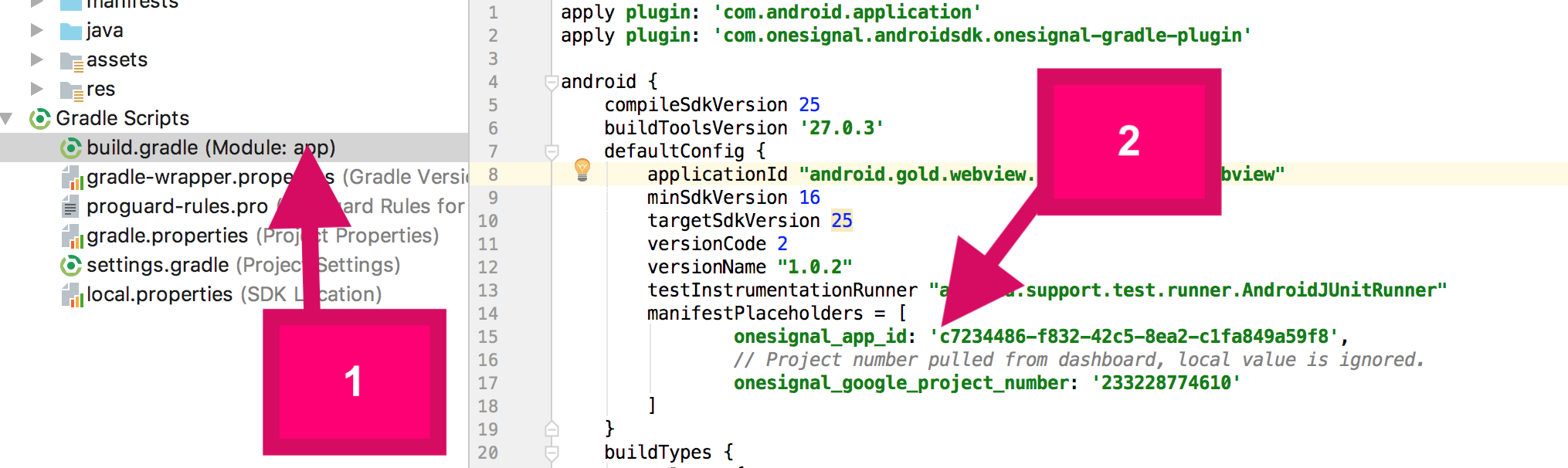
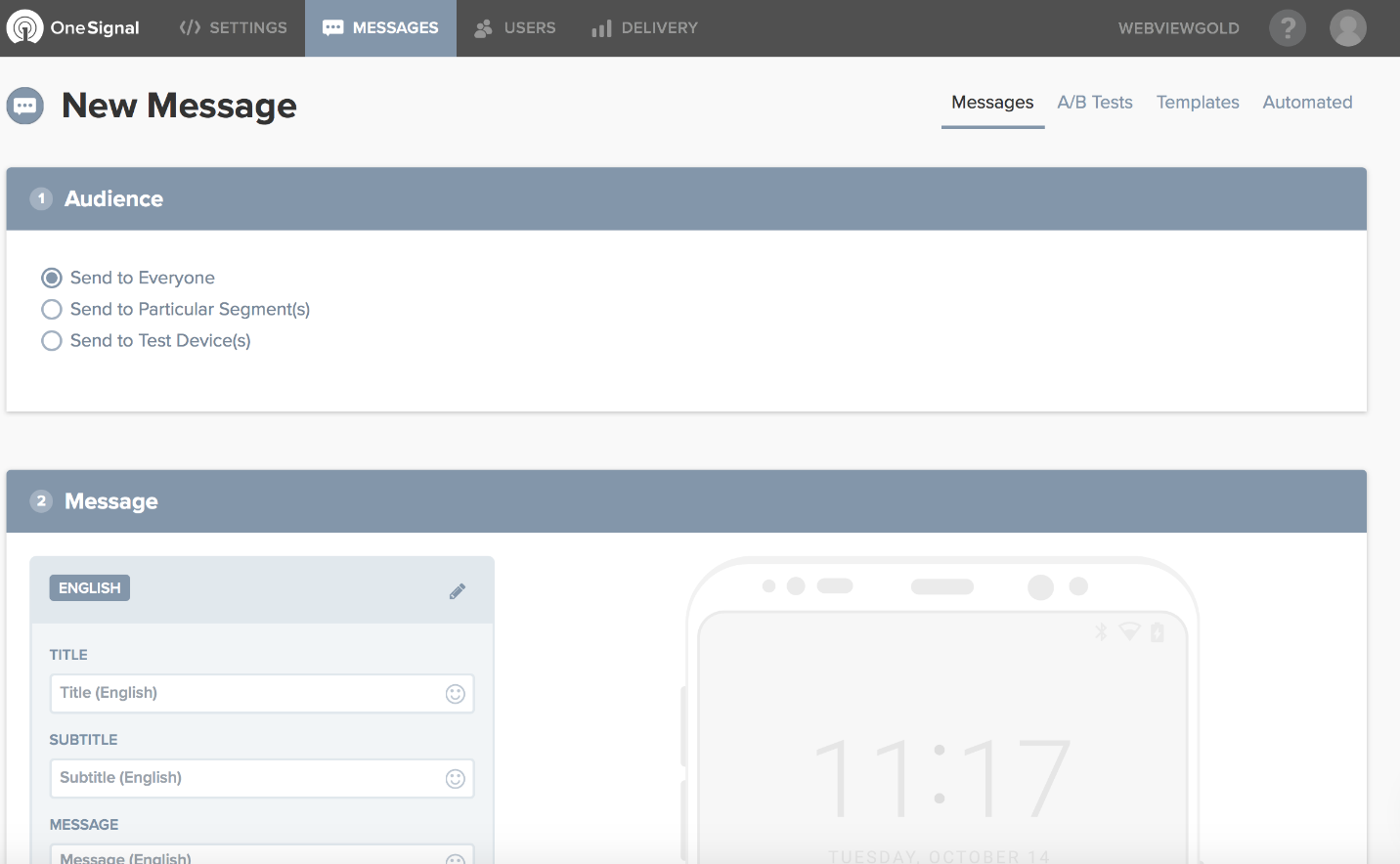
It could be used for different situations (countdowns, happy hours (like in the example code), reminders, ...).  
To cancel all scheduled local push notifcations run:

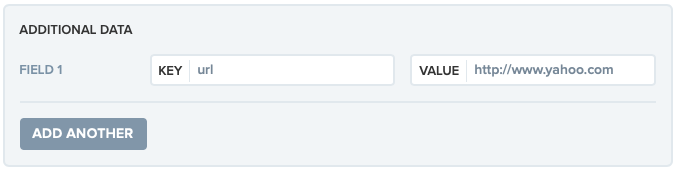
<script>

window.location.href = "sendlocalpushmsg://push.send.cancel";

</script>

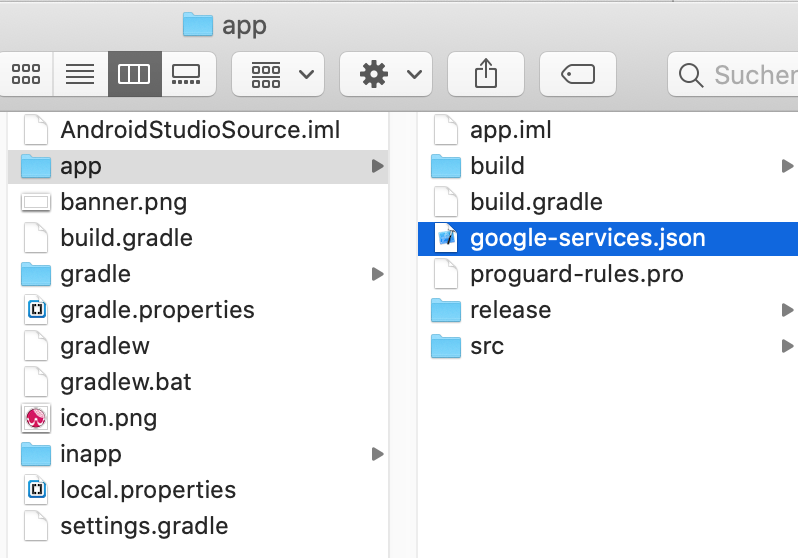
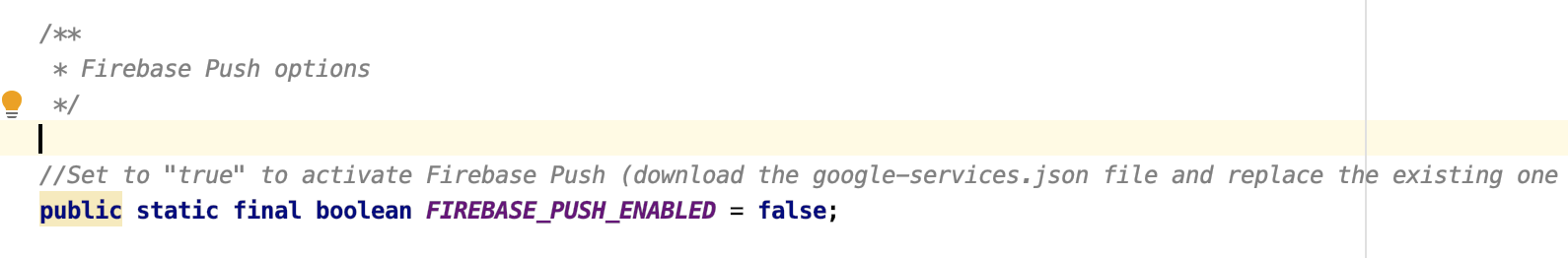
**Option 2:** Use OneSignal Remote Push Notifications

If you want to use [OneSignal](https://www.onesignal.com/) in your app, activate these options in **Config.java** (by switching the values from **false** to **true**):  
[](https://www.webviewgold.com/docs/android/assets/img/onesignal_config.png)  
  
Please set your OneSignal App ID in **build.gradle** file afterward:  
  
  
[](https://www.webviewgold.com/docs/android/assets/img/onesignal_appid.png)  
  
  
[Set up](https://documentation.onesignal.com/docs/generate-a-google-server-api-key) the OneSignal service for your Android app. Yay, you are **done**. You can send push notifications now:  
  
[](https://www.onesignal.com/)

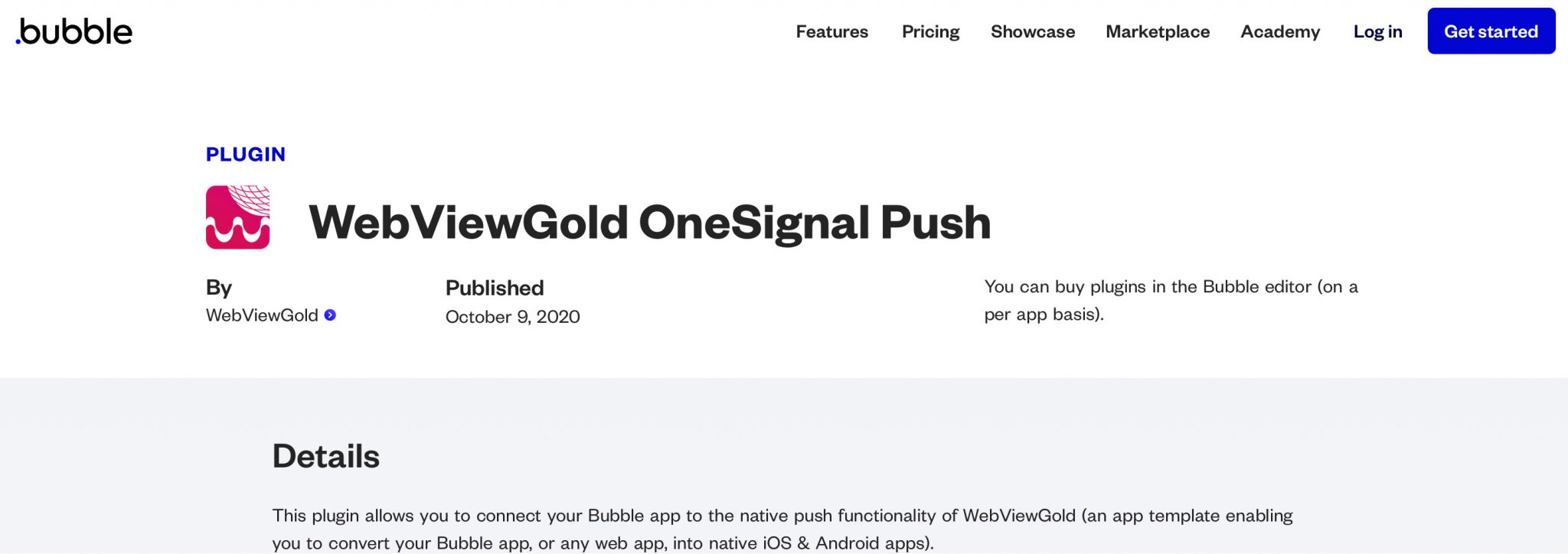
If you want to use OneSignal URL Deeplinking (opens a specific URL when clicking on your notification), please add the desired URL with a custom field called **url**:  
  
[](https://www.webviewgold.com/docs/android/assets/img/onesignal_deeplinking.png)

On the OneSignal.com Backend, you can fill in this key-value-pair window after clicking on "Advanced settings". Using the OneSignal API, you can add the "url" field to the API request (<https://documentation.onesignal.com/docs/onesignal-api>).  
  
Do you want to get **each individual user ID**on your server for further processing and **individual user** push messages? Just activate the **PUSH\_ENHANCDE\_WEBVIEW\_URL** option in Confing.java (by switching the value from false to true) in order to append ?onesignal\_push\_id=XYZ to your WebView URL. If your WebView URL is https://www.example.org, WebViewGold will call https://www.example.org?onesignal\_push\_id=XYZ instead. Only your FIRST URL request will get that GET variable, so save it in a session or in a cookie to access it on your linked pages.

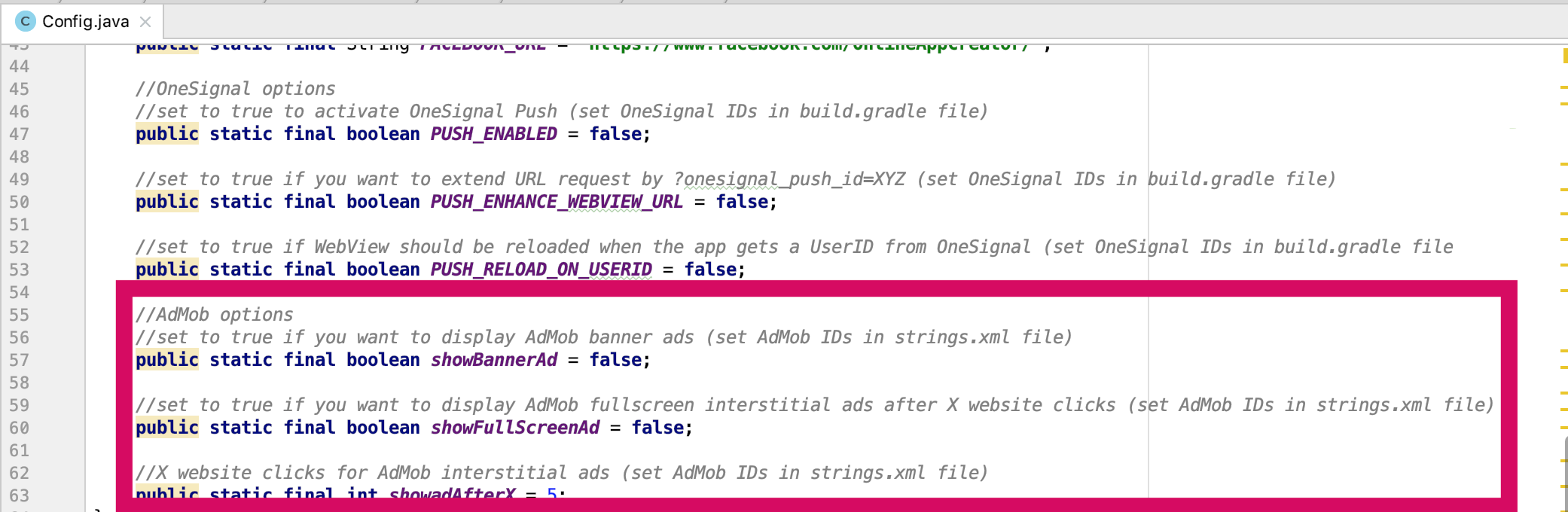
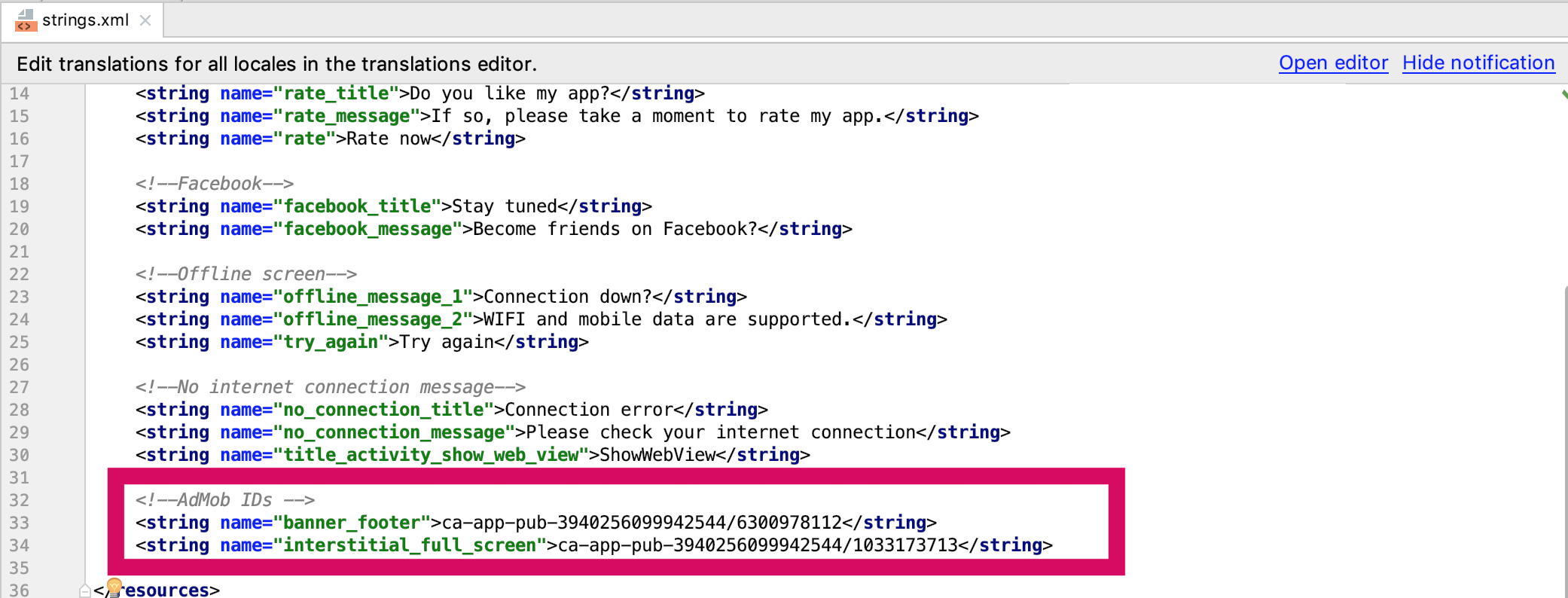
**Option 3:** Use Firebase Remote Push Notifications

**1. Create a Firebase project**Before you can add Firebase to your WebViewGold app, you have to create a Firebase project in your [Firebase Console](https://console.firebase.google.com/).  
  
**2. Register your app with Firebase**After you created your Firebase project, you can add your WebViewGold app to it.  
  
a. In the center of the Firebase console's project overview page, click the Android icon to launch the setup workflow.  
  
b. If you've already added an app to your Firebase project, click **Add app** to display the platform options. Enter your app's package name as changed/defined [in this step](https://www.webviewgold.com/docs/android/?downloaded=true&update=no#publish). Make sure that you enter the package name that your app is using. You cannot add or modify this value after you register your app with your Firebase project.  
  
**4. Click the *Register app* button.  
  
5. Download google-services.json**Click Download google-services.json to obtain your Firebase Android config file (google-services.json) and replace the sample file already included in the WebViewGold project buy using the Mac Finder or the Windows Explorer:  
  
[](https://www.webviewgold.com/docs/android/assets/img/firebase1.png)  
  
**6. Activate Firebase in WebViewGold**Go to Config.java file and turn the **FIREBASE\_PUSH\_ENABLED** variable to **true**.  
  
[](https://www.webviewgold.com/docs/android/assets/img/firebase2.png)  
  
**7. That's it!**  
Do you want to get **each individual user ID** on your server for further processing and **individual user** push messages? Just activate the **FIREBASE\_PUSH\_ENHANCE\_WEBVIEW\_URL** option in Confing.java (by switching the value from false to true) in order to append ?firebase\_push\_id=XYZ to your WebView URL. If your WebView URL is https://www.example.org, WebViewGold will call https://www.example.org?firebase\_push\_id=XYZ instead. Only your FIRST URL request will get that GET variable, so save it in a session or in a cookie to access it on your linked pages.

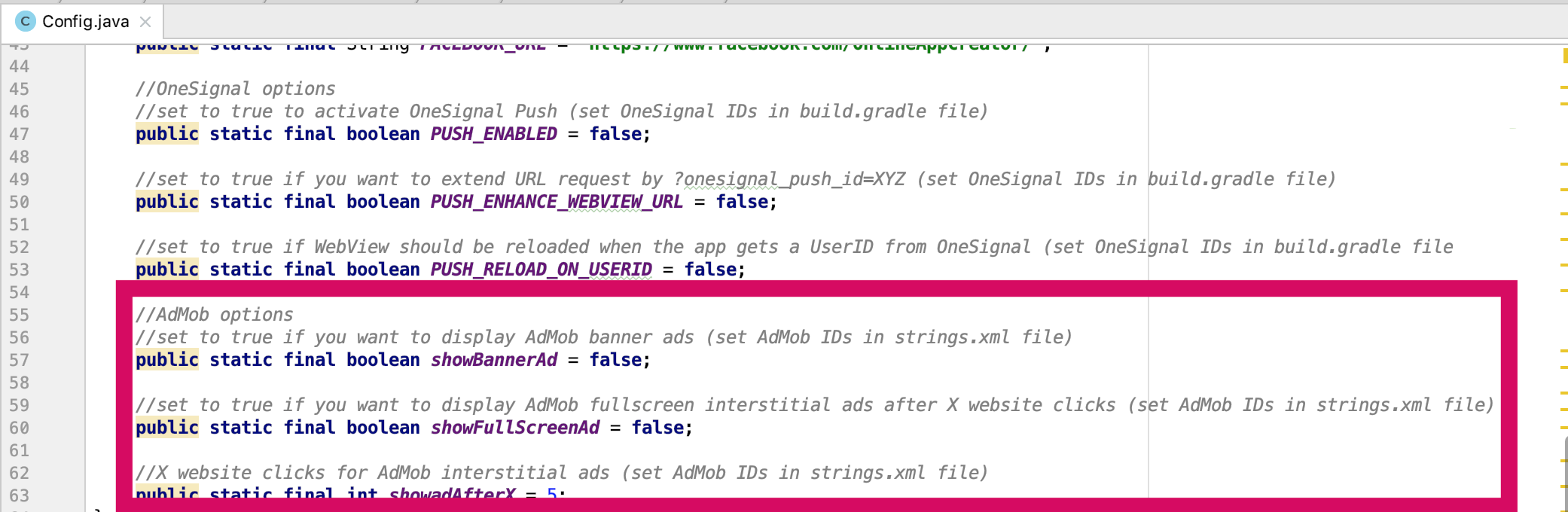
### Option 4: Use bubble.io Push Notifications

1. Create a **bubble.io app**  
2. Install the [**WebViewGold OneSignal Push**](https://bubble.io/plugin/webviewgold-onesignal-push-1602240916803x713805488222896100) plugin:   
[](https://bubble.io/plugin/webviewgold-onesignal-push-1602240916803x713805488222896100)  
3. Follow the **OneSignal instructions (see above)** and the **plugin instructions (see** [**here**](https://bubble.io/plugin/webviewgold-onesignal-push-1602240916803x713805488222896100) **and** [**here**](https://www.youtube.com/watch?v=C0HBXlMSK2E)**)**.

**Banner and Interstitial ads via AdMob API**

AdMob offers the possibility of monetizing mobile projects. It is a Google-provided service for integrating Google ads into mobile applications. The AdMob program allows mobile application developers to incorporate ads on iOS and Android platforms. The ad space provided by the app developer is paid by clicks (CPC, Cost Per Click) or impressions (CPM, Cost Per Mille). WebViewGold is ready for activating AdMob ads by default, and it allows you to monetize your WebView-based applications. Make sure to buy an [Extended License](https://www.webviewgold.com/docs/android/?downloaded=true&update=no#credits) of WebViewGold if you want to make money with apps based on WebViewGold.  
  
Open **Config.java** file:  
  
[](https://www.webviewgold.com/docs/android/assets/img/admob1.png)  
  
You can activate and deactivate banner and/or interstitial ads and define the interval of appearing interstitial ads. Please set your AdMob IDs in the **strings.xml** file before launching your app:  
  
[](https://www.webviewgold.com/docs/android/assets/img/admob2.png)  
  
For more information, please consider the [official AdMob SDK documentation](https://developers.google.com/admob/) as well (but skip the setup parts as WebViewGold already includes the framework).

**Banner and Interstitial ads via Facebook Audience Network Ads API**

The Facebook Audience Network offers the possibility of monetizing mobile projects. It is a Facebook-provided service for integrating Facebook ads into mobile applications. The Audience Network program allows mobile application developers to incorporate ads on iOS and Android platforms. The ad space provided by the app developer is paid by clicks (CPC, Cost Per Click) or impressions (CPM, Cost Per Mille). WebViewGold is ready for activating Facebook Audience Network ads by default, and it allows you to monetize your WebView-based applications.  
  
Open **Config.java** file and activate these 3 variables:  
  
[](https://www.webviewgold.com/docs/android/assets/img/admob1.png)  
  
Moreover, set **USE\_FACEBOOK\_ADS** to **true**. In the last setup step, please set your Facebook Ad IDs in the **strings.xml** file before launching your app.

**Image, Music, And File Downloader API**

If you offer to download or view documents or other downloads in your web app, our WebViewGold Download API will help you for the tremendous and native user experience of your app.

Save **images** to the camera roll of the user by calling links with savethisimage://?url= prefix:

<a href="savethisimage://?url=IMAGE URL HERE">Link</a>

Save **PDF files** by calling links with .pdf suffix:

<a href="https://www.example.org/file.pdf">Link</a>

Save **Music files** by calling links with .mp3, .mp4 or .wav suffix:

<a href="https://www.example.org/file.mp3">Link mp3</a>

<a href="https://www.example.org/file.mp4">Link mp4</a>

<a href="https://www.example.org/file.wav">Link wav</a>

You can **add more file formats** that should trigger the file downloader functionality by adding them to the *downloadableExtension* list in the *Config.java* file.

**Reset App API**

Sometimes only a restart helps. This is true in life as well as in digital life. Click on this link to reset your users' app to app factory settings (by clearing all cookies and cache):

<a href="reset://">Link</a>

Insert the <https://www.onlineappcreator.com/reset_test.html> URL into your WebViewGold app for a demo.

**Show/hide the loading sign manually**

The native Android loading indicator of WebViewGold will be triggered **automatically** when a page starts or finishes loading. However it could be useful to know that it **can be triggered manually** as well (e.g., you could show the activity indicator as well when waiting in your live chat queue, or some cloud server action is running in the background, and you want to visualize the process on the client):

<a href="spinneron://">Show the loading spinner</a>

<a href="spinneroff://">Hide the loading spinner</a>

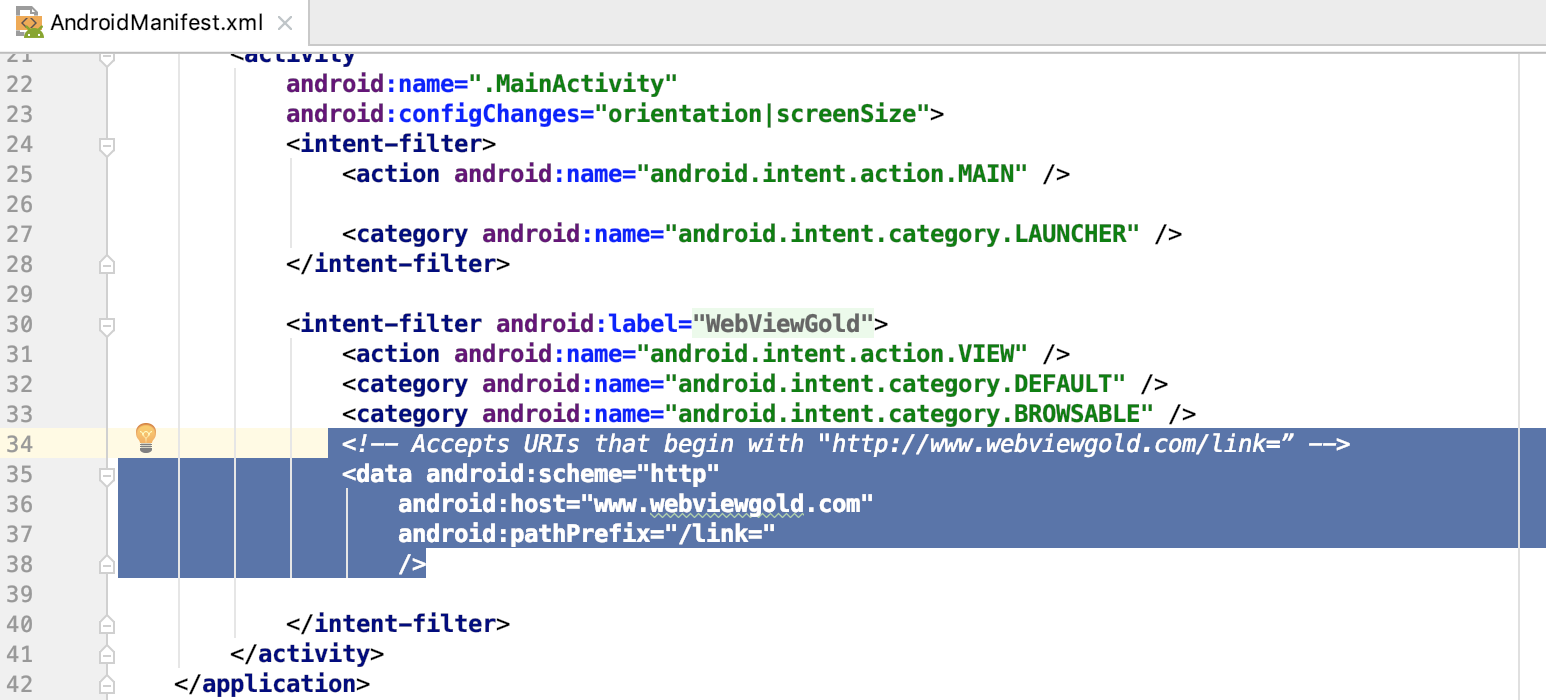
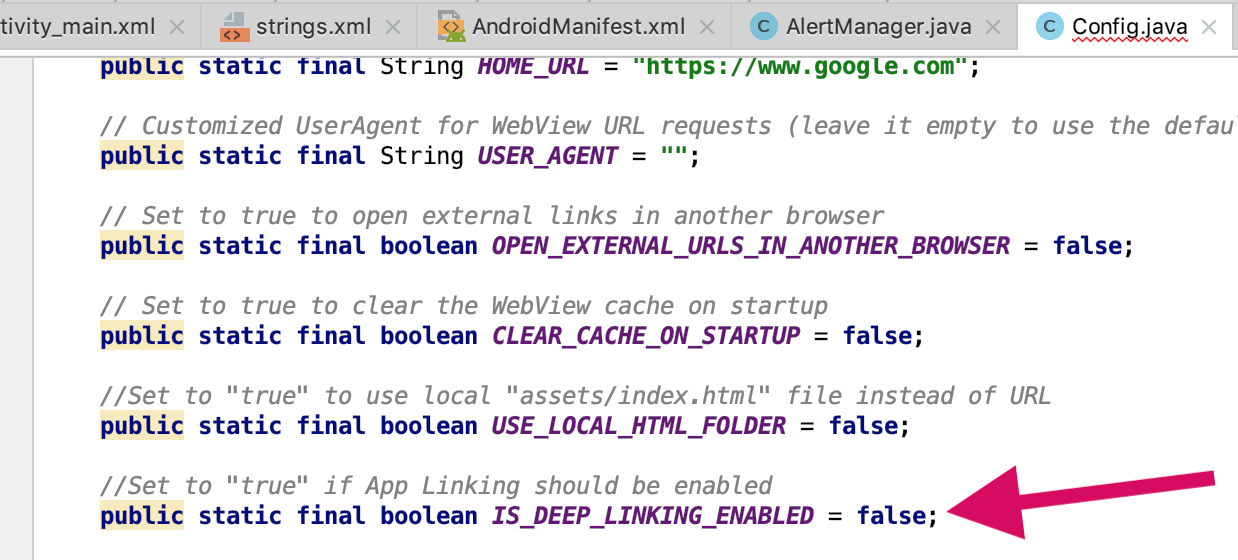
**UUID API**

WebViewGold allows passing a **non-personal, unique device ID** (UUID / "Universally Unique Identifier") to JavaScript, which **can be used to identify an Android device uniquely**. This can be useful, for example, to save settings or values on the server-side and restore them on the client-side. Please respect local and international **privacy regulations** if you use this API.

<a href="get-uuid://">Get UDID</a>

<a href="#" onClick="alert(uuid);">Show UDID</a>

**App Links API (take users to a specific page directly in your app)**

WebViewGold allows configuring a link scheme like www.webviewgold.com/link= which can be triggered from Mail app, browser apps, ... and opens the attached link in the WebView app. Please change the scheme to your own app name (like www.example.org/link= instead of www.webviewgold.com/link=) in **AndroidManifest.xml**:  
  
[](https://www.webviewgold.com/docs/android/assets/img/applinks1.png)  
  
Afterward, activate the **IS\_DEEP\_LINKING\_ENABLED** option in **Config.java** file:  
  
[](https://www.webviewgold.com/docs/android/assets/img/applinks2.png)  
  
Last but not least, you must verify your domain ownership: [**Go to the Google guide**](https://developer.android.com/training/app-links/verify-site-associations)   
**Done! :)** Example format:

http://www.webviewgold.com/link=https://www.google.com

(this link would open www.google.com in WebView app).

Do you want to test the mechanism? In the simulator, it can be challenging to check App Links. Open the *Terminal* app (Mac) or *Command Prompt* (Windows) and paste:

adb shell at startup -a android.intent.action.VIEW -d "https://www.webviewgold.com/link=https://www.example.org" webviewgold.myappname

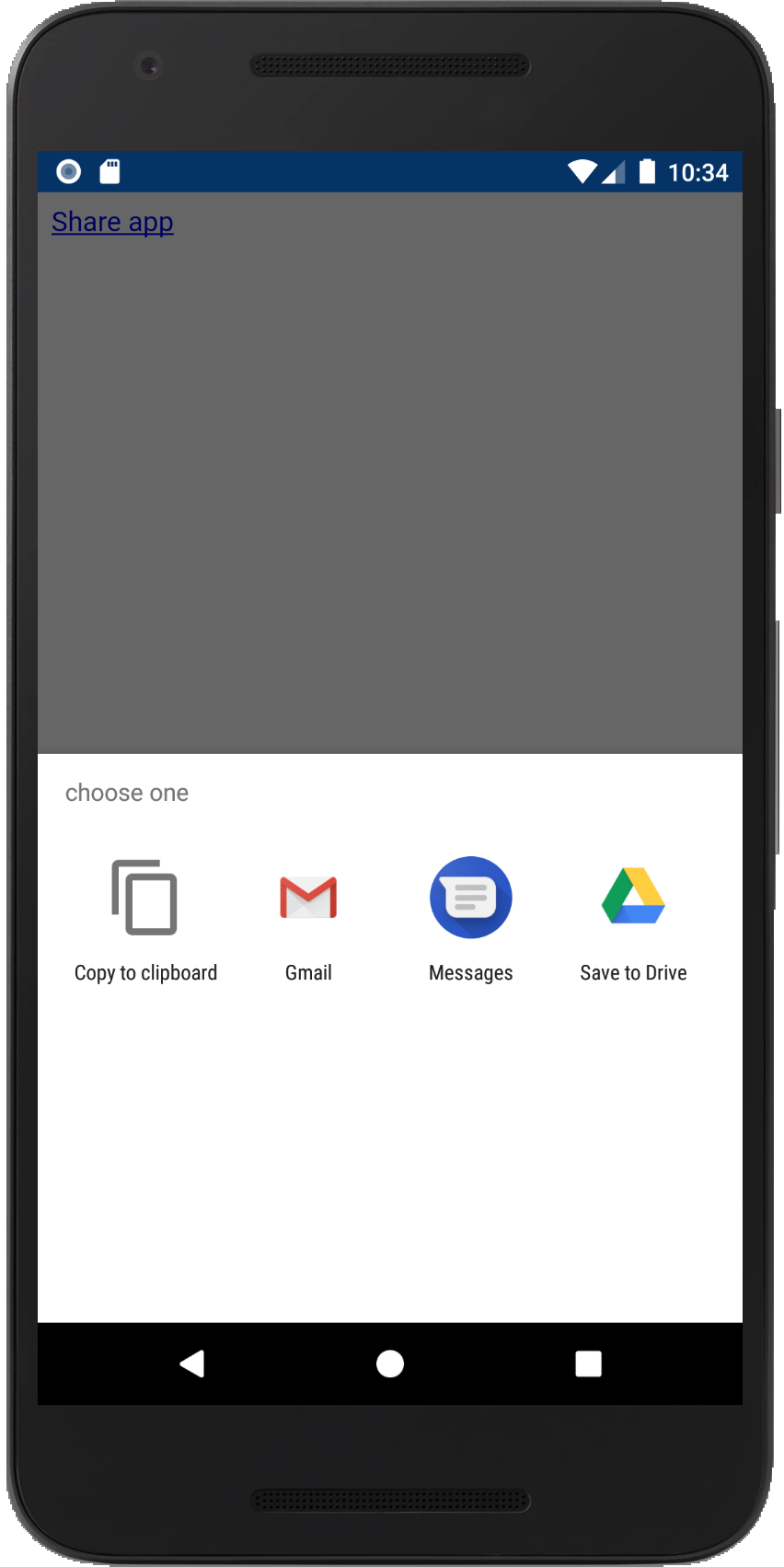
Replace *webviewgold.myappname* with the package name of your app and replace *www.webviewgold.com* with your domain.

**App Sharing Dialog API**

The Sharing Dialog functionality of WebViewGold gives your app users the ability to share the download link of your application with their friends, family, and colleagues. It's an incredibly smooth and effective way of expanding the user base of your application.

WebViewGold allows triggering a native App Sharing Dialog dialog (WhatsApp, SMS, Mail, ...) by calling shareapp:// links:

<a href="shareapp://sharetext?=Hey, check out this cool app at https://play.google.com/apps/...">Share app</a>

Clicking on such a link will trigger the sharing dialog:  
  
[](https://www.webviewgold.com/docs/android/assets/img/share_dialog.png)

**In-App Purchase & In-App Subscription API**

Within some apps, you can buy additional content or services. These kinds of purchases are called "in-app purchases". They can be an attractive source of revenue for developers and are very convenient for customers as they use existing accounts and payment sources for settlement. WebViewGold allows triggering Google Play In-App Purchases. Make sure to own an [Extended License](https://www.webviewgold.com/docs/android/?downloaded=true&update=no#credits) of WebViewGold if you plan to use this feature in an end product.  
  
Afterward, just link to this kind of URL:

<a href="inapppurchase://?package=purchase\_package\_identifier\_name\_here&successful\_url=https://www.google.com&disableadmob=true">Buy In-App Purchase</a>

Alternatively, use this kind of URL for subscription products:

<a href="inappsubscription://?package=purchase\_package\_identifier\_name\_here&successful\_url=https://www.google.com&expired\_url=https://www.yahoo.com">Start In-App Subscription</a>

In this URL example, https://www.google.com should be called after the successful subscription activation, and https://www.yahoo.com should be called as soon as the subscription is expired.  
  
You can identify the user server-side. e.g., the site /buy\_now.php redirects to In-App-Purchase API and that API redirects to /thanks.php, you can still access the user/session cookies server-sided and identify the user who just bought that In-App-Purchase. In this use case, please make sure that you deactivate the CLEAR\_CACHE\_ON\_STARTUP in the Config.java file in order to keep cookies activated by the In-App Purchase process.

**QR Code Scanner API**

QR codes are two-dimensional versions of the barcode. QR stands for "quick response". The term refers to the immediate access to the hidden information in the code. Both their design and their functions can be personalized, and they are ideal for connecting conventional print media with any and interactive online offer. WebViewGold offers a built-in QR code scanner that allows scanning these codes and opening the encrypted URL in the WebView afterward.  
  
Link to this kind of URL to trigger the QR Code Scanner:

<a href="qrcode://">Scan QR Code</a>

**Build and publish your app**

Your app is ready, and now it's time to upload it to the Google Play Store and reach an audience of millions? Exciting! The following section will show you how to do it - whether you want to do it yourself or have some work done for you.

**Option A) Full App Publishing Service**

Sometimes you just want to get more comfortable. Especially when you have a lot to do anyway, or more important decisions to make, or an important deadline. Treat yourself to some support - the [appsubmitter.io](https://www.appsubmitter.io/) service is the first choice if you want to have your app published by experts - no certificate stress, no annoying screenshot making, no upload problems, or anything else. Of course, via your own developer account,

**Option B) Manual Publishing Process**

You don't want to use our pleasant submission service, but you want to deal with it yourself? Sure, follow this guide:

In the first step, please change the **package name** to any unique package name.  
**Do a full backup of your source code folder before renaming the package name.**

Afterward, please consider a professional User Experience/UX Review on [UXreviewer.io](https://www.uxreviewer.io/) or similar services for best Play Store approval results. Such a UX review also helps you to achieve better customer retention in general for your product with suggestions for improvement.