Purpose:

The intent of this document is to provide a fast, replicable process to get a Firebase project off the ground. Upon completion of the steps outline here, you'll have a hello world app that can authenticate to firebase when run from debug or release. You can build your app from there.

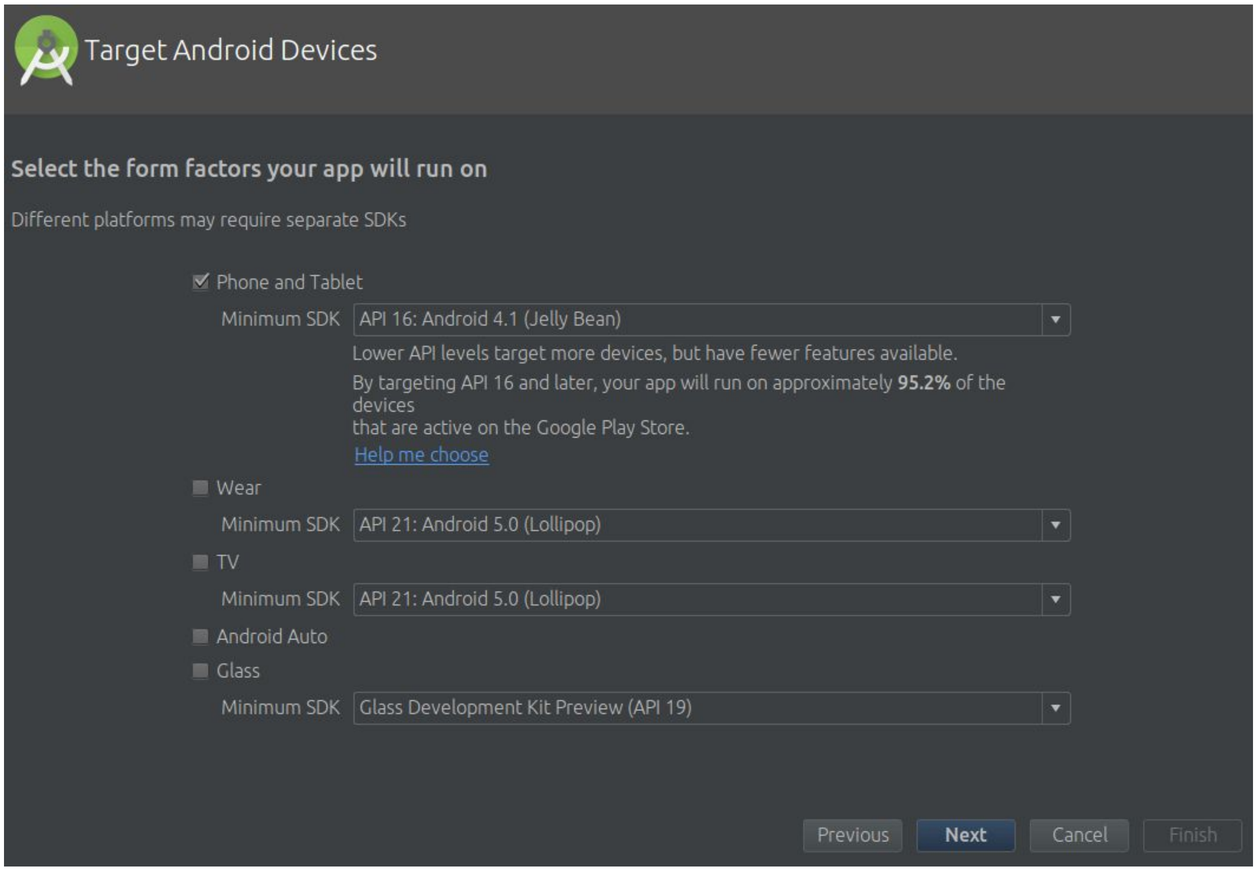
Assumptions:

We're building a project for release, not just poking around. Our intent is to actually have a release APK. If you're planning other build variants and you know what they're going to be, it's a good idea to get all of this setup early.

You have a Firebase account already setup.

Getting started:

You'll need a fresh new Android project. To support Firebase, set the MinSDK to 16.


Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☑ Phone and Tablet
Minimum SDK: API 16: Android 4.1 (Jelly Bean)
Lower API levels target more devices, but have fewer features available.
By targeting API 16 and later, your app will run on approximately **95.2%** of the devices
that are active on the Google Play Store.
Help me choose

☐ Wear
Minimum SDK: API 21: Android 5.0 (Lollipop)

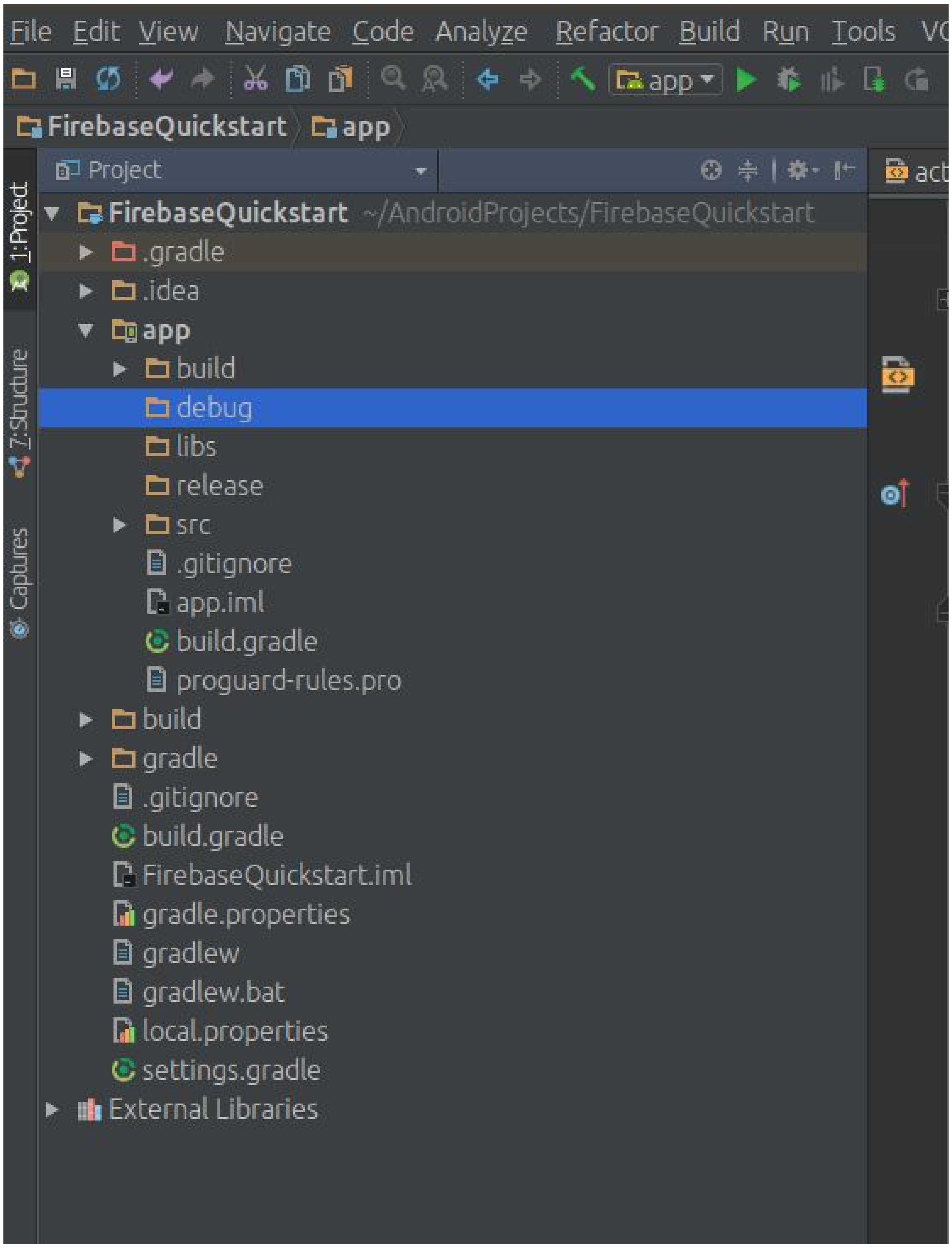
☐ TV
Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Glass
Minimum SDK: Glass Development Kit Preview (API 19)

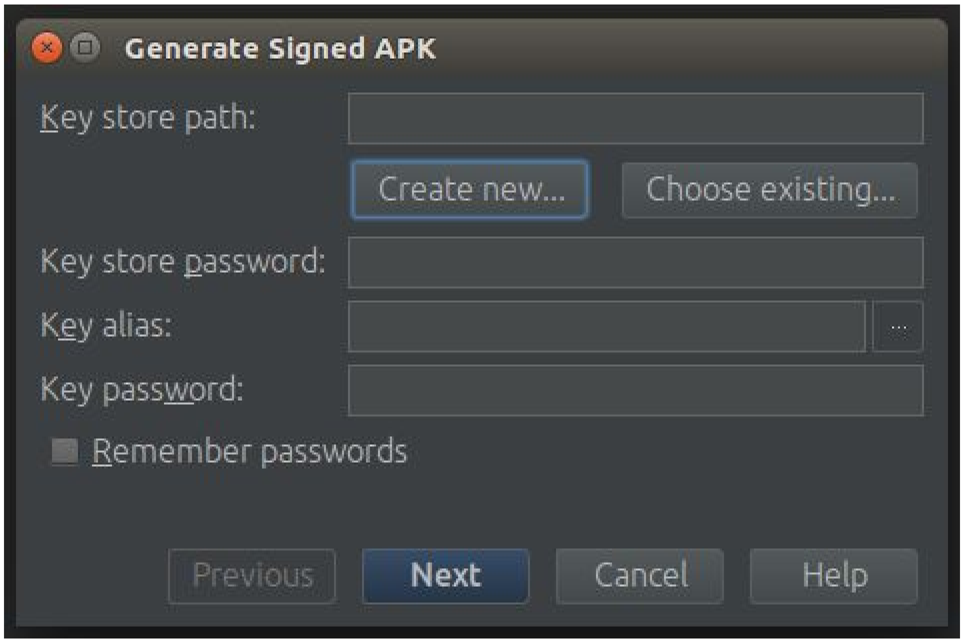
Setting up directories and build variants:

Go to project view, and build the **release** and **debug** directories
Right click on app, then New/Directory



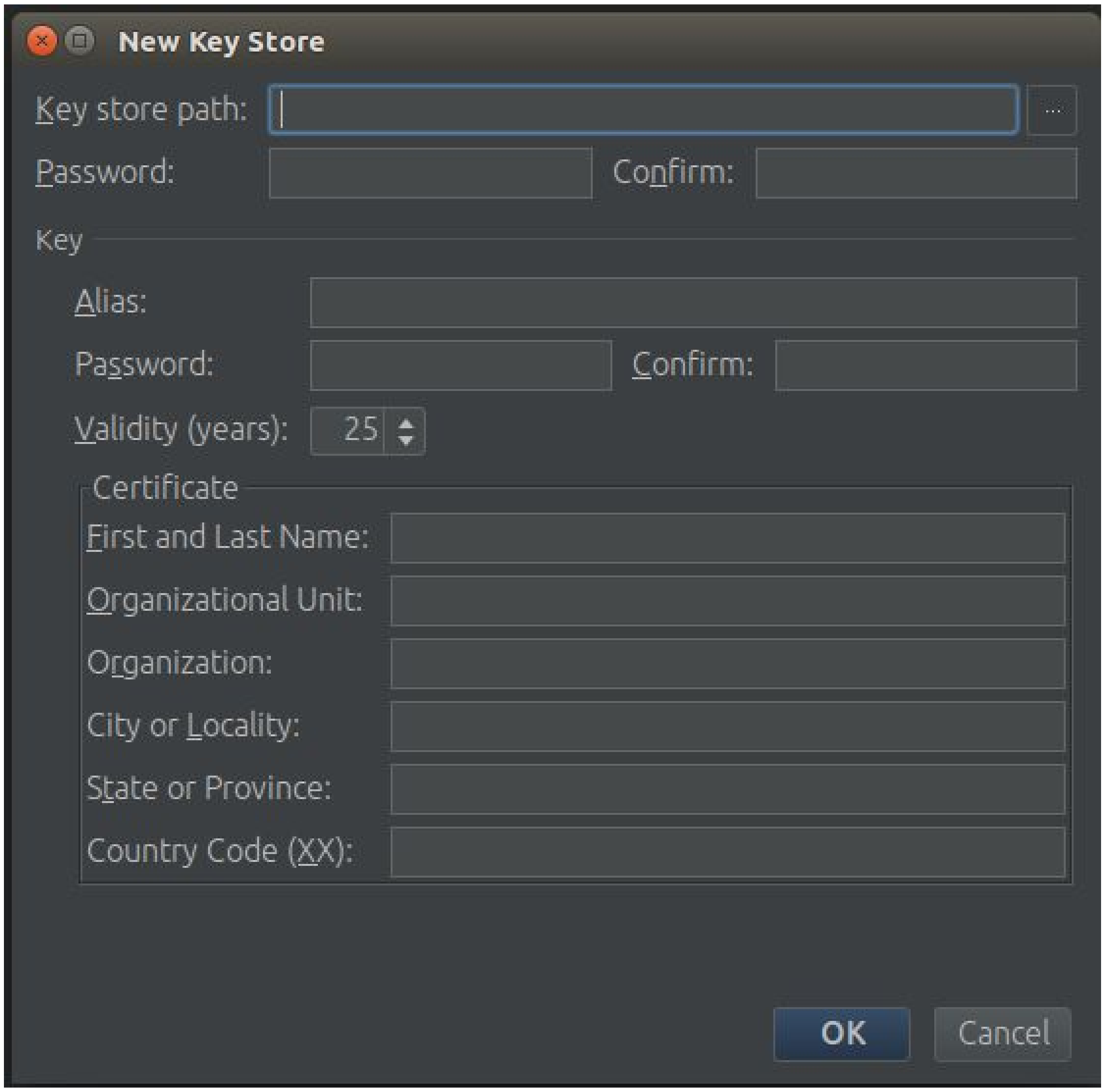
Generate Signed APK

Build Menu/Generate Signed APK



Create Keystore

Click Create new



New Key Store

Key store path:

Password: Confirm:

Key

Alias:

Password: Confirm:

Validity (years): 25

Certificate

First and Last Name:

Organizational Unit:

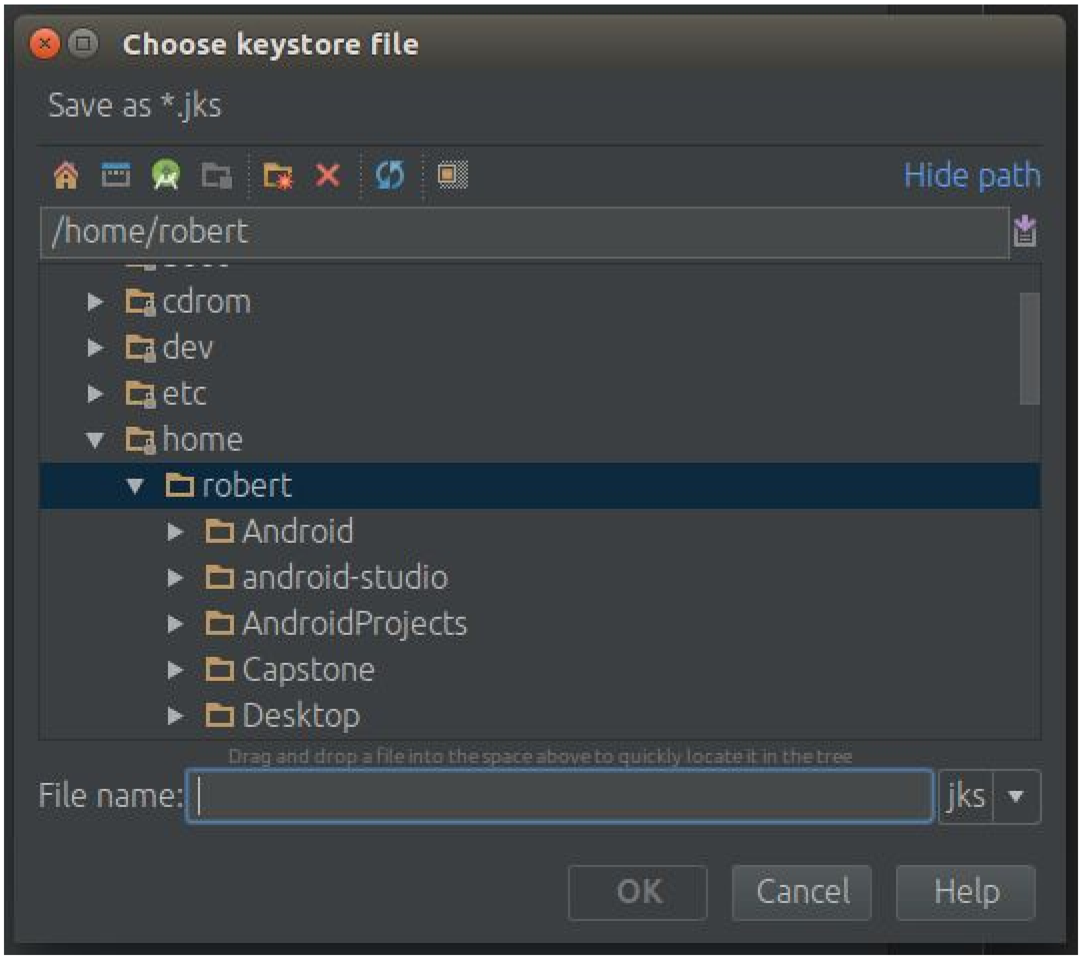
Organization:

City or Locality:

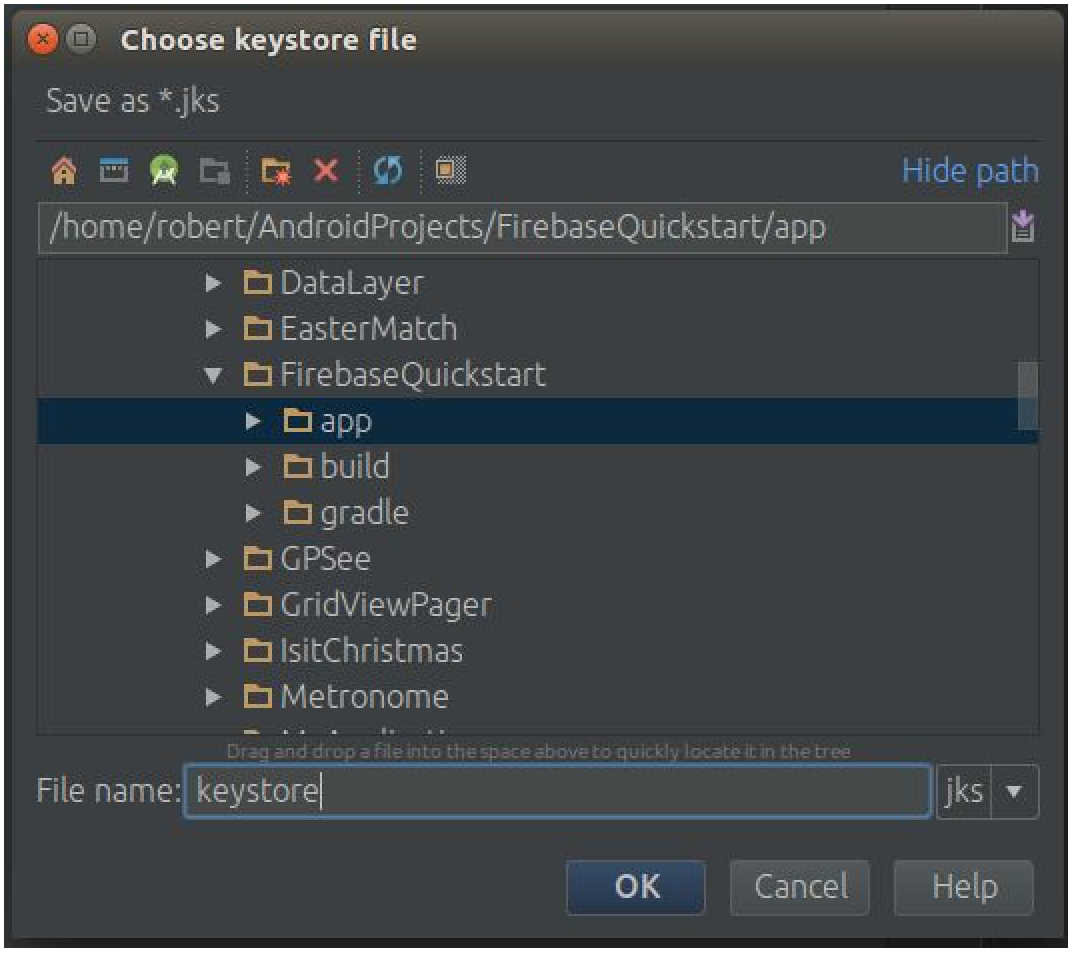
State or Province:

Country Code (XX):

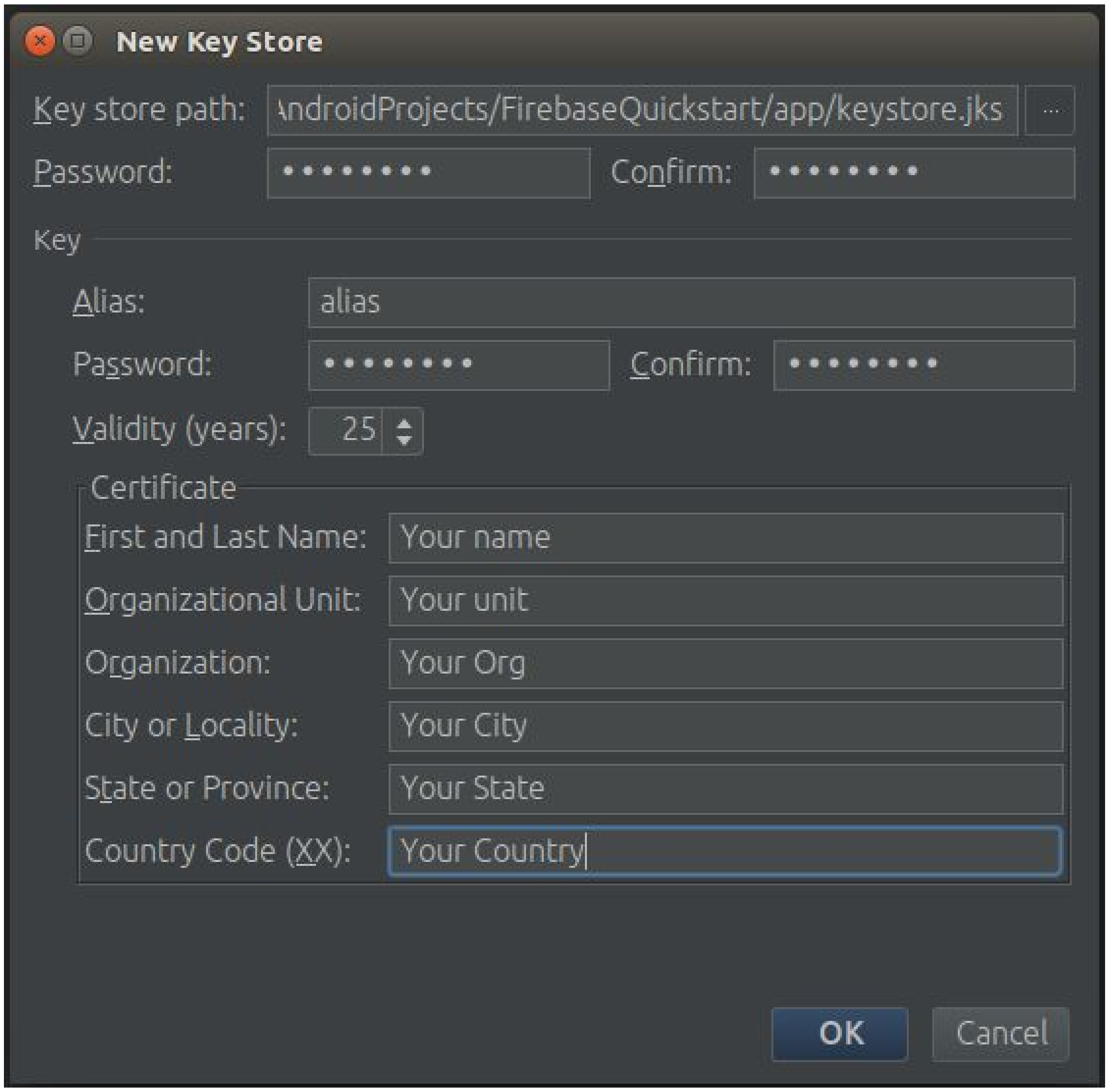
Click the … button next to the keystore path.



Click the little Android Studio icon (third from the left) and Studio will select the directory for the project. Click on the app directory, give the keystore a name, and click ok.



Give your keystore a password.
Give your keystore and alias and password
Fill out the Certificate Fields



New Key Store

Key store path: \ndroidProjects/FirebaseQuickstart/app/keystore.jks
Password: ••••••••• Confirm: •••••••••

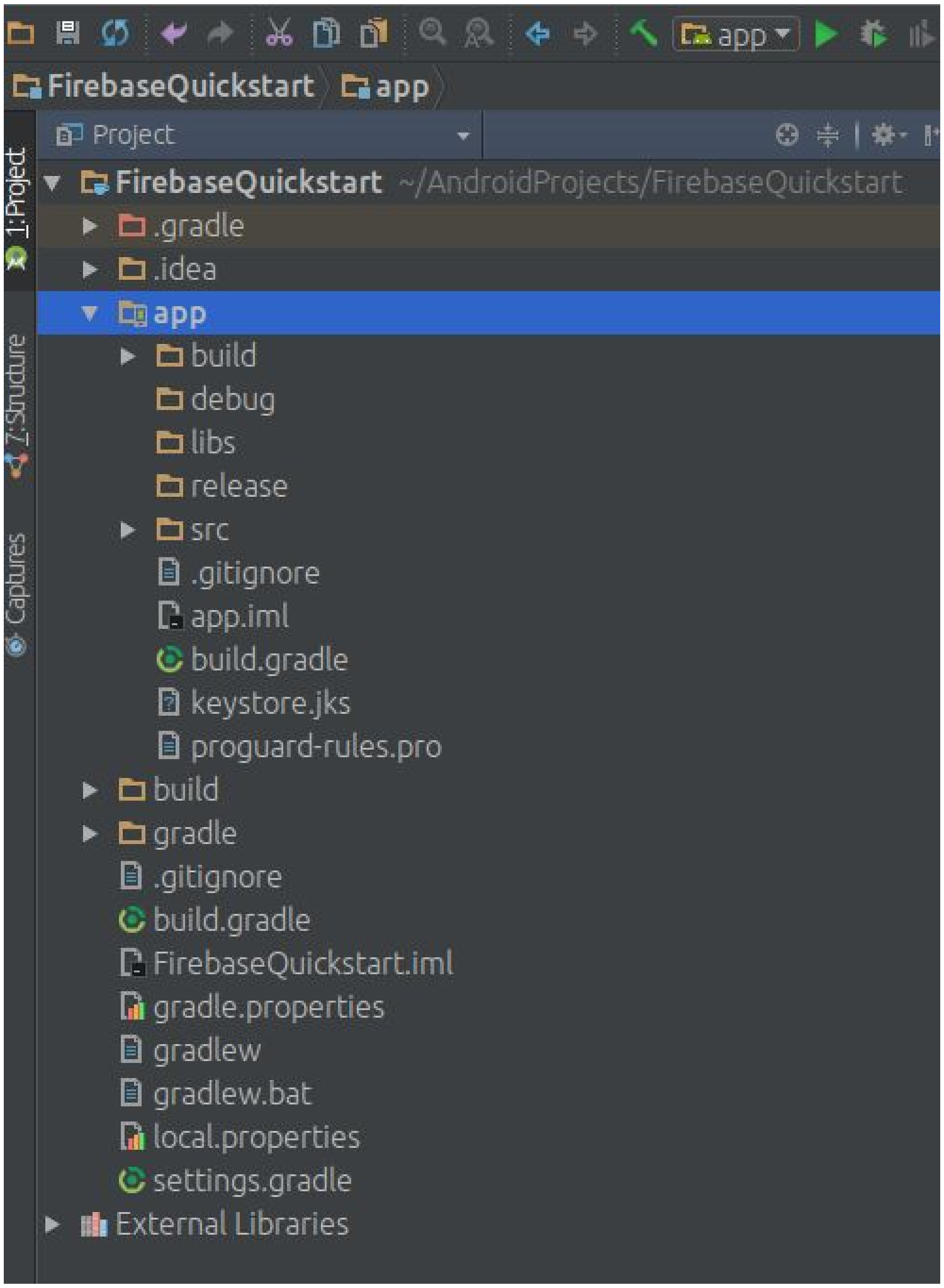
Key
Alias: alias
Password: ••••••••• Confirm: •••••••••
Validity (years): 25

Certificate
First and Last Name: Your name
Organizational Unit: Your unit
Organization: Your Org
City or Locality: Your City
State or Province: Your State
Country Code (XX): Your Country

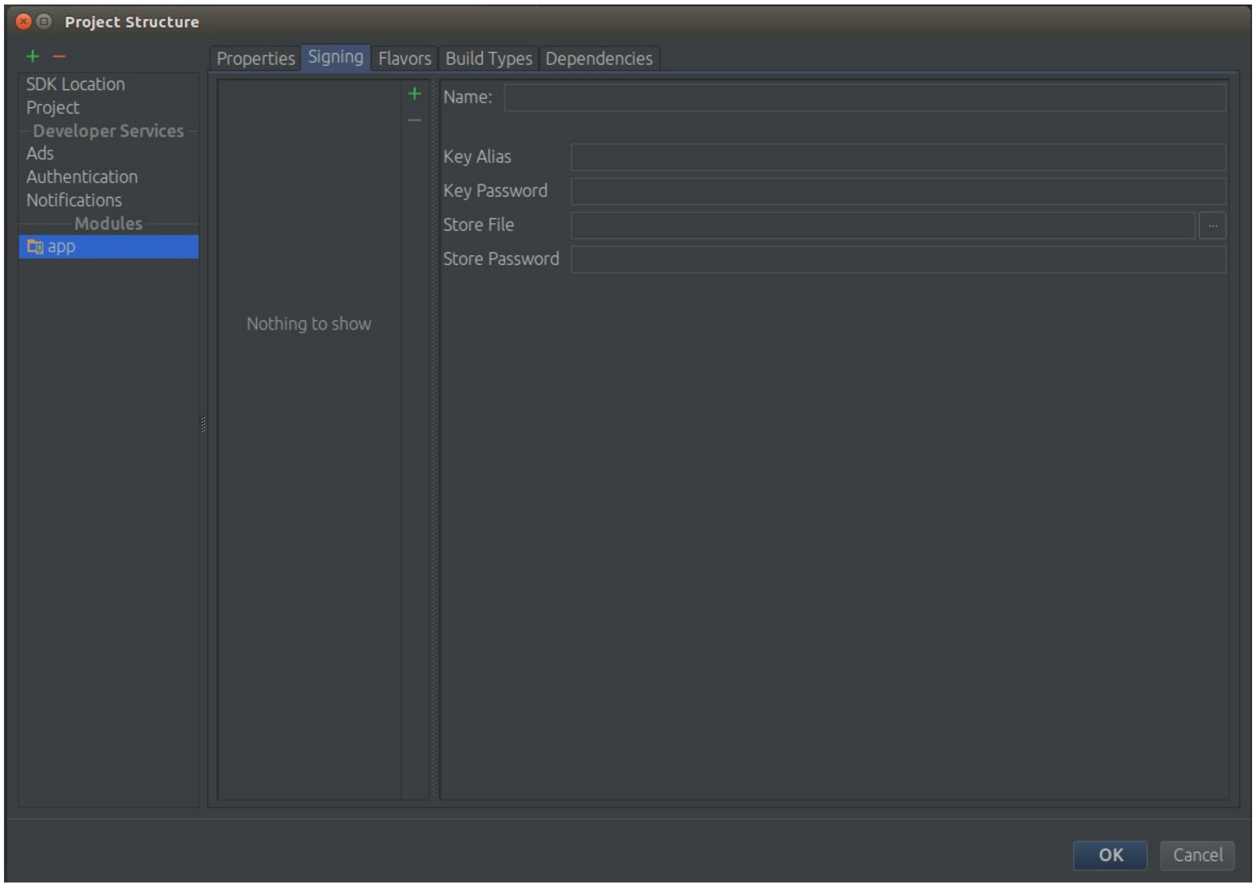
Click OK

This Generate Signed APK window will pop back up. Hit cancel. We'll be creating a signing config, not generating the APK here.

Create Signing Config

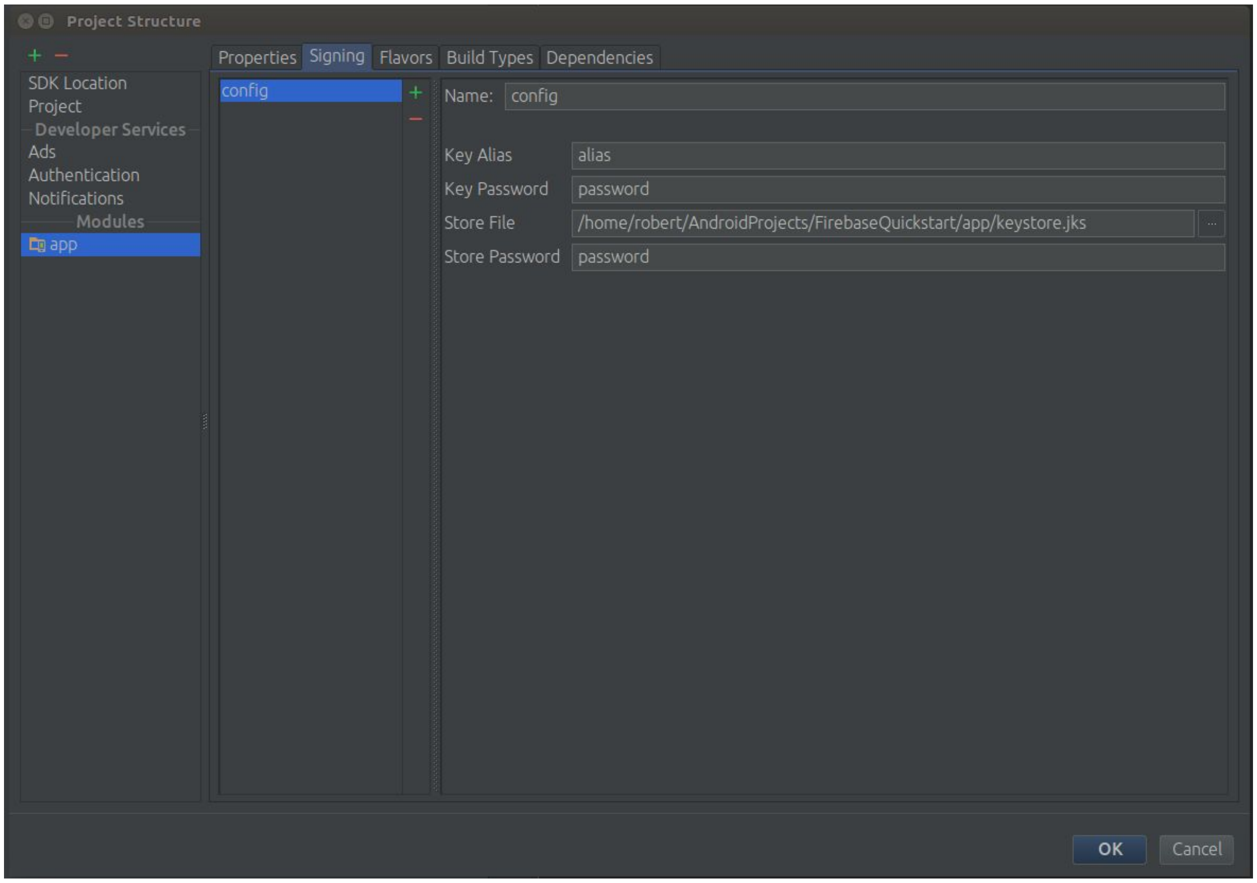


Right click on your app module and click Open Module Settings.



Go to the signing tab and click the green +

Fill out the fields with the alias and store information we just created in the previous steps.



Name: config

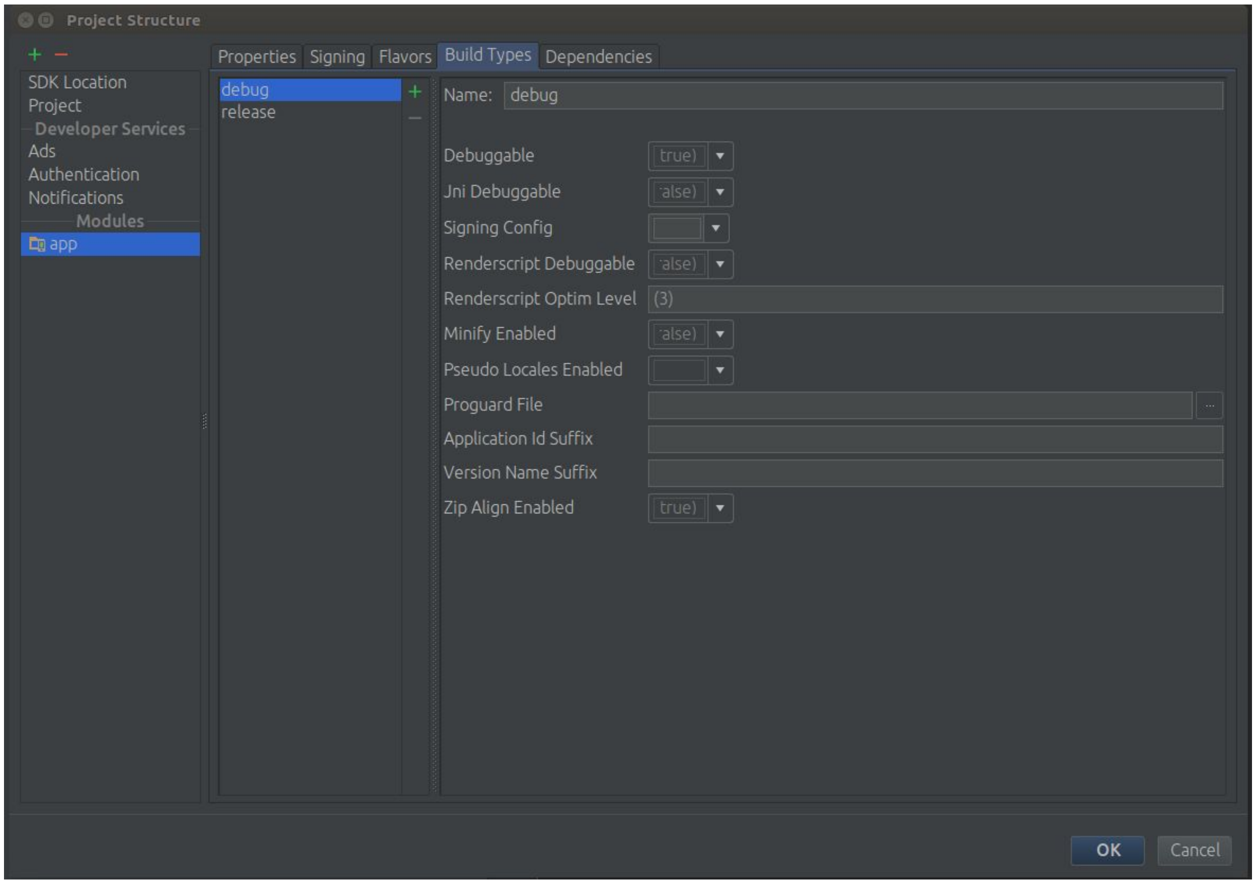
Key Alias: alias

Key Password: password

Store File: /home/robert/AndroidProjects/FirebaseQuickstart/app/keystore.jks

Store Password: password

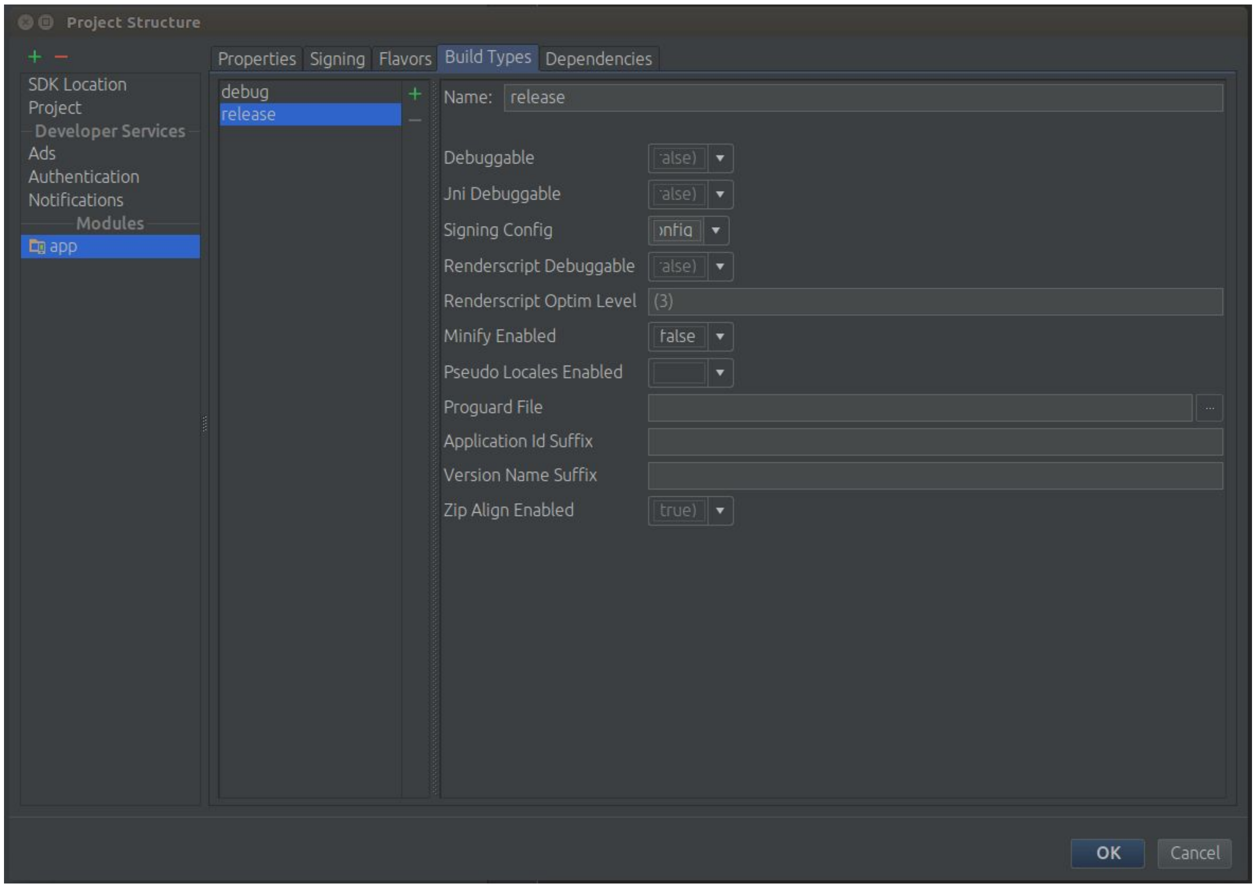
Click the Build Types tab



Name: debug

Debuggable (true)
Jni Debuggable (false)
Signing Config
Renderscript Debuggable (false)
Renderscript Optim Level (3)
Minify Enabled (false)
Pseudo Locales Enabled
Proguard File
Application Id Suffix
Version Name Suffix
Zip Align Enabled (true)

Select the release variant, and then select "config" in the dropdown menu next to "Signing Config"


Project Structure

SDK Location
Project
Developer Services
Ads
Authentication
Notifications
Modules
app

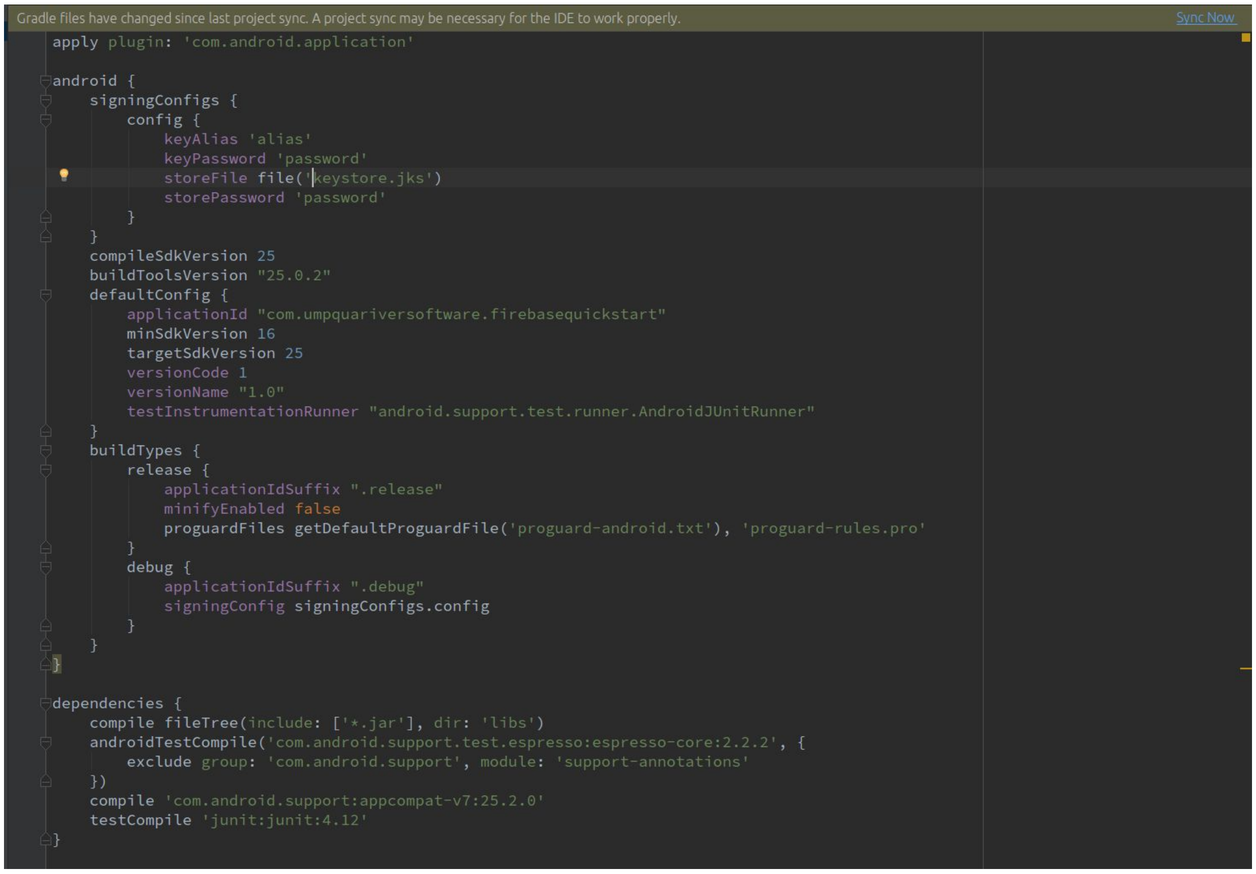
Properties | Signing | Flavors | Build Types | Dependencies

debug
release

Name: release

Debuggable alse)
Jni Debuggable alse)
Signing Config onfig
Renderscript Debuggable alse)
Renderscript Optim Level (3)
Minify Enabled false
Pseudo Locales Enabled
Proguard File
Application Id Suffix
Version Name Suffix
Zip Align Enabled true)

Modify build.gradle
Add applicationIdSuffix to release and debug build types
Change the storeFile to a relative path (remove everything aside from keystrore.jks)

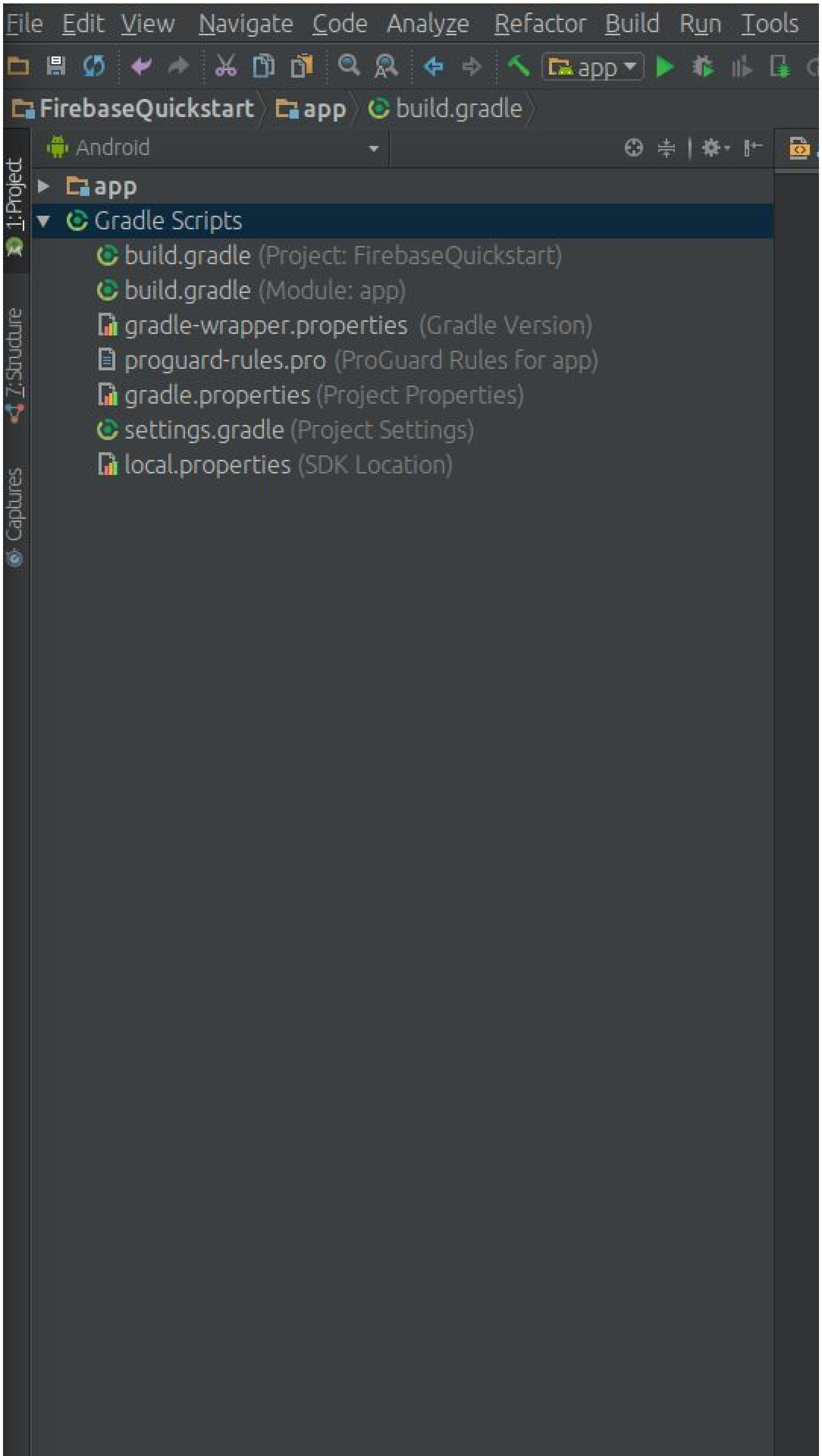

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.
Sync Now

```
apply plugin: 'com.android.application'

android {
    signingConfigs {
        config {
            keyAlias 'alias'
            keyPassword 'password'
            storeFile file('keystore.jks')
            storePassword 'password'
        }
    }
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.umpquariversoftware.firebasequickstart"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            applicationIdSuffix ".release"
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
        debug {
            applicationIdSuffix ".debug"
            signingConfig signingConfigs.config
        }
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.2.0'
    testCompile 'junit:junit:4.12'
}
```

Gradle files have changed, so click Sync Now in the upper right hand corner.

Checkpoint.

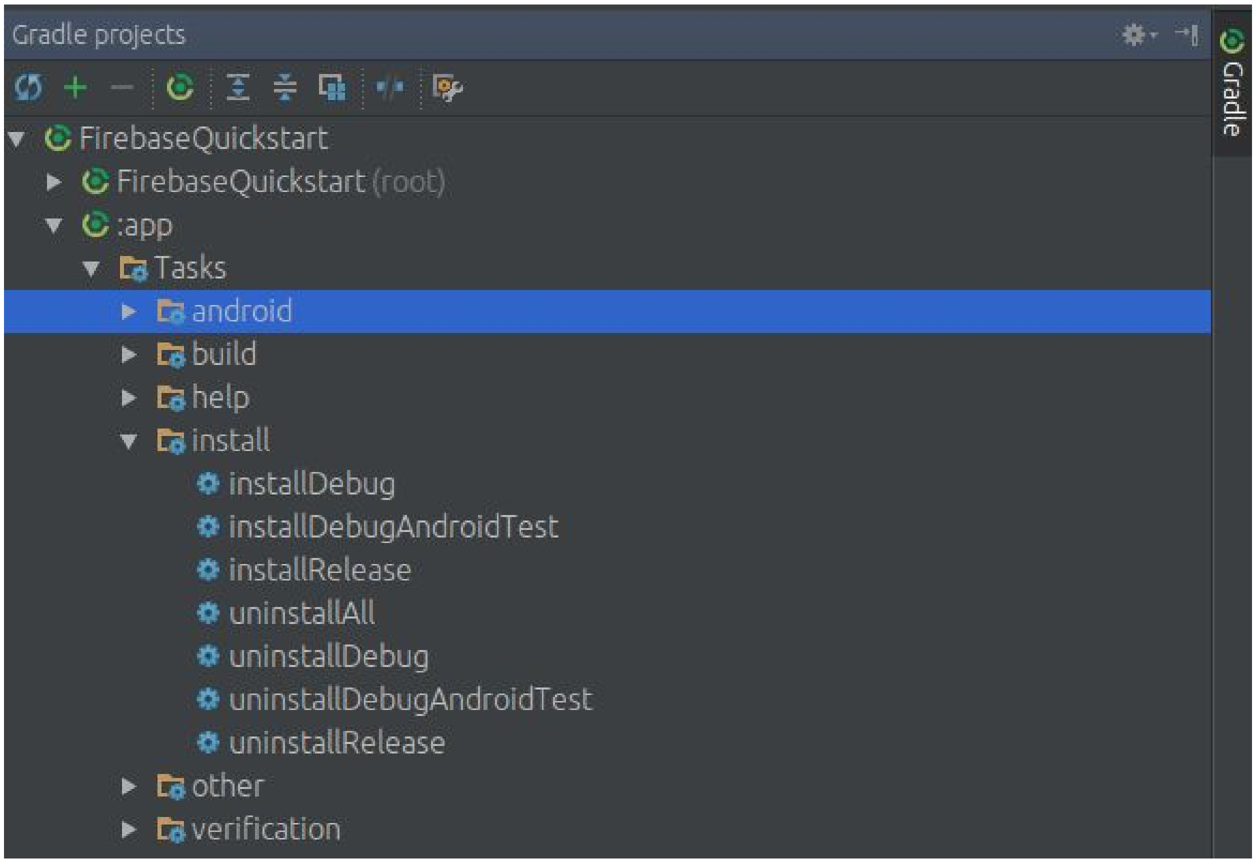
You should now have a debug and release variant.
To verify, click the Build Menu, and click Select Build Variant. On the left, your build variants will show up.


File Edit View Navigate Code Analyze Refactor Build Run Tools
app
FirebaseQuickstart app build.gradle
Android
1: Project
7: Structure
Captures

app
Gradle Scripts
build.gradle (Project: FirebaseQuickstart)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

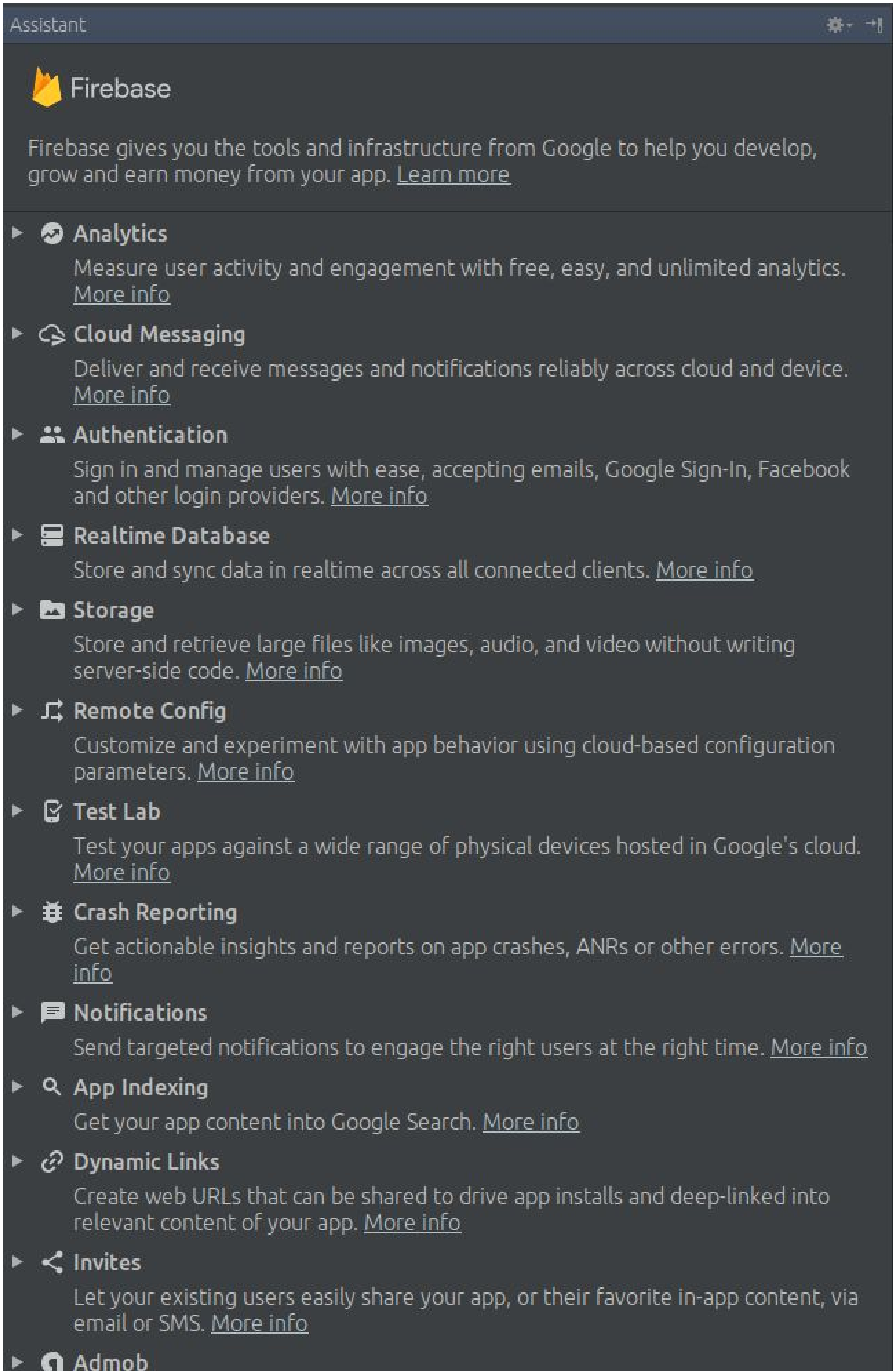
Both debug and release should show up in the drop down menu. Verify you can run both of these by selecting it in the drop down, and then clicking the green Run button at the top of Studio.

You'll also have an installRelease gradle task at this point.
Find this on the right side of AS by clicking the Gradle button. Under app/tasks/install you should see an installRelease task. Verify this pushes a signed APK to your device.



If all of your checks completed successfully, you're ready to start talking to Firebase.

In the Tools Menu, select Firebase. The firebase assistant comes up on the right.


Assistant

Firebase

Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more

Analytics
Measure user activity and engagement with free, easy, and unlimited analytics. More info

Cloud Messaging
Deliver and receive messages and notifications reliably across cloud and device. More info

Authentication
Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. More info

Realtime Database
Store and sync data in realtime across all connected clients. More info

Storage
Store and retrieve large files like images, audio, and video without writing server-side code. More info

Remote Config
Customize and experiment with app behavior using cloud-based configuration parameters. More info

Test Lab
Test your apps against a wide range of physical devices hosted in Google's cloud. More info

Crash Reporting
Get actionable insights and reports on app crashes, ANRs or other errors. More info

Notifications
Send targeted notifications to engage the right users at the right time. More info

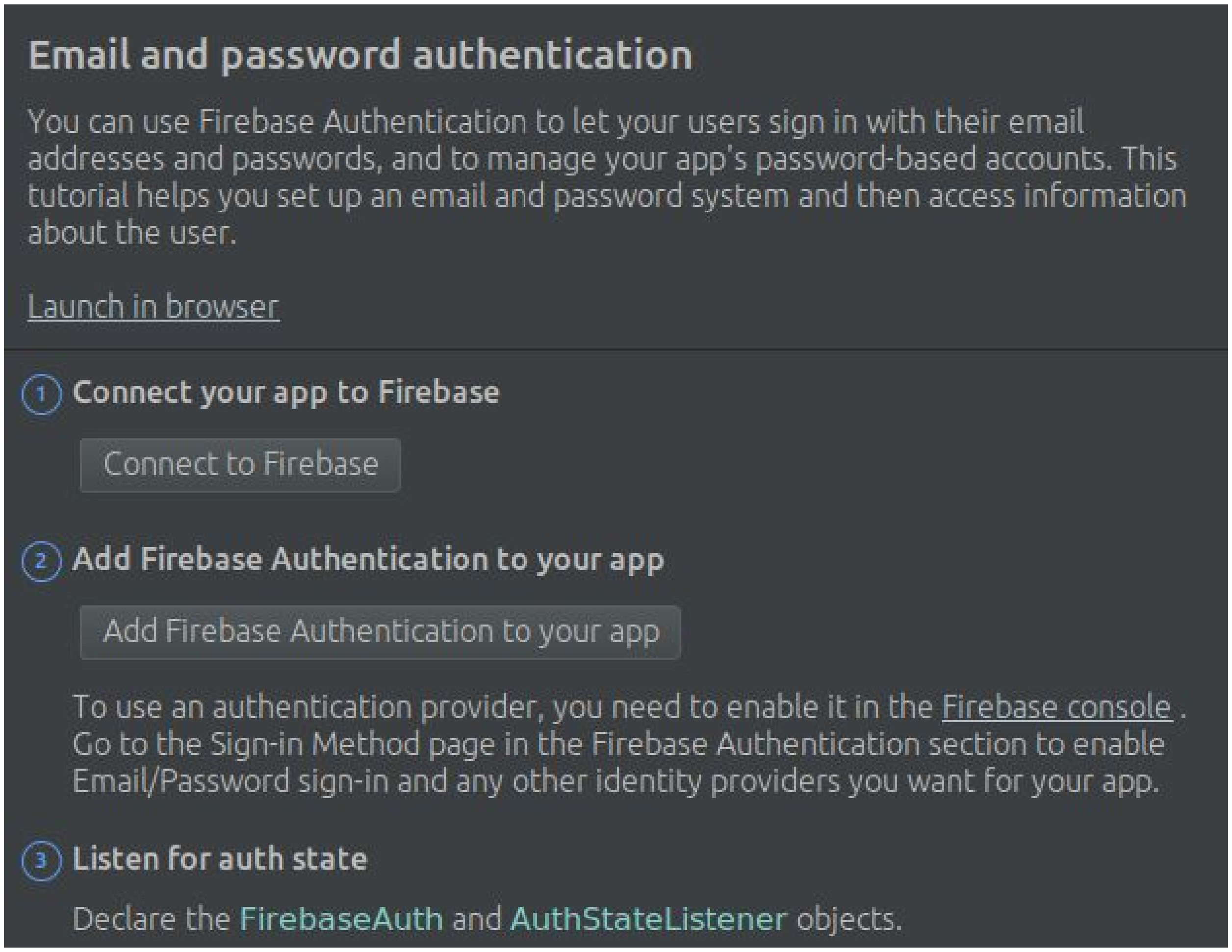
App Indexing
Get your app content into Google Search. More info

Dynamic Links
Create web URLs that can be shared to drive app installs and deep-linked into relevant content of your app. More info

Invites
Let your existing users easily share your app, or their favorite in-app content, via email or SMS. More info

Admob

Click on Authentication,and click email and password authentication. The assistant brings up the steps. Click connect to Firebase.



Email and password authentication

You can use Firebase Authentication to let your users sign in with their email addresses and passwords, and to manage your app's password-based accounts. This tutorial helps you set up an email and password system and then access information about the user.

Launch in browser

1. Connect your app to Firebase

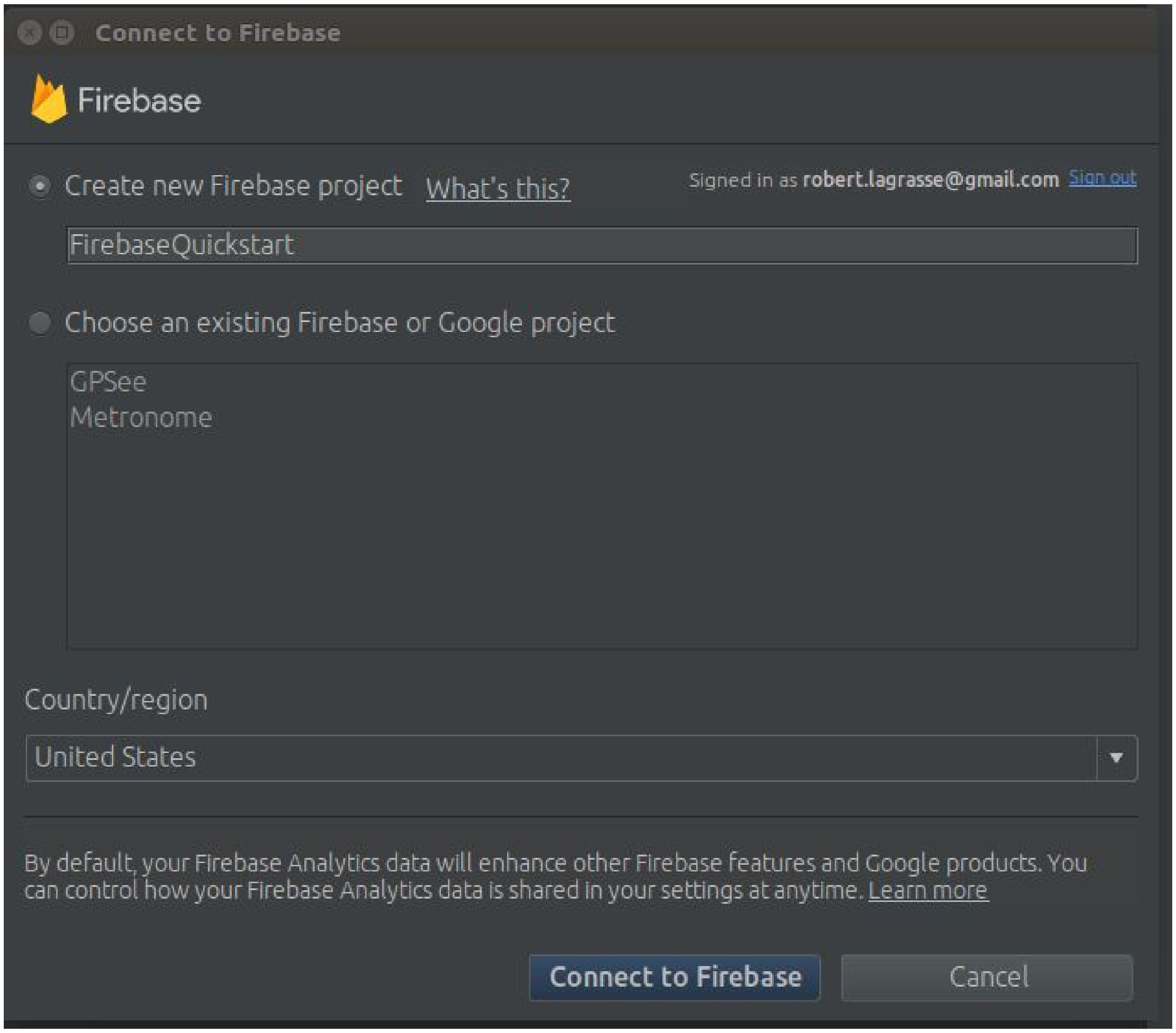
2. Add Firebase Authentication to your app

To use an authentication provider, you need to enable it in the Firebase console . Go to the Sign-in Method page in the Firebase Authentication section to enable Email/Password sign-in and any other identity providers you want for your app.

3. Listen for auth state

Declare the FirebaseAuth and AuthStateListener objects.

Click Connect to Firebase



● Create new Firebase project What's this?

Signed in as robert.lagrasse@gmail.com Sign out

FirebaseQuickstart

○ Choose an existing Firebase or Google project

GPSee
Metronome

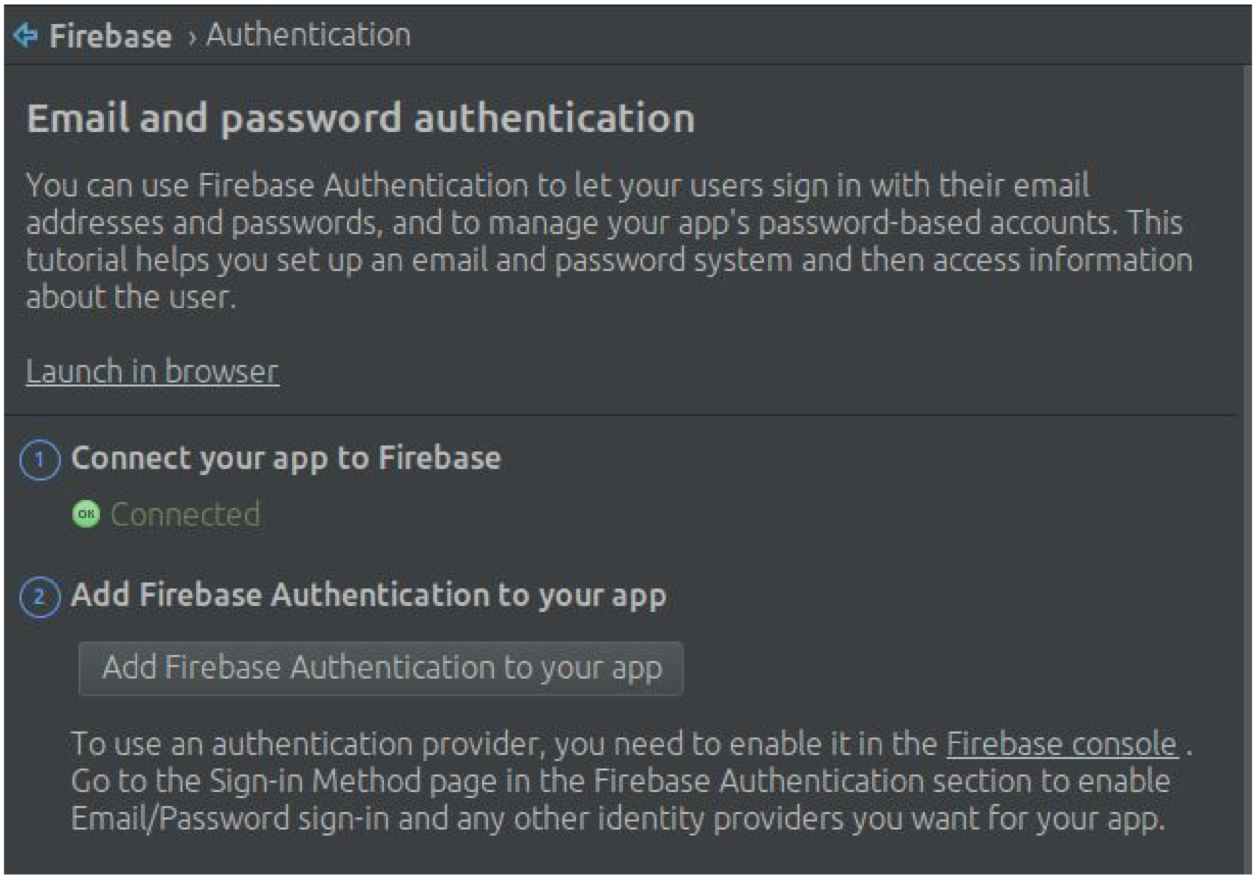
Country/region: United States

By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime. Learn more

Select Create new Firebase project, and give it a name (this is the default).
Click Connect to Firebase

It takes a minute. Let it do it’s thing.

In the Assistant, click Step 2 and add Firebase Authentication to your app.



Firebase › Authentication

Email and password authentication

You can use Firebase Authentication to let your users sign in with their email addresses and passwords, and to manage your app's password-based accounts. This tutorial helps you set up an email and password system and then access information about the user.

Launch in browser

1. Connect your app to Firebase

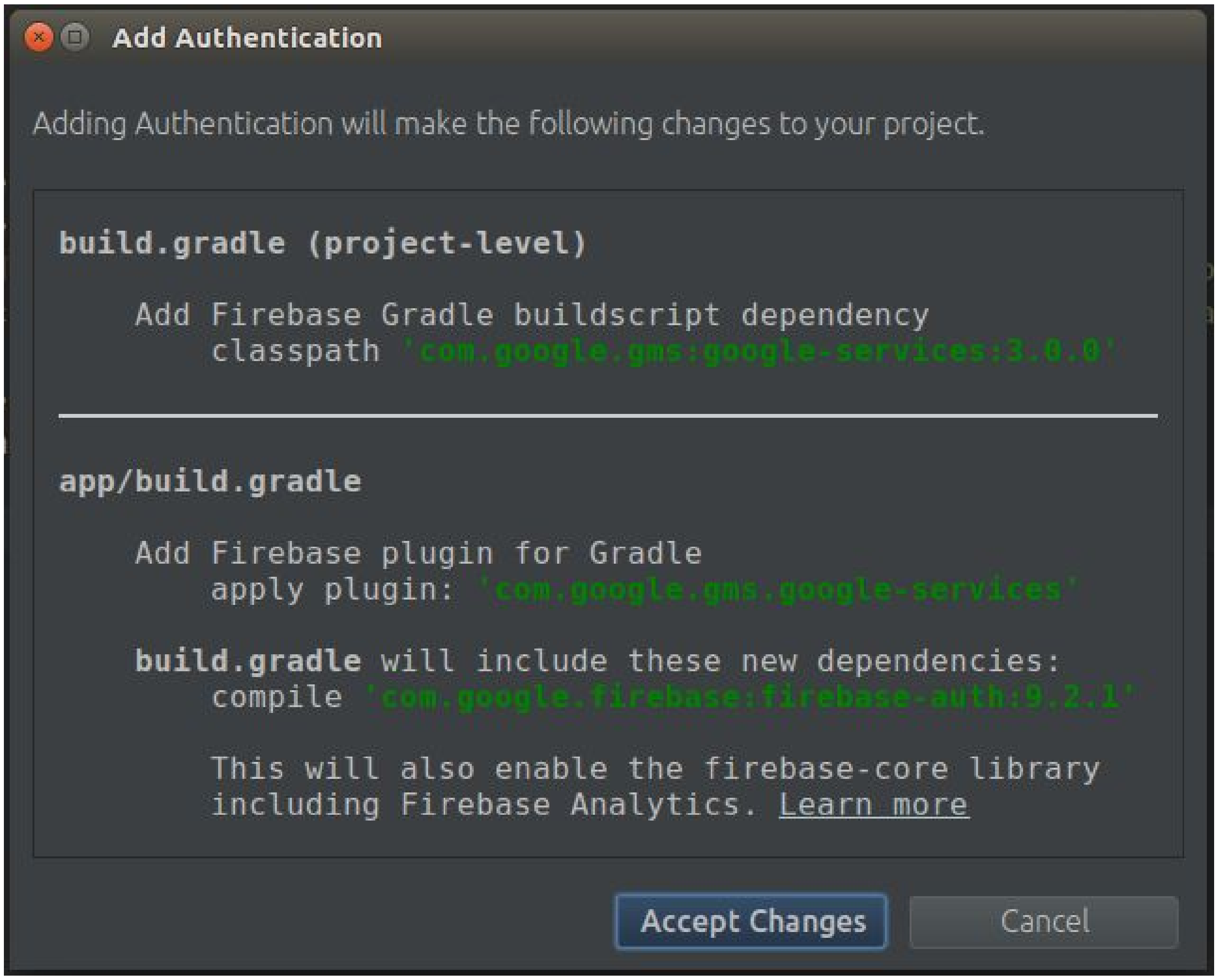
   Connected

2. Add Firebase Authentication to your app

   Add Firebase Authentication to your app

   To use an authentication provider, you need to enable it in the Firebase console . Go to the Sign-in Method page in the Firebase Authentication section to enable Email/Password sign-in and any other identity providers you want for your app.

This screen pops up:



Adding Authentication will make the following changes to your project.

```
build.gradle (project-level)

    Add Firebase Gradle buildscript dependency
        classpath 'com.google.gms:google-services:3.0.0'

app/build.gradle

    Add Firebase plugin for Gradle
        apply plugin: 'com.google.gms.google-services'

    build.gradle will include these new dependencies:
        compile 'com.google.firebase:firebase-auth:9.2.1'

        This will also enable the firebase-core library
        including Firebase Analytics. Learn more
```

Let the wizard add the dependencies by clicking Accept Changes

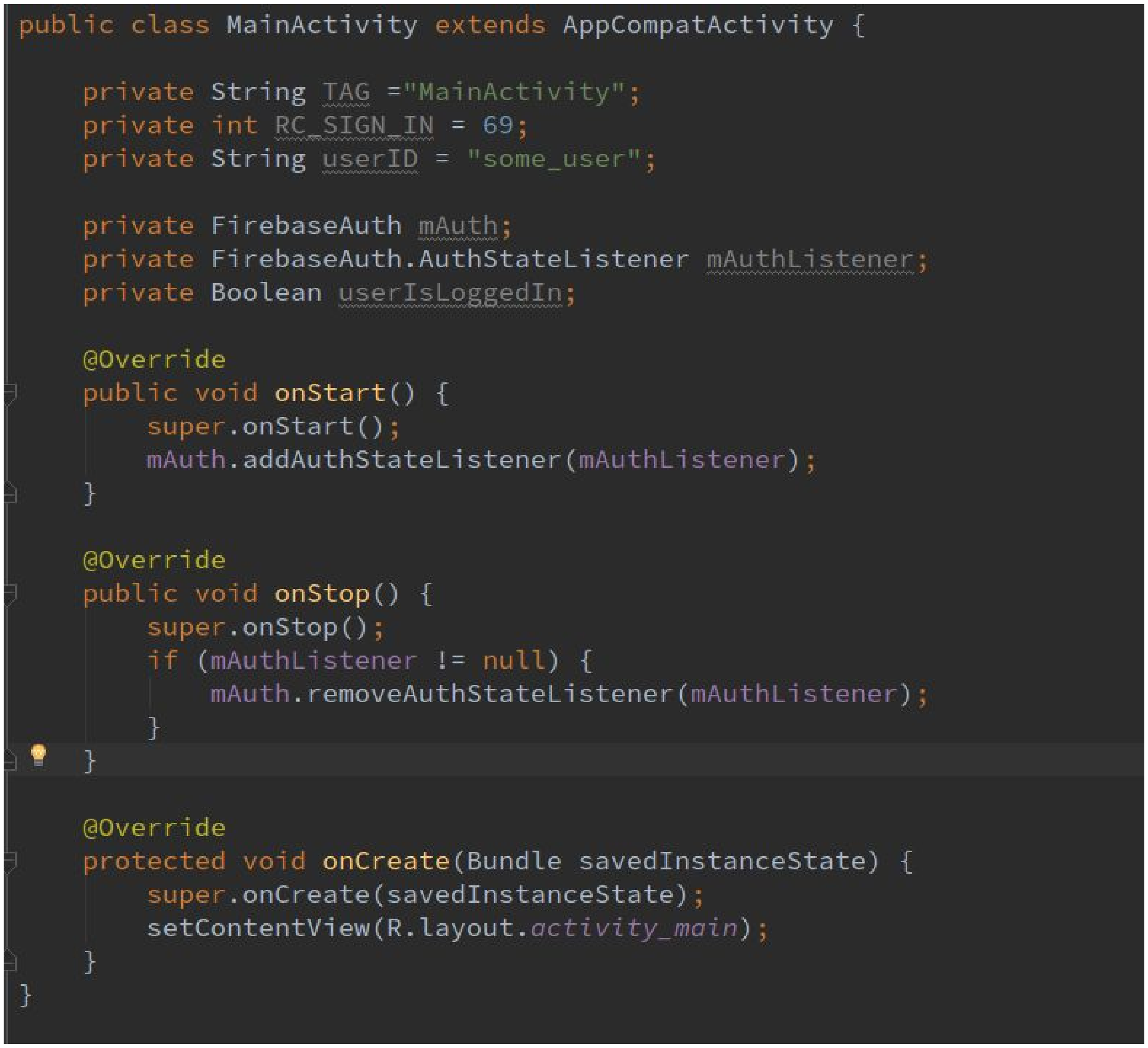
We depart from the Wizard at this point.

Declare these at the method level:

```
private String TAG ="MainActivity";
private int RC_SIGN_IN = 69;
private String userID = "some_user";

private FirebaseAuth mAuth;
private FirebaseAuth.AuthStateListener mAuthListener;
private Boolean userIsLoggedIn;
```

Override onStart()/onStop() as directed by the wizard, or just paste this in:

```
@Override
public void onStart() {
  super.onStart();
  mAuth.addAuthStateListener(mAuthListener);
}

@Override
public void onStop() {
  super.onStop();
  if (mAuthListener != null) {
     mAuth.removeAuthStateListener(mAuthListener);
  }
}
```



```
public class MainActivity extends AppCompatActivity {

    private String TAG ="MainActivity";
    private int RC_SIGN_IN = 69;
    private String userID = "some_user";

    private FirebaseAuth mAuth;
    private FirebaseAuth.AuthStateListener mAuthListener;
    private Boolean userIsLoggedIn;

    @Override
    public void onStart() {
        super.onStart();
        mAuth.addAuthStateListener(mAuthListener);
    }

    @Override
    public void onStop() {
        super.onStop();
        if (mAuthListener != null) {
            mAuth.removeAuthStateListener(mAuthListener);
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Basing the next steps on the directions found here:
https://github.com/firebase/FirebaseUI-Android/blob/master/auth/README.md

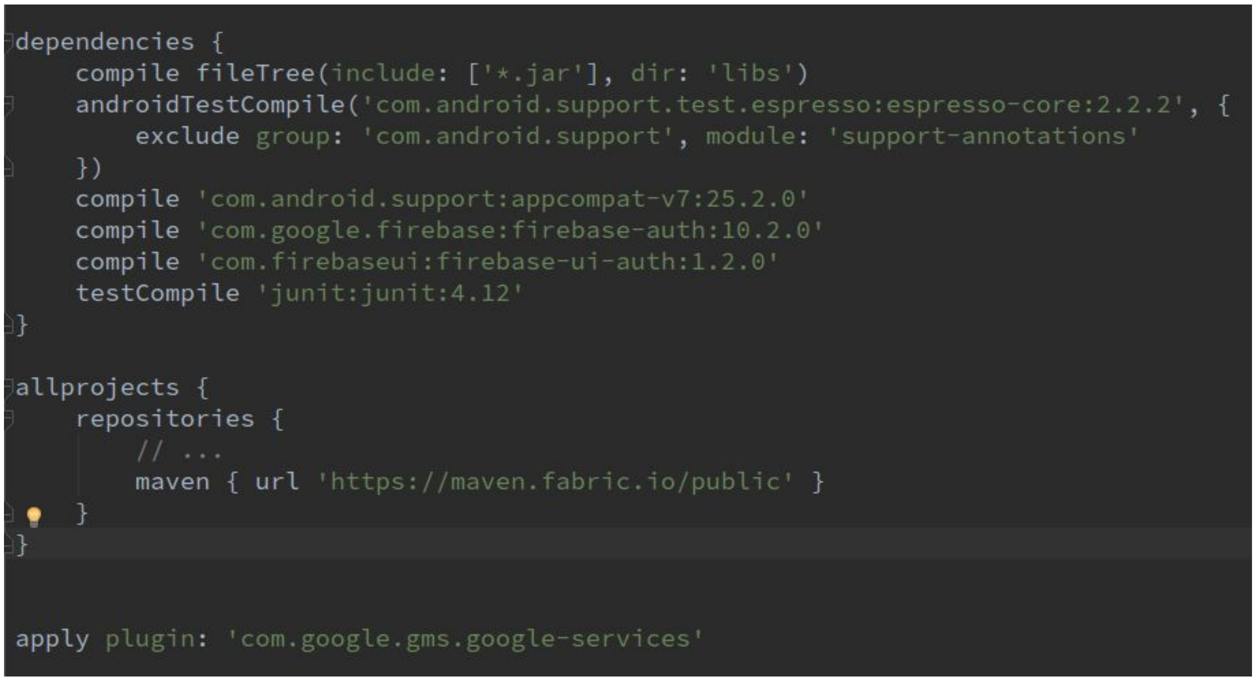
There are far more details in that document. Use them. This is just basic stuff to get you off the ground.

Modify build.gradle:

```
dependencies {
   // ...
   compile 'com.firebaseui:firebase-ui-auth:1.2.0'
}
```

and add the Fabric repository

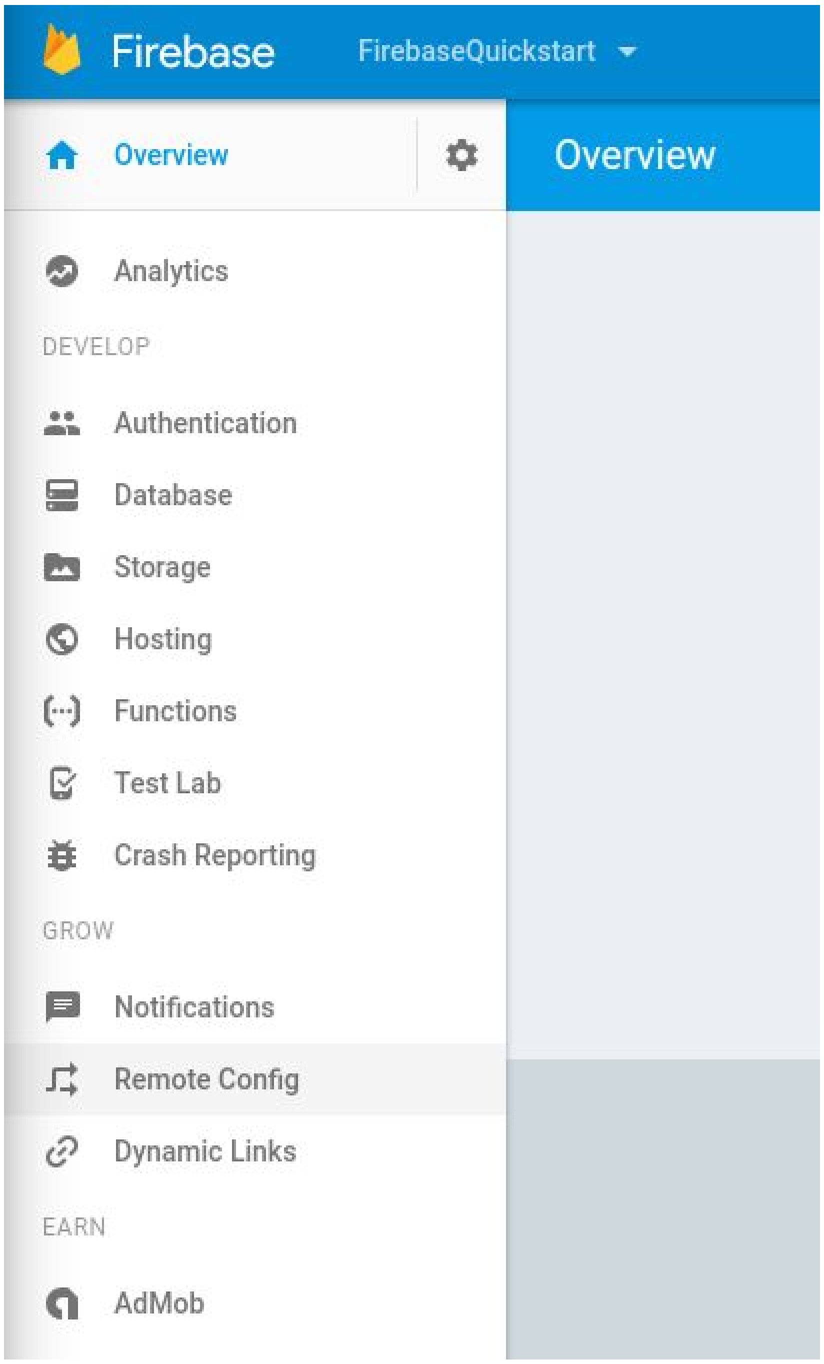
```
allprojects {
   repositories {
      // ...
      maven { url 'https://maven.fabric.io/public' }
   }
}
```



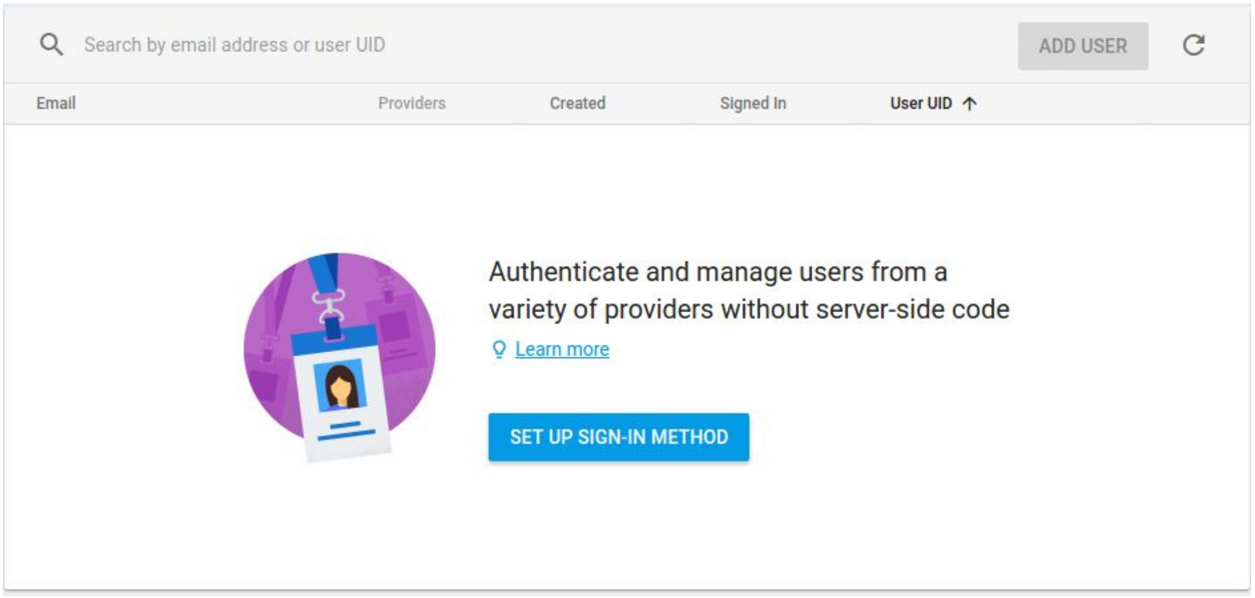
```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.2.0'
    compile 'com.google.firebase:firebase-auth:10.2.0'
    compile 'com.firebaseui:firebase-ui-auth:1.2.0'
    testCompile 'junit:junit:4.12'
}

allprojects {
    repositories {
        // ...
        maven { url 'https://maven.fabric.io/public' }
    }
}


apply plugin: 'com.google.gms.google-services'
```

Gradle Sync as you’ve changed things.

Enable Authentication via email and Google in Firebase console for this project.

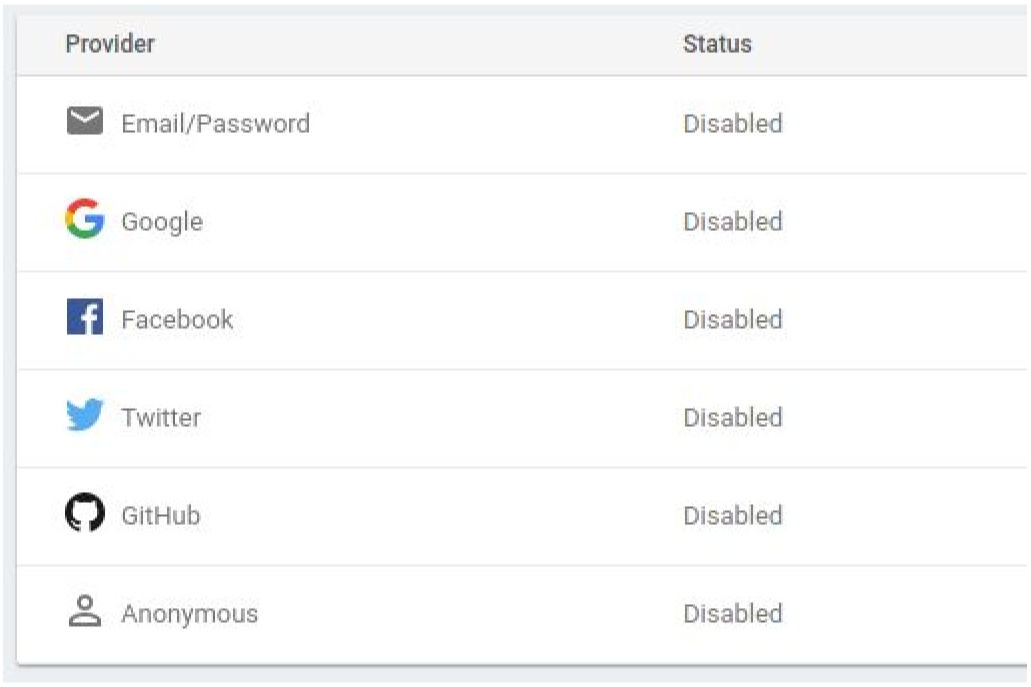


In the Firebase console, Click Authentication



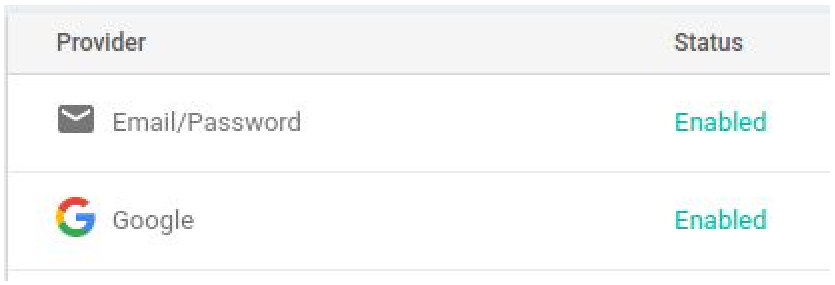
Authenticate and manage users from a variety of providers without server-side code

Click the SET UP SIGN-IN METHOD



Provider	Status
Email/Password	Disabled
Google	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

Click the Disable status next to Email/Password and enable it on the next screen. Repeat for Google.



Provider	Status
Email/Password	Enabled
Google	Enabled

Both providers are now enabled.

Now paste this method in your activity:

```
private void authenticateUser(){

   /**
    * Authenticates user, and watches for changes.
    *
    * **/

   mAuth = FirebaseAuth.getInstance();
   userIsLoggedIn = false;

   mAuthListener = new FirebaseAuth.AuthStateListener() {
       @Override
       public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
           FirebaseUser user = firebaseAuth.getCurrentUser();
           if (user != null) {
               // User is signed in
               userIsLoggedIn = true;
               userID = user.getUid();
               Toast.makeText(MainActivity.this, "Welcome, " + user.getDisplayName()
                       , Toast.LENGTH_SHORT).show();
           } else {
               userIsLoggedIn = false;
               startActivityForResult(
                       AuthUI.getInstance()
                               .createSignInIntentBuilder()
                               .setProviders(Arrays.asList(new
AuthUI.IdpConfig.Builder(AuthUI.EMAIL_PROVIDER).build(),
                                       new AuthUI.IdpConfig.Builder(AuthUI.GOOGLE_PROVIDER).build()))
                               .build(),
                       RC_SIGN_IN);
```

```
            }
        }
    };
}
```

Call the authenticateUser() method from onCreate. If the user is not logged in, the FirebaseUI login activity takes over. If they are logged in, they get a toast. Replace the toast with whatever your program needs to get going.

You can and probably should build response handlers for the different types of login responses.

Drop in a method to let them sign out. Wire up to a button or menu option later:

```
public void signoutUser() {
  AuthUI.getInstance()
          .signOut(this)
          .addOnCompleteListener(new OnCompleteListener<Void>() {
              public void onComplete(@NonNull Task<Void> task) {
                  // user is now signed out
                  // do something if you need to
                  finish();
              }
          });
}
```

Pushing any version of this should now connect to firebase for authentication.

Running the app will prompt the user to select an account, and then request permissions to view basic profile information.