

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

Firebase Integration





Agenda

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



Database



Goodbye RDBMS...

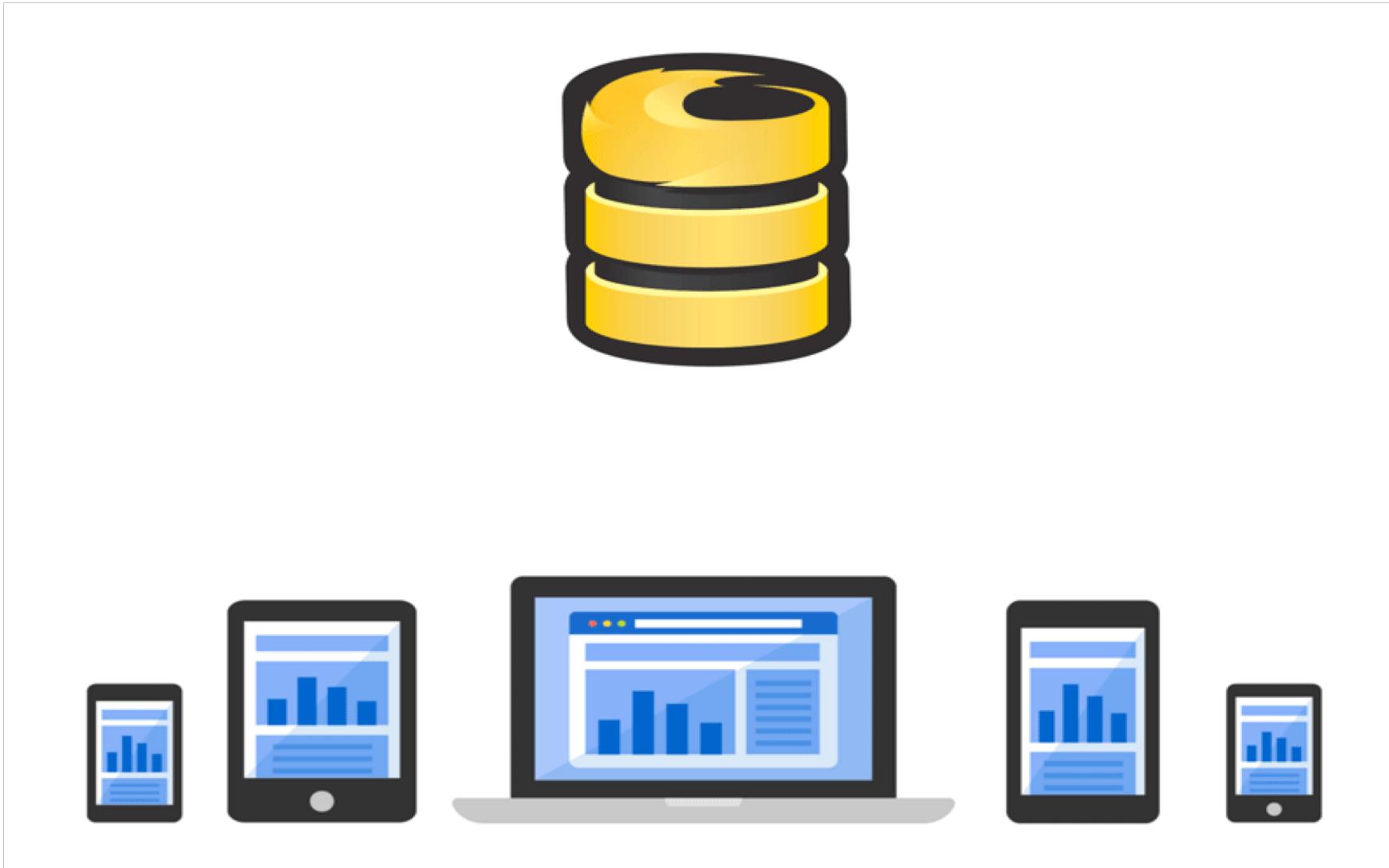


Real Time Database

- ❑ 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");`

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, Performance, GROW, Notifications, and Remote Config. The main area displays the database structure for a project named "coffeematefbi". The database has two main nodes: "coffees" and "user-coffees". Under "user-coffees", there is a child node with a key "1SQVbMgN5bcLg9JXgG0ts3DNOAV2" which contains a child node "-Kpt-S0GNxFpsOBSzS-". This child node holds five data fields: "address" (Co. Waterford X91 W29R Ireland), "favourite" (false), "googlephoto" (a URL to a Google User Content image), "latitude" (52.24), and "longitude" (-7.16).

Authentication

Database

Storage

Hosting

Functions

Test Lab

Crash Reporting

Performance

GROW

Notifications

Remote Config

https://coffeematefbi.firebaseio.com/

Default security rules require users to be authenticated.

LEARN MORE DISMISS

coffeematefbi

- + coffees
- user-coffees
 - 1SQVbMgN5bcLg9JXgG0ts3DNOAV2
 - -Kpt-S0GNxFpsOBSzS-
 - 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...

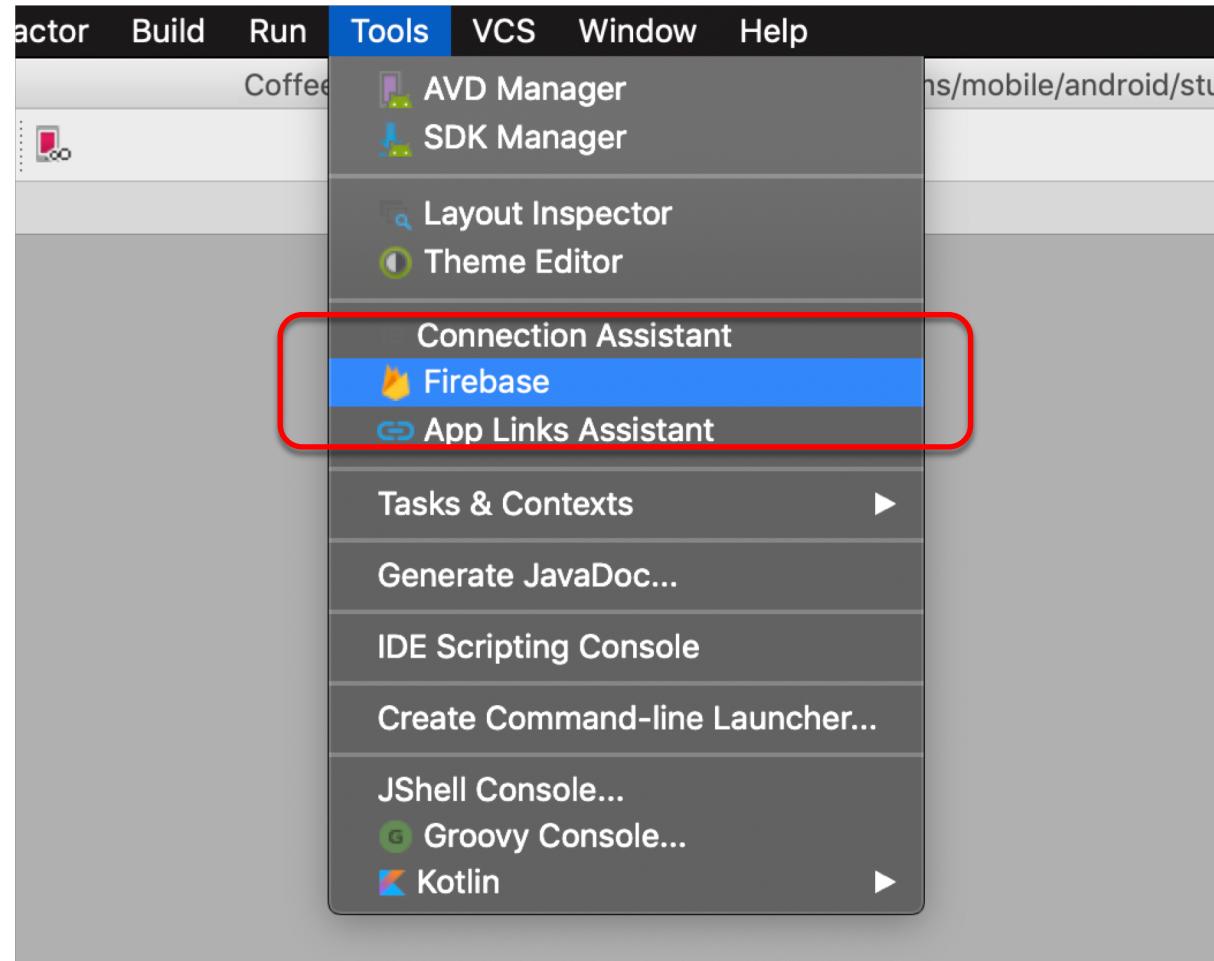


CoffeeMate.9.0

Setup & Code Highlights



1. Connect your App to Firebase *





1. Connect your App to Firebase *

The screenshot shows the Android Studio Assistant tool with the 'Assistant' tab selected. The main content area displays the Firebase services:

- Firebase**: A brief introduction stating "Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app." with a [Learn more](#) link.
- Analytics**: Describes measuring user activity and engagement with free, easy, and unlimited analytics. [More info](#)
- Cloud Messaging**: Describes delivering and receiving messages and notifications reliably across cloud and device. [More info](#)
- Authentication** (highlighted with a red border): Describes sign-in and management of users with ease, supporting emails, Google Sign-In, Facebook and other login providers. [More info](#)
- Realtime Database**: Describes storing and syncing data in realtime across all connected clients. [More info](#)
- Storage**: Describes storing and retrieving large files like images, audio, and video without writing server-side code. [More info](#)
- Remote Config**: Describes customizing and experimenting with app behavior using cloud-based configuration parameters. [More info](#)
- Test Lab**: Describes testing apps against a wide range of physical devices hosted in Google's cloud. [More info](#)
- Crash Reporting**: Describes getting actionable insights and reports on app crashes, ANRs or other errors. [More info](#)
- App Indexing**: Describes getting app content into Google Search. [More info](#)
- Dynamic Links**

At the bottom right of the Assistant window, it shows "465 ms".



1. Connect your App to Firebase *

The screenshot shows a web-based guide for setting up Firebase Authentication. At the top, it says "Email and password authentication". Below that, a paragraph explains how to use Firebase Authentication for user sign-in. A "Launch in browser" button is present. The main content is organized into three numbered steps:

- ① Connect your app to Firebase**
A "Connect to Firebase" button is shown.
- ② Add Firebase Authentication to your app**
A "Add Firebase Authentication to your app" button is shown. Below it, text explains that to use an authentication provider, you need to enable it in the [Firebase console](#). It also mentions going to the Sign-in Method page in the Firebase Authentication section to enable Email/Password sign-in and other identity providers.
- ③ Check current auth state**
Text instructions say to declare an instance of `FirebaseAuth`:

```
private FirebaseAuth mAuth;
```

In the `onCreate()` method, initialize the `FirebaseAuth` instance:

```
mAuth = FirebaseAuth.getInstance();
```



1. Connect your App to Firebase *

Sign in with Google

Choose an account
to continue to [Android Studio](#)

Joshua Everett Drohan
joshuajdrohan@gmail.com
Signed out

David Drohan
daveydrohan@gmail.com

Noah Drohan
noahtldrohan@gmail.com

Sign in with Google

Android Studio wants to access
your Google Account
 daveydrohan@gmail.com

This will allow [Android Studio](#) to:

- View and manage your data across Google Cloud Platform services
- View and manage your applications deployed on Google App Engine
- View and manage your Actions on Google.
- View and administer all your Firebase data and settings

Make sure you trust Android Studio
You may be sharing sensitive info with this site or app.
Learn about how Android Studio will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your [Google Account](#).

[Learn about the risks](#)

[Cancel](#) [Allow](#)



1. Connect your App to Firebase *

A screenshot of a web browser window titled "developer.android.com/studio/login?success=true". The page displays a "Success!" message, stating "You've signed in to Android Studio." and "To continue, go back to Android Studio." A large green circle with a white checkmark is prominently displayed. The browser's address bar shows the URL. The top navigation bar includes links for "Developers", "Platform", "Android Studio", "Google Play", "Android Jetpack", "Docs", and "News".

Success!

You've signed in to Android Studio.

To continue, go back to Android Studio.

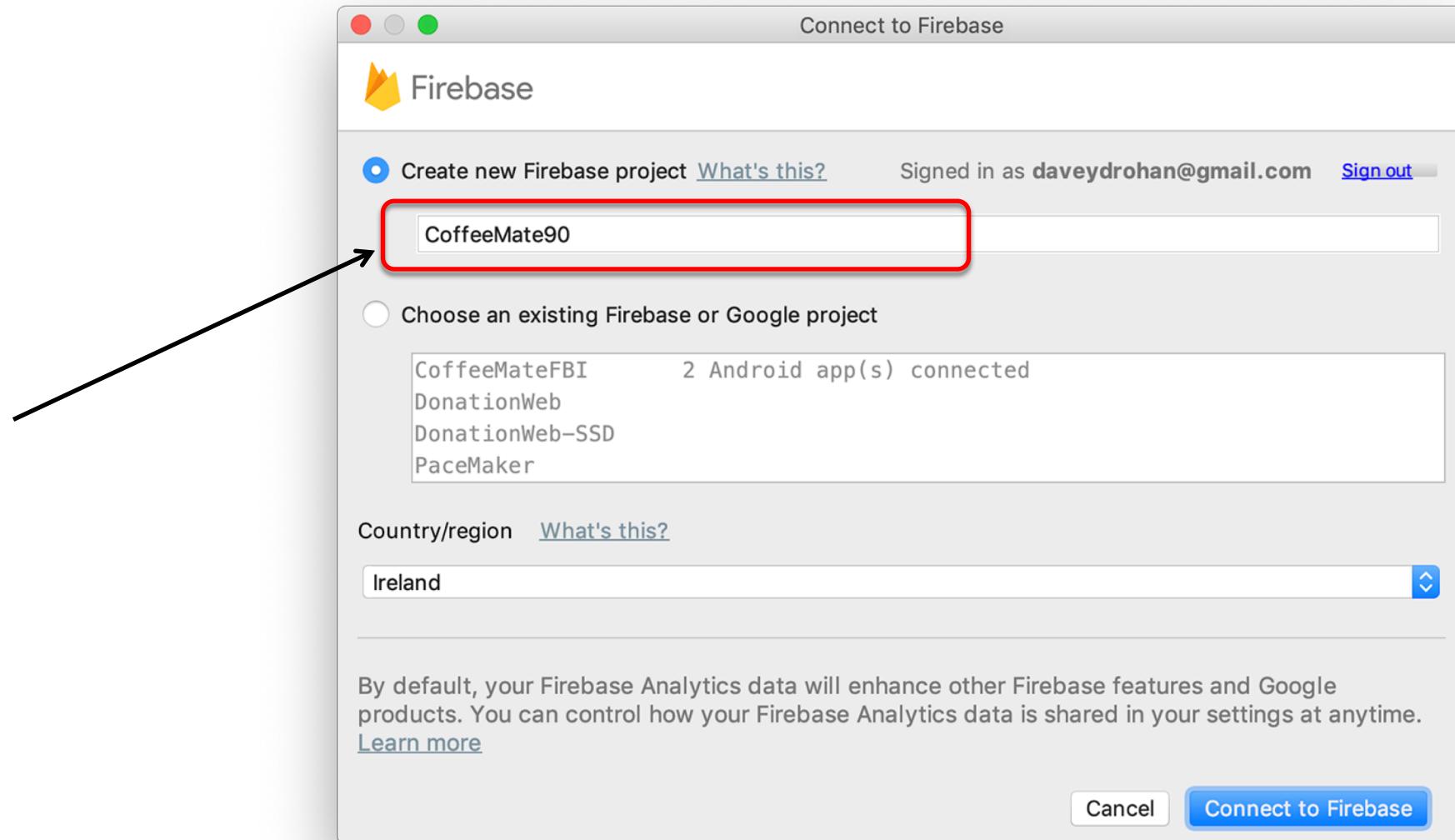
Explore Google services you can now use in your Android app:

 [Firebase](#)

 [Google Cloud Platform](#)



1. Connect your App to Firebase *





1. Connect your App to Firebase *

Connect to Firebase

Create new Firebase project [What's this?](#)

Signed in as **daveydrohan@gmail.com** [Sign out](#)

CoffeeMate FullFat Server

Choose an existing Firebase or Google project

CoffeeMateFBI 2 Android app(s) connected
DonationWeb
DonationWeb-SSD
PaceMaker

Country/region [What's this?](#)

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 anytime.
[Learn more](#)

e ① Connect your app to Firebase

✓ Connected



2. Add Firebase to your Android App *

The screenshot shows the official Firebase website at firebase.google.com. The page features a yellow header with the Firebase logo and navigation links for Products, Use Cases, Pricing, Docs, Support, a search bar, and a 'GO TO CONSOLE' button. Below the header is a large graphic of a smartphone displaying code snippets related to Firebase services like FIRAuth and FIRDatabase. The main text on the page reads: 'Firebase helps mobile app teams succeed.' There are two calls-to-action: 'GET STARTED' and 'WATCH THE VIDEO'. At the bottom, there are three promotional cards: 'Build apps fast, without managing infrastructure', 'Backed by Google, trusted by top apps', and 'One platform, with products that work better together'.



2. Add Firebase to your Android App *

Welcome to Firebase!

Tools from Google for developing great apps, engaging with your users and earning more through mobile ads.

[Learn more](#) [Documentation](#) [Support](#)

Recent projects

- [Add project](#)
- [Explore a demo project](#)
- DonationWeb**
donationweb-vue
- DonationWeb-SSD**
donationweb-ssd
- CoffeeMateFBI**
coffeematefb1
- PaceMaker**
pacemaker-a88f1



2. Add Firebase to your Android App *

Welcome to Firebase!

Tools from Google for developing great apps, engaging with your users and earning more through mobile ads.

[Learn more](#) [Documentation](#) [Support](#)

Recent projects

Add project

CoffeeMate FullFat Server
coffeemate-fullfat-server-46ff4

CoffeeMateFBI
coffeematefbi

DonationWeb
donationweb-vue

DonationWeb-SSD
donationweb-ssd

PaceMaker
pacemaker-a88f1



2. Add Firebase to your Android App *

The screenshot shows the Firebase Project Overview page for the project 'CoffeeMate FullFat Server'. The left sidebar contains links for Project Overview, Authentication, Database, Storage, Hosting, Functions, and ML Kit. The main area displays a message: 'You need to choose data sharing settings for this project' with a 'Choose data sharing settings' button. Below this, there's a section for 'CoffeeMate FullFat Server' showing a 'Spark plan' and a list of connected apps: 'ie.cmff' and another app represented by a blue icon. A black arrow points from the text 'Add your first Android app' in the slide content down to the 'ie.cmff' app entry. At the bottom, it says 'Waiting for Analytics data...'.



2. Add Firebase to your Android App *

The screenshot shows the Firebase Project Overview page for the 'CoffeeMate FullFat Server' project. The left sidebar includes links for Project Overview, Authentication, Database, Storage, Hosting, Functions, ML Kit, Quality, Analytics, and Spark (Free \$0/month). The main area displays a message: 'You need to choose data sharing settings for this project' with a 'Choose data sharing settings' button. Below this, there's a section for the app 'ie.cmff' with a purple Android icon, a gear icon, and a 'Continue SDK setup' button. A black arrow points from the text 'Add Firebase to your Android App' in the slide content down to the 'Continue SDK setup' button. At the bottom, it says 'Waiting for Analytics data...'.



2. Add Firebase to your Android App *

The screenshot shows the Firebase console's 'Add Firebase to your Android app' wizard. Step 1, 'Register app', is completed with the package name 'ie.cmff'. Step 2, 'Download config file', is active, with a blue arrow pointing to the 'Download google-services.json' button. Below it, instructions say to switch to the Project view in Android Studio to see the project root directory. A blue arrow points to the 'google-services.json' file in the Android Studio project structure. The project structure shows the following hierarchy:

- MyApplication (~/Desktop/My)
- .gradle
- .idea
- app
 - build
 - libs
 - src
 - .gitignore
 - app.iml
 - build.gradle
 - google-services.json
 - proguard-rules.pro
- gradle

At the bottom, there are 'Previous' and 'Next' buttons.



2. Add Firebase to your Android App *

The screenshot shows the Firebase console for a project named "CoffeeMate FullFat Server". The "Overview" tab is selected. On the left, there's a sidebar with "Register app" and "Download config file" steps completed, and "Add Firebase SDK" step 3 in progress. The main content area provides instructions for adding the Google services plugin to the build.gradle files. It includes code snippets for the Project-level build.gradle and App-level build.gradle.

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

```
buildscript {  
    dependencies {  
        // Add this line  
        classpath 'com.google.gms:google-services:4.0.1'  
    }  
}
```

App-level build.gradle (<project>/<app-module>/build.gradle):

```
dependencies {  
    // Add this line  
    implementation 'com.google.firebaseio:firebase-core:16.0.1'  
}  
...  
// Add to the bottom of the file  
apply plugin: 'com.google.gms.google-services'
```

Includes Analytics by default ⓘ

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



2. Add Firebase to your Android App *

The screenshot shows the Firebase console for a project named "CoffeeMate FullFat Server". It displays two build.gradle files:

```
Project-level build.gradle (<project>/build.gradle):
buildscript {
    dependencies {
        // Add this line
        classpath 'com.google.gms:google-services:4.0.1'
    }
}

App-level build.gradle (<project>/<app-module>/build.gradle):
dependencies {
    // Add this line
    implementation 'com.google.firebaseio:firebase-core:16.0.1'
}
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

Below the files, a message says: "Finally, press 'Sync now' in the bar that appears in the IDE:". A yellow status bar at the bottom left indicates "Gradle files have changed sir" and contains a blue "Sync now" button.

At the bottom, there are "Previous" and "Next" buttons, and a step indicator "4 Run your app to verify installation".



2. Add Firebase to your Android App *

The screenshot shows a browser window for the Firebase console at console.firebaseio.google.com/u/0/project/coffeemate-fullfat-server-4. The page title is "CoffeeMate FullFat Server - Overview - Firebase con...". The main content is titled "Add Firebase to your Android app" and includes a checklist:

- ✓ Register app
Android package name: ie.cmff
- ✓ Download config file
- ✓ Add Firebase SDK
- 4 Run your app to verify installation

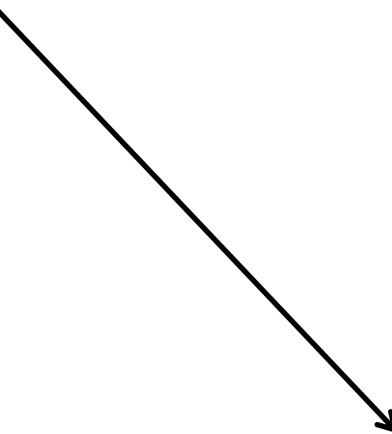
A green success message at the bottom states: "Congratulations, you've successfully added Firebase to your app!"

At the bottom, there are "Previous" and "Continue to the console" buttons. To the right of the checklist is a blue smartphone icon with a yellow Firebase logo on its screen, set against a background of floating stars.



2. Add Firebase to your Android App *

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



The screenshot shows the 'API key' settings page in the Google Cloud Platform. The URL is `console.cloud.google.com/apis/credentials/key/185ba235-94a2-4ef...`. The project is 'CoffeeMate FullFat Project'. The tab 'Application restrictions' is selected. Under 'Application restrictions', the option 'Android apps' is selected. Below that, there's a section titled 'Restrict usage to your Android apps (Optional)' with a note about getting package name and SHA-1 certificate fingerprint. A command-line interface box shows the command `$ keytool -list -v -keystore mystore.keystore`. Below it, a table lists package names and SHA-1 certificate fingerprints. One entry for 'ie.cmff' is highlighted with a red box. At the bottom, there's a note about a 5-minute delay for changes to take effect and 'Save' and 'Cancel' buttons.

Package name	SHA-1 certificate fingerprint
ie.cmff	AA:BB:CC:DD:EE:FF

[+ Add package name and fingerprint](#)



CoffeeMate.9.0 Dependencies (Dec/2018) *

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'com.android.support:appcompat-v7:28.0.0'  
    implementation 'com.android.support:support-v4:28.0.0'  
    implementation 'com.android.support:design:28.0.0'  
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
  
    implementation 'com.makeramen:roundedimageview:2.2.1'  
    implementation "com.android.support:recyclerview-v7:28.0.0"  
    implementation "com.android.support:cardview-v7:28.0.0"  
  
    implementation 'com.google.android.gms:play-services-auth:15.0.1'  
    implementation 'com.google.android.gms:play-services-maps:15.0.1'  
    implementation 'com.google.android.gms:play-services-location:15.0.1'  
  
    implementation 'com.shobhitpuri.custombuttons:google-signin:1.0.0'  
  
    implementation 'com.google.firebaseio:firebase-core:16.0.1'  
    implementation 'com.google.firebaseio:firebase-auth:16.0.1'  
    implementation 'com.google.firebaseio:firebase-database:16.0.1'  
  
    implementation 'com.firebaseioui:firebase-ui-database:4.0.0'  
  
    implementation 'com.github.bumptech.glide:glide:4.8.0'  
    annotationProcessor 'com.github.bumptech.glide:compiler:4.8.0'  
}
```



FirebaseManager – our Database utility class *

```
public class FirebaseManager {  
    private static final String TAG = "coffeemate";  
    public DatabaseReference mFirebaseDatabase;  
    public static String mFirebaseUserId;  
    public FirebaseListener mFirebaseListener;
```

❑ Our Firebase db reference.

```
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);  
}
```

❑ Binding to our db instance.



FirebaseManager – usage *

```
public FirebaseAuth mFirebaseAuth;  
public FirebaseUser mFirebaseUser;  
public FirebaseManager mFirebaseManager;
```



```
@Override  
public void onCreate() {  
    super.onCreate();  
    Log.v("coffeemate", "CoffeeMate App Started");  
    mInstance = this;  
    mFirebaseManager = new FirebaseManager();  
    mFirebaseManager.open();  
}
```

- ❑ Our utility class reference (inside our Application object class).

- ❑ Creating & ‘Opening’ a connection to our Firebase db.



FirebaseManager – Querying the database *

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

```
public Query getAllCoffees()
{
    return mFirebaseDatabase.child("user-coffees")
        .child(mFirebaseUserId)
        .orderByChild("rating");
}

public Query getFavouriteCoffees()
{
    return mFirebaseDatabase.child("user-coffees")
        .child(mFirebaseUserId)
        .orderByChild("favourite")
        .equalTo(true);
}
```

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



FirebaseManager – usage (in CoffeeFragment) *

```
@Override  
public void onResume() {  
    super.onResume();  
  
    if (!favourites)  
        query = app.mFirebaseManager.getAllCoffees();  
    else  
        query = app.mFirebaseManager.getFavouriteCoffees();  
  
    updateView(query);  
}
```

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



FirebaseManager – our Database utility class *

- Adding an ‘ValueEventListener’ and triggering our callback

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



FirebaseManager – usage (in EditFragment) *

```
@Override  
public void onCreate(Bundle savedInstanceState)  
{  
    super.onCreate(savedInstanceState);  
  
    if(getArguments() != null)  
        app.mFirebaseManager.getACoffee(  
            getArguments().getString("coffeekey"))  
    );  
}
```

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



FirebaseManager – usage (in EditFragment) *

```
//@Override  
public void updateUI() {...}  
  
@Override  
public void onSuccess(DataSnapshot dataSnapshot) {  
    aCoffee = dataSnapshot.getValue(Coffee.class);  
    Log.v("coffeemate", "getACoffee() snapshot is " + aCoffee );  
    if(aCoffee != null)  
        updateUI();  
}  
  
@Override  
public void onFailure() {  
    Log.v("coffeemate", "Unable to Read/Update Coffee Data");  
    Toast.makeText(getActivity(), "Unable to Read/Update Coffee Data",  
        Toast.LENGTH_LONG).show();  
}
```

- 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 displays a composite screenshot of a mobile application and a web-based Firebase console interface.

Mobile Application (Left):

- Header:** "Favourite Coffee's" with a home icon.
- Item List:**
 - Item 1:** "Firebase Cuppa" (star icon), "€1.49", "The Fire Station", "3.5*", "X" button.
- Bottom Bar:** "ddrohan.github.io" and a blue circular icon with a white envelope symbol.

Firebase Console (Right):

- Project Overview:** "coffeeamate-fullfat-serve-46ff4" is selected.
- Database:** Shows the "Database" tab with the following structure:
 - Root Level:** "coffees", "user-coffees".
 - under "user-coffees":**
 - Node R9Q...4dmB3:**
 - Marker:** "name: base Cuppa lite"
 - Address:** "1 Matties Hill, Roanmore Cres, Waterford, X91 K..."
 - Price:** 1.49
 - Rating:** 3.5
 - Shop:** "The Fire Station"
 - UID:** "R9Q...4dmB3"
 - User Token:** "1134...00000125"
 - Node -LThfulnOATf24d-8jGI:**
 - Address:** "Waterford Shopping Centre, Paddy Browne's Rd, L..."
 - Marker:** "name: test coffee lite"



Firebase Console (actual data)

The image displays a side-by-side comparison between a mobile application interface and the Firebase Database console.

Mobile Application (Left): A screenshot of a smartphone showing a mobile application titled "Favourite Coffee's". The screen lists a single coffee shop entry: "Firebase Cuppa lite" located "The Fire Station" with a price of "€1.49" and a rating of "3.5". There is a red "X" button to remove the item. The bottom of the screen shows the URL "ddrohan.github.io" and a blue circular icon with a white envelope.

Firebase Database Console (Right): A screenshot of the Firebase Database interface for the project "CoffeeMate FullFat Server". The database structure is as follows:

```
coffeemate-fullfat-server-46ff4
  ├── coffees
  ├── user-coffees
  │   ├── R90[REDACTED]dmB3
  │   |   ├── -LTcFs7smvvkiHoo6rHU
  |   |   |   ├── address: "1 Matties Hill, Roanmore Cres, Waterford, X91 K..."
  |   |   |   ├── favourite: true
  |   |   |   ├── googlephoto: "https://lh5.googleusercontent.com/-AXr-7Z4gX7k/..."
  |   |   |   ├── marker
  |   |   |   |   └── name: "Firebase Cuppa lite"
  |   |   |   ├── price: 1.49
  |   |   |   ├── rating: 3.5
  |   |   |   ├── shop: "The Fire Station"
  |   |   |   ├── uid: "R90[REDACTED]dmB3"
  |   |   |   └── usertoken: "1134[REDACTED]25"
  |
  |   ├── -LThfulnOATf24d-8jGI
  |   |   ├── address: "Waterford Shopping Centre, Paddy Browne's Rd, L..."
  |   |   ├── favourite: false
  |   |   ├── googlephoto: "https://lh5.googleusercontent.com/-AXr-7Z4gX7k/..."
  |   |   ├── marker
  |   |   |   └── name: "test coffee lite"
```



Cloud Firestore



Firebase Console – Cloud Firestore

The screenshot shows the Firebase console interface for the project "CoffeeMateFBI". The left sidebar includes sections for DEVELOP (Authentication, Database, Storage, Hosting, Functions), STABILITY (Crashlytics, Performance, Test Lab), ANALYTICS (Dashboard, Events, Audiences, Attribution), and GROW (Predictions, Cloud Messaging, Remote Config). The main area is titled "Database" and displays two database options: "Realtime Database" and "Cloud Firestore (BETA)". A red box highlights the "Cloud Firestore (BETA)" section, which describes it as "The next generation of the Real-time Database with more powerful queries and automatic scaling". Below this, a note states "Default security rules require users to be authenticated." The right side shows a hierarchical view of the database structure under "coffeematefbi", including "coffees", "user-coffees" (with child nodes like "1SQVbMgN5bcLg9JXgG0ts3DNOAV2" and "C3PpngqTI8OxuHvNtG4wZsVMhZv2"), and "users" (with child node "1SQVbMgN5bcLg9JXgG0ts3DNOAV2" containing fields like "favCoffeeImageURL", "userEmail", "userId", and "userName").

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

Cloud Firestore BETA
The next generation of the Real-time Database with more powerful queries and automatic scaling

Default security rules require users to be authenticated.

coffeematefbi

- + coffees
- user-coffees
 - + 1SQVbMgN5bcLg9JXgG0ts3DNOAV2
 - + C3PpngqTI8OxuHvNtG4wZsVMhZv2
 - + wotuVu5IZhh0HRcCHRkJgsWh9y1
- users
 - 1SQVbMgN5bcLg9JXgG0ts3DNOAV2
 - + favCoffeeImageURL: "iVBORw0KGgoAAAANSUhEUgAAAHgAAAB4CAIAAC2BqGFAAA..."
 - + userEmail: "daveydrohan@gmail.com"
 - + userId: "1SQVbMgN5bcLg9JXgG0ts3DNOAV2"
 - + userName: "David_Drohan"



What is it?

- ❑ Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps—at a global scale.
- ❑ Though this may sound like something similar to the **Realtime Database**, Firestore brings many new things to the platform that makes it into something completely different from Realtime Database.



Improved Querying and Data Structure

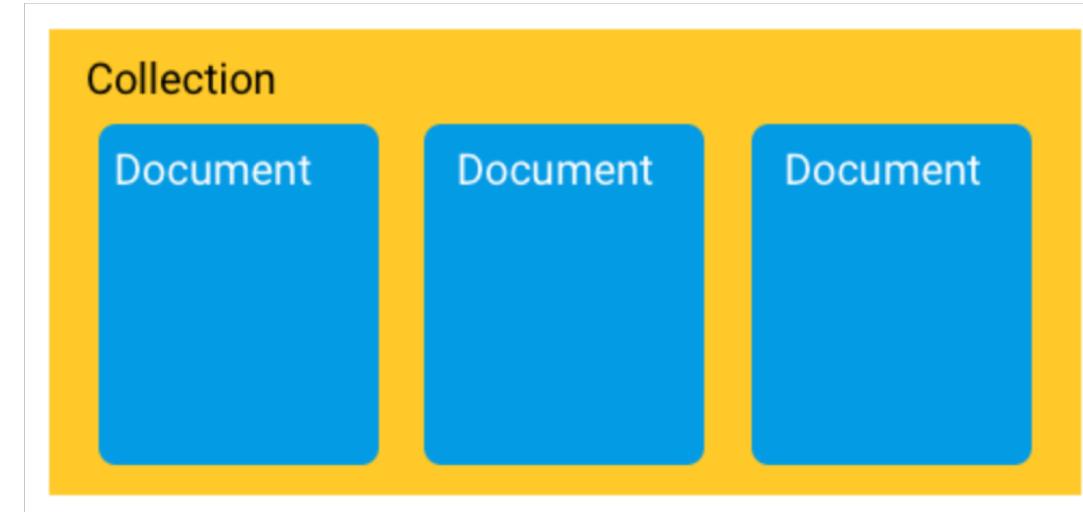
- ❑ Where Realtime Database stores data in the form of a giant JSON tree, Cloud Firestore takes a much more structured approach.
- ❑ Firestore keeps its data inside objects called **documents**. These documents consist of key-value pairs and can contain any kind of data, from strings to binary data to even objects that resemble JSON trees (Firestore calls it maps).

(Sounds a bit like mongodb to me ☺)



Improved Querying and Data Structure

- ❑ The documents, in turn, are grouped into **collections**.

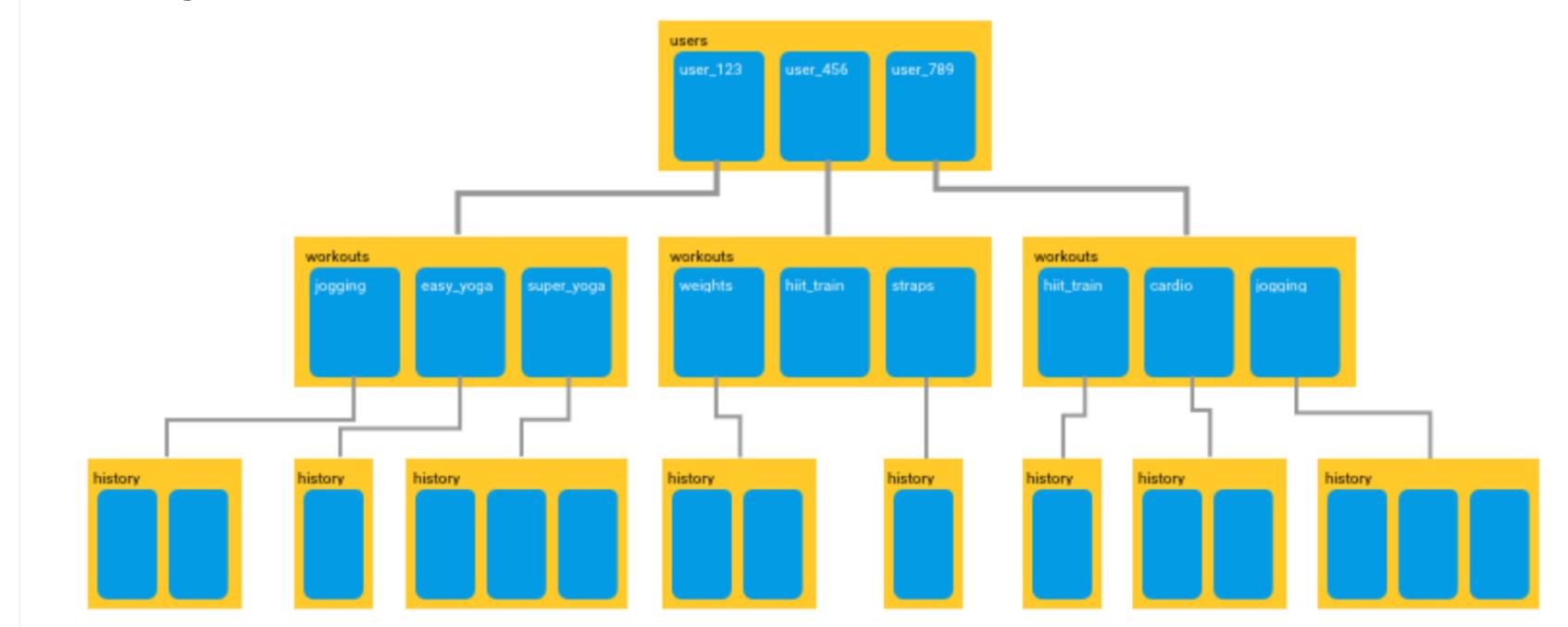


- ❑ Firestore database can consist of multiple collections that can contain documents pointing towards sub-collections.
- ❑ These sub-collections can again contain documents that point to other sub-collections, and so on.



Improved Querying and Data Structure

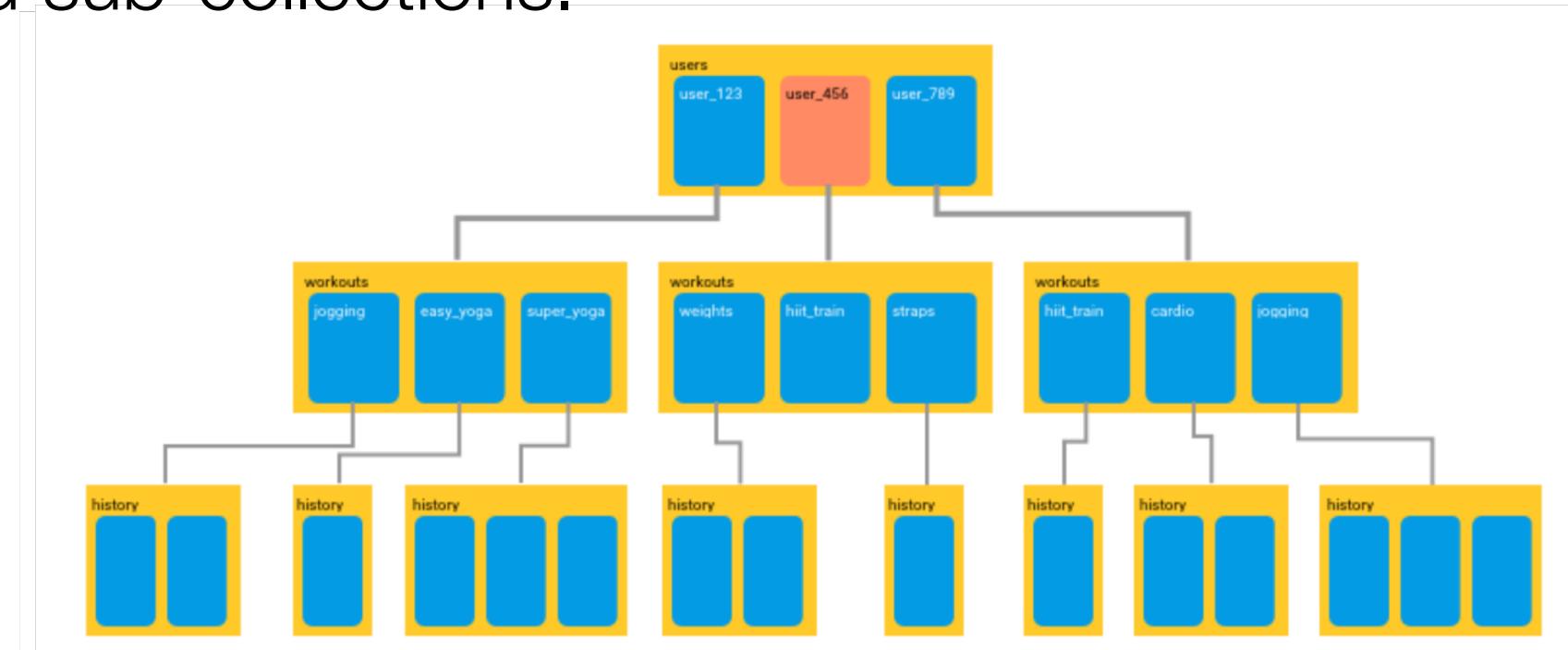
- You can build hierarchies to store related data and easily retrieve any data that you need using queries. All queries can scale with the size of your result set, so your app is ready to scale from day one.





Improved Querying and Data Structure

- ❑ Firestore's queries are *shallow*. By this, I mean to say that in Firestore, you can simply fetch any document that you want without having to fetch all of the data that is contained in any of its linked sub-collections.





Better Scalability

- ❑ Though Firebase's Realtime Database is capable of scaling, things will start to get crazy when your app becomes really popular or if your database becomes really massive.
- ❑ Cloud Firestore is based on Googles Cloud infrastructure. This allows it to scale much more easily and to a greater capacity than the Realtime Database.



Multi-Region Database

- ❑ In Firestore, your data is automatically copied to various regions. So if one data center goes offline due to some unforeseen reason, you can be sure that your app's data is still safe somewhere else.
- ❑ Firestore's multi-region database also provides strong consistency. Any changes to your data will be mirrored across every copy of your database.



Different Pricing Model

- ❑ The Realtime Database charges its users based on the amount of data that you have stored in the database.
- ❑ Cloud Firestore also charges you for the same, but the cost is significantly lower than that of Realtime Database and instead of basing the cost on the amount of data stored, Firestore's pricing is driven by the number of reads/writes that you perform.



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 <small>✓ Free usage from Spark plan included*</small>
Free Products A/B Testing, Analytics, App Indexing, Authentication (except Phone Auth), Cloud Messaging (FCM), Crashlytics, Dynamic Links, Invites, Performance Monitoring, Predictions, and Remote Config.	<small>✓ Included</small>	<small>✓ Included Free</small>	<small>✓ Included Free</small>
Realtime Database Simultaneous connections ? GB stored GB downloaded Multiple databases per project	100 1 GB 10 GB/month <small>✗</small>	100k 2.5 GB 20 GB/month <small>✗</small>	100k/database \$5/GB \$1/GB <small>✓</small>



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 <small>✓ Free usage from Spark plan included*</small>
Free Products A/B Testing, Analytics, App Indexing, Authentication (except Phone Auth), Cloud Messaging (FCM), Crashlytics, Dynamic Links, Invites, Performance Monitoring, Predictions, and Remote Config.	<small>✓ Included</small>	<small>✓ Included Free</small>	<small>✓ Included Free</small>
Cloud Firestore Stored data Bandwidth Document writes Document reads Document deletes	1 GB total 10GB/month 20K/day 50K/day 20K/day	2.5 GB total 20GB/month 100K/day 250K/day 100K/day	\$0.18/GB Google Cloud Pricing \$0.18/100K \$0.06/100K \$0.02/100K



Realtime Database



Cloud Firestore



Some important points though...

- ❑ 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>
- ❑ <https://hackernoon.com/introduction-to-firebase-218a23186cd7>
- ❑ Demo app available at <https://goo.gl/WBP5fR>



Questions?

