

# REAL-TIME IMAGE SHARING

**JERRY JALAVA**

Senior System Architect, Google  
Developer Expert, Authorised Trainer

[JERRY@QVIK.FI](mailto:JERRY@QVIK.FI) | @W\_I



**QVIK**



# CLIENT ASKS FOR POC...

- Users should be able to easily sign-in (which is required)
- Users should be able to upload images from their camera
- Other users should see these images immediately
- Users should be able to manage their own images
- Users should be able to like images of others



**DEADLINE IS SET FOR  
YESTERDAY**

# SO WHAT DO WE NEED TO DO

- Some server to handle our uploads and other api requests
- Scalable storage system for storing the images
- A mechanism to notify all users at once about new images, and fetch them
- Build the Authentication and Listing views to Android client

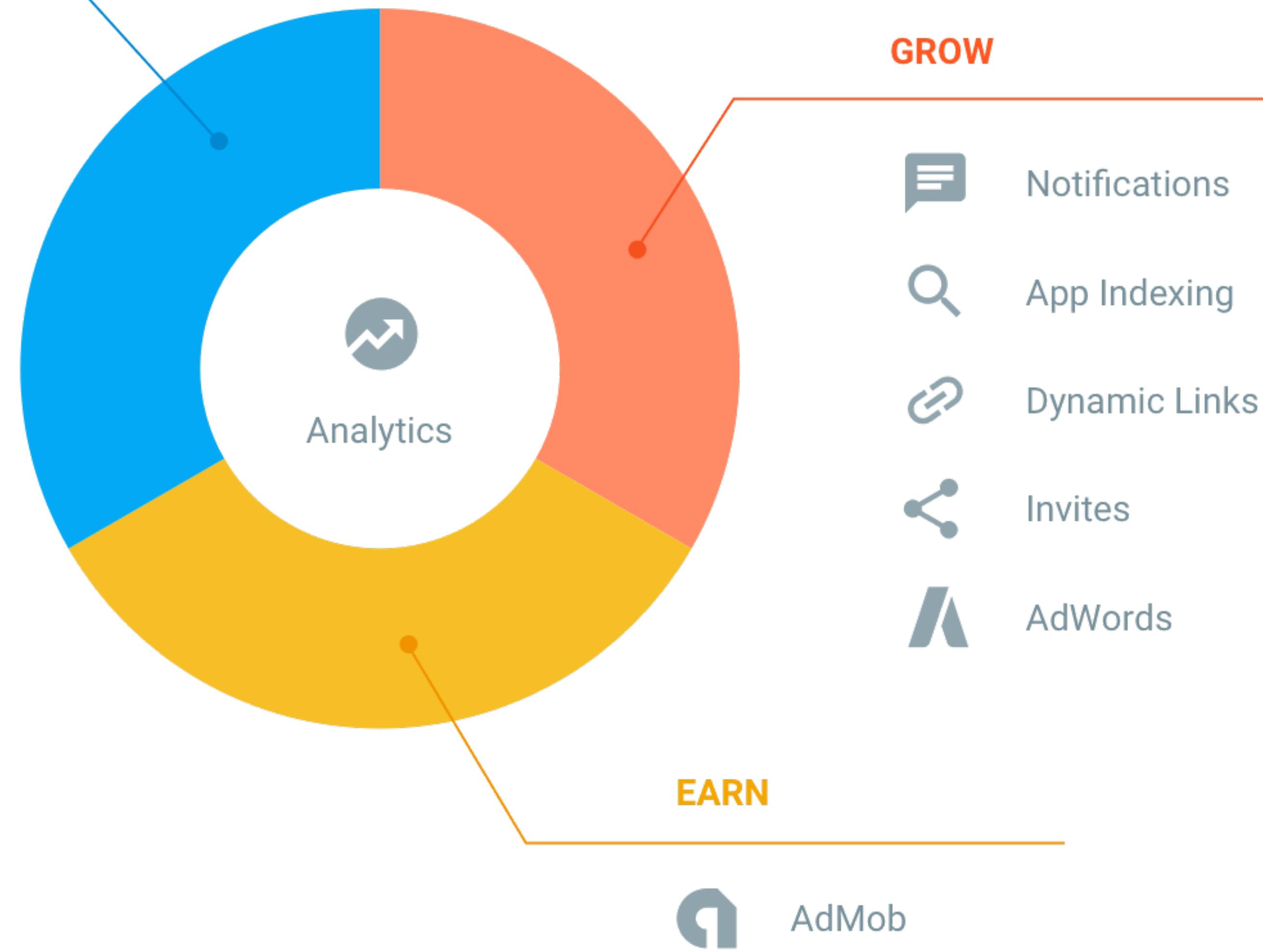


**BUT ALL THAT WILL TAKE  
AT LEAST DAYS**

WHAT IF I TOLD YOU,  
**NOT NECESSARILY**

## DEVELOP

-  Realtime Database
-  Authentication
-  Cloud Messaging
-  Storage
-  Hosting
-  Remote Config
-  Test Lab
-  Crash Reporting



<https://firebase.google.com>

# LETS JUST USE FOLLOWING

- Firebase Auth
- Firebase Storage
- Firebase Realtime Database
- FirebaseUI  
(<https://github.com/firebase/firebaseui>)



# NOW WE HAVE SOLVED

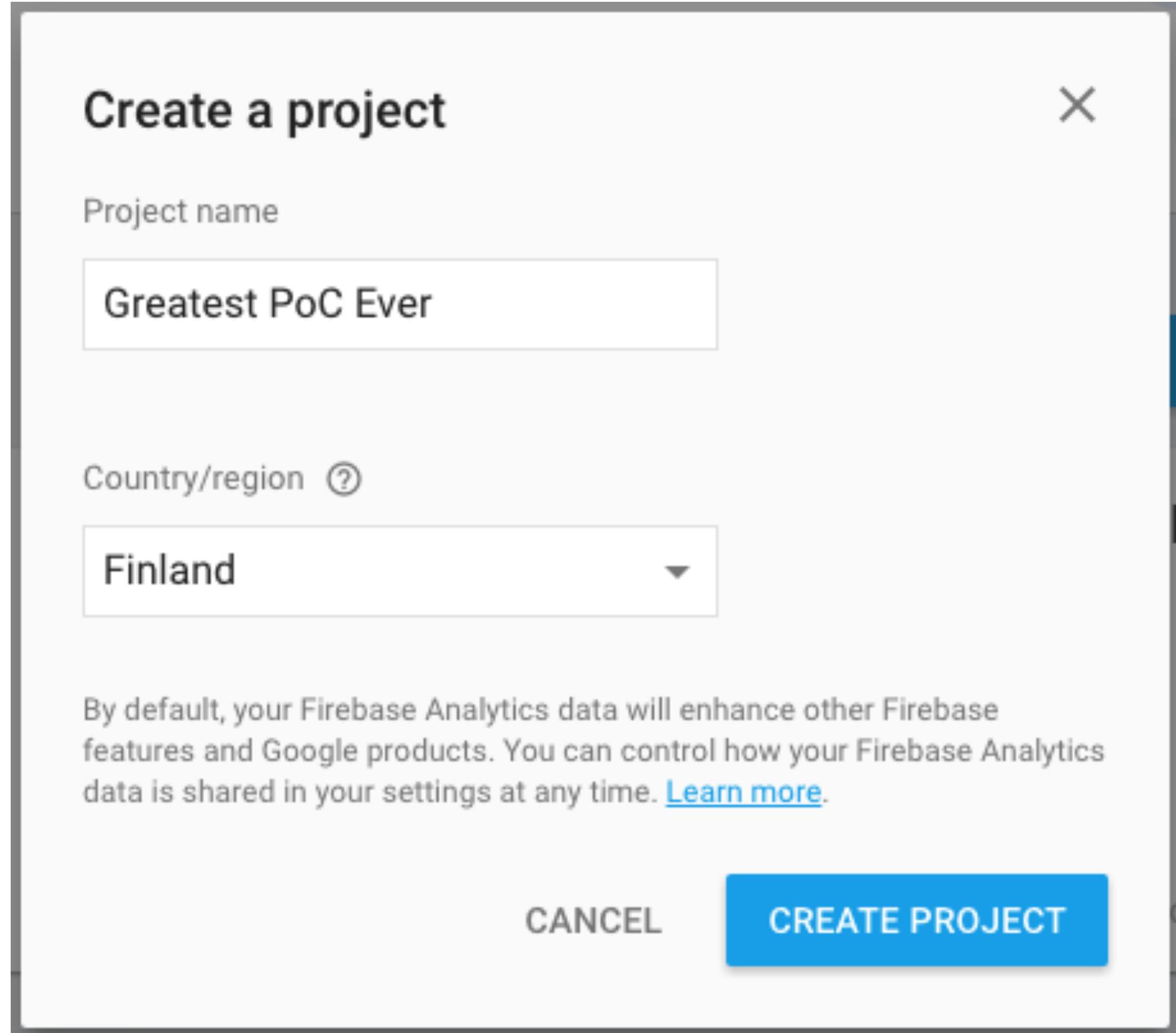
- Some server to handle our uploads and other api requests
- Scalable storage system for storing the images
- A mechanism to notify all users at once about new images, and fetch them
- For UI, we will use ready components as much as possible



**LETS START  
BUILDING**

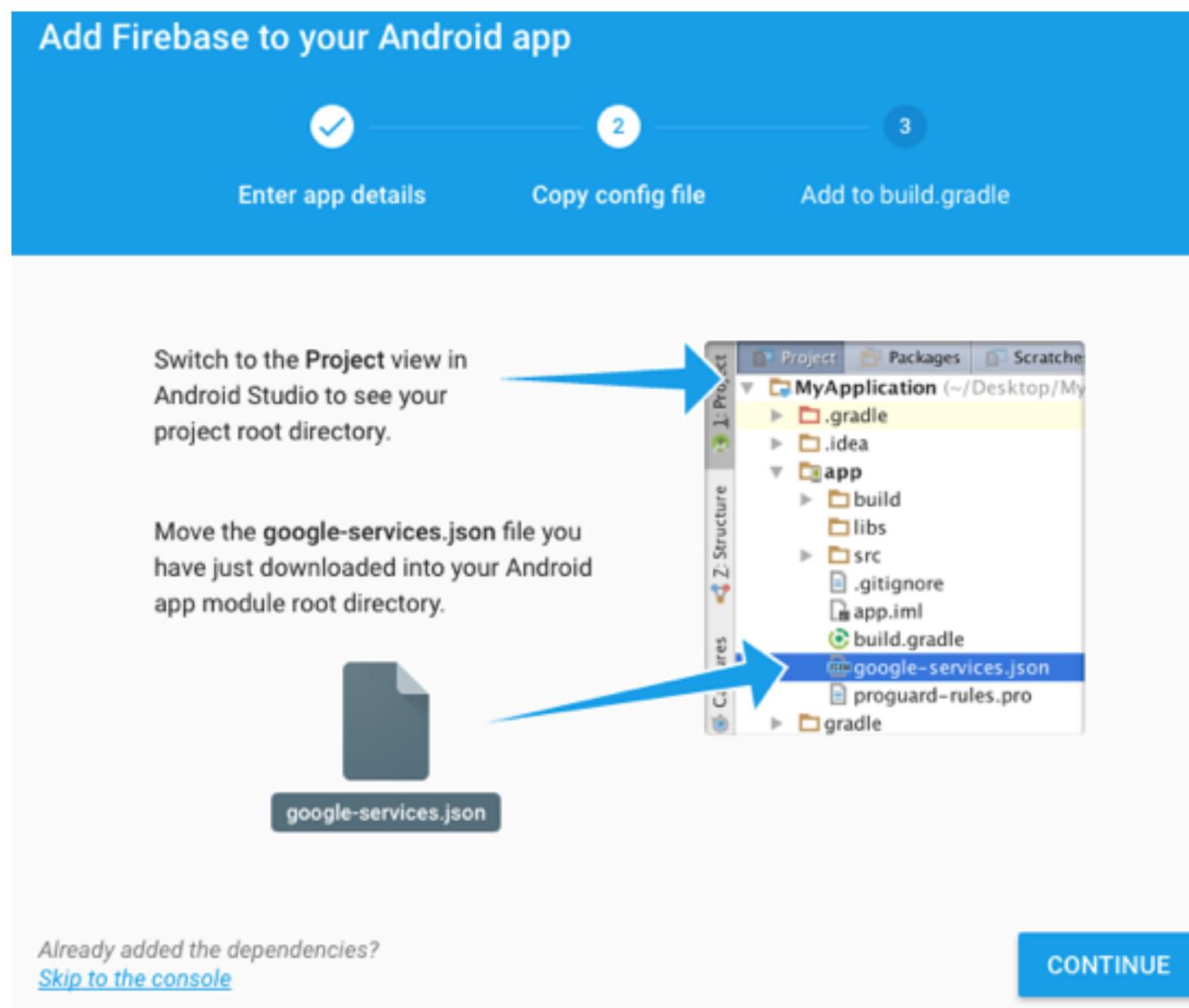
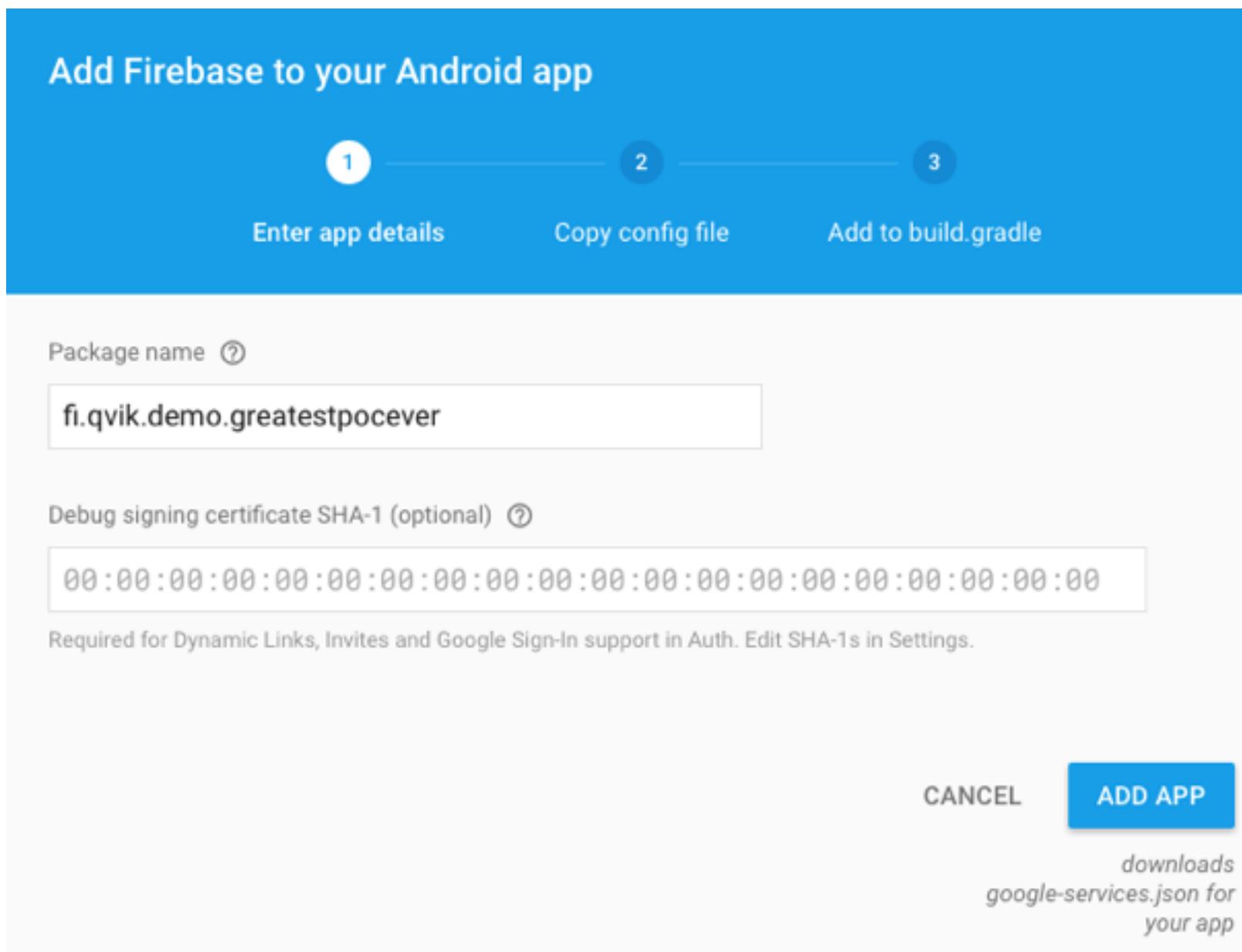
## 1. LETS ADD FIREBASE TO OUR APP

We head to <https://console.firebaseio.google.com/> and create new project



# 1. LETS ADD FIREBASE TO OUR APP

Next we create and download Firebase configuration for our app



The Google services plugin for [Gradle](#) loads the google-services.json file that you just downloaded. Modify your build.gradle files to use the plugin.

1. Project-level build.gradle (<project>/build.gradle):

```
buildscript {  
    dependencies {  
        // Add this line  
        classpath 'com.google.gms:google-services:3.0.0'  
    }  
}
```
2. App-level build.gradle (<project>/<app-module>/build.gradle):

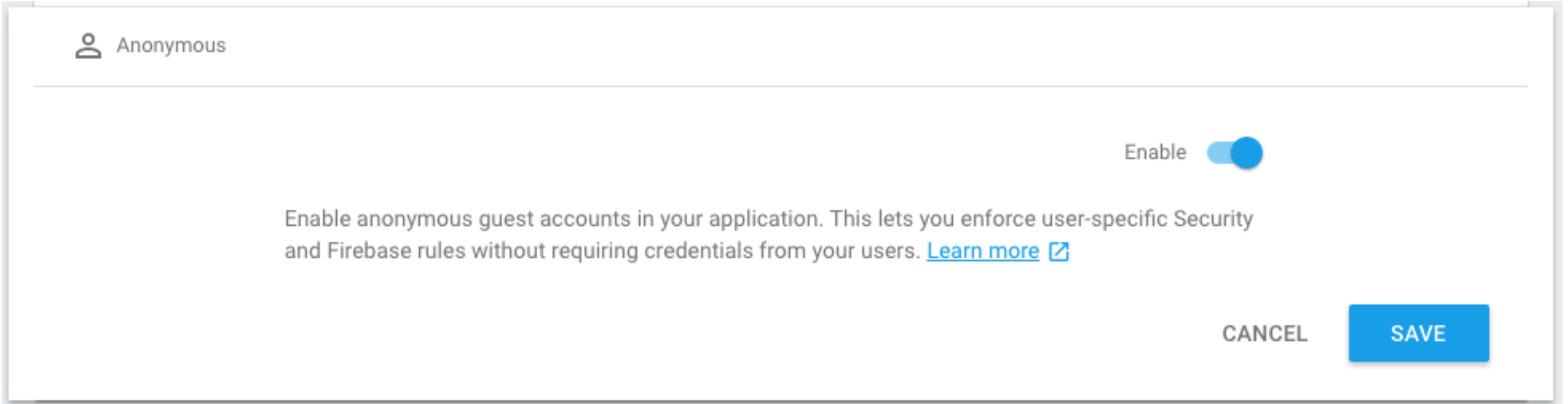
```
...  
// Add to the bottom of the file  
apply plugin: 'com.google.gms.google-services'  
includes Firebase Analytics by default
```
3. Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed since Sync now FINISH

```
{  
  "project_info": {  
    "project_number": "567854574506",  
    "firebase_url": "https://greatest-poc-ever.firebaseio.com",  
    "project_id": "greatest-poc-ever",  
    "storage_bucket": "greatest-poc-ever.appspot.com"  
  },  
  "client": [  
    {
```

## 2. LETS ENABLE AUTHENTICATION

We head to Authentication / Sign-In method config and enable Anonymous auth



### **3. LETS BUILD THE BASIC UI**

(Create some amazing UI by writing layout XML)

## 4. FETCH THE ADDED IMAGES WITH LIVE CHANGES

Here is just one simple way of listening for changes in our images collection

```
private FirebaseDatabase mDatabase;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Initialize Firebase Database
    mDatabase = FirebaseDatabase.getInstance();
    // Get reference to our images database collection
    DatabaseReference imagesRef = mDatabase.getReference("images");
    // Listen for
    imagesRef.limitToFirst(10).addChildEventListener(new ChildEventListener() {
        @Override
        public void onChildAdded(DataSnapshot dataSnapshot, String previousChildName) {
            // New image has been added...
        }

        @Override
        public void onChildRemoved(DataSnapshot dataSnapshot) {}

        @Override
        public void onChildChanged(DataSnapshot dataSnapshot, String previousChildName) {}

        @Override
        public void onChildMoved(DataSnapshot dataSnapshot, String previousChildName) {}

        @Override
        public void onCancelled(DatabaseError error) {}
    });
}
```

## 5. UPLOAD IMAGE TO STORAGE

Here is just one simple way of uploading a file to Firebase storage

```
private void upload(final Uri fileUri) {
    // Initialize Firebase Storage
    StorageReference mStorageRef = FirebaseStorage.getInstance().getReference();
    // Get a reference to store file at photos/<FILENAME>.jpg
    final StorageReference photosRef = mStorageRef.child("photos")
        .child(fileUri.getLastPathSegment());
    // Upload file to Firebase Storage
    photosRef.putFile(fileUri)
        .addOnSuccessListener(this, new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                // Upload succeeded, update details to our images collection...
            }
        });
}
```

**AND WE ARE DONE**  
**WELL, ALMOST**



LIVE DEMO

**SCAN THE QR CODE TO INSTALL**

[HTTPS://GITHUB.COM/JERRYJJ/FIREBASE-STORAGE-  
DEMO-ANDROID](https://github.com/jerryjj/firebase-storage-demo-android)



**QVIK**  
**CREATES**  
**MEANINGFUL**  
**SERVICES!**  
**WOULD YOU LIKE TO**  
**JOIN US?**

# QVIK

THANK YOU

[www.qvik.fi](http://www.qvik.fi)

