

Mobile Application Development

Produced
by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology

<http://www.wit.ie>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE





Android & Firebase

Part 4

Firebase Integration





Google Services Overview

- ❑ Overview of **Google Play Services** and Setup
- ❑ Detailed look at
 - Google+ Sign-in and Authentication (Part 1)
 - Location & Geocoding (Part 2)
 - Google Maps (Part 3)
- Firebase Integration (Part 4) (NEW)



Google Services Overview

- ❑ Detailed look at
 - **Firebase Integration (Part 4) (NEW)**



Agenda

- ❑ Firebase history
- ❑ The all new Firebase
- ❑ Real-time database
- ❑ Authentication
- ❑ Storage
- ❑ Remote config
- ❑ Hosting
- ❑ Crash reporting
- ❑ Test lab
- ❑ Firebase cloud messaging
- ❑ Dynamic links
- ❑ App indexing
- ❑ Analytics
- ❑ CoffeeMate Highlights & Demos along the way...



History of firebase

- ❑ Originally firebase was an online chat message integration service.
- ❑ Later the real time architecture was separated to create firebase database in 2012.
- ❑ Firebase surfaced as a popular choice when [Parse.com](#) went down.
- ❑ Google acquired firebase in 2014 and added a whole lot of features in *Google IO-2016*.
- ❑ Firebase is now a complete BaaS solution.



Introducing Firebase



What Is It?

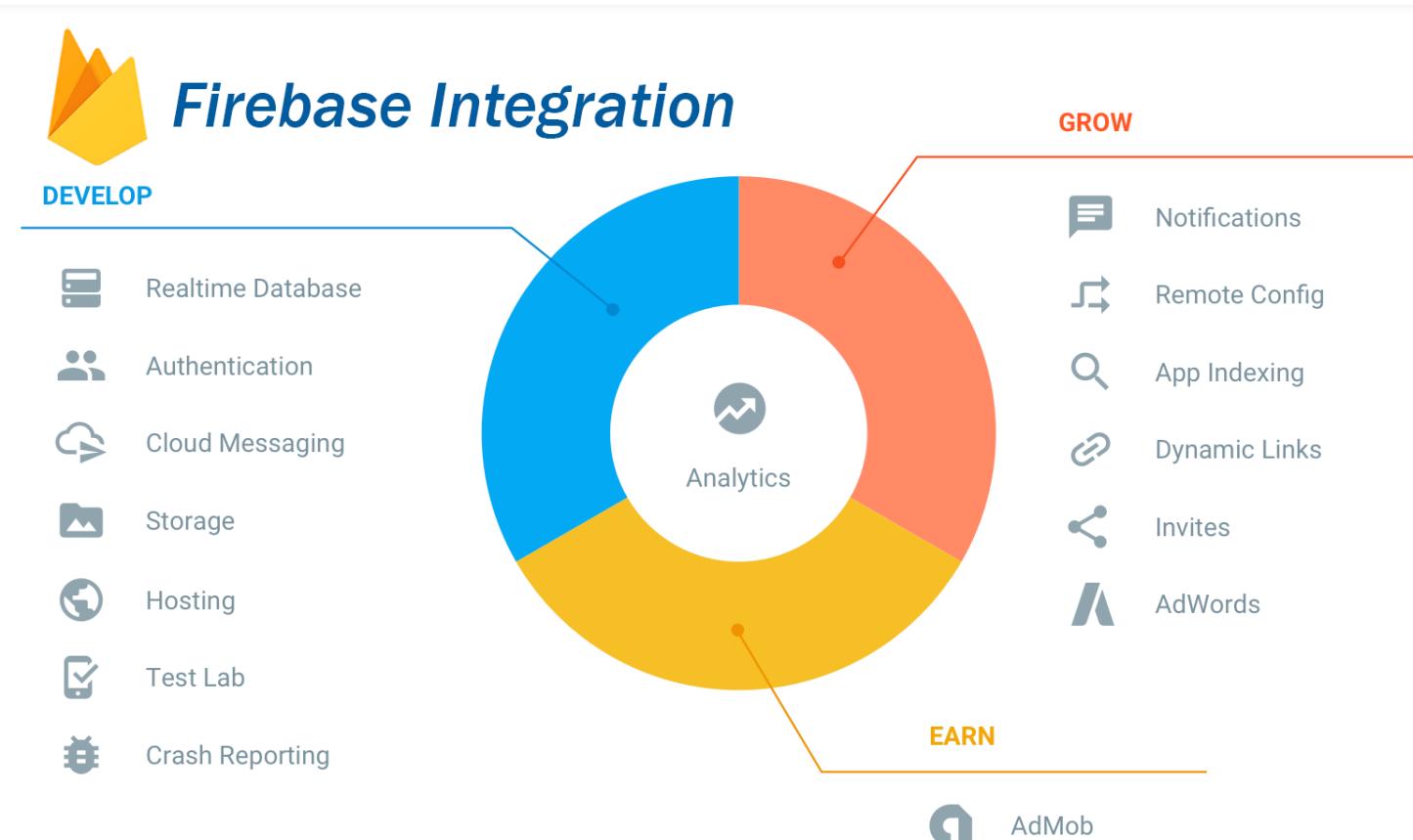
- Cloud database ?
- Another name for Google app engine?
- Cross platform solution ?
- Another name for GCM ?
- Analytics ?
- Virtual private server ?
- @^#%\$&!<*>^%(\$....





What It Is ... *

- ❑ A complete BaaS solution that includes
 - Real time JSON database
 - Authentication
 - Cloud storage
 - Cloud messaging
 - Crash reporting and analytics
 - And much much more





Firebase Products

Develop & test your app



Realtime Database

Store and sync app data in milliseconds



Crash Reporting

Find and prioritize bugs; fix them faster



Authentication

Authenticate users simply and securely



Cloud Functions

Run mobile backend code without managing servers



Cloud Storage

Store and serve files at Google scale



Hosting

Deliver web app assets with speed and security



Test Lab for Android

Test your app on devices hosted by Google



Performance Monitoring

Gain insight into your app's performance

Grow & engage your audience



Google Analytics

Get free and unlimited app analytics



Cloud Messaging

Send targeted messages and notifications



Dynamic Links

Drive growth by using deep links with attribution



Remote Config

Modify your app without deploying a new version



Invites

Make it easy to share your app and content



App Indexing

Drive search traffic to your mobile app



AdMob

Maximize revenue with in-app ads



AdWords

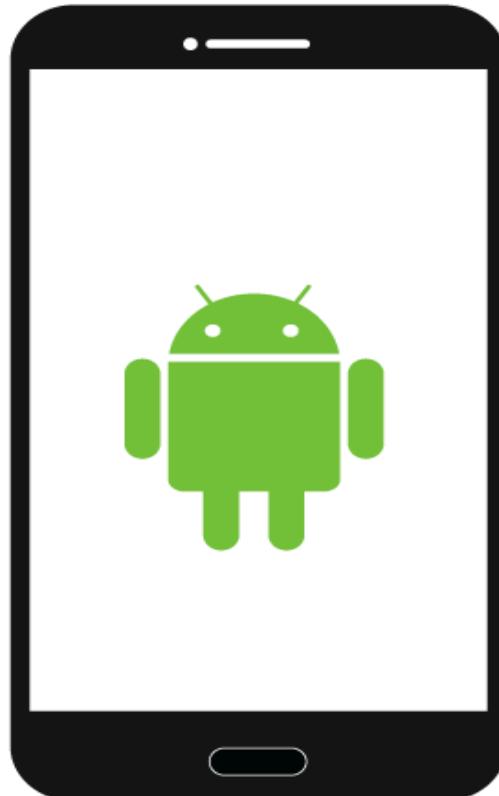
Drive installs with targeted ad campaigns



Idea For An App

What people think...

Create an android app
what's the big deal??





Idea For An App *

In Reality



Web app



backend



In Reality

- That's a lot to learn
- A lot of code
- A lot of concerns
- A lot of resources
- A lot of maintenance
- And the most dangerous of all – a lot of unknowns



And What About... *

Infrastructure

Scalability

Security

Reporting

Analytics

Pricing

Monetization

Testing

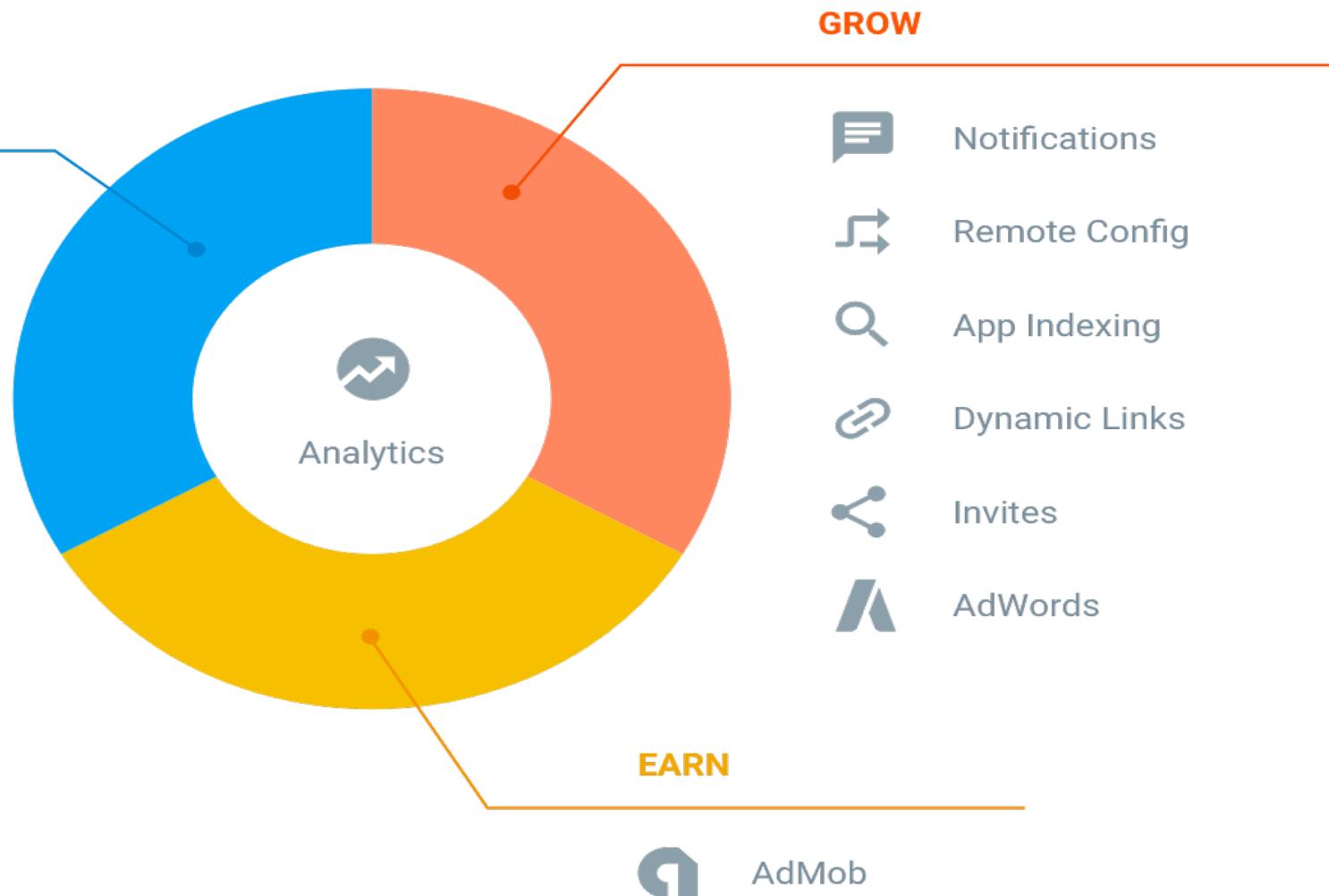
Authentication



Enter The New Firebase

DEVELOP

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





Database



Goodbye RDBMS...



Real Time Database

- ❑ Unlike RDBMS, data is stored as JSON. It is no-SQL JSON database.
- ❑ What makes it real time is its ability to notify listeners of any change in data.
- ❑ Whenever any change is made in the JSON database structure, the firebase SDK notifies the app.
- ❑ So you can forget about REST API calls, connectivity checks, 3rd party libraries and polling.



The Core Magic Of Firebase *





Saving Data In Firebase

- ❑ Get database reference (your base node)
 - `DatabaseReference mDBRef = FirebaseDatabase.getInstance().getReference()`

 - ❑ Point it to the right JSON node
 - `mDBRef = mDBRef.child("mydb").child("table1");`

 - ❑ Set the value at the pointed node
 - `mDBRef.child('row1').setValue(myPOJO);`

 - ❑ Or push the value to create a unique key and set the value
 - `mDBRef.push().setValue(myPOJO);`
- Create a Key
- Firebase takes care of the Key



View Your Saved Data

- ❑ Log on to <https://firebase.google.com>.
- ❑ Go to console and select your project.
- ❑ Hit database and select data tab

The screenshot shows the Firebase console interface. On the left, a sidebar lists various services: Authentication, Database (which is selected and highlighted with a red box), Storage, Hosting, Functions, Test Lab, Crash Reporting, and Performance. Below this is a 'GROW' section with Notifications and Remote Config. The main area shows a database tree under the project 'coffeematefbi'. The 'user-coffees' node contains a child node '1SQVbMgN5bcLg9JXgG0ts3DNOAV2', which in turn has a child node '-Kpt-S0GNxNFpsOBSzS-'. This node contains the following data:

- address: "Co. Waterford X91 W29R Ireland"
- favourite: false
- googlephoto: "https://lh5.googleusercontent.com/-AXr-7Z4gX7k/..."
- latitude: 52.24
- longitude: -7.16



Retrieving Data

- ❑ Data is retrieved via callbacks listeners.
- ❑ There are mainly 2 types of listeners
 - [ValueEventListener](#) – get entire child structure.
 - [ChildEventListener](#) – get only the child that got changed / added or deleted.
- ❑ Attach these listeners to [Database reference](#) we saw earlier.
- ❑ Both of these listeners are called once, so app can get the data and prepare UI



Attaching Data Listeners

- ❑ ValueEventListeners can be added in 2 ways:
 - **addListenerForSingleValueEvent (valEvListener);** //next slide
 - ◆ *Get the entire data snapshot only once*
 - **addValueEventListener(valEvListener);**
 - ◆ *Get entire data snapshot whenever there is a change in any of the child nodes*
- ❑ ChildEventListener can be added in only one way:
 - **addChildEventListener(childEvListener)**
 - ◆ *Get updates as child nodes*



Retrieving Data As POJO

```
public ChildEventListener childEvListener = new ChildEventListener() {  
  
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {  
        MyPOJO m = dataSnapshot.getValue(MyPOJO.class);  
    }  
  
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}  
  
    public void onChildRemoved(DataSnapshot dataSnapshot) {}  
  
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}  
  
    public void onCancelled(DatabaseError databaseError) {}  
};
```



Retrieving Data As POJO

```
public ValueEventListener valEvListener = new ValueEventListener() {  
    @Override  
    public void onDataChange(DataSnapshot dataSnapshot) {  
        MyPOJO m = dataSnapshot.getValue(MyPOJO.class);  
    }  
  
    @Override  
    public void onCancelled(DatabaseError databaseError) {}  
};
```



Filtering Data

- ❑ Set DatabaseReference to the right parent node.
 - Db = base.child('Institute').child('year');
- ❑ To get all students with major = 'Computing'.
 - Set orderByChild('major').
 - Set equalTo('Computing');
 - Add listeners

```
Db.orderByChild('major').equalTo('Computing')  
.addChildEventListener(childEvListener);
```



Pagination

- ❑ Create a [Query](#) object or keep using [DatabaseReference](#).
Query is super class of DatabaseReference class.
- ❑ Set `orderBy` some child key (and filter if needed)
- ❑ Set limits (`startAt`, `limit` etc.)
- ❑ Attach data listeners
- ❑ And done. Appropriate callback will be called when operation is complete.



Pagination Code

```
// class level
final int limit = 50;
int start = 0;

// event level
Query userListQuery = userDBRef.orderByChild("email")
.limitToFirst(limit).startAt(start);
userListQuery.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot d) {
        // Do something
        start += (limit+1);
```



Firebase Realtime Database



Quick DEMO...



CoffeeMateFBI 1.0

**Setup
&
Code
Highlights**



1. Create a Firebase Project *

The screenshot shows the Firebase console homepage. At the top, there's a navigation bar with various links like Lynda.com, Android Development, and a search bar for 'console.firebaseio.google.com'. Below the header, a large banner features a cartoon character holding a pencil, with the text 'Welcome back to Firebase' and 'Continue building your apps with Firebase using some of the resources below.' Below the banner, there are links for Documentation, Sample code, API reference, and Support. A red box highlights the 'CREATE NEW PROJECT' button, which is blue with white text. To the right of it is another button labeled 'IMPORT GOOGLE PROJECT'. On the left, there's a section titled 'Your projects using Firebase' showing a project named 'PaceMaker' with its URL 'pacemaker-a88f1.firebaseio.com'.



1. Create a Firebase Project

The screenshot shows a web browser window for the Firebase console at console.firebaseio.google.com. The title bar includes the URL and several tabs. The main page has a "Welcome back to Firebase" message and links for "Documentation" and "Sample code". A modal dialog box is open in the center, titled "Create a project". It contains fields for "Project name" (set to "CoffeeMate FBI") and "Country/region" (set to "Ireland"). Below these fields is a note about Firebase Analytics data sharing, with a "Learn more" link. At the bottom of the dialog are "CANCEL" and "CREATE PROJECT" buttons.

Welcome back to Firebase

Continue building your apps with F

some of the resources below.

Documentation <> Sample code

Your projects using Firebase

PaceMaker

pacemaker-a88f1.firebaseio.com

Create a project

Project name

CoffeeMate FBI

Country/region ②

Ireland

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 any time. [Learn more](#).

CANCEL CREATE PROJECT

IMPORT GOOGLE PROJECT



1. Create a Firebase Project *

The screenshot shows the Firebase console homepage. At the top, there's a navigation bar with various links like Lynda.com, Android Development, and a search bar for 'console.firebaseio.google.com'. Below the header, a blue banner says 'Welcome back to Firebase' with a small illustration of a person working on a large smartphone. There are links for Documentation, Sample code, API reference, and Support. In the center, under 'Your projects using Firebase', there are two project cards. The first card, 'CoffeeMate FBI', is highlighted with a red rounded rectangle. It contains the project name, a preview thumbnail, and the URL 'coffeemate-fbi.firebaseio.com'. The second card, 'PaceMaker', also has a preview thumbnail and the URL 'pacemaker-a88f1.firebaseio.com'. At the bottom right of the page are two buttons: 'CREATE NEW PROJECT' and 'IMPORT GOOGLE PROJECT'.



1. Create a Firebase Project - Overview

The screenshot shows the Firebase console's Overview page for the "CoffeeMate FBI" project. The left sidebar lists various services: Analytics, Authentication, Database, Storage, Hosting, Test Lab, and Crash Reporting under the "DEVELOP" section; and Notifications and Remote Config under the "GROW" section. At the bottom, there's a "Spark Free \$0/month" plan and an "UPGRADE" button. The main content area features a welcome message "Welcome to Firebase! Get started here." and three prominent buttons: "Add Firebase to your iOS app" (with an iOS icon), "Add Firebase to your Android app" (with an Android icon), and "Add Firebase to your web app" (with a code icon). Below these are two smaller sections: "Discover Firebase" with an illustration of a person working on a laptop with a chart, and another section showing mobile devices and badges.



1. Create a Firebase Project - Overview

The screenshot shows the Firebase console's Overview page for the project "CoffeeMate FBI". The left sidebar lists various services: Analytics, Authentication, Database, Storage, Hosting, Test Lab, Crash Reporting, Notifications, and Remote Config. A "Spark" plan is shown as free \$0/month with an "UPGRADE" button. The main area displays four service cards: Analytics (with an illustration of a person at a laptop), Authentication (with an illustration of user profiles), Database (with an illustration of servers), and Storage (with an illustration of a smartphone and files). Each card has a "Learn more" link and a "GET STARTED" button.

console.firebaseio.google.com/project/coffeemate-fbi/overview

Firebase CoffeeMate FBI Overview Go to docs

Overview

Analytics

DEVELOP

- Authentication
- Database
- Storage
- Hosting
- Test Lab
- Crash Reporting

GROW

- Notifications
- Remote Config

Spark Free \$0/month UPGRADE

Analytics
Get detailed analytics to measure and analyse how users engage with your app

[Learn more](#) [GET STARTED](#)

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

[Learn more](#) [GET STARTED](#)

Database
Store and sync data in real time across all connected clients

[Learn more](#) [GET STARTED](#)

Storage
Store and retrieve user-generated content like images, audio and video without server-side code

[Learn more](#) [GET STARTED](#)



2. Add Firebase to your Android App *

Welcome to Firebase! Get started here.

iOS Add Firebase to your iOS app

Add Firebase to your Android app (highlighted with a red box)

</> Add Firebase to your web app

Discover Firebase

Spark Free \$0/month **UPGRADE**



2. Add Firebase to your Android App – Web Key *

The screenshot shows the Firebase Console settings for the project "CoffeeMate FBI". The "Web API Key" field is highlighted with a red box. The key value is "AlzaSyAe...K7ITP8".

Project name: CoffeeMate FBI

Public-facing name: CoffeeMate FBI

Project ID: coffeemate-fbi

Web API Key: AlzaSyAe...K7ITP8

Your apps: There are currently no apps in the project. CoffeeMate FBI

Add Firebase to your iOS app (iOS icon)

Add Firebase to your Android app (Android icon)

Add Firebase to your web app (HTML/CSS icon)



2. Add Firebase to your Android App *

Add Firebase to your Android app

1 2 3

Register app Download config file Add Firebase SDK

A An OAuth2 client already exists for this package name and SHA-1 in another project. You can omit the SHA-1 for now and [read more about this situation and how to resolve it.](#)

Android package name ?

App nickname (optional) ?

Debug signing certificate SHA-1 (optional) ?

Required for Dynamic Links, Invites and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

CANCEL REGISTER APP

in project **CoffeeMateFBI**

Remember to update/add to your google maps key if you change your package name (or else your map won't work!)



2. Add Firebase to your Android App *

Add Firebase to your Android app

1 Register app 2 Download config file 3 Add Firebase SDK

Android Studio instructions Alternatives: [Unity](#) [C++](#)

[Download google-services.json](#)

2. Switch to the Project view in Android Studio to see your project root directory.

3. Move the google-services.json file you have just downloaded into your Android app module root directory.

google-services.json

Already added the dependencies?
[Skip to the console](#)

CONTINUE



2. Add Firebase to your Android 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.

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

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

```
...  
// Add to the bottom of the file  
apply plugin: 'com.google.gms.google-services'  
include Firebase Analytics by default
```

Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed sit Sync now

FINISH



CoffeeMateFBI.1.0 Dependencies (July/2017) *

```
dependencies {  
    compile fileTree(include: ['*.jar'], dir: 'libs')  
    compile project(':volley')  
    compile 'com.android.support:appcompat-v7:25.4.0'  
    compile 'com.android.support:support-v4:25.4.0'  
    compile 'com.android.support:design:25.4.0'  
    compile 'com.makeramen:roundedimageview:2.2.1'  
    compile 'com.android.support.constraint:constraint-layout:1.0.2'  
    compile 'com.google.code.gson:gson:2.7'  
    compile 'com.google.android.gms:play-services-auth:11.0.2'  
    compile 'com.google.android.gms:play-services-maps:11.0.2'  
    compile 'com.google.android.gms:play-services-location:11.0.2'  
  
    compile 'com.google.firebaseio:firebase-core:11.0.2'  
    compile 'com.google.firebaseio:firebase-auth:11.0.2'  
    compile 'com.google.firebaseio:firebase-database:11.0.2'  
    compile 'com.firebaseui:firebase-ui-database:2.1.0'  
    testCompile 'junit:junit:4.12'  
}
```



FBDBManager – our Firebase Database utility class *

```
public class FBDBManager {  
  
    private static final String TAG = "coffeemate";    □ Our Firebase db reference.  
    public DatabaseReference mFirebaseDatabase;  
    public static String mFBUserId;  
    public FBDBListener mFBDBListener;  
  
    public void open() {  
        //Set up local caching  
        FirebaseDatabase.getInstance().setPersistenceEnabled(true);  
  
        //Bind to remote Firebase Database  
        mFirebaseDatabase = FirebaseDatabase.getInstance().getReference();  
        Log.v(TAG, "Database Connected :" + mFirebaseDatabase);  
    }  
}
```



FBDBManager – usage *

```
public class CoffeeMateApp extends Application
{
    private RequestQueue mRequestQueue;
    private static CoffeeMateApp mInstance;
    //...

    public FBDBManager mFBDBManager;

    //...
    @Override
    public void onCreate() {
        super.onCreate();
        Log.v("coffeemate", "CoffeeMate App Started");
        mInstance = this;
        mRequestQueue = Volley.newRequestQueue(getApplicationContext());
        Log.v("coffeemate", "Adding Firebase offline persistence...");
        mFBDBManager = new FBDBManager();
        mFBDBManager.open();
    }
}
```

- ❑ Our utility class reference (inside our Application object class).
- ❑ Creating & ‘Opening’ a connection to our Firebase db.



FBDBManager – our Firebase Database utility class *

- ❑ ‘Query’ing the ‘*mFBUserId*’ node of the ‘*user-coffees*’ node.

```
public Query getAllCoffees()
{
    Query query = mFirebaseDatabase.child("user-coffees").child(mFBUserId)
        .orderByChild("rating");

    return query;
}

public Query getFavouriteCoffees()
{
    Query query = mFirebaseDatabase.child("user-coffees").child(mFBUserId)
        .orderByChild("favourite").equalTo(true);

    return query;
}
```

- ❑ As above, but only where the ‘*favourite*’ field is ‘*true*’.



FBDBManager – usage (in CoffeeFragment) *

```
@Override  
public void onResume() {  
    super.onResume();  
  
    query = app.mFBDBManager.getAllCoffees();  
  
    if(favourites)  
        query = app.mFBDBManager.getFavouriteCoffees();  
  
    updateUI(query);  
}
```

- Returning a ‘query’ of all coffees, (or favourite coffees) which we pass to our custom firebaseUI adapter via *updateUI()*



FBDBManager – our Firebase Database utility class *

- Adding an ‘ValueEventListener’ and triggering our callback

```
public void getACoffee(final String coffeeKey)
{
    mFirebaseDatabase.child("user-coffees").child(mFBUserId).child(coffeeKey)
        .addValueEventListener(
            new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    Log.v(TAG, "The read Succeeded: " + dataSnapshot.toString());
                    mFBDBListener.onSuccess(dataSnapshot);
                }
                @Override
                public void onCancelled(DatabaseError firebaseError) {
                    Log.v(TAG, "The read failed: " + firebaseError.getMessage());
                    mFBDBListener.onFailure();
                }
            });
}
```



FBDBManager – usage (in EditFragment) *

```
@Override  
public void onResume() {  
    super.onResume();  
  
    app.mFBDBManager.attachListener(this);  
  
    if(getArguments() != null) {  
        coffeeKey = getArguments().getString("coffeeKey");  
        app.mFBDBManager.getACoffee(coffeeKey);  
    }  
    titleBar = (TextView) getActivity()  
        .findViewById(R.id.recentAddedBarTextView);  
    titleBar.setText("Update a Coffee");  
}
```

- ☐ Retrieving the ‘coffeeKey’ from the bundle and calling ‘getACoffee()’ - which in turn triggers the callback on the edit screen



Firebase Console (actual data)

The image shows the Firebase Realtime Database console on the left and an Android emulator on the right.

Firebase Realtime Database Screenshot:

- The database structure under "user-coffees" includes nodes for multiple coffee shops, such as "1SQVbMgN5bcLg9JXgG0ts3DNOAV2" and "C3PpngqTi8OxuHvNtG4wZsVMhZv2".
- The node "1SQVbMgN5bcLg9JXgG0ts3DNOAV2" is highlighted with a red box and contains the following data:
 - KpRYrfoaJaqJZGt9Mx6
 - address: "191 Hennessy's Road Waterford X91 PXA4"
 - favourite: false
 - googlephoto: "https://lh5.googleusercontent.com/-AXr-7Z4gX7k/..."
 - latitude: 52.25
 - longitude: -7.126
 - name: "Firebase Coffee"
 - price: 1.99
 - rating: 2
 - shop: "Fire Station"
 - uid: "1SQVbMgN5bcLg9JXgG0ts3DNOAV2"
 - usertoken: "113437677814759908125"
 - KpRZ845g8sLB4TDNo0w
 - address: "5 Lismore Park Waterford Ireland"
 - favourite: false
 - googlephoto: "https://lh5.googleusercontent.com/-AXr-7Z4gX7k/..."
 - latitude: 52.25
 - longitude: -7.1399983
 - name: "Coffee Latte"
 - price: 2.49
 - rating: 2
 - shop: "Aldi"
 - uid: "1SQVbMgN5bcLg9JXgG0ts3DNOAV2"
 - usertoken: "113437677814759908125"

Android Emulator Screenshot:

- The emulator displays the "Recently Added Coffee's" screen from the "CoffeeMateFBI" app.
- The list shows two items:
 - 1. Firebase Coffee (Fire Station) - €1.99 (2.9+)
 - 2. Coffee Latte (Aldi) - €2.49 (2.9+)



Firebase Console (actual data)

The screenshot shows the Firebase Realtime Database interface and an Android emulator side-by-side.

Firebase Realtime Database Screenshot:

- The database structure under `user-coffees` includes nodes for multiple users (e.g., `1SQVbMgN5bcLg9JXgG0ts3DNOAV2`, `-KpRYrfoAJaqJZGt9Mx6`, `-KpRZ845g8sLB4TDN0oW`, `C3PpngqTl8OxuHvNtG4wZsVMhZv2`, `wotuVu5IZjh0HRcCHRkJgsWh9y1`).
- The node `-KpRYrfoAJaqJZGt9Mx6` contains detailed information about a coffee shop:
 - `address: "191 Hennessy's Road Waterford X91 PXA4"`
 - `favourite: true` (highlighted with a red box)
 - `googlephoto: "https://lh5.googleusercontent.com/-AXr-7Z4gX7k/..."`
 - `latitude: 52.25`
 - `longitude: -7.126`
 - `name: "Firebase Coffee"`
 - `price: 1.99`
 - `rating: 2`
 - `shop: "Fire Station"`
 - `uid: "1SQVbMgN5bcLg9JXgG0ts3DNOAV2"`
 - `usertoken: "113437677814759908125"`

Android Emulator Screenshot:

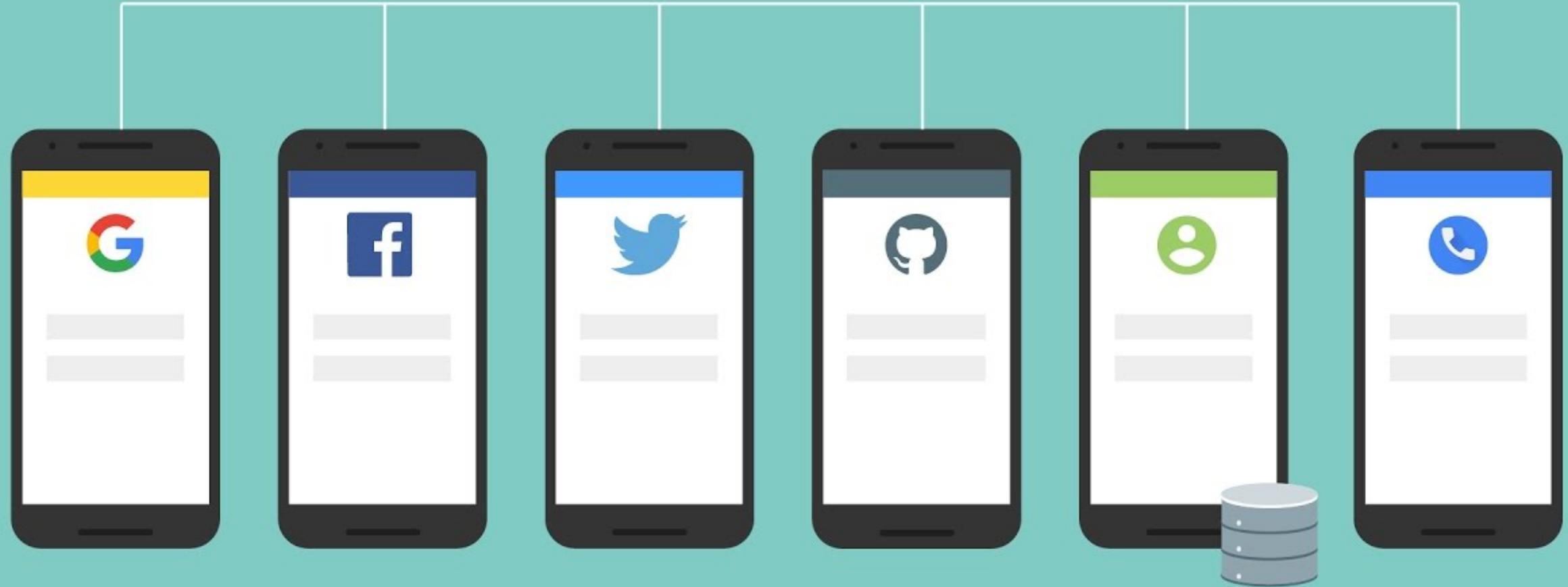
- The emulator displays the `CoffeeMateFBI` application.
- The screen shows a list titled `Recently Added Coffee's` with two items:
 - `Firebase Coffee` (`Fire Station`) - marked with a yellow star (highlighted with a red box)
 - `Coffee Latte` (`Aldi`)
- The bottom of the screen shows the URL `ddrohan.github.io` and a blue info icon.



Pricing

<https://firebase.google.com/pricing/>

Products	Spark Plan Generous limits for hobbyists Free	Flame Plan Fixed pricing for growing apps \$25/month	Blaze Plan Calculate pricing for apps at scale Pay as you go
Free Products Authentication (except Phone Auth), Analytics, App Indexing, Dynamic Links, Invites, Remote Config, Cloud Messaging (FCM), Performance Monitoring, and Crash Reporting.	Included	Included Free	Included Free
Realtime Database Simultaneous connections	100	100K/instance	100K/instance
GB stored	1 GB	2.5 GB	\$5/GB
GB downloaded	10 GB/month	20 GB/month	\$1/GB
Automated backups			

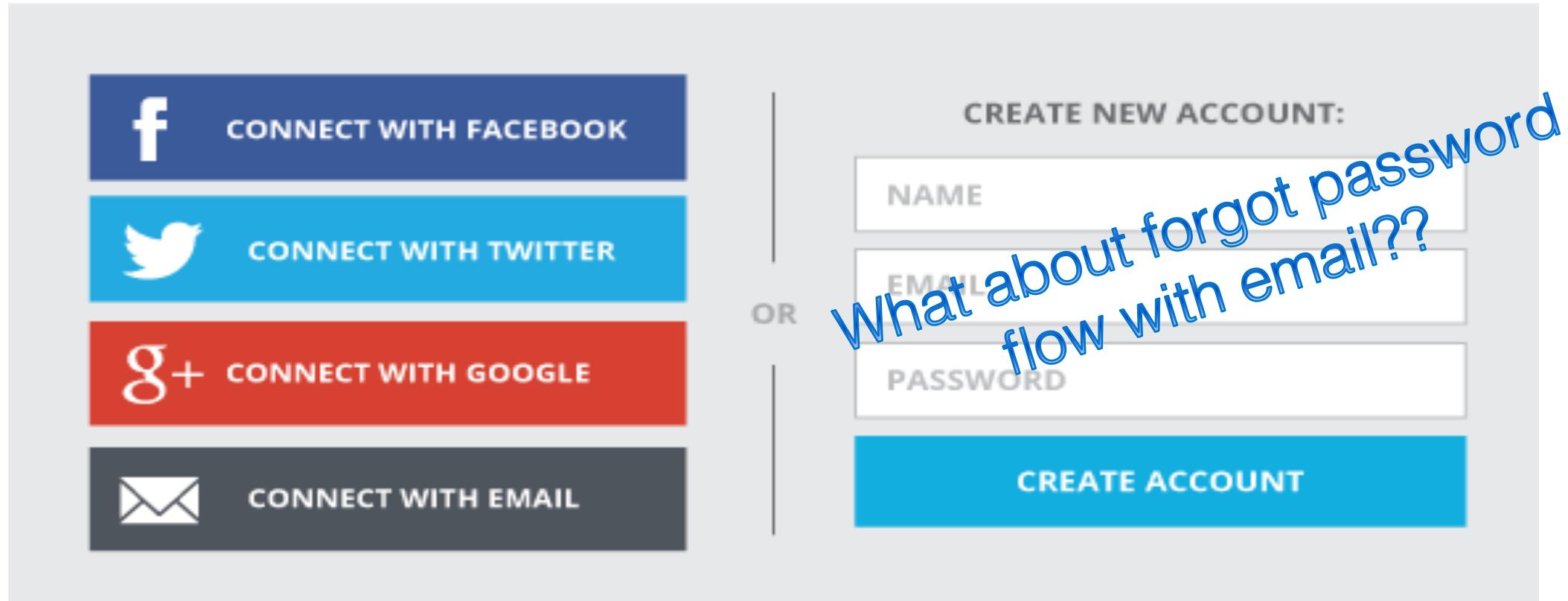


All goodness bundled as one...
Authentication



Firebase Authentication *

A type of screen present in almost all apps these days...



How long would it take you to develop this???



Firebase Authentication

- ❑ Integrate easily with popular identity providers like google, twitter, facebook and more





Firebase Authentication

- ❑ Of course you will need to register your app with individual service providers.
- ❑ Minimal client side handling, integrates seamlessly with firebase
- ❑ Ready made ‘forgot password’ flow with customizable email template



Firebase Authentication



Quick DEMO...



CoffeeMateFBI 1.0

**Setup
&
Code
Highlights**



1. Setup your Sign-In Method *

The screenshot shows the Firebase console interface for setting up sign-in methods. The left sidebar has a red box around the 'Authentication' item under the 'DEVELOP' section. The main content area has a red box around the 'SIGN-IN METHOD' tab in the navigation bar. Below the navigation bar is a search bar and a table for managing users. To the right of the table is a descriptive text and a 'SET UP SIGN-IN METHOD' button.

console.firebaseio.google.com/project/coffeemate-fbi/authentication/use

Firebase CoffeeMate FBI

Overview Analytics Authentication

DEVELOP

Database Storage Hosting Test Lab Crash Reporting

GROW

Notifications Remote Config

Spark Free \$0/month UPGRADE

Authentication

USERS SIGN-IN METHOD EMAIL TEMPLATES

Search by email address or user UID

Email Providers Created Signed In User UID ↑

ADD USER

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

[Learn more](#)

SET UP SIGN-IN METHOD



1. Setup your Sign-In Method *

The screenshot shows the Firebase Authentication screen. On the left, there's a sidebar with links like Overview, Analytics, Database, Storage, Hosting, Test Lab, Crash Reporting, Notifications, and Remote Config. The Authentication link is selected. The main area is titled "Authentication" and has tabs for USERS, SIGN-IN METHOD (which is selected), and EMAIL TEMPLATES. Below this, it says "Sign-in providers". A table lists providers with their status:

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

At the bottom, it says "OAuth redirect domains" with a help icon.



1. Setup your Sign-In Method *

The screenshot shows the Firebase Authentication screen. On the left sidebar, 'Authentication' is selected under 'DEVELOP'. The main area displays the 'SIGN-IN METHOD' tab. It lists 'Email/Password' as 'Disabled'. Below this, the 'Google' provider is listed with its status as 'Enabled'. A red box highlights the 'Enable' toggle switch for the Google provider. A note below states: 'Google sign-in is automatically configured on your connected iOS and web apps. To set up Google sign-in for your Android apps, you need to add the [SHA1 fingerprint](#) for each app on your [Project Settings](#)'. There are also sections for 'Whitelist client IDs from external projects (optional)' and 'Web SDK configuration (optional)'.



1. Setup your Sign-In Method *

The screenshot shows the Firebase Authentication console for the project "CoffeeMate FBI". The left sidebar includes links for Overview, Analytics, DEVELOP (Authentication, Database, Storage, Hosting, Test Lab, Crash Reporting), GROW (Notifications, Remote Config), and Spark (Free \$0/month). The main panel displays the "Authentication" section with "Provider" and "Status" columns. Under "Provider", "Email/Password" is listed as "Disabled". Below it, "Google" is listed with its status as "Enabled" (indicated by a blue toggle switch). A note states: "Google sign-in is automatically configured on your connected iOS and web apps. To set up Google sign-in for your Android apps, you need to add the [SHA1 fingerprint](#) for each app on your [Project Settings](#)". There are sections for "Whitelist client IDs from external projects (optional)" and "Web SDK configuration (optional)". At the bottom right, there are "CANCEL" and "SAVE" buttons, with "SAVE" being highlighted by a red rectangle.

Provider	Status
Email/Password	Disabled
Google	Enabled
Facebook	Disabled
Twitter	Disabled
Github	Disabled



2. Introduce Authentication Flow *

```
private void firebaseAuthWithGoogle(GoogleSignInAccount acct) {
    Log.v(TAG, "firebaseAuthWithGoogle:" + acct.getEmail());

    AuthCredential credential = GoogleAuthProvider.getCredential(acct.getIdToken(), null);
    app.mFirebaseAuth.signInWithCredential(credential)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                Log.v(TAG, "signInWithCredential:onComplete:" + task.isSuccessful());
                validateFirebaseUser();
                // If sign in fails, display a message to the user. If sign in succeeds
                // the auth state listener will be notified and logic to handle the
                // signed in user can be handled in the listener.
                if (!task.isSuccessful()) {
                    Log.v(TAG, "signInWithCredential", task.getException());
                    Toast.makeText(Login.this, "Authentication failed.",
                        Toast.LENGTH_SHORT).show();
                }
            }
        });
}
```



2. Introduce Authentication Flow *

```
private void validateFirebaseUser()
{
    Log.v(TAG, "Calling validateFirebaseUser() " );
    if(app.mFirebaseUser == null)
        app.mFirebaseUser = FirebaseAuth.getInstance().getCurrentUser();

    app.mFBDBManager.checkUser(app.mFirebaseUser.getUid(),
                               app.mFirebaseUser.getDisplayName(),
                               app.mFirebaseUser.getEmail());
}
```



2. Introduce Authentication Flow *

```
//Check to see if the Firebase User exists in the Database
//if not, create a new User
public void checkUser(final String userid,final String username,final String email) {
    Log.v(TAG, "checkUser ID == " + userid);
    mFirebaseDatabase.child("users").child(userid).addSingleValueEventListener(
        new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                mFBDBListener.onSuccess(dataSnapshot);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                mFBDBListener.onFailure();
            }
        });
}
```



2. Introduce Authentication Flow *

```
@Override  
public void onSuccess(DataSnapshot dataSnapshot) {  
    if(dataSnapshot.exists()){  
        Log.v(TAG, "User found : ");  
    }  
    else{  
        Log.v(TAG, "User not found, Creating User on Firebase");  
        User newUser = new User(app.mFirebaseUser.getUid(),  
                               app.mFirebaseUser.getDisplayName(),  
                               app.mFirebaseUser.getEmail(), null);  
        app.mFBDBManager.m FirebaseDatabase.child("users")  
            .child(app.mFirebaseUser.getUid())  
            .setValue(newUser);  
    }  
    app.mFBDBManager.mFBUserId = app.mFirebaseUser.getUid();  
    startHomeScreen();  
}
```



Firebase Console – Authenticated Users *

The screenshot shows the Firebase Authentication console interface. On the left, a sidebar lists various services: Overview, Analytics, Authentication (which is selected and highlighted with a red box), Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, AdMob, and Spark (Free \$0/month). The main area is titled 'Authentication' and contains tabs for 'USERS', 'SIGN-IN METHOD', and 'TEMPLATES'. A search bar at the top allows searching by email address, phone number, or user UID. Below the search bar is an 'ADD USER' button. A table displays the user information for 'daveydrohan@gmail.com', which was created on '18 Jul 2017' via 'Google'. The user's unique identifier ('User ID') is partially visible as '1SQVbMgN5bcLg9JXgG0ts3DNO...'. A red box highlights both the 'Authentication' sidebar item and the user row in the table.

Username	Provider	Created	Updated	User ID
daveydrohan@gmail.com	G	18 Jul 2017	18 Jul 2017	1SQVbMgN5bcLg9JXgG0ts3DNO...



Firebase Console – Authenticated Users *

The screenshot shows the Firebase Authentication screen. On the left, a sidebar lists various services: Overview, Analytics, Authentication (which is selected and highlighted with a red box), Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, AdMob, and Spark (Free \$0/month). The main area is titled 'Authentication' and contains tabs for 'USERS', 'SIGN-IN METHOD', and 'TEMPLATES'. A search bar at the top of the table says 'Search by email address, phone number or user UID'. Below it is a table with columns: Email/Phone, Provider, Created, Last Signed In, and User UID. Three rows of data are shown, each corresponding to a Google account (Gmail) with a timestamp of '18 Jul 2017' and a unique User UID. A red box highlights the table containing the user data.

Email/Phone	Provider	Created	Last Signed In	User UID
daveydrohan@gmail.com	G	18 Jul 2017	18 Jul 2017	1SQVbMgN5bcLg9JXgG0ts3DNO...
noahldrohan@gmail.com	G	18 Jul 2017	18 Jul 2017	C3PpngqTl8OxuhvNtG4wZsVMhZ...
joshuajdrohan@gmail.com	G	18 Jul 2017	18 Jul 2017	wotuVu5lZh0HRcCHRkJgsWh9y1



Firebase Console – User Data in db *

The screenshot shows the Firebase Realtime Database console. On the left, a sidebar lists various services: Overview, Analytics, Authentication (highlighted with a red box), Database (highlighted with a red box), Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, AdMob, and EARN. The main area is titled "Realtime Database" and shows the "DATA" tab selected. A modal window displays the URL <https://coffeematefbi.firebaseio.com/>. Inside the modal, a message states "Default security rules require users to be authenticated." with "FIND OUT MORE" and "DISMISS" buttons. Below this, the database structure is shown under the "coffeematefbi" root node:

```
coffeematefbi
  users
    1SQVbMgN5bcLg9JXgG0ts3DNOAV2
      userEmail: "daveydrohan@gmail.com"
      userId: "1SQVbMgN5bcLg9JXgG0ts3DNOAV2"
      userName: "David Drohan"
```



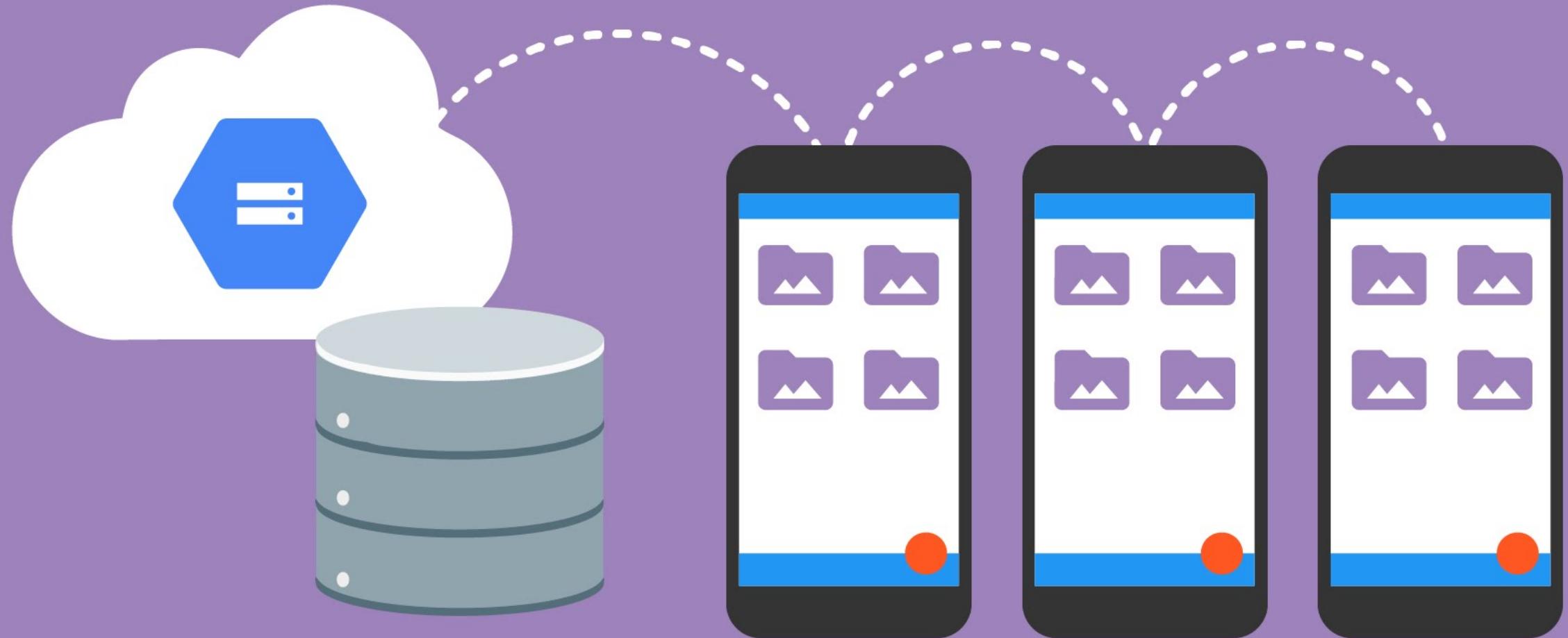
Firebase Console – User Data in db *

The screenshot shows the Firebase Realtime Database console for the project "coffeematefbi". The left sidebar lists various services: Overview, Analytics, Authentication (highlighted with a red box), Database (highlighted with a red box), Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, EARN, and AdMob. The main area displays the database structure under the "DATA" tab. A modal window is open, showing the URL <https://coffeematefbi.firebaseio.com/>. It contains a warning message: "Default security rules require users to be authenticated." with "FIND OUT MORE" and "DISMISS" buttons. Below the modal, the database structure is shown:

```
coffeematefbi
  +-- users
    +-- 1SQVbMgN5bcLg9JXgG0ts3DNOAV2
      +-- userEmail: "daveydrohan@gmail.com"
      +-- userId: "1SQVbMgN5bcLg9JXgG0ts3DNOAV2"
      +-- userName: "David Drohan"
    +-- C3PpngqTI8OxuHvNtG4wZsVMhZv2
      +-- userEmail: "noahldrohan@gmail.com"
      +-- userId: "C3PpngqTI8OxuHvNtG4wZsVMhZv2"
      +-- userName: "Noah Drohan"
    +-- wotuVu5IZjh0HRcCHRkJgsWh9y1
      +-- userEmail: "joshuajdrohan@gmail.com"
      +-- userId: "wotuVu5IZjh0HRcCHRkJgsWh9y1"
      +-- userName: "Joshua Everett Drohan"
```



Storage



Sync files and folders seamlessly...



Firebase Storage

- ❑ Store user's image, audio, video and other content in the cloud easily without worrying about the network quality.
- ❑ Firebase adds Google security to file uploads and downloads.
- ❑ Backed by Google Cloud Storage.
- ❑ Petabyte scalability available if your app goes viral.



Upload Files To Firebase

❑ After adding gradle dependencies. Get Storage reference

- `StorageReference storage = FirebaseStorage.getInstance().getReferenceFromUrl('gs:<bucket>');`
- `storage = storage.child('myimages/user1234pic.jpg');`

❑ Easily change reference.

- `storage.getParent(), getRoot(), getPath(), getName(), getBucket()` etc.

❑ Upload from memory, stream or from SD card

- `storage.putBytes(imageData); // as byte array.`
- `storage.putStream(stream); // as input stream`
- `Storage.putFile(fileURI): / Uri of the local file to be uploaded`

`gs://coffeematefbi.appspot.com`



More On File Upload

❑ Put file metadata

- `StorageMetadata metadata = new StorageMetadata.Builder().setContentType("image/jpg").build()`
- `storage.putFile(file, metadata)`

❑ Manage uploads

- `UploadTask task = storage.putFile(file, metadata)`
- `task.pause(), resume(), cancel()...`

❑ Monitor upload

- `OnProgressListener, OnPausedListener, OnSuccessListener, OnFailureListener`



Download Files From Firebase

❑ Download to a local file

- `File localFile = File.createTempFile('myfile', 'jpg');`
- `storage.getFile(localFile).addOnSuccessListener(...);`

❑ Download to memory

- `Final long ONE_MEG = 1024*1024;`
- `storage.getBytes(ONE_MEG).addOnSuccessListener(new OnSuccessListener<byte[]>() {`
`public void onSuccess(byte[] bytes){`
`}`
`});`

❑ Point StorageReference to desired file and call `getMetadata()` to get metadata.



Firebase Storage



Quick Example...

<https://github.com/kotlintpoint/Firebase-Storage-Upload-Download>



FirebaseStorageEx App *

- ❑ On first look, a very basic app to Upload/Download a file
- ❑ What it actually does is
 - Using System Intents, allow you to select a file from any source on your device, including your google drive
 - Allow you to name that file and upload it to your **Firebase Storage** space.
 - Allow you to download a file once the correct filename is supplied from your **Firebase Storage** and display it using third party apis (glide/picasso).



FirebaseStorageEx App - Setup *

Update your Firebase Storage rules to allow reading and writing without authentication (for demo purposes)

```
service firebase.storage {  
  match /b/{bucket}/o {  
    match /{allPaths=**} {  
      allow read, write: if request.auth != null;  
    }  
  }  
}
```

BEFORE

```
service firebase.storage {  
  match /b/{bucket}/o {  
    match /{allPaths=**} {  
      allow read, write: if true;  
    }  
  }  
}
```

AFTER



FirebaseStorageEx App - Setup *

Don't forget to register your app with your Firebase Project

Add Firebase to your Android app

1 2 3

Register app Download config file Add Firebase SDK

Android package name ?

A red box highlights this input field.

App nickname (optional) ?

Debug signing certificate SHA-1 (optional) ?

Required for Dynamic Links, Invites and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

CANCEL REGISTER APP

in project **CoffeeMateFBI**



FirebaseStorageEx App - Setup *

Don't forget to register your app with your Firebase Project

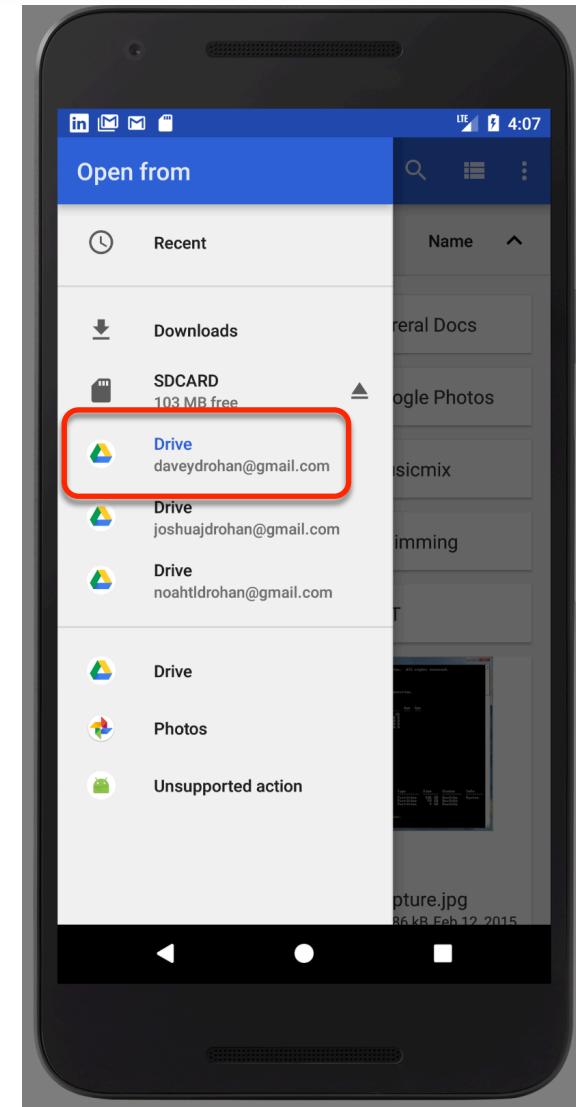
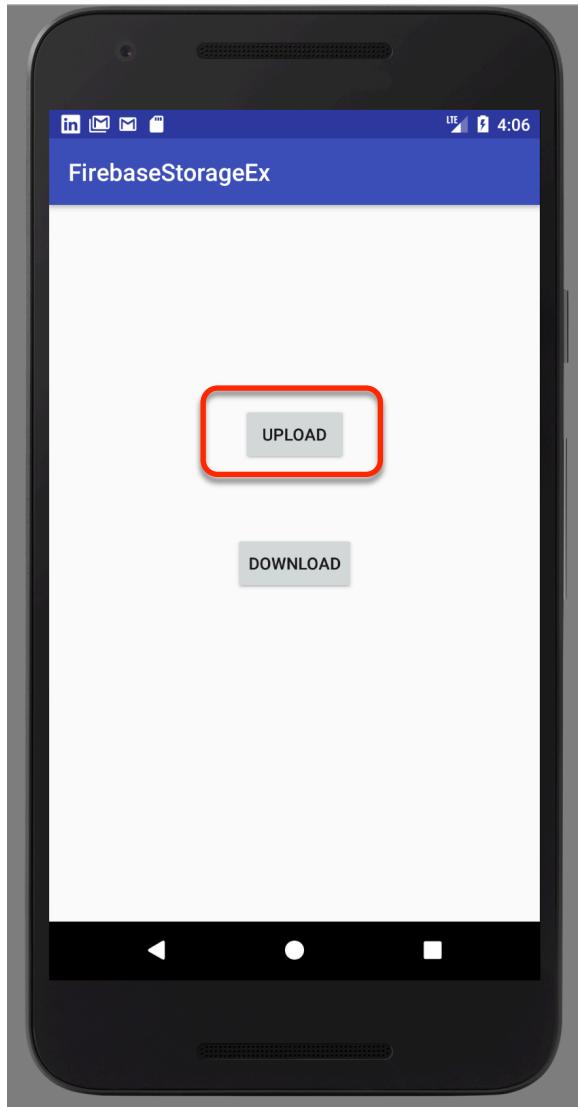
The screenshot shows the Firebase console's Overview page for the project "CoffeeMateFBI". The left sidebar lists various services: Analytics, Authentication, Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, EARN, and AdMob. The main content area is titled "Overview" and shows "CoffeeMateFBI mobile apps" with two entries:

App	Analytics (last 30 days)	Monthly active users	Estimated revenue
CoffeeMate ie.cmfbi	14	\$0	
ie.wit.firebaseiostorageex	0	\$0	

Below the apps, there's a section for "Crashes (30 days)" showing 0 users impacted and 0 instances. At the bottom right, there's a button to "Add another app".

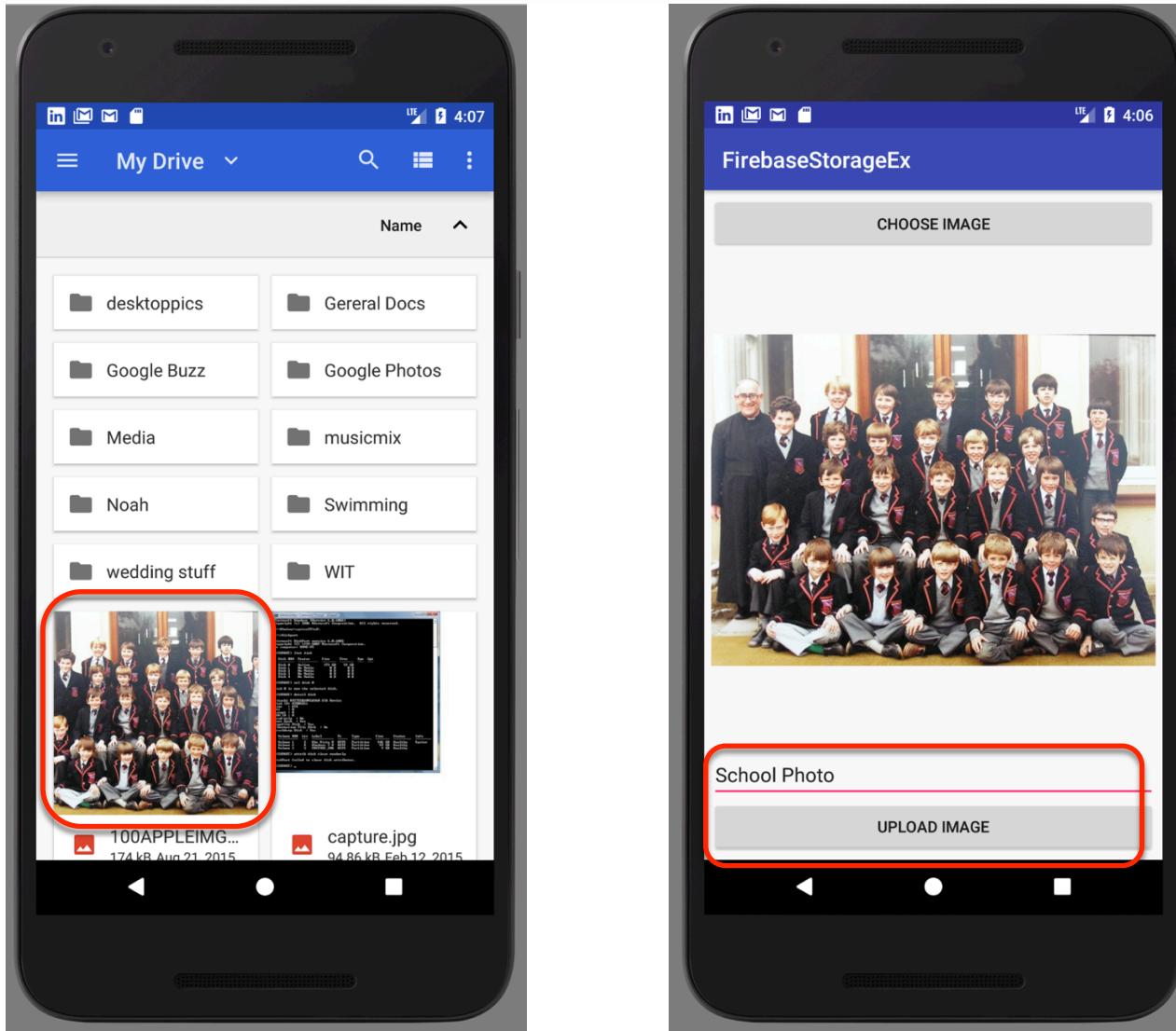


FirebaseStorageEx App - Upload *





FirebaseStorageEx App - Upload *



FirebaseStorageEx App – Upload

The screenshot shows the Firebase Storage console interface. On the left, a sidebar lists various services: Overview, Analytics, DEVELOP, Authentication, Database (highlighted with a red box), Hosting, Functions, Test Lab, Crash Reporting, Performance, GROW, Notifications, Remote Config, Dynamic Links, EARN, AdMob, Spark (Free \$0/month), and UPGRADE. The main area is titled 'Storage' and shows a list of files under 'gs://coffeematefbi.appspot.com/images'. There is one file named 'School Photo' which is being uploaded, indicated by a progress bar. A blue 'UPLOAD FILE' button is visible at the top right. To the right of the file list, a modal window displays the details for 'School Photo': Name (School Photo), Size (170.07 KB), Type (image/jpeg), Created (4 Aug 2017, 16:03:40), and Updated (4 Aug 2017, 16:03:40). The modal also includes sections for File location and Other metadata.

Name	Size	Type	Last modified
School Photo	1...	imag...	4 Au...

School Photo

Name: School Photo

Size: 170.07 KB

Type: image/jpeg

Created: 4 Aug 2017, 16:03:40

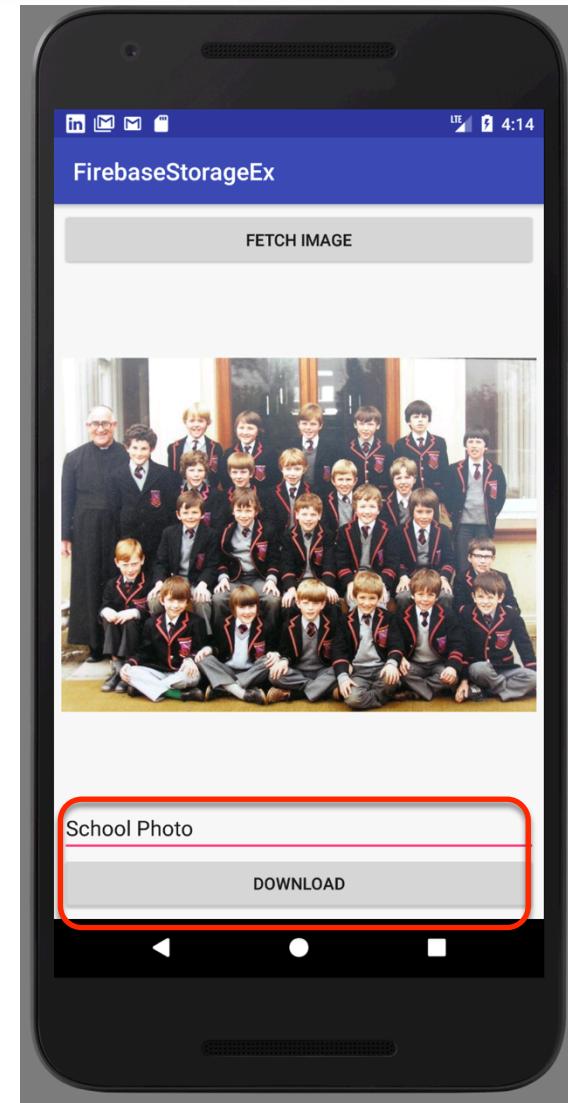
Updated: 4 Aug 2017, 16:03:40

File location

Other metadata



FirebaseStorageEx App - Download *





FirebaseStorageEx App – uploadImage()*

```
private void uploadImage() {
    if(file!=null)
    {
        FirebaseStorage storage=FirebaseStorage.getInstance();
        StorageReference reference=storage.getReferenceFromUrl("gs://coffeematefb.firebaseio.com");
        StorageReference imagesRef=reference.child("images/"+editTextName.getText().toString());
        UploadTask uploadTask = imagesRef.putFile(file);
        uploadTask.addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                pd.dismiss();
                Toast.makeText(ImageUploadActivity.this, "Error : "+e.toString(), Toast.LENGTH_SHORT).show();
            }
        }).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                pd.dismiss();
                Toast.makeText(ImageUploadActivity.this, "Uploading Done!!!", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```



FirebaseStorageEx App - fetchImage() *

```
private void fetchImage() {  
    FirebaseStorage storage=FirebaseStorage.getInstance();  
    // Points to the root reference  
    StorageReference storageRef = storage.getReferenceFromUrl("gs://coffeematefbi.appspot.com");  
    // Points to "images" Directory  
    StorageReference imagesRef = storageRef.child("images");  
    // Points to "images/'filename'.jpg"  
    // Note that you can use variables to create child values  
    String fileName = editTextName.getText().toString();  
    StorageReference fileNameRef = imagesRef.child(fileName);  
  
    // File path is "images/'filename'.jpg"  
    String path = fileNameRef.getPath();  
    // File name is "'filename'.jpg"  
    String name = fileNameRef.getName();  
    // Points to "images"  
    imagesRef = fileNameRef.getParent();  
  
    Glide.with(this /* context */)  
        .using(new FirebaseImageLoader())  
        .load(fileNameRef)  
        .into(imageView);  
}
```



FirebaseStorageEx App - downloadImage() *

```
private void downloadImage() {  
  
    FirebaseStorage storage=FirebaseStorage.getInstance();  
    // Create a storage reference from our app  
    StorageReference storageRef = storage.getReferenceFromUrl("gs://coffeematefbi.appspot.com");  
  
    storageRef.child("images/"+editTextName.getText().toString()).getBytes(Long.MAX_VALUE)  
        .addOnSuccessListener(new OnSuccessListener<byte[]>() {  
            @Override  
            public void onSuccess(byte[] bytes) {  
                // Use the bytes to display the image  
                String path=Environment.getExternalStorageDirectory()+"/"+editTextName.getText().toString();  
                try {  
                    FileOutputStream fos=new FileOutputStream(path);  
                    fos.write(bytes);  
                    fos.close();  
                    Toast.makeText(ViewDownloadActivity.this, "Success!!!", Toast.LENGTH_SHORT).show();  
  
                } catch (IOException e) {  
                    e.printStackTrace();  
                    Toast.makeText(ViewDownloadActivity.this, e.toString(), Toast.LENGTH_SHORT).show();  
                }  
                pd.dismiss();  
            }  
        }).addOnFailureListener(new OnFailureListener() {  
            @Override  
            public void onFailure(@NonNull Exception exception) {  
                // Handle any errors  
                pd.dismiss();  
                Toast.makeText(ViewDownloadActivity.this, exception.toString()+"!!!", Toast.LENGTH_SHORT).show();  
            }  
        });  
}
```



Pricing

Products	Spark Plan Generous limits for hobbyists Free	Flame Plan Fixed pricing for growing apps \$25/month	Blaze Plan Calculate pricing for apps at scale Pay as you go
Free Products Authentication (except Phone Auth), Analytics, App Indexing, Dynamic Links, Invites, Remote Config, Cloud Messaging (FCM), Performance Monitoring, and Crash Reporting.	Included	Included Free	Included Free
Storage	5 GB 1 GB/day 20K/day 50K/day	50 GB 50 GB/day 100K/day 250K/day	\$0.026/GB \$0.12/GB \$0.10/thousand \$0.01/thousand



Hosting



The new VPS in town...



Firebase Hosting

- ❑ Firebase hosting provides a fast and secure static hosting for your web app.
- ❑ It is a production grade web content hosting.
- ❑ No need to look for any other VPS solution
- ❑ Hosting gives a subdomain of “firebaseapp.com”
- ❑ You can also get a custom domain.
- ❑ Install firebase CLI and deploy in a matter of seconds.
- ❑ Also suited for progressive web apps



Deploy Your Site In Seconds

- ❑ Install npm.
- ❑ Install firebase CLI
 - `npm install -g firebase-tools`
- ❑ Initialize your app
 - `firebase init`
- ❑ Set root folder and put your static content there.
- ❑ Deploy
 - `firebase deploy`



Firebase Hosting



Quick DEMO...



S W R

console.firebaseio.google.com/project/coffeematefbi/hosting

kollect.ie WTC – ROCA Sports The Pirate Bay Bofl 365 Online Android Development Electronic Marksheets lynda.com solution.md Triathlon Ireland Apple Developer >

CoffeeMateFBI – Hosting – Firebase console

Firebase CoffeeMateFBI Overview Go to docs :

Analytics

DEVELOP

Authentication

Database

Storage

Hosting

Functions

Test Lab

Crash Reporting

Performance

GROW

Notifications

Remote Config

Dynamic Links

Spark Free \$0/month UPGRADE

DASHBOARD USAGE

HOSTING

Deploy web and mobile web apps in seconds using a secure global content delivery network

Learn more View the docs

GET STARTED

The screenshot shows the Firebase Hosting dashboard for the project "CoffeeMateFBI". The left sidebar has a red box around the "Hosting" item, which is currently selected. The main area features a large globe icon and text about deploying web and mobile apps using a secure global content delivery network. A "GET STARTED" button is visible at the bottom right of the main panel.



S W R E

console.firebaseio.google.com/project/coffeematefbi/hosting

kollect.ie WTC – ROCA Sports The Pirate Bay Bofl 365 Online Android Development Electronic Marksheets lynda.com solution.md Triathlon Ireland Apple Developer >

CoffeeMateFBI – Hosting – Firebase console

Firebase CoffeeMateFBI ▾

Overview Hosting DASHBOARD USAGE ?

Analytics

DEVELOP

Authentication

Database

Storage

Hosting

Functions

Test Lab

Crash Reporting

Performance

GROW

Notifications

Remote Config

Dynamic Links

Spark Free \$0/month UPGRADE

Set up hosting

1 Install 2 Deploy

To host your site, you need to install Firebase command line tools using npm ([Node.js](#)).

Install Firebase tools: `$ npm install -g firebase-tools`

Doesn't work? You may need to [change npm permissions](#).

If you've previously installed Firebase command line tools, run the install command again to make sure that you have the latest version.

CANCEL CONTINUE



S W R E

console.firebaseio.google.com/project/coffeematefbi/hosting

kollect.ie WTC – ROCA Sports The Pirate Bay Bofl 365 Online Android Development Electronic Marksheets lynda.com solution.md Triathlon Ireland Apple Developer >

CoffeeMateFBI – Hosting – Firebase console

Firebase CoffeeMateFBI ▾

Overview Hosting DASHBOARD USAGE

Analytics

DEVELOP

- Authentication
- Database
- Storage
- Hosting**
- Functions
- Test Lab
- Crash Reporting
- Performance

GROW

- Notifications
- Remote Config
- Dynamic Links

Spark Free \$0/month UPGRADE

Set up hosting

1 Install 2 Deploy

Open a terminal window and navigate to or create a directory for your site

Sign in to Google: `$ firebase login`

Initiate your project: `$ firebase init`

Add your static files to your deploy directory (the default is public)

Deploy your website: `$ firebase deploy`

FINISH



S W R E

console.firebaseio.google.com/project/coffeematefbi/hosting/main

kollect.ie WTC – ROCA Sports The Pirate Bay Bofl 365 Online Android Development Electronic Marksheets lynda.com solution.md Triathlon Ireland Apple Developer >

CoffeeMateFBI – Hosting – Firebase console

Firebase CoffeeMateFBI Overview Go to docs :

Analytics

DEVELOP

Authentication

Database

Storage

Hosting

Functions

Test Lab

Crash Reporting

Performance

GROW

Notifications

Remote Config

Dynamic Links

Spark Free \$0/month UPGRADE

Hosting

DASHBOARD USAGE

Domain

Domain	Type	Status
coffeematefbi.firebaseioapp.com	Default	

CONNECT DOMAIN

Deployment history

Status	Time:	Deploy	Files
Waiting for your first deploy			

INSTRUCTIONS

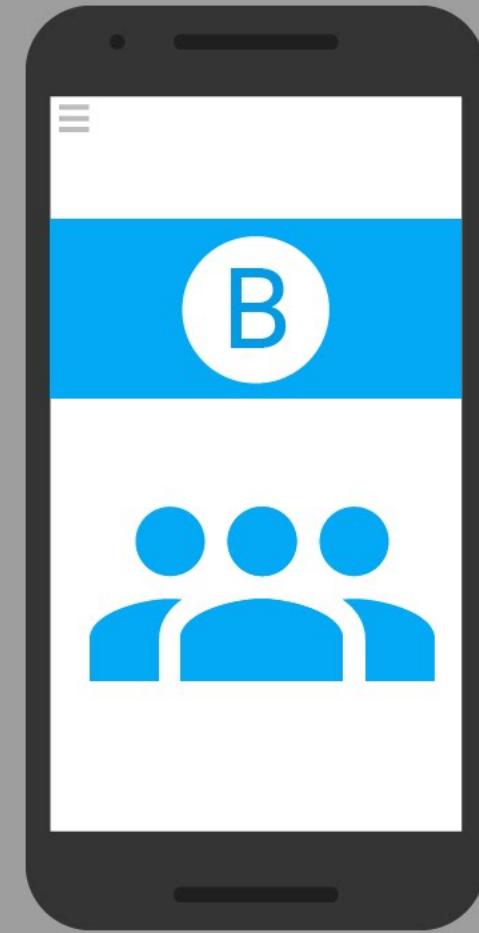


Pricing

Products	Spark Plan Generous limits for hobbyists Free	Flame Plan Fixed pricing for growing apps \$25/month	Blaze Plan Calculate pricing for apps at scale Pay as you go
Free Products Authentication (except Phone Auth), Analytics, App Indexing, Dynamic Links, Invites, Remote Config, Cloud Messaging (FCM), Performance Monitoring, and Crash Reporting.	Included	Included Free	Included Free
Hosting GB stored GB transferred Custom domain & SSL	1 GB 10 GB/month 	10 GB 50 GB/month 	\$0.026/GB \$0.15/GB



Remote Config



Update your App without
actually updating it



Remote Config

- Change behavior and config without app update.
- Set in-app default values that app uses to work.
- Override these values from firebase console when app is published.
- Change configuration on the fly.
- Change configuration based on condition. Ideal for A/B testing



Customize Experience

- ❑ Create segments and customize experience
- ❑ Background = blue (default green) if
 - Country = India &&
 - OS Type = Android
- ❑ Discount = 30% (default 5%) if
 - OS Type = iOS
 - Version = 2.3.0



The “2 Line Code” cliché

- ❑ Fetch parameters from remote config server
 - `mFirebaseRemoteConfig.fetch(300).addOnCompleteListener(...)`
- ❑ Make the fetched parameters available to your app
 - `mFirebaseRemoteConfig.activateFetched();`
- ❑ And all is done for your app



Remote Config



Quick DEMO... TBD



S W R E

console.firebaseio.google.com/project/coffeematefbi/config

kollect.ie WTC – ROCA Sports The Pirate Bay Bofl 365 Online Android Development Electronic Marksheets lynda.com solution.md Triathlon Ireland Apple Developer >

CoffeeMateFBI – Remote Config – Firebase console

Firebase CoffeeMateFBI ▾

Overview

Analytics

DEVELOP

- Authentication
- Database
- Storage
- Hosting
- Functions
- Test Lab
- Crash Reporting
- Performance

GROW

- Notifications
- Remote Config**
- Dynamic Links

Spark
Free \$0/month

UPGRADE

Remote Config

PARAMETERS



Customise and experiment with app behaviour using server-side configuration parameters

[Learn more](#) [View the docs](#)

ADD YOUR FIRST PARAMETER

Example: holiday_promotion_enabled = true



S W R

console.firebaseio.google.com/project/coffeematefbi/config

kollect.ie WTC – ROCA Sports The Pirate Bay Bofl 365 Online Android Development Electronic Marksheets lynda.com solution.md Triathlon Ireland Apple Developer >

CoffeeMateFBI – Remote Config – Firebase console

Firebase CoffeeMateFBI Overview Go to docs ?

Analytics

DEVELOP

- Authentication
- Database
- Storage
- Hosting
- Functions
- Test Lab
- Crash Reporting
- Performance

GROW

- Notifications
- Remote Config
- Dynamic Links

Spark Free \$0/month UPGRADE

Remote Config

PARAMETERS

This is the key you'll pass to the Remote Config SDK, for example:
config.getBoolean("holiday_promo_enabled");

experiment with app server-side configuration

Parameter key	Default value	Add value for condition ▾
Example: holiday_promo_enabled	(empty string)	Other empty values ▾

CANCEL ADD PARAMETER



FirebaseStorageEx App *





Crash Reporting



And finding the root cause



Crash Reporting

- ❑ Monitor fatal and non-fatal errors.
- ❑ Errors are grouped into clusters of similar stack traces.
- ❑ Find bug cause easily - Report consists of stack trace, device characteristics, performance etc.
- ❑ Enable email alerts to be notified of a new crash anytime.
- ❑ Integrate with analytics to find order of events that caused the crash.
- ❑ Seamless across android and iOS.



Implement Crash Reporting

- ❑ Connect your app with firebase console.
- ❑ Add firebase dependency into gradle.
- ❑ Add custom logs wherever required.
 - `FirebaseCrash.log("SQLite syntax error");`
- ❑ Report server side errors with
[Google Stackdriver Error reporting.](#)
- ❑ See console.



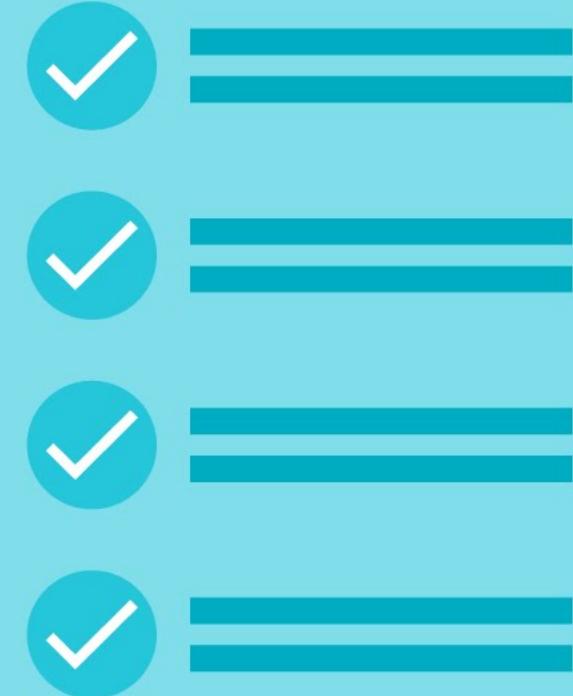
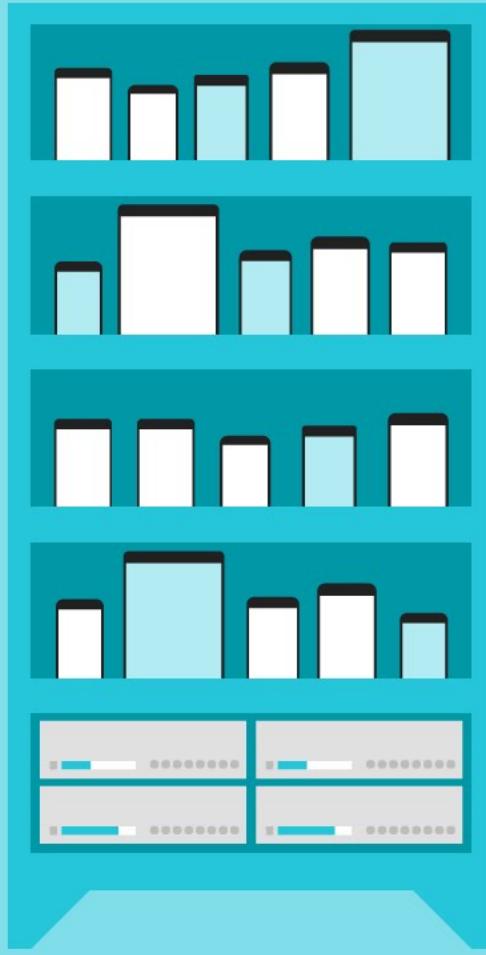
Crash reporting with proguard

- ❑ In **proguarded** apps the stack trace is not sufficient to identify the root cause.
- ❑ Usually we view the mapping files and try to make sense out of it.
- ❑ All of this is done by firebase. Just upload the mapping files and you will see meaningful stacktraces.



Demo

Stack trace in firebase console



Test Lab for Android

Test virtually on
popular REAL
devices



Firebase test lab for android

- ❑ Cloud based infrastructure for testing android apps.
- ❑ Test across variety of devices and device configurations.
- ❑ Test results include logs, videos and screenshots.
- ❑ Integrate with android studio and CI systems easily.



Demo

Show integration in android studio

Test lab results

Show video

Show logs



Pricing

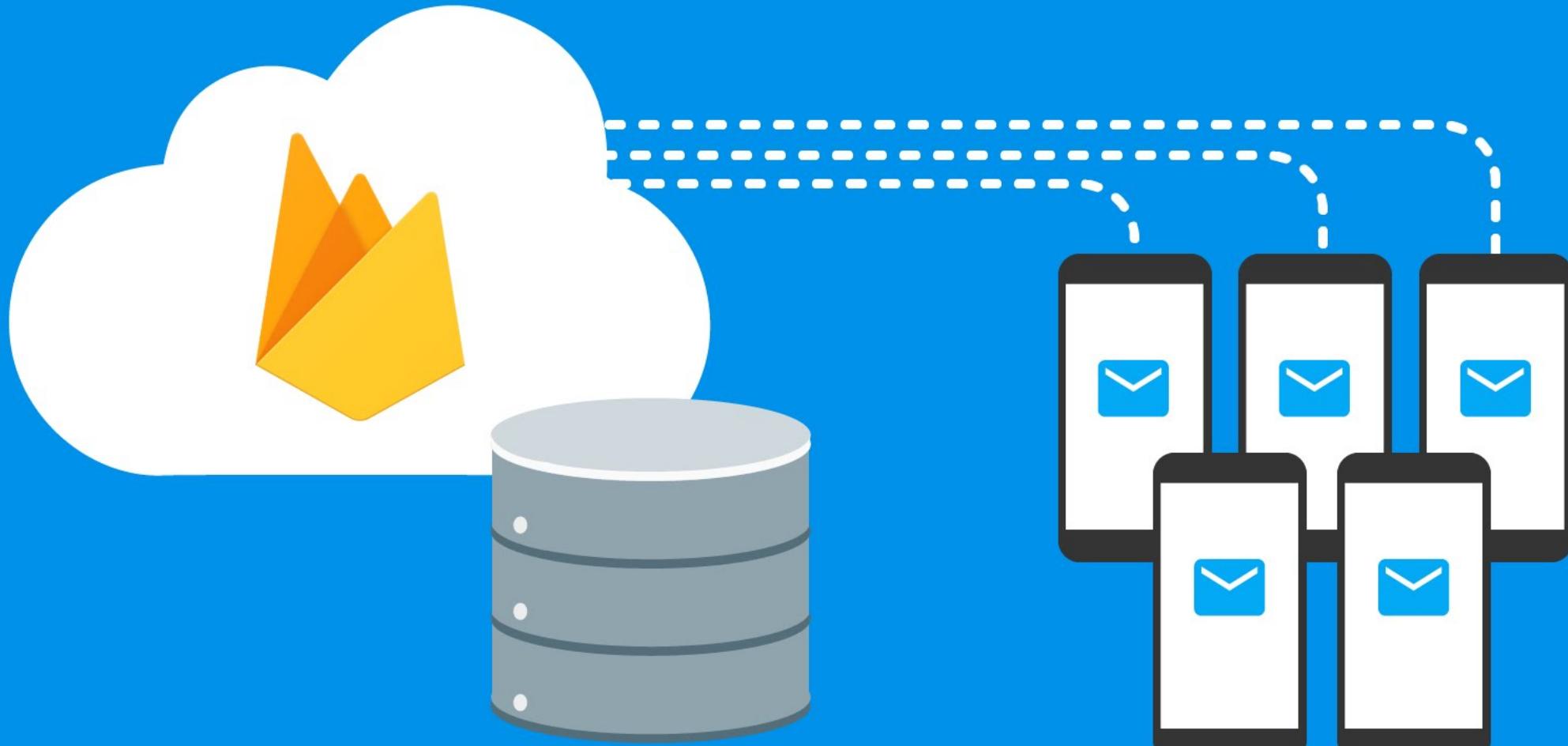
SPARK Free	FLAME \$25 per month	BLAZE Pay as you go
Generous limits for hobbyists	Predictable pricing for growing apps	Commodity pricing for apps at scale

Included Free

Analytics, App Indexing, Authentication, Cloud Messaging, Crash Reporting, Dynamic Links, Invites, Notifications & Remote Config



Test Lab	Daily quota or hourly rate	15 tests per day ³	15 tests per day ³	hourly, per device ⁴
----------	----------------------------	-------------------------------	-------------------------------	---------------------------------



Cloud Messaging

The awesome user engagement and more



Firebase Cloud Messaging

- ❑ It is a cross-platform messaging solution that reliably delivers messages at no cost.
- ❑ Send notifications that are displayed to your user. Or send data messages which can be consumed by application code.
- ❑ Distribute messages to your client app in any of three ways — to single devices, to groups of devices, or to devices subscribed to topics.
- ❑ Send acknowledgments from devices back to your server over FCM's reliable and battery-efficient connection channel.
- ❑ 95% of the messages delivered in under 250 milliseconds.



How does FCM works?





FCM message structure

```
{  
  "to" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...",  
  "notification" : {  
    "body" : "great match!",  
    "title" : "Portugal vs. Denmark",  
    "icon" : "myicon"  
  },  
  "data" : {  
    "nick" : "Mario",  
    "room" : "PortugalVSDenmark"  
  }  
}
```



Set up FCM with android app

gradle dependency : compile 'com.google.firebaseio:firebase-messaging:x.x.x'

A service that extends `FirebaseInstanceIdService` is required to handle the creation, rotation, and updation of registration tokens which is required to send the notification or message to specific devices or for creating device groups.

```
<service  
    android:name=". MyFirebaseInstanceIdService">  
    <intent-filter>  
        <action android:name="com.google.firebaseio.INSTANCE_ID_EVENT"/>  
    </intent-filter>  
</service>
```

```
@Override  
public void onTokenRefresh() {  
    String refreshedToken = FirebaseInstanceId.getInstance().getToken();  
    sendRegistrationToServer(refreshedToken);  
}
```

Send FCM from firebase console



- ❑ Open the **Notifications** tab of the Firebase console and select **New Message**.
- ❑ Enter the message text.
- ❑ Select **Single Device** for the message target.
- ❑ In the field labeled **FCM Registration Token**, enter the registration token you obtained in a previous section of this guide.



Receive message

Manifest declaration:-

A service that extends **FirebaseMessagingService** is required if you want to do any message handling beyond receiving notifications on apps in the background.

To receive notifications in foregrounded apps, to receive data payload, to send upstream messages, and so on, you must extend this service.

```
<service  
    android:name=".MyFirebaseMessagingService">  
    <intent-filter>  
        <action android:name="com.google.firebaseio.MESSAGING_EVENT"/>  
    </intent-filter>  
</service>
```



Receive a Message

- By overriding the method **onMessageReceived**, you can perform actions based on the received RemoteMessage object and get the message data:

```
@Override  
public void onMessageReceived(RemoteMessage remoteMessage) {  
  
    Log.d(TAG, "From: " + remoteMessage.getFrom());  
  
    Log.d(TAG, "Message data payload: " + remoteMessage.getData());  
  
    Log.d(TAG, "Message Notification Body: " +  
        remoteMessage.getNotification().getBody());  
  
}
```



Topic messaging

Subscribe the client app to a topic:

```
FirebaseMessaging.getInstance().subscribeToTopic("Golf");
```

HTTP POST request Send to a single topic:

URL: <https://fcm.googleapis.com/fcm/send>

```
{
  "to": "/topics/Golf",
  "data": {
    "message": "Woods made history!!",
  }
}
```

Topic messaging to multiple topics



Send to a multiple topic:

```
{  
    "condition": "Golf" in topics || 'Cricket' in topics,  
    "data": {  
        "message": "Subsidy worth $7mn allotted!"  
    }  
}
```

HTTP response:

```
//Success example:  
{  
    "message_id": "1023456"  
}
```



Device group messaging

Before sending messages to a device group, you must:

- ❑ Obtain registration tokens for each device you want to add to the group.
- ❑ Create the [notification_key](#), which identifies the device group by mapping a particular group (typically a user) to all of the group's associated registration tokens.
- ❑ To create a device group, send a POST request that provides a name for the group, and a list of registration tokens for the devices. FCM returns a new [notification_key](#) that represents the device group



Device group messaging

- ❑ HTTP post to : <https://android.googleapis.com/gcm/notification>

project_id:SENDER_ID

```
{  
    "operation": "create",  
    "notification_key_name": "appUser-Chris",  
    "registration_ids": ["token1", "token2", ..... "token 9"]  
}
```

- ❑ Response format

```
{  
    "notification_key": "APA91bGHXQBB...9QgnYOEURwm0I3lmyqzk2TXQ"  
}
```



Send an upstream message

To initiate an upstream message, the client app sends a request containing the following:

- ❑ The address of the receiving app server in the format **SENDER_ID@gcm.googleapis.com**.
- ❑ A **message ID** that should be unique for each sender ID.
- ❑ The message data comprising the key-value pairs of the message's payload.

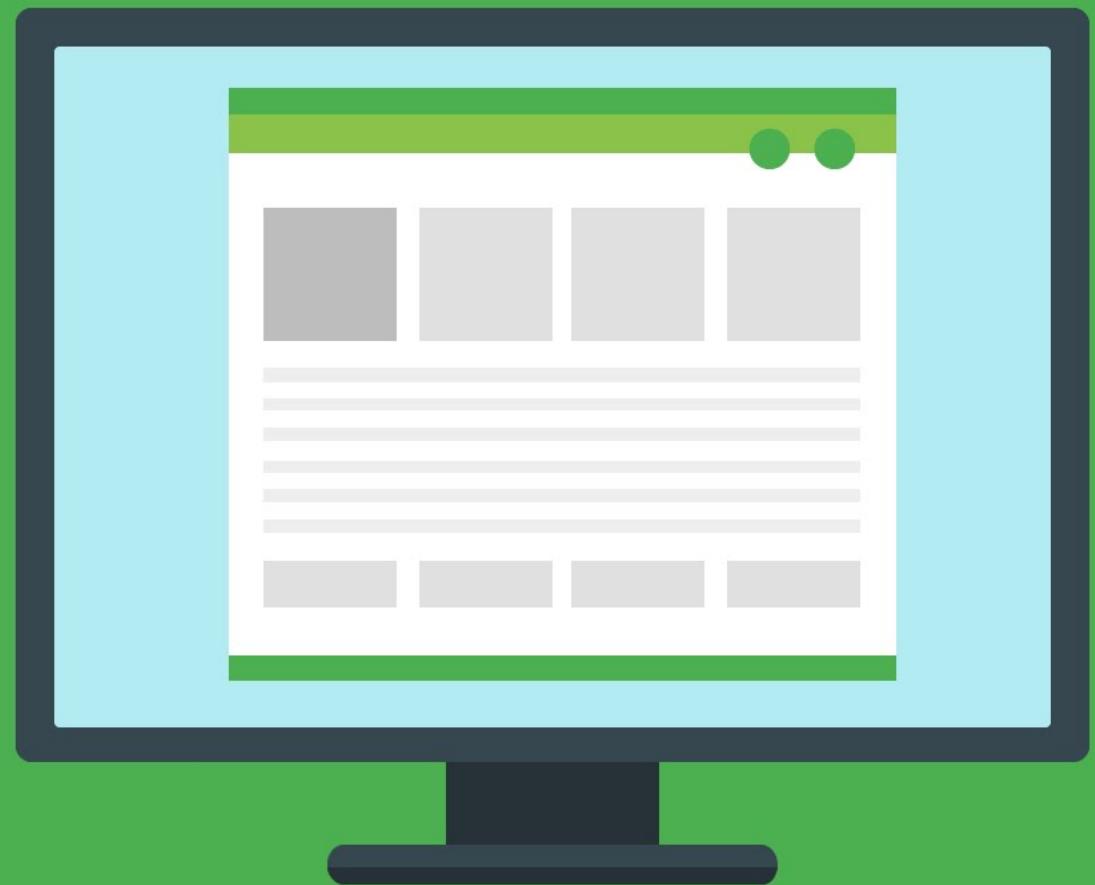
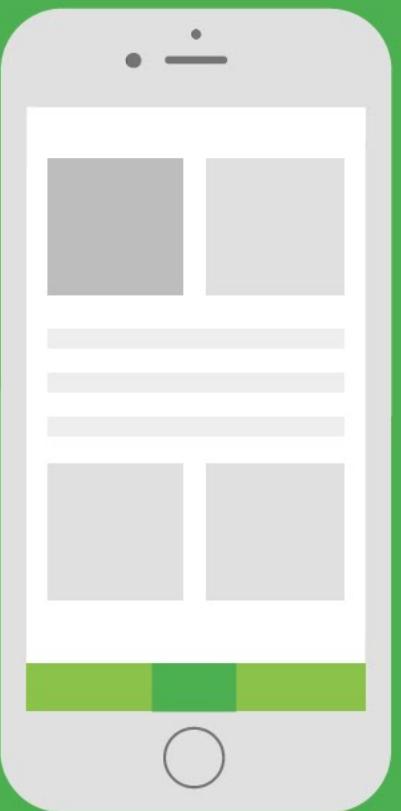
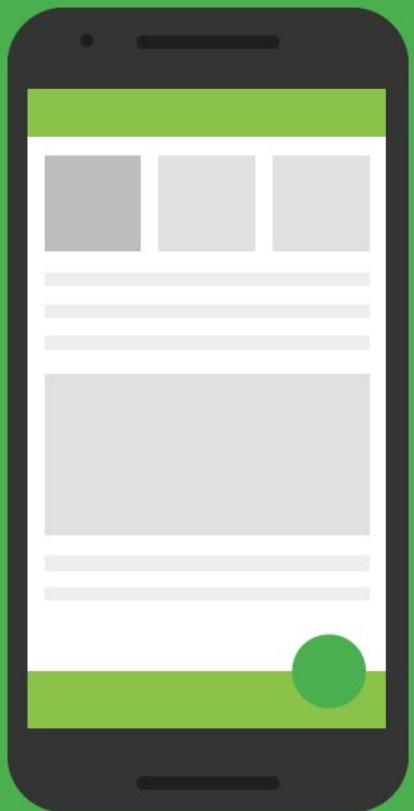
```
FirebaseMessaging fm = FirebaseMessaging.getInstance();
fm.send(new RemoteMessage.Builder(SENDER_ID + "@gcm.googleapis.com")
    .setMessageId(Integer.toString(msgId.incrementAndGet()))
    .addData("my_message", "Hello World")
    .addData("my_action", "SAY_HELLO")
    .build());
```

- ❑ Check the status of upstream messages with callbacks **onMessageSent** and **onSendError**.



Migrating from GCM

- ❑ In the Firebase console, select **Import Google Project**.
- ❑ All the permissions required by FCM are now added automatically by library.
- ❑ `com.google.android.gms.gcm.GcmReceiver` is already with FCM added automatically. No need to add in manifest.
- ❑ Remove registration.
- ❑ GcmPubSub methods have been integrated and simplified in the **FirebaseMessaging** class.



Dynamic Links

Works the way it's
supposed to



Firebase Dynamic Links

- ❑ When users open a Dynamic Link into an app that is not installed, the app's Play Store page opens, where users can install the app. After users install and open the app, the app displays the deep-linked content.
- ❑ Dynamic Links work seamlessly across iOS, Android, and web apps.
- ❑ If your app needs to be upgraded to open a link, your app automatically opens the link (with appropriate content) after upgrading.
- ❑ Create short links using the Firebase console.
- ❑ Free, unlimited use.

Create a Dynamic Link



Gradle dependency: 'com.google.firebaseio:firebase-invites:x.x.x'

- ❑ In the Firebase Console, open the **Dynamic Links** section. Accept the terms of service if you are prompted to do so.
- ❑ Take note of your project's Dynamic Links domain, which is displayed at the top of the Dynamic Links page. You need your project's Dynamic Links domain to programmatically create Dynamic Links. A Dynamic Links domain looks like *app_code.app.goo.gl*.

You can also create a Dynamic Link programmatically by constructing a URL with the following form:

- ❑ *https://domain/?link=your_deep_link&apn=package_name[&amv=minimum_version][&al=android_link][&afl=fallback_link]*

Open Dynamic Links in your app



- you must call **getInvitation()** in every activity that might be launched by the link, even though the link might be available from the intent using **getIntent().getData()**.
- Calling **getInvitation()** retrieves the link and clears that data so it is only processed once by your app.

intent-filter>

```
<action android:name="android.intent.action.VIEW"/>
<category android:name="android.intent.category.DEFAULT"/>
<category android:name="android.intent.category.BROWSABLE"/>
<data android:host="www.example.com" android:scheme="http"/>
<data android:host="www.example.com" android:scheme="https"/>
</intent-filter>
```

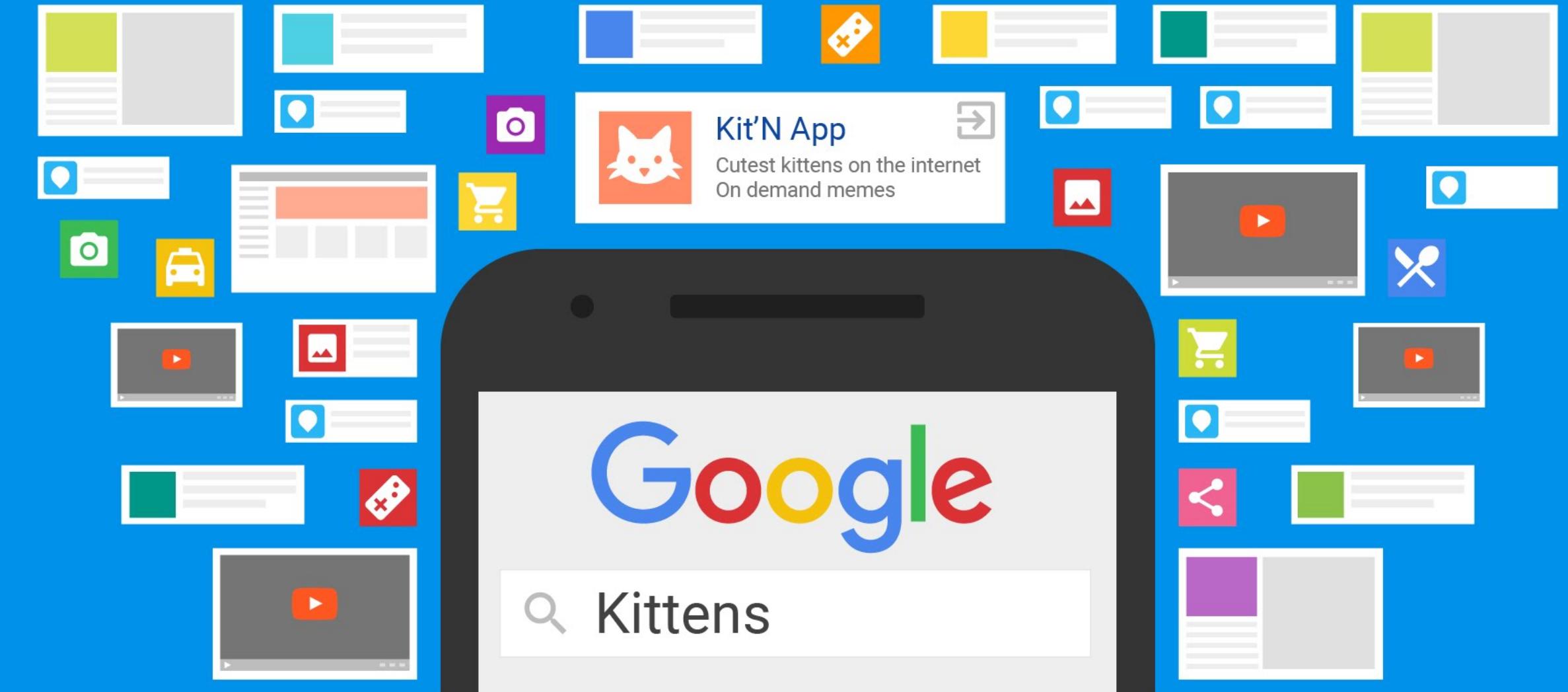
Handle deep links



```
mGoogleApiClient = new GoogleApiClient.Builder(this).enableAutoManage(this, this)
    .addApi(AppInvite.API).build();
```

```
boolean autoLaunchDeepLink = false;
AppInvite.AppInviteApi.getInvitation(mGoogleApiClient, this, autoLaunchDeepLink)
    .setResultCallback(
        new ResultCallback<AppInviteInvitationResult>() {

            @Override
            public void onResult(@NonNull AppInviteInvitationResult result) {
                if (result.getStatus().isSuccess()) {
                    Intent intent = result.getInvitationIntent();
                    String deepLink = AppInviteReferral.getDeepLink(intent);
                }
            });
});
```



App Indexing

Measure your Success



App Indexing

- ❑ Firebase App Indexing gets your app into Google Search.
- ❑ If users have your app installed, they can launch your app and go directly to the content they searched for.
- ❑ App Indexing re-engages your app users by helping them find both public and personal content right on their device, even offering query **auto completions** to help them more quickly find what they need.
- ❑ If users don't yet have your app, relevant queries trigger an install card for your app in Search results.

Gradle dependencies: **compile 'com.google.firebaseio.firebaseio-appindexing:x.x.x'**



Enable public content

1. Associate your site and your app

Associate your site with your app using **Digital Asset Link** and build your app against **API level 23** or higher for the Android platform.

Digital Asset Links take the form of a JSON file called **assetlinks.json** in your web server's well-known directory. Configure the following fields in your assetlinks.json file:

```
[  
  {  
    "relation": ["delegate_permission/common.handle_all_urls"],  
    "target" : { "namespace": "android_app",  
                "package_name": "com.recipe_app",  
                "sha256_cert_fingerprints": ["hash_of_app_certificate"] }  
  }]
```

Upload your **assetlinks.json** file and make sure the read access is set as visible to everyone at <https://<your-site>/.well-known/assetlinks.json>.



Enable public content

2 . Route incoming links

```
<activity  
    android:name=".client.RecipeActivity"  
<intent-filter android:label="@string/app_name" android:autoVerify="true">  
    <action android:name="android.intent.action.VIEW" />  
    <category android:name="android.intent.category.DEFAULT" />  
    <category android:name="android.intent.category.BROWSABLE" />  
  
    <data android:scheme="http"  
        android:host="recipe-app.com"  
        android:pathPrefix="/recipe" />  
</intent-filter>  
</activity>
```



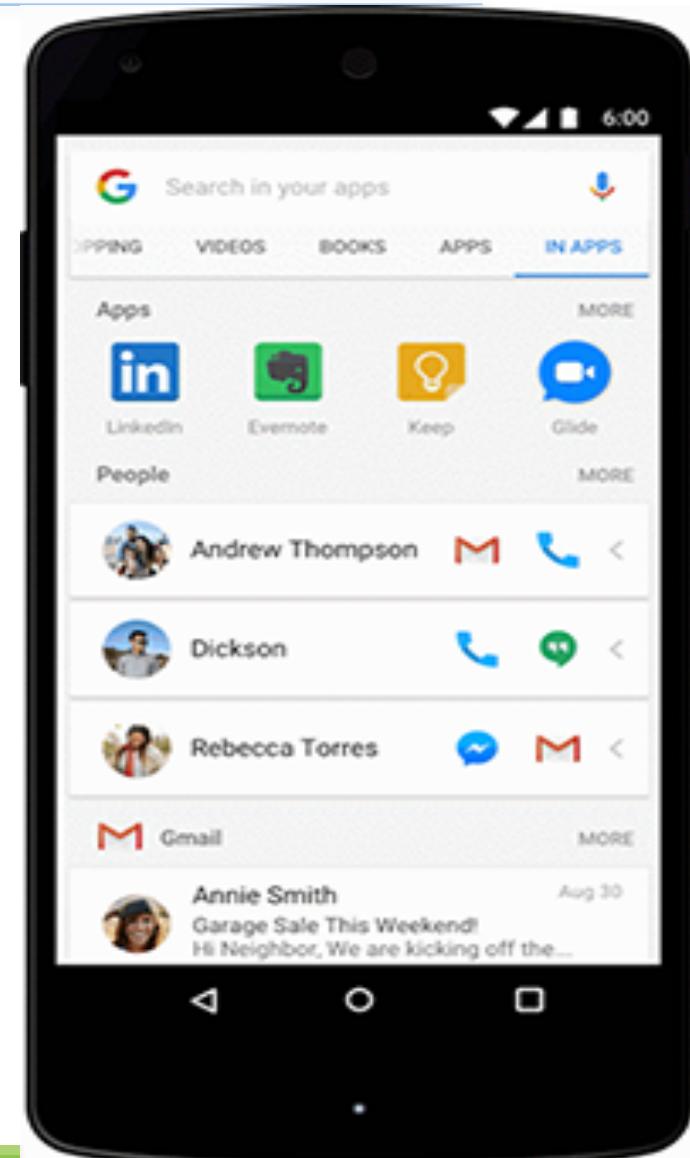
Handle incoming URLs

```
Intent intent = getIntent();
String action = intent.getAction();
String data = intent.getDataString();
if (Intent.ACTION_VIEW.equals(action) && data != null)
{
    // process your link
    String recipId = data.substring(data.lastIndexOf("/") + 1);
}
```

Enable Personal Content Indexing



- Create an IntentService.
- To include items in the app's personal content index, create **Indexable** objects in the same class.
- Set the unique URL and title for each indexable object through the `setUrl()` and `setName()` setters, respectively. **These properties are required.**
- To create your indexable objects more conveniently, use existing builders whenever possible (e.g., for messages, documents, and spreadsheets).
- For content types that don't have custom builders, use the generic `Indexable.Builder()`.



Generate and refresh the index



- ❑ Use intent filters to add user's existing personal content to the on-device index and allow Google Play services to periodically call `AppIndexingService` to update the index
- ❑ With this periodic update, the on-device index accurately reflects any changes that happen to the indexed content over time.

```
<service android:name=".client.AppIndexingService"  
        android:exported="true"  
        android:permission="com.google.android.gms.permission.APPINDEXING">  
    <intent-filter>  
        <action android:name="com.google.firebaseio.appindexing.UPDATE_INDEX" />  
    </intent-filter>  
</service>
```



Remove/update the personal content

```
FirebaseAppIndex.getInstance().remove(noteUrl);  
FirebaseAppIndex.getInstance().update(noteUrl);  
FirebaseAppIndex.getInstance().removeAll();
```

Define Indexable objects for the following types of content:

- Content that's important to users, such as favorites or content that they may want to access more frequently. For example, documents or songs that they've bookmarked or marked for offline use.
- Content generated in your app, not just accessed by it. For example, contacts that are created by users directly in your app, and stored by your app, not contacts from the phone's directory.



Analytics



Measure your Success



Thank you

Any questions?

Demo app available at
<https://goo.gl/WBP5fR>



Backup slides



Some important points

- ❑ Do not think RDBMS, think JSON. How data should be structured is very important.
- ❑ Firebase has a recycler view, that integrates with real time database smoothly without any listeners.
- ❑ Test lab which is available in paid plan (Blaze), is an amazing feature for testing your app on different real and virtual devices.
- ❑ Set developer mode to true when testing remote config.



Firebase Setup



Goodbye RDBMS...



Adding Gradle Dependency

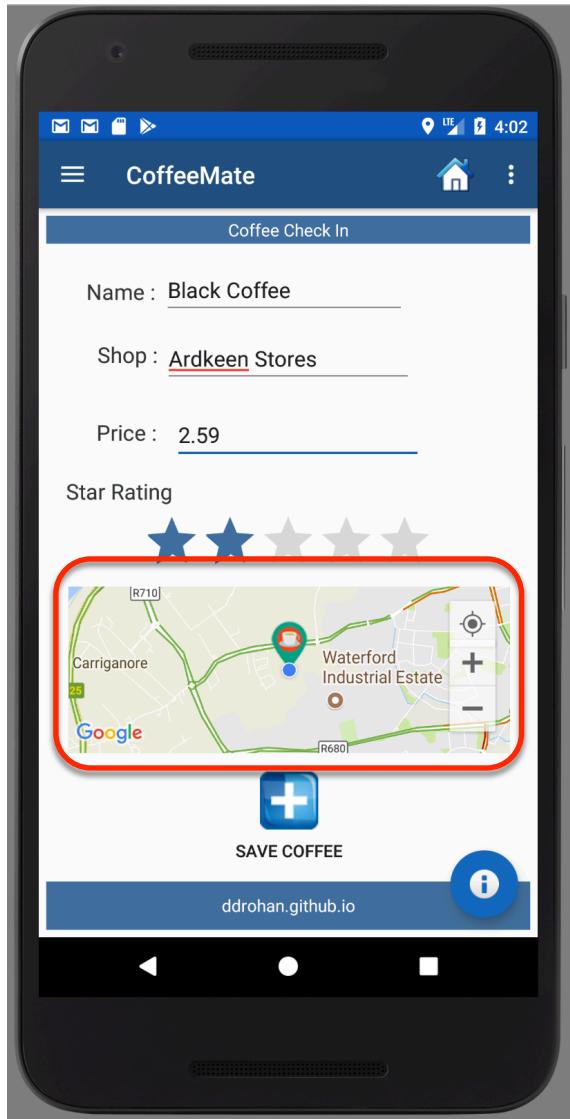
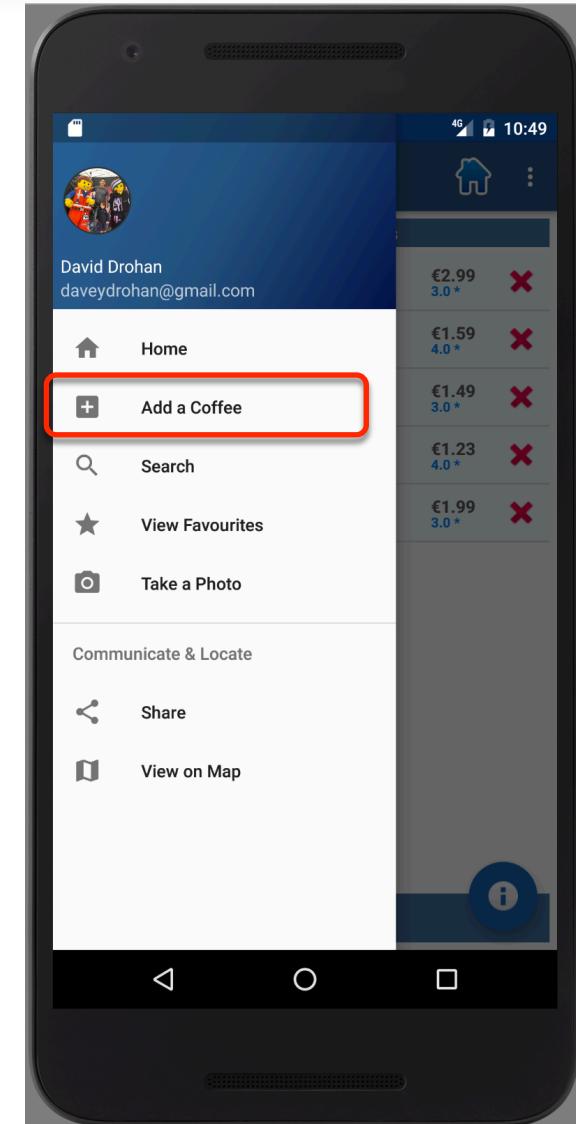
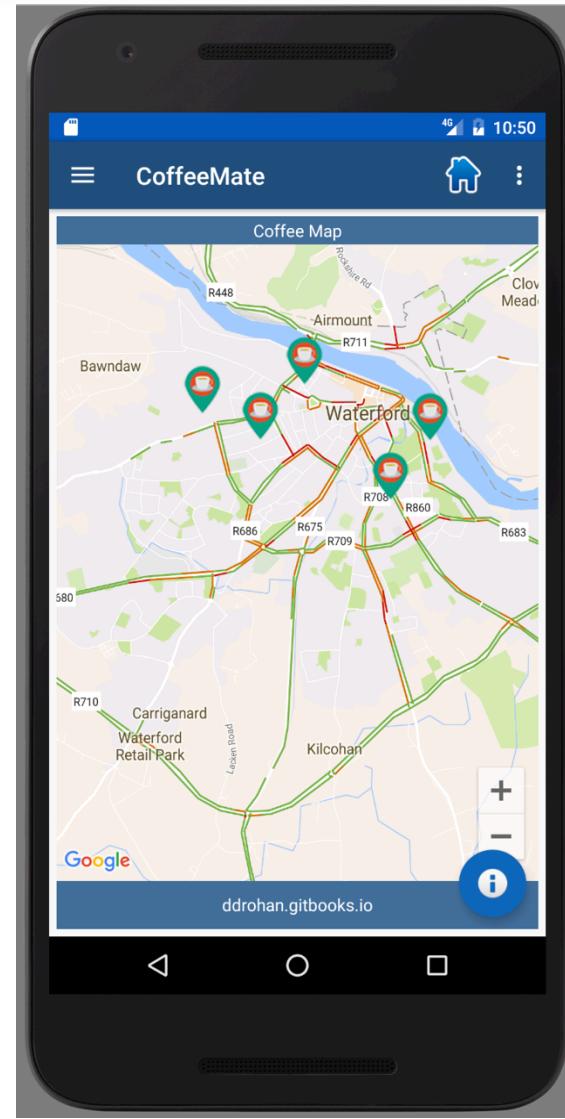
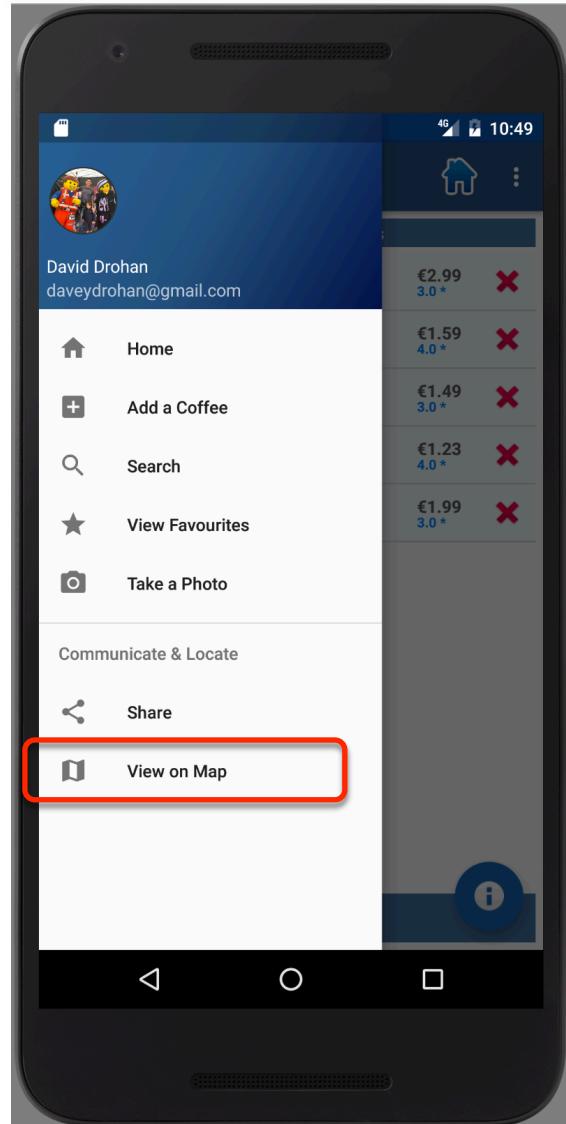
Add this in your app Gradle dependency

'com.google.firebaseio:firebase-database:X.X.X'

And sync the project



CoffeeMateFBI 1.0 *





References & Links

- ❑ [Presentation by Kaushal Dhruw & Shakti Moyal 2016](#)
- ❑ <https://firebase.google.com>



Questions?