

# Mobile Application Development

---

Produced  
by

David Drohan ([ddrohan@wit.ie](mailto: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...



# Agenda

---

- ❑ Firebase history
- ❑ The all new Firebase
- ❑ Real-time database
- ❑ Authentication
- ❑ Storage
- ❑
- ❑
- ❑ 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 ?
- @^#%\$&!<\*>^%(\$....



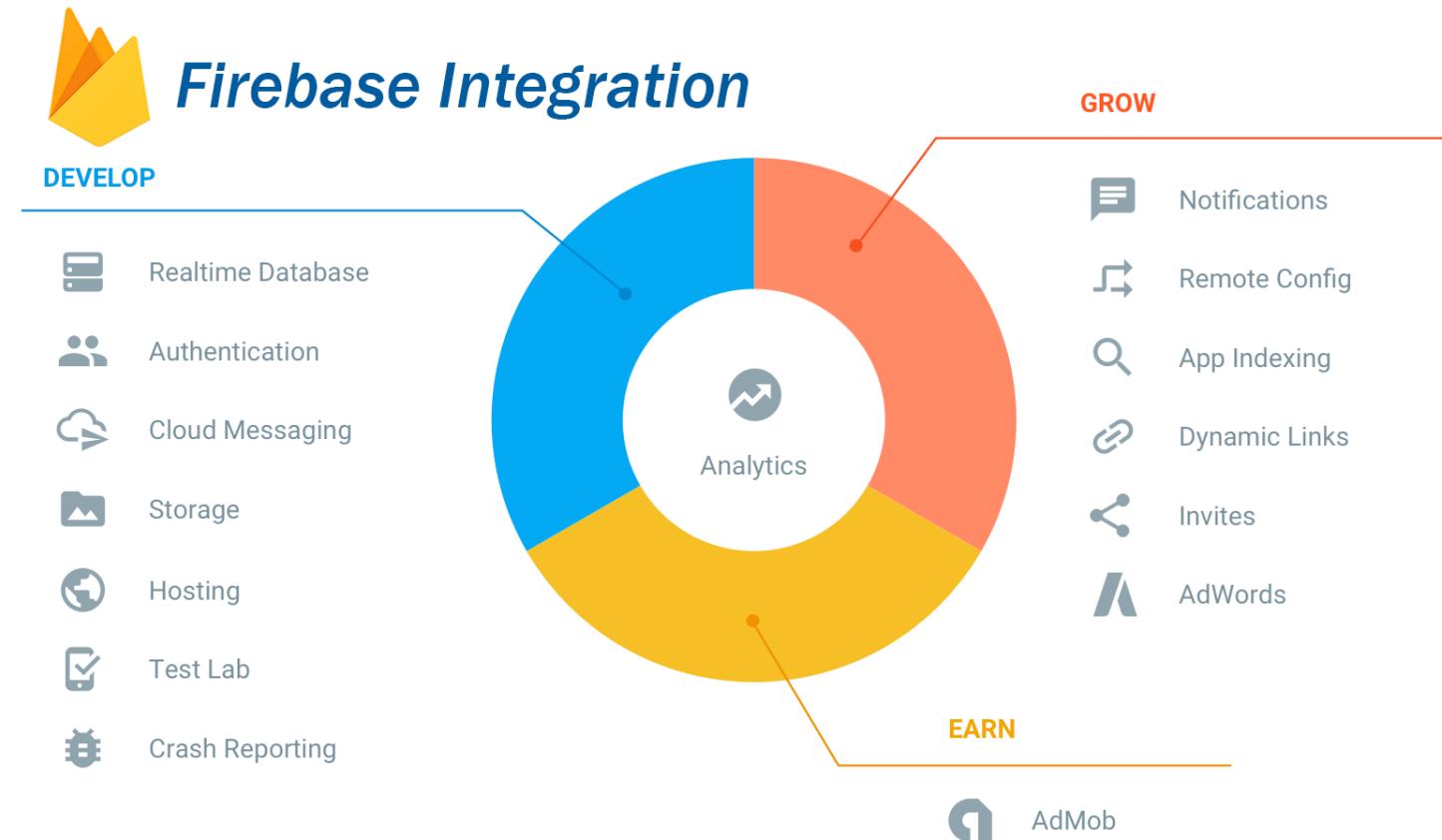


# What It Is ... \*

❑ A complete BaaS solution that

includes

- Real time JSON database
- Authentication
- Cloud storage
- Cloud messaging
- Crash reporting and analytics
- And a lot 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



# So... You have an Idea For An App...

What people think...

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





# Idea For An App \*

In Reality





# 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

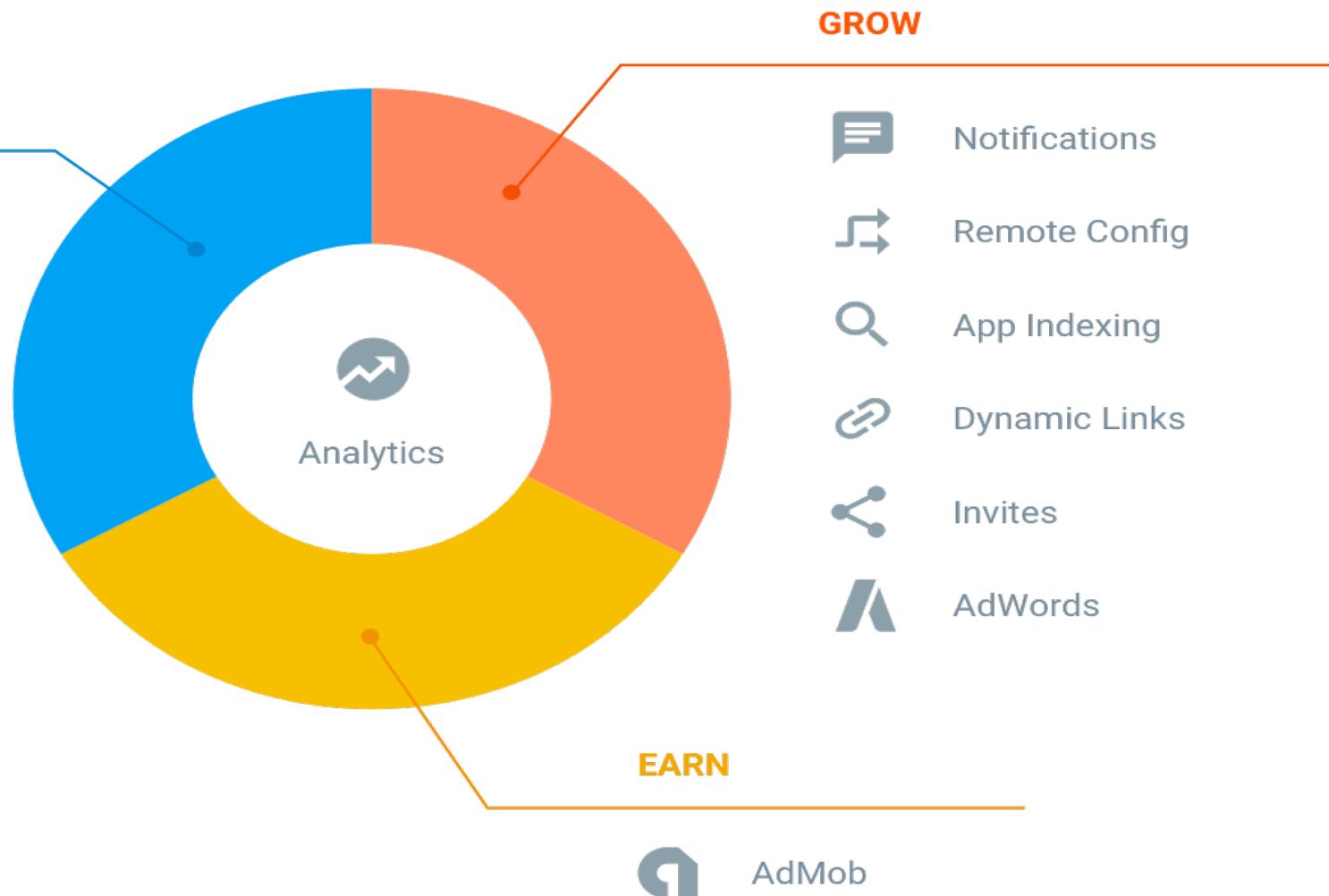
Authentication



# Enter The New Firebase

## DEVELOP

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





# Database



Goodbye RDBMS...



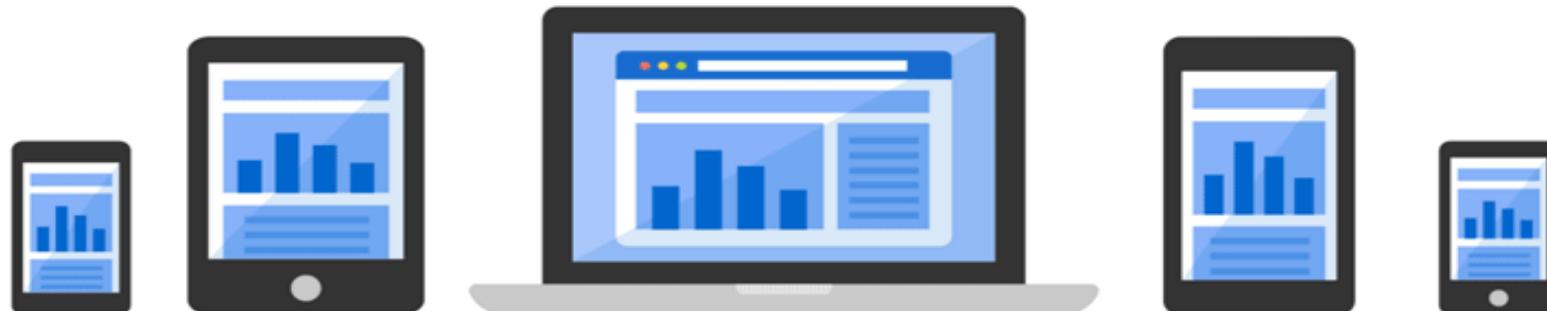
# Real Time Database

---

- ❑ 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, 3<sup>rd</sup> 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");`

Create a Key

## ❑ Set the value at the pointed node

- `mDBRef.child('row1').setValue(myPOJO);`

Firebase takes  
care of the Key

## ❑ Or push the value to create a unique key and set the value

- `mDBRef.push().setValue(myPOJO);`



# 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 solution.md. Below the navigation is a blue header bar with the Firebase logo and a 'Welcome back to Firebase' message. In the center, there's a large graphic of a person working on a smartphone. Below the graphic, there are links for Documentation, Sample code, API reference, and Support. On the left, there's a sidebar titled 'Your projects using Firebase' which lists a project named 'PaceMaker' with a URL 'pacemaker-a88f1.firebaseio.com'. At the bottom right, there are two buttons: 'CREATE NEW PROJECT' (highlighted with a red box) and 'IMPORT GOOGLE PROJECT'.



# 1. Create a Firebase Project

The screenshot shows the Firebase console interface on a Mac OS X desktop. The title bar includes standard OS X icons and the URL `console.firebaseio.google.com`. The main navigation bar has items like Lynda.com, Android Development, Electronic Marksheet, etc. A central modal dialog box is open, 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. In the background, the main Firebase dashboard shows a project named "PaceMaker" with the URL `pacemaker-a88f1.firebaseio.com`.



# 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. Underneath, there are links for Documentation, Sample code, API reference, and Support. A red box highlights the first project card, 'CoffeeMate FBI'. The page lists two projects:

- CoffeeMate FBI** (highlighted with a red border)
  - Project URL: coffeeamate-fbi.firebaseio.com
- PaceMaker**
  - Project URL: pacemaker-a88f1.firebaseio.com

At the bottom right, there are buttons for 'CREATE NEW PROJECT' and 'IMPORT GOOGLE PROJECT'.



# 1. Create a Firebase Project - Overview

The screenshot shows the Firebase console's Overview page for a project named "CoffeeMate FBI". 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 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 "Learn more" and "GET STARTED" buttons.



## 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 interface. On the left, a sidebar lists various services: Analytics, Authentication, Database, Storage, Hosting, Test Lab, Crash Reporting, Notifications, Remote Config, Dynamic Links, EARN, and AdMob. The main area is titled 'Settings' and has tabs for GENERAL, CLOUD MESSAGING, ANALYTICS, ACCOUNT LINKING, and SERVICE ACCOUNTS. Under the GENERAL tab, there's a section for 'Your project' with fields for Project name (CoffeeMate FBI), Public-facing name (CoffeeMate FBI), Project ID (coffeeamate-fbi), and Web API Key (AlzaSy...). Below this, under 'Your apps', it says 'There are currently no apps in the project. CoffeeMate FBI'. At the bottom, there are three 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). The URL in the browser bar is [console.firebaseio.google.com/project/coffeamate-fbi/settings/general/](https://console.firebaseio.google.com/project/coffeamate-fbi/settings/general/).



## 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 screenshot shows the Firebase Console interface with the URL `console.firebaseio.google.com/project/coffeemate-fbi-e815e/settings/general/`. The main navigation bar includes links for Lynda.com, Android Development, Electronic Marksheets, lynda.com, solution.md, Triathlon Ireland, Apple Developer, TeamViewer, MapMyRun, Garmin Connect, WIT Catalogue, OneDrive, Google Drive, and more.

The left sidebar lists various Firebase services: Analytics, Authentication, Database, Storage, Hosting, Test Lab, Crash Reporting, Notifications, Remote Config, Dynamic Links, AdMob, and EARN.

The central area displays a modal window titled "Add Firebase to your Android app" with three steps: "Enter app details", "Copy config file", and "Add to build.gradle". Step 3 is currently active.

The instructions in the modal state: "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."

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

```
buildscript {  
    dependencies {  
        // Add this line  
        classpath 'com.google.gms:google-services:3.0.0'  
    }  
}
```

Step 2: "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
```

Step 3: "Finally, press "Sync now" in the bar that appears in the IDE:"

A progress bar at the bottom indicates "Gradle files have changed since last sync" and features a blue "Sync now" button. A "FINISH" button is located at the bottom right of the modal.



# 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:**

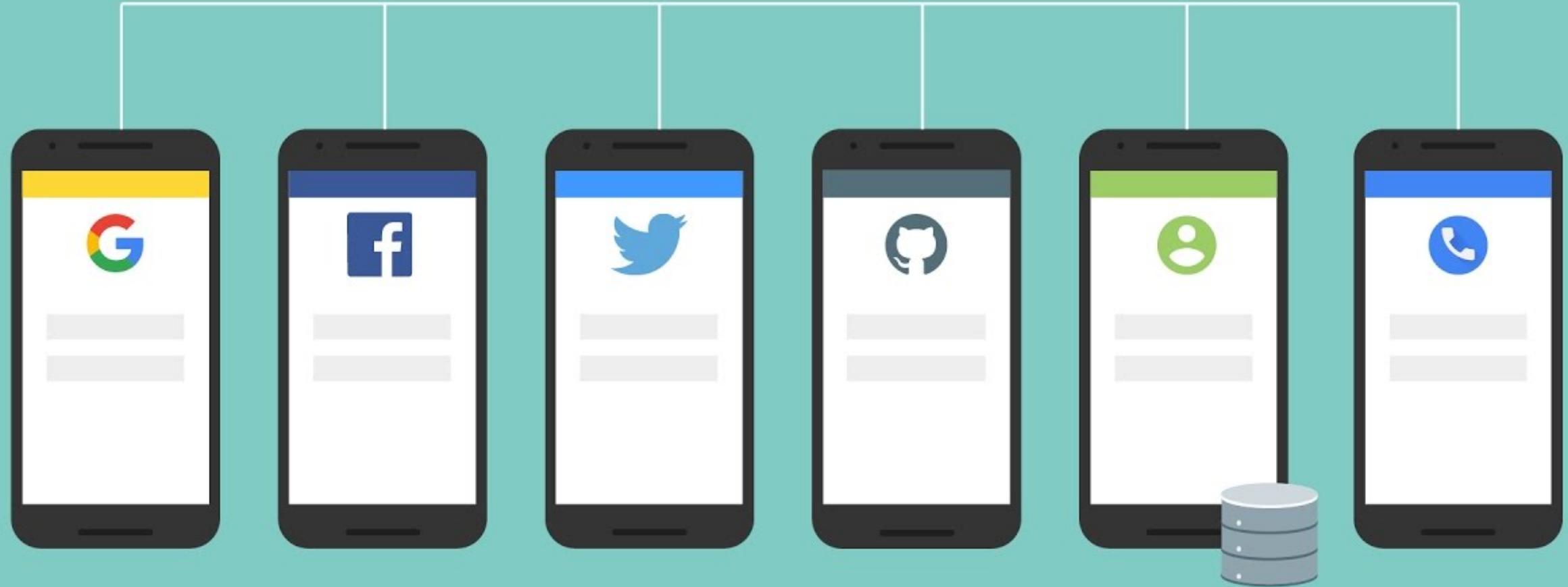
- The emulator displays the `CoffeeMateFBI` application.
- The app shows a list of recently added coffee items:
  - Firebase Coffee** (highlighted with a red box)
    - Shop: Fire Station
    - Rating: 2.0+
    - Price: €1.99
  - Coffee Latte**
    - Shop: Aldi
    - Rating: 2.0+
    - Price: €2.49
- The footer of the app shows the URL `ddrohan.github.io`.



# Pricing

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

| Products  | Spark Plan<br>Generous limits for hobbyists<br>Free              | Flame Plan<br>Fixed pricing for growing apps<br>\$25/month                               | Blaze Plan<br><a href="#">Calculate pricing for apps at scale</a><br>Pay as you go    |
|---|--|--|---|
| <b>Free Products</b><br>Authentication (except Phone Auth),<br>Analytics, App Indexing, Dynamic Links,<br>Invites, Remote Config, Cloud Messaging<br>(FCM), Performance Monitoring, and Crash<br>Reporting. | <span style="color: green;">✓ Included</span>                    | <span style="color: green;">✓ Included Free</span>                                       | <span style="color: green;">✓ Included Free</span>                                    |
| <b>Realtime Database</b><br>Simultaneous connections <span style="color: gray;">?</span><br>GB stored<br>GB downloaded<br>Automated backups   | 100<br><br>10 GB/month<br><br><span style="color: red;">✗</span> | 100K/instance<br><br>2.5 GB<br><br>20 GB/month<br><br><span style="color: red;">✗</span> | 100K/instance<br><br>\$5/GB<br><br>\$1/GB<br><br><span style="color: green;">✓</span> |

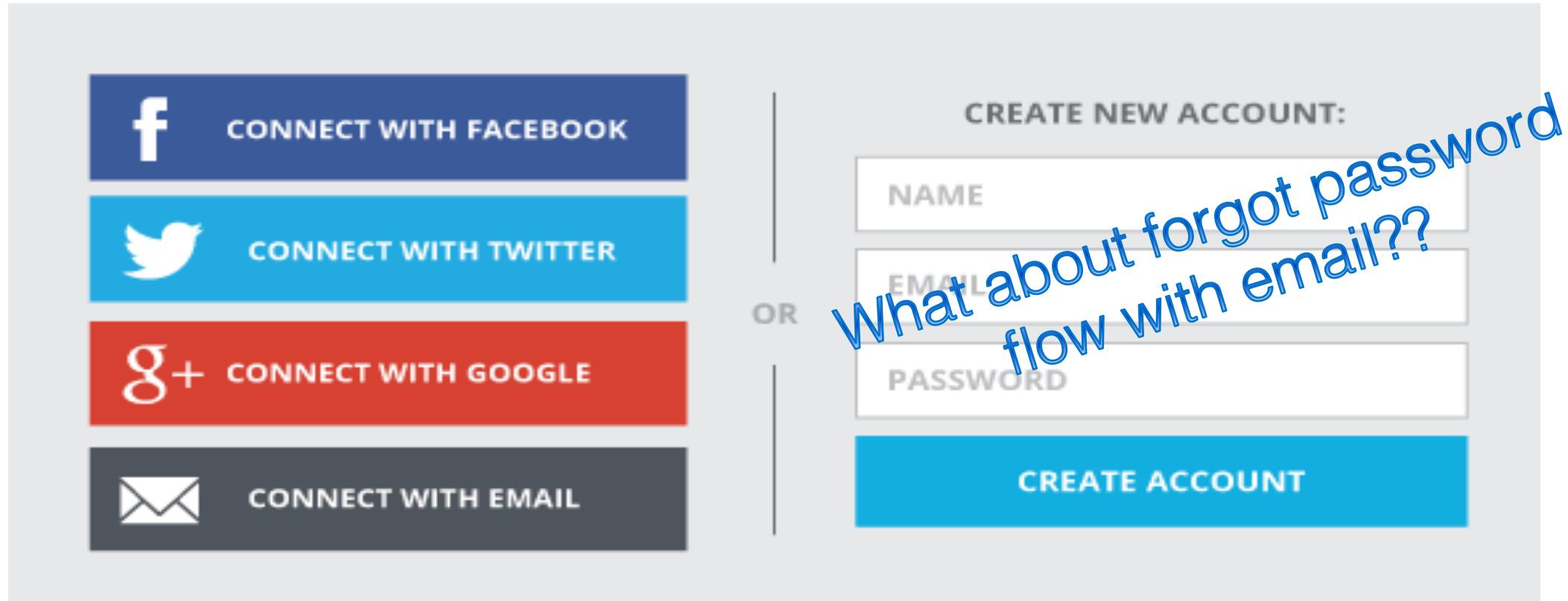


All goodness bundled as one...  
**Authentication**



# 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...



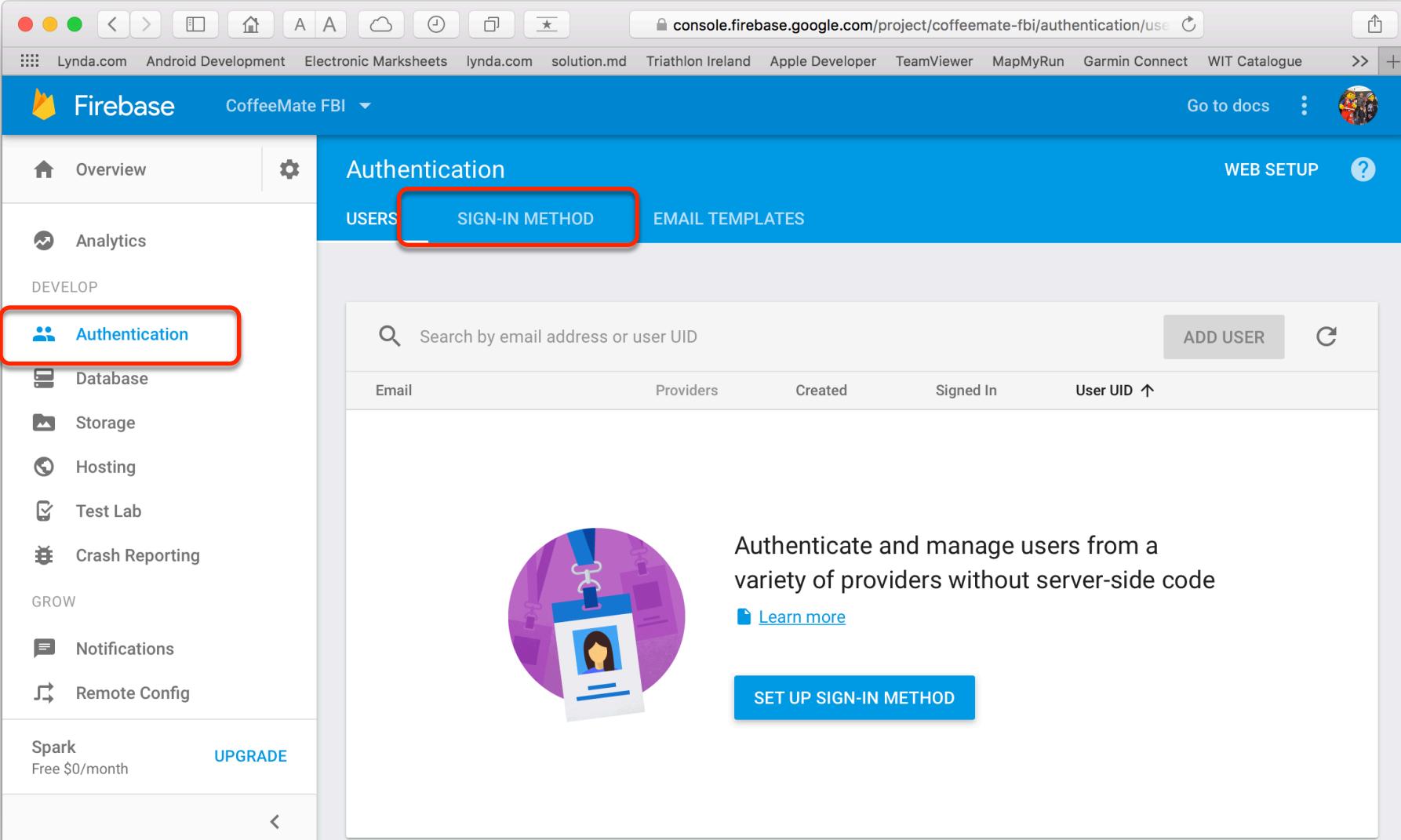
---

# 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

DEVELOP

Authentication 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 ADD USER

| Email         | Providers     | Created       | Signed In     | User UID ↑    |
|---------------|---------------|---------------|---------------|---------------|
| [Placeholder] | [Placeholder] | [Placeholder] | [Placeholder] | [Placeholder] |

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 console interface for setting up authentication methods. The left sidebar includes links for Overview, Analytics, Database, Storage, Hosting, Test Lab, Crash Reporting, Notifications, and Remote Config. The Authentication link is selected. The main content area is titled 'Authentication' and shows three tabs: USERS, SIGN-IN METHOD (which is selected), and EMAIL TEMPLATES. Below this, a section titled 'Sign-in providers' lists several providers: Email/Password (Disabled), Google (disabled and highlighted with a red box), Facebook (Disabled), Twitter (Disabled), GitHub (Disabled), and Anonymous (Disabled). At the bottom, there is a link for 'OAuth redirect domains'.

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



# 1. Setup your Sign-In Method \*

The screenshot shows the Firebase Authentication screen. On the left sidebar, 'Authentication' is selected under the 'DEVELOP' section. The main area displays the 'SIGN-IN METHOD' tab. It lists 'Email/Password' as disabled and 'Google' as enabled. A red box highlights the 'Enable' toggle switch for the Google provider. Below the providers, there is a note about automatically configuring Google sign-in for iOS and web apps, and instructions to add SHA1 fingerprints for Android apps. There are also sections for whitelisting client IDs and configuring the Web SDK.

| Provider       | Status   |
|----------------|----------|
| Email/Password | Disabled |
| Google         | Enabled  |

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](#).

Whitelist client IDs from external projects (optional) [?](#)

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, Database, Storage, Hosting, Test Lab, Crash Reporting, Notifications, and Remote Config. A "UPGRADE" button is visible at the bottom of the sidebar.

The main panel displays the "Authentication" provider settings. The "Email/Password" provider is listed as "Disabled". The "Google" provider is listed and has its "Enable" toggle switch turned on. Below the providers, there is a note about automatically configuring Google sign-in for iOS and web apps, and instructions to add an SHA1 fingerprint for Android apps via Project Settings. There are sections for "Whitelist client IDs from external projects (optional)" and "Web SDK configuration (optional)".

At the bottom right of the main panel, there are "CANCEL" and "SAVE" buttons. The "SAVE" button is highlighted with 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).addListenerForSingleValueEvent(
        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 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 shows a table of users. A single user entry is highlighted with a red box:

| Email                 | Provider | Created     | Last Signed In | User ID                      |
|-----------------------|----------|-------------|----------------|------------------------------|
| daveydrohan@gmail.com | G        | 18 Jul 2017 | 18 Jul 2017    | 1SQVbMgN5bcLg9JXgG0ts3DNO... |

Below the table, there are buttons for 'ADD USER' and a search bar. The bottom right corner of the table row contains navigation icons for rows per page (50), page number (1-1 of 1), and arrows.



# 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). The entire table area is also highlighted with a red box.

| 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 \*

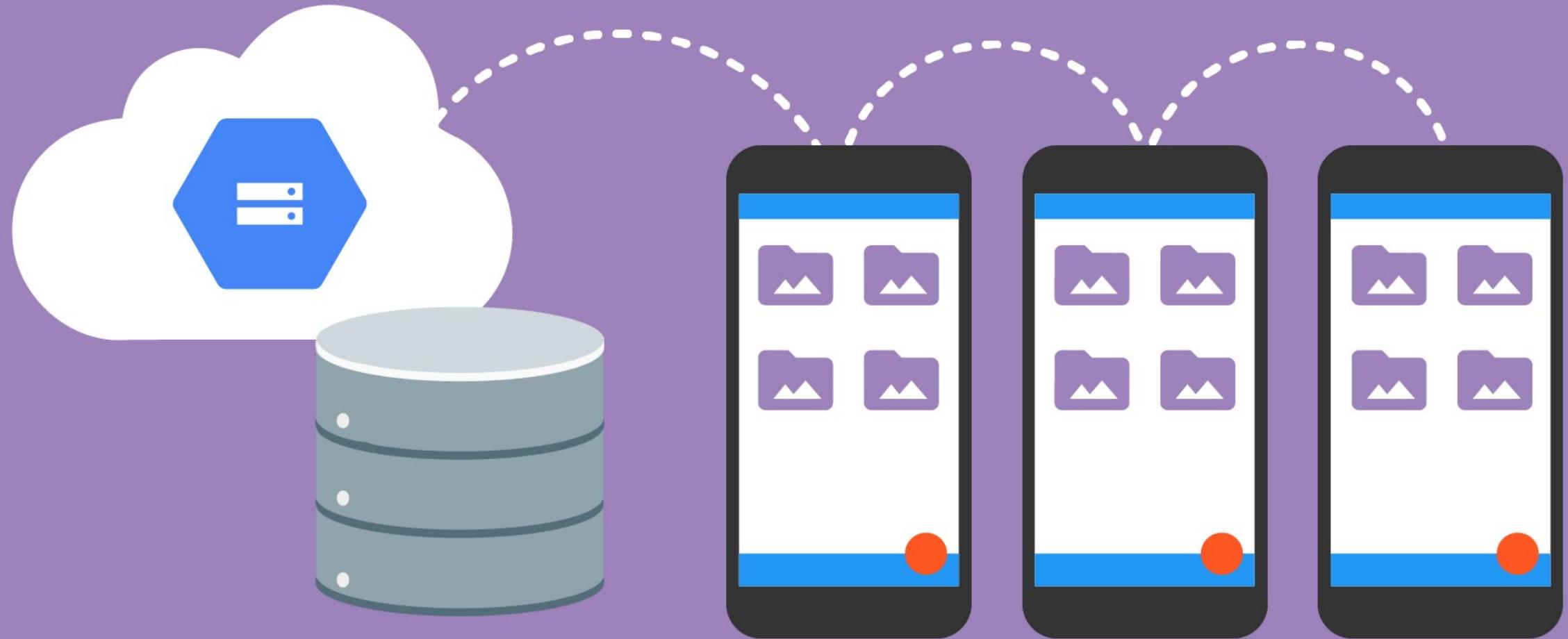
The screenshot shows the Firebase Realtime Database interface. On the left, a sidebar lists various services: Overview, Analytics, Authentication (which is highlighted with a red box), Database (also highlighted with a red box), Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, EARN, and AdMob. Below this is a plan section for 'Spark' (Free \$0/month) with an 'UPGRADE' button. The main area is titled 'Realtime Database' and shows the 'DATA' tab selected. It displays a hierarchical database structure under the project name 'coffeematefbi'. A specific node 'users' is expanded, showing three child nodes with user data:

- 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"

A red box highlights the 'Database' item in the sidebar, and another red box highlights the 'users' node in the tree view.



# 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 Register app    2 Download config file    3 Add Firebase SDK

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



# FirebaseStorageEx App - Setup \*

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

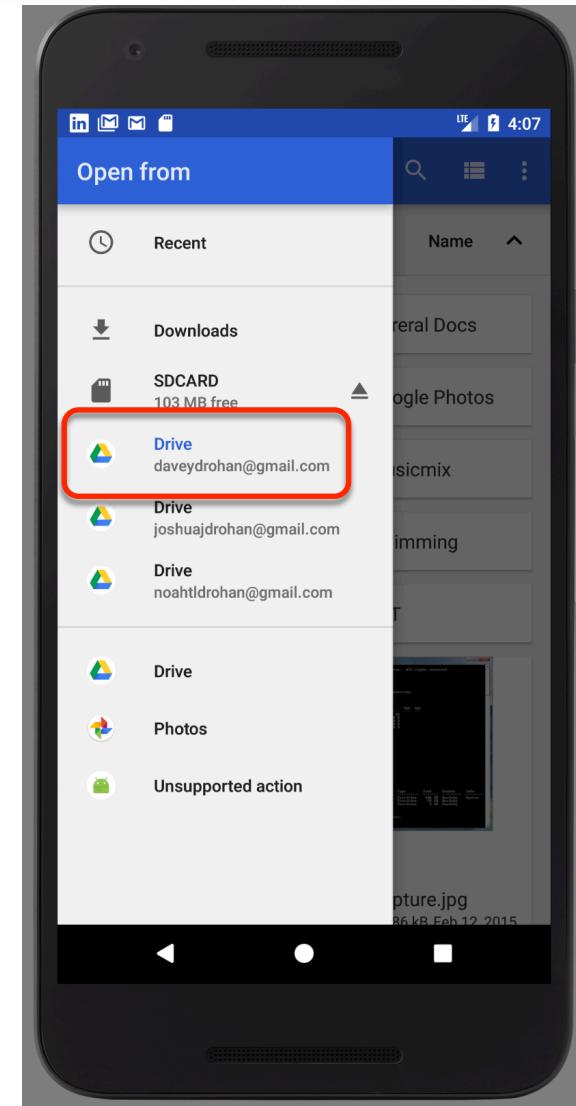
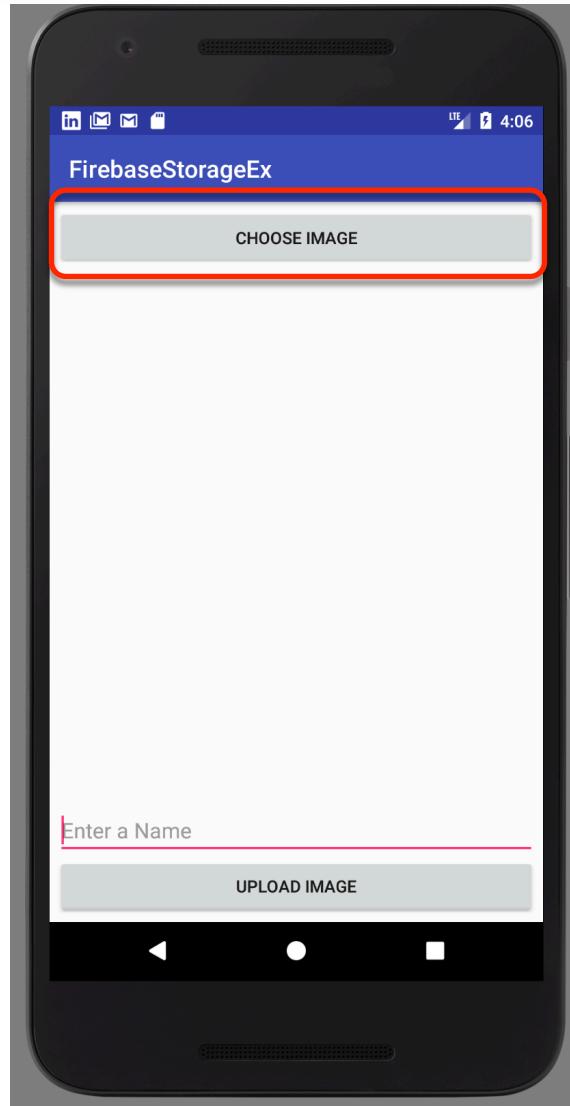
The screenshot shows the Firebase console's Overview page for the project "CoffeeMateFBI". The left sidebar lists various services: Analytics, Authentication, Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, Remote Config, Dynamic Links, EARN, and AdMob. The 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<br>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".

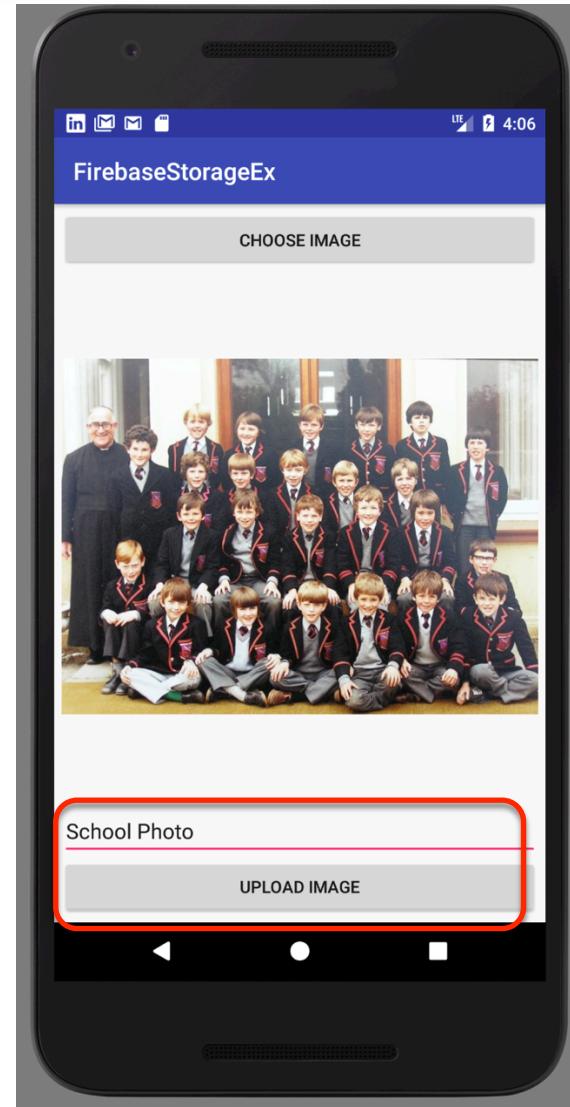
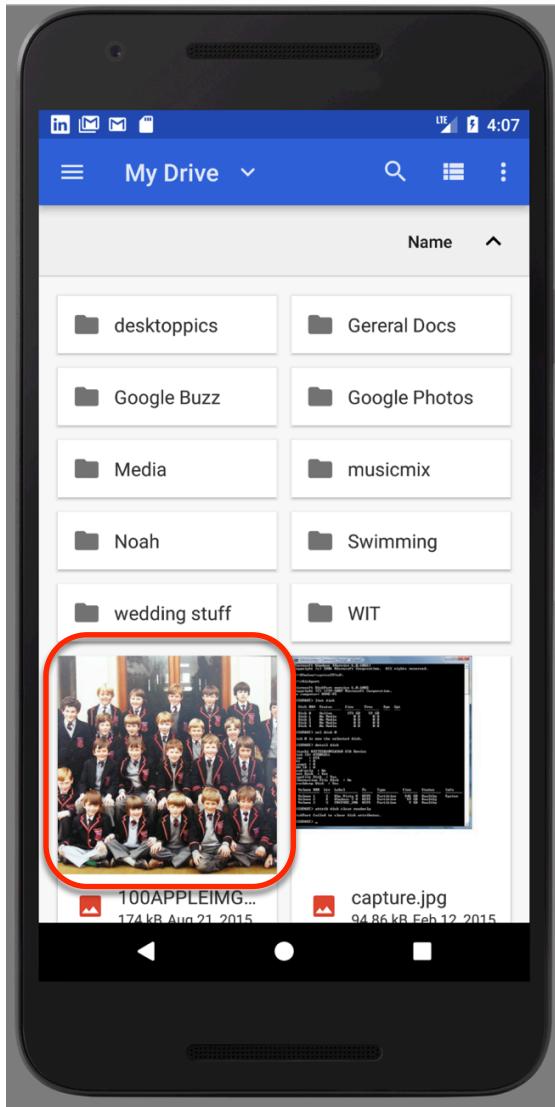


# FirebaseStorageEx App - Upload \*





# 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 (which is 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 'images'. A file named 'School Photo' is selected, showing its details in a modal window. The modal displays the following information:

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

**School Photo**

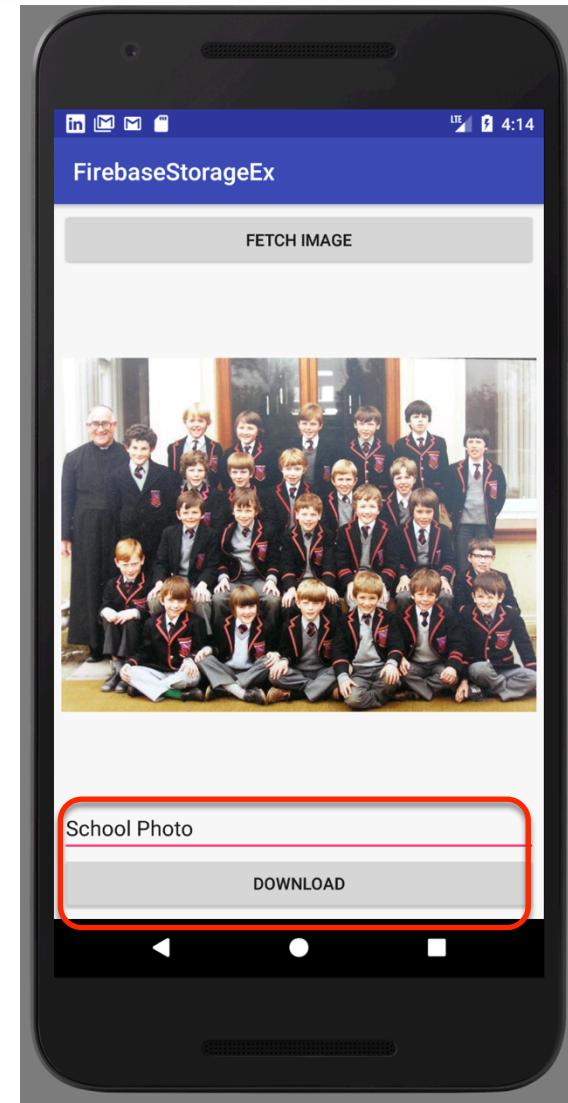
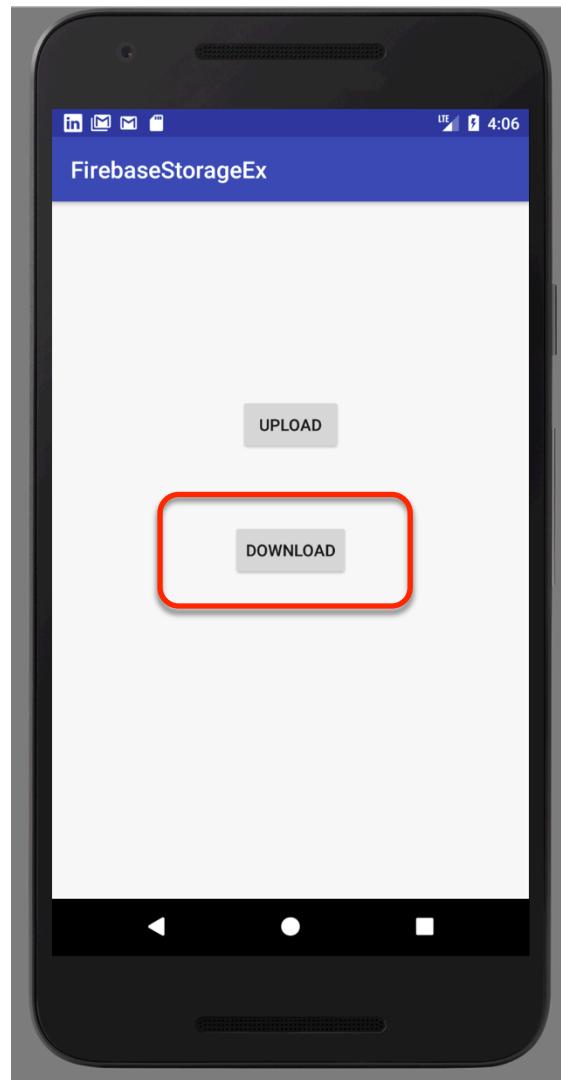
Name: School Photo  
Size: 170.07 KB  
Type: image/jpeg  
Created: 4 Aug 2017, 16:03:40  
Updated: 4 Aug 2017, 16:03:40

File location  
Other metadata

At the top right of the main storage view, there is a blue 'UPLOAD FILE' button.



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



# 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. (FirebaseUI)
- ❑ Test lab which is available in paid plan (Blaze), is an amazing feature for testing your app on different real and virtual devices (next section)
- ❑ Set developer mode to true when testing Remote Config (next section).



# References & Links

---

- ❑ [Presentation by Kaushal Dhruw & Shakti Moyal 2016](#)
- ❑ <https://firebase.google.com>
- ❑ Demo app available at <https://goo.gl/WBP5fR>



---

# Questions?