

# 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





# Agenda

---

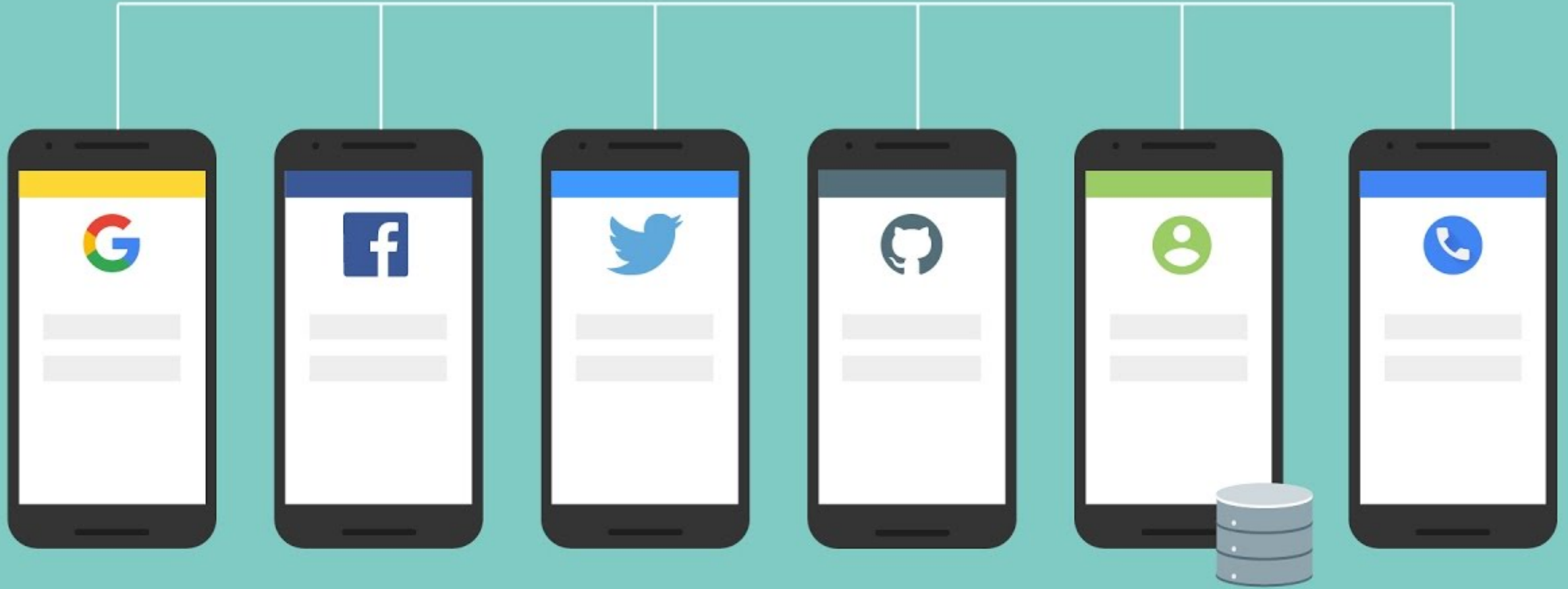
- ❑ Firebase history
- ❑ The all new Firebase
- ❑ Real-time database
- ❑ Authentication
- ❑ Storage
- ❑ Remote config
- ❑ Hosting
- ❑ Crash reporting
- ❑ Test lab
- ❑ Firebase cloud messaging
- ❑ Dynamic links
- ❑ App indexing
- ❑ Analytics
- ❑ CoffeeMate Highlights & Demos along the way...



# Agenda

---

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



All goodness bundled as one...

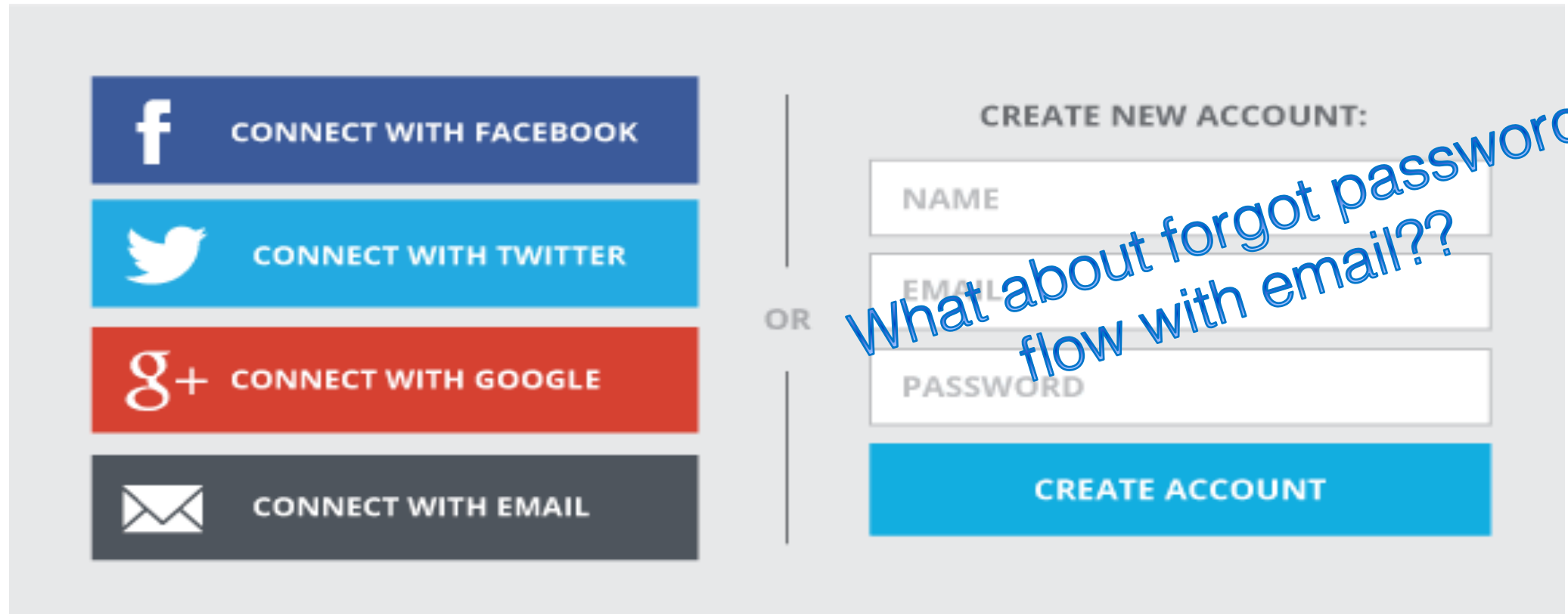


Authentication



# Firebase Authentication \*

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



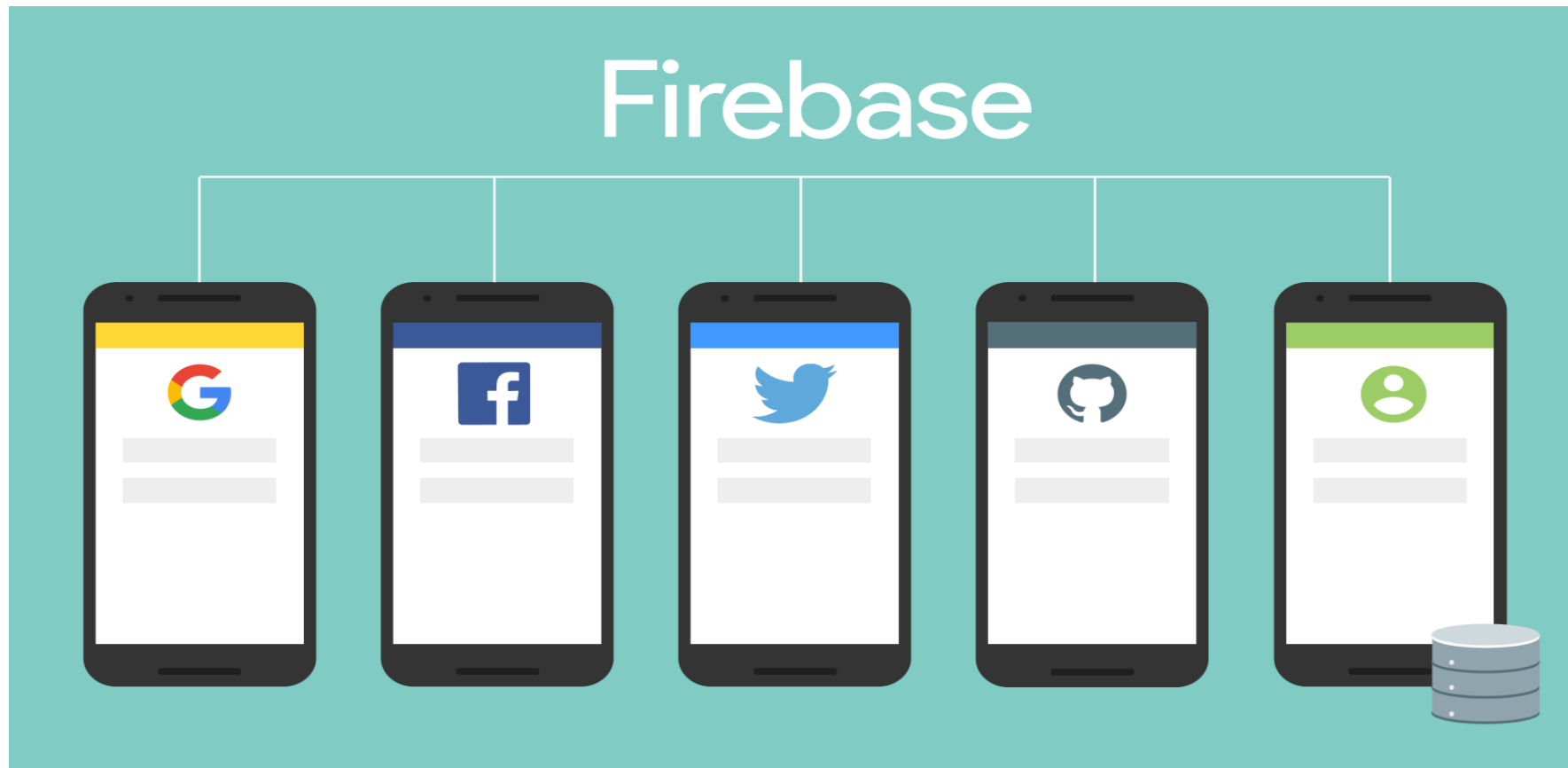
How long would it take you to develop this???



# 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 Authentication console for a project named 'CoffeeMate FBI'. The 'SIGN-IN METHOD' tab is highlighted with a red box. The left sidebar also has the 'Authentication' menu item highlighted with a red box. The main content area shows a search bar, a table with columns for Email, Providers, Created, Signed In, and User UID, and a 'SET UP SIGN-IN METHOD' button.

Search by email address or user UID ADD USER ↻

Email	Providers	Created	Signed In	User UID ↑
 <p>Authenticate and manage users from a variety of providers without server-side code</p> <p><a href="#">Learn more</a></p> <p><b>SET UP SIGN-IN METHOD</b></p>				



# 1. Setup your Sign-In Method \*

The screenshot shows the Firebase Authentication console for a project named 'CoffeeMate FBI'. The 'SIGN-IN METHOD' tab is selected, displaying a table of sign-in providers. The 'Google' provider is highlighted with a red box. All providers are currently disabled.

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 console for a project named 'CoffeeMate FBI'. The 'SIGN-IN METHOD' tab is active, displaying a table of sign-in providers. The 'Email/Password' provider is listed as 'Disabled'. The 'Google' provider is expanded, showing an 'Enable' toggle switch that is currently turned on (blue). Below the toggle, there is a note: '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 two optional settings: 'Whitelist client IDs from external projects (optional)' and 'Web SDK configuration (optional)', both with dropdown arrows.

Provider	Status
Email/Password	Disabled
Google	Enabled



# 1. Setup your Sign-In Method \*

The screenshot shows the Firebase Authentication console for the project 'CoffeeMate FBI'. The 'Authentication' section is active, and the 'Google' provider is being configured. The 'Enable' toggle is turned on. Below the toggle, there is a text block explaining that Google sign-in is automatically configured for iOS and web apps, but for Android apps, a SHA1 fingerprint must be added to the Project Settings. There are also dropdown menus for 'Whitelist client IDs from external projects (optional)' and 'Web SDK configuration (optional)'. At the bottom right of the configuration panel, there are 'CANCEL' and 'SAVE' buttons, with the 'SAVE' button 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.getId(),  
                                app.mFirebaseUser.getDisplayName(),  
                                app.mFirebaseUser.getEmail());  
}
```





## 2. Introduce Authentication Flow \*

```
//Check to see if the Firestore 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.getId(),
                                app.mFirebaseUser.getDisplayName(),
                                app.mFirebaseUser.getEmail(), null);
        app.mFBDBManager.mFirebaseDatabase.child("users")
            .child(app.mFirebaseUser.getId())
            .setValue(newUser);
    }
    app.mFBDBManager.mFBUserId = app.mFirebaseUser.getId();

    startHomeScreen();
}
```



# Firebase Console – Authenticated Users \*

The screenshot shows the Firebase Authentication console for a project named 'CoffeeMateFBI'. The left sidebar contains various service categories: Overview, Analytics, DEVELOP (with 'Authentication' highlighted), Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, GROW (with Notifications, Remote Config, and Dynamic Links), and EARN (with AdMob). The main content area is titled 'Authentication' and has tabs for 'USERS', 'SIGN-IN METHOD', and 'TEMPLATES'. The 'USERS' tab is active, displaying a search bar and a table of users. The table has columns for 'Email', 'Provider', 'Created', 'Signed in', and 'User ID'. A single user is listed with the email 'daveydrohan@gmail.com', provider 'Google', and creation/sign-in dates of '18 Jul 2017'. The user ID is '1SQVbMgN5bcLg9JXgG0ts3DN0...'. A red box highlights the 'Authentication' menu item in the sidebar and the user list table in the main content area.

Email	Provider	Created	Signed in	User ID
daveydrohan@gmail.com	Google	18 Jul 2017	18 Jul 2017	1SQVbMgN5bcLg9JXgG0ts3DN0...



# Firebase Console – Authenticated Users \*

The screenshot shows the Firebase Console interface for a project named 'CoffeeMateFBI'. The 'Authentication' section is active, displaying a table of users. The 'Authentication' menu item in the left sidebar is highlighted with a red box. The table of users is also highlighted with a red box. The table has the following data:

email	provider	Created	Signed in	User ID
daveydrohan@gmail.com	G	18 Jul 2017	18 Jul 2017	1SQVbMgN5bcLg9JXgG0ts3DN0...
noahdrohan@gmail.com	G	18 Jul 2017	18 Jul 2017	C3PpngqT180xuHvNtG4wZsVMhZ...
joshuajdrohan@gmail.com	G	18 Jul 2017	18 Jul 2017	wotuVu5iZjhGHRcCHRkJgsWh9y1



# Firebase Console – User Data in db \*

The screenshot shows the Firebase Console interface for a project named 'CoffeeMateFBI'. The left sidebar contains a navigation menu with 'Database' highlighted in a red box. The main content area displays the 'Realtime Database' data view for the path 'coffeematefbi/users'. A red box highlights the following JSON data:

```
coffeematefbi
├── users
│   └── 1SQVbMgN5bcLg9JXgG0ts3DN0AV2
│       ├── userEmail: "daveydrohan@gmail.com"
│       ├── userId: "1SQVbMgN5bcLg9JXgG0ts3DN0AV2"
│       └── userName: "David Drohan"
```



# Firebase Console – User Data in db \*

The screenshot shows the Firebase Console interface for a project named 'CoffeeMateFBI'. The 'Database' menu item is highlighted with a red box. The main content area displays the Realtime Database data for the path 'coffeeMatefbi/users'. The data is structured as follows:

```
coffeeMatefbi
├── users
│   ├── 1SQVbMgN5bcLg9JXgG0ts3DN0AV2
│   │   ├── userEmail: "daveydrohan@gmail.com"
│   │   ├── userId: "1SQVbMgN5bcLg9JXgG0ts3DN0AV2"
│   │   └── userName: "David Drohan"
│   ├── C3PpngqTI80xuHvNtG4wZsVMhZv2
│   │   ├── userEmail: "noah1drohan@gmail.com"
│   │   ├── userId: "C3PpngqTI80xuHvNtG4wZsVMhZv2"
│   │   └── userName: "Noah Drohan"
│   └── wotuVu5IZjhh0HRcCHRkJgsWh9y1
│       ├── userEmail: "joshua1drohan@gmail.com"
│       ├── userId: "wotuVu5IZjhh0HRcCHRkJgsWh9y1"
│       └── userName: "Joshua Everett Drohan"
```



---

Thanks!

A simple line drawing of a smiling person with their arms raised in a celebratory gesture. The person has a round face with a wide smile, two dots for eyes, and a few strands of hair. Their arms are raised, with hands open. A small '©' symbol is visible at the bottom right of the drawing.



# 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. (FirestoreUI)
- ❑ 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?