Mobile Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Android & Firebase
Part 4

Firebase Integration

Google Services Overview

- Overview of Google Play Services and Setup
- Detailed look at
  - Google Sign-in and Authentication (Part 1)
  - Location & Geocoding (Part 2)
  - Google Maps (Part 3)

  - Firebase Integration (Part 4) (NEW)

Google Services Overview

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

Agenda

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

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.

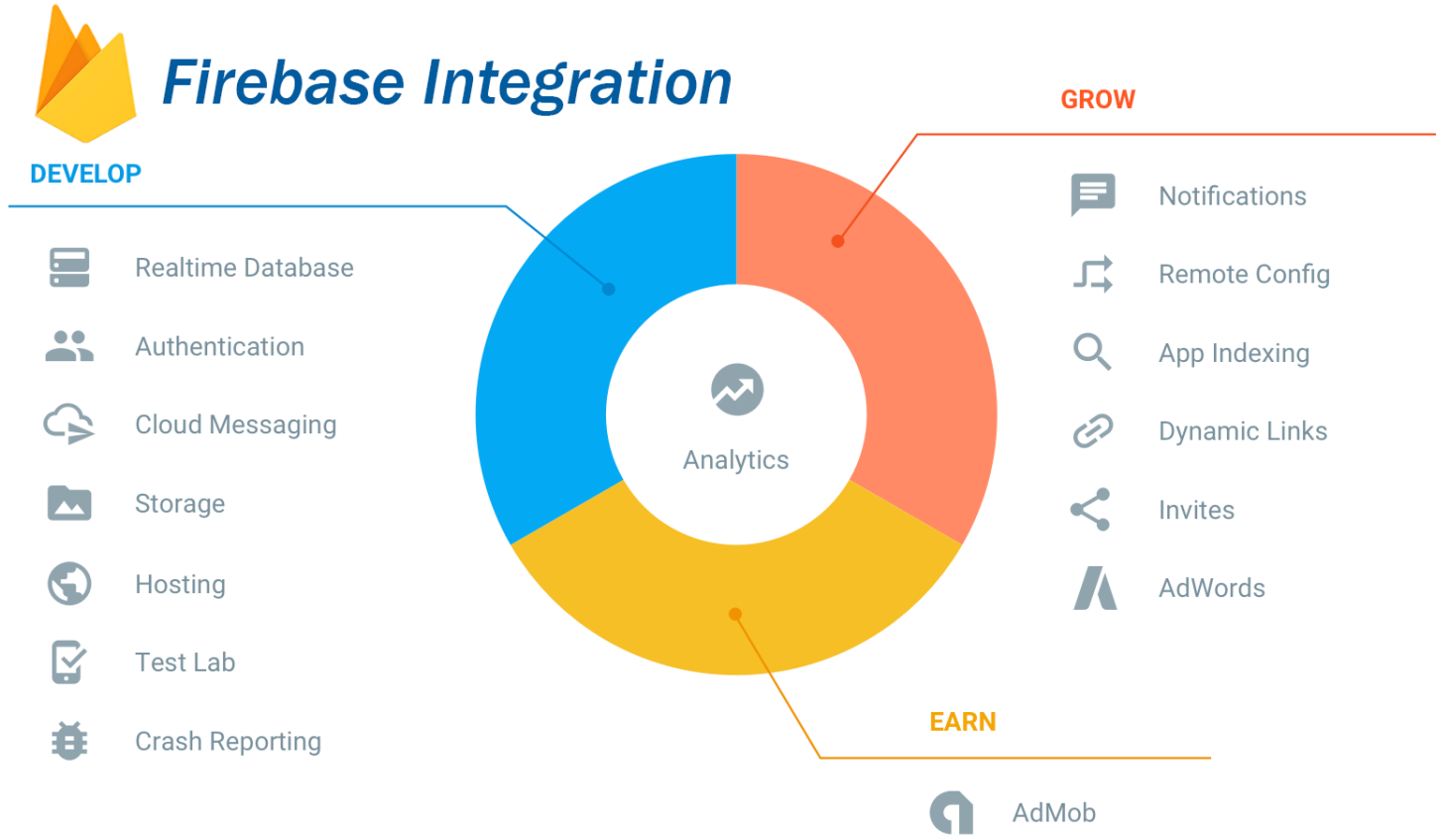
Introducing Firebase

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

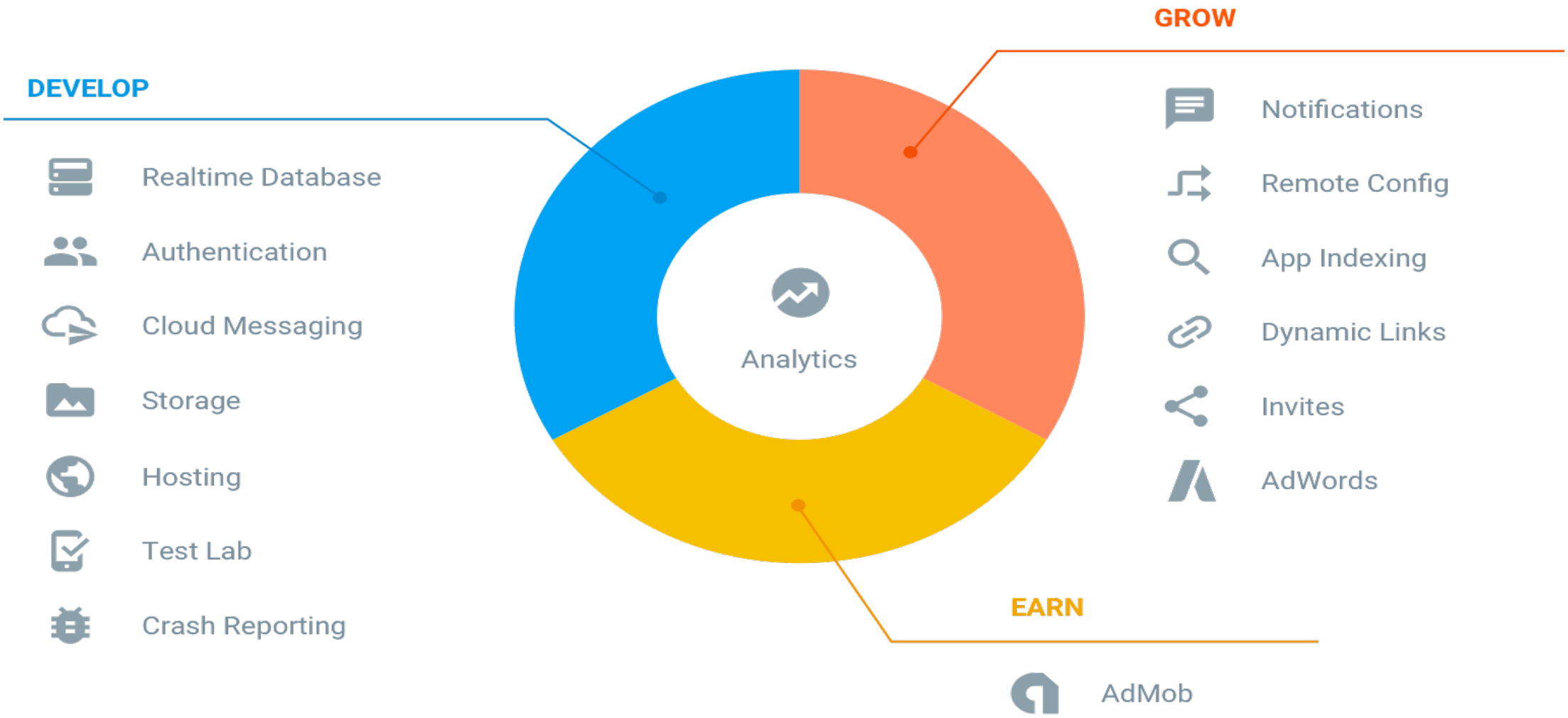
backend

In Reality

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

And What About… *

Enter The New Firebase



Database

Goodbye RDBMS…

Real Time Database

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

The Core Magic Of Firebase *

Saving Data In Firebase

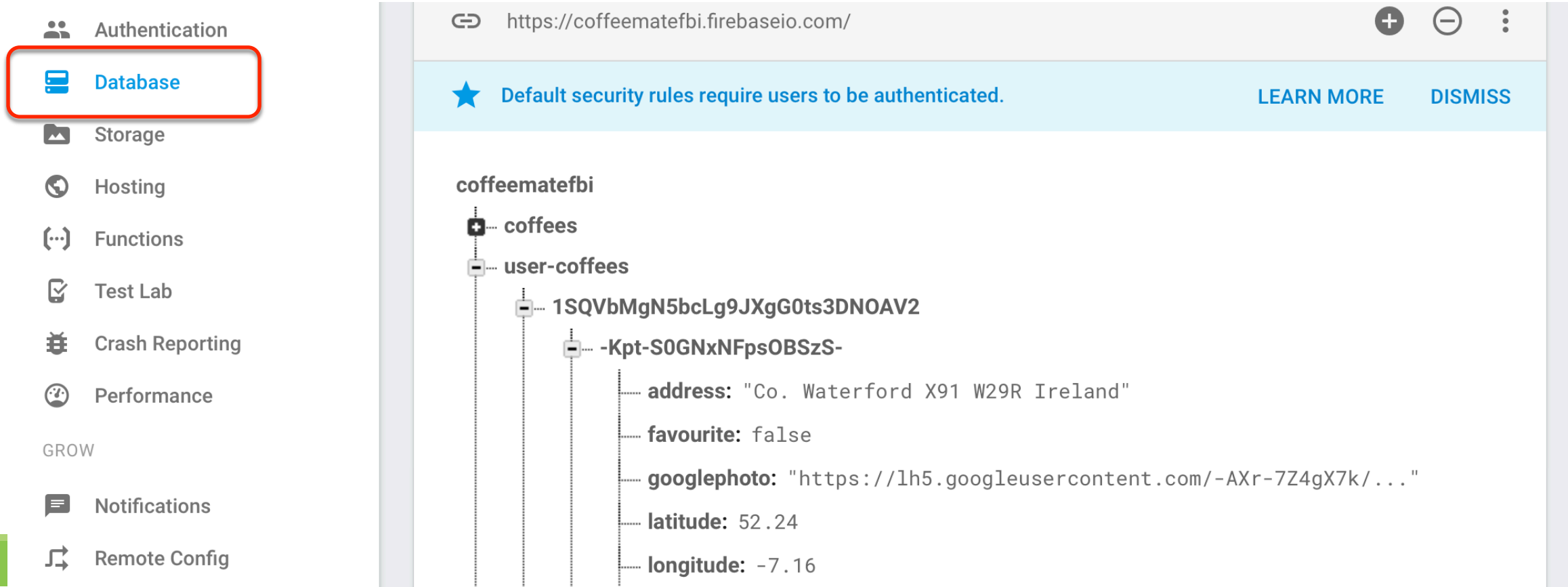
- Get database reference (your base node)
  - DatabaseReference mDBRef = FirebaseDatabase.*getInstance*().getReference();
- Point it to the right JSON node
  - mDBRef = mDBRef.child("mydb").child("table1");
- Set the value at the pointed node
  - mDBRef.child('row1').setValue(myPOJO);
- Or push the value to create a unique key and set the value
  - mDBRef.push().setValue(myPOJO);

Create a Key

Firebase takes care of the Key

View Your Saved Data

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



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);
```

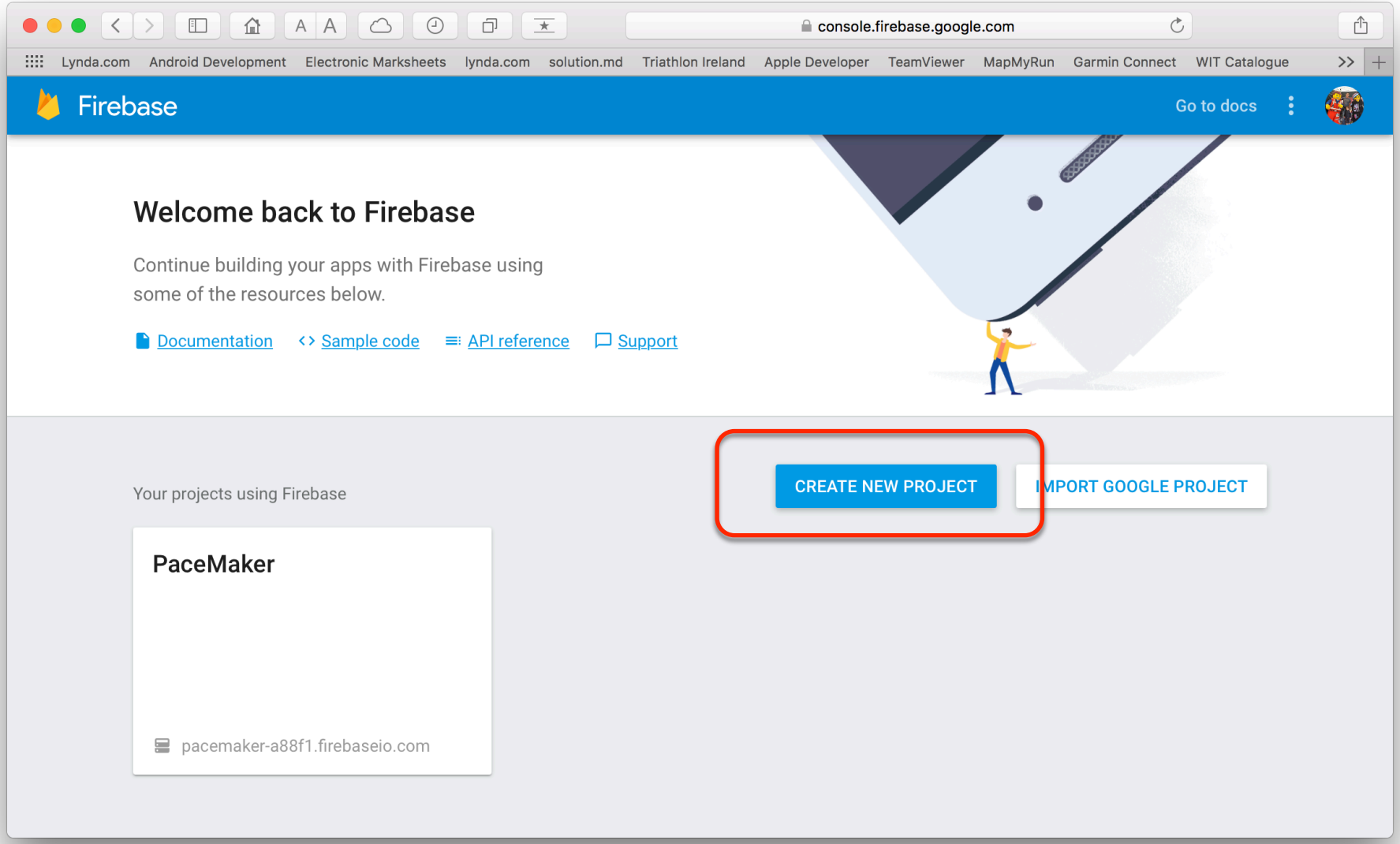
Firebase Realtime Database

Quick DEMO…

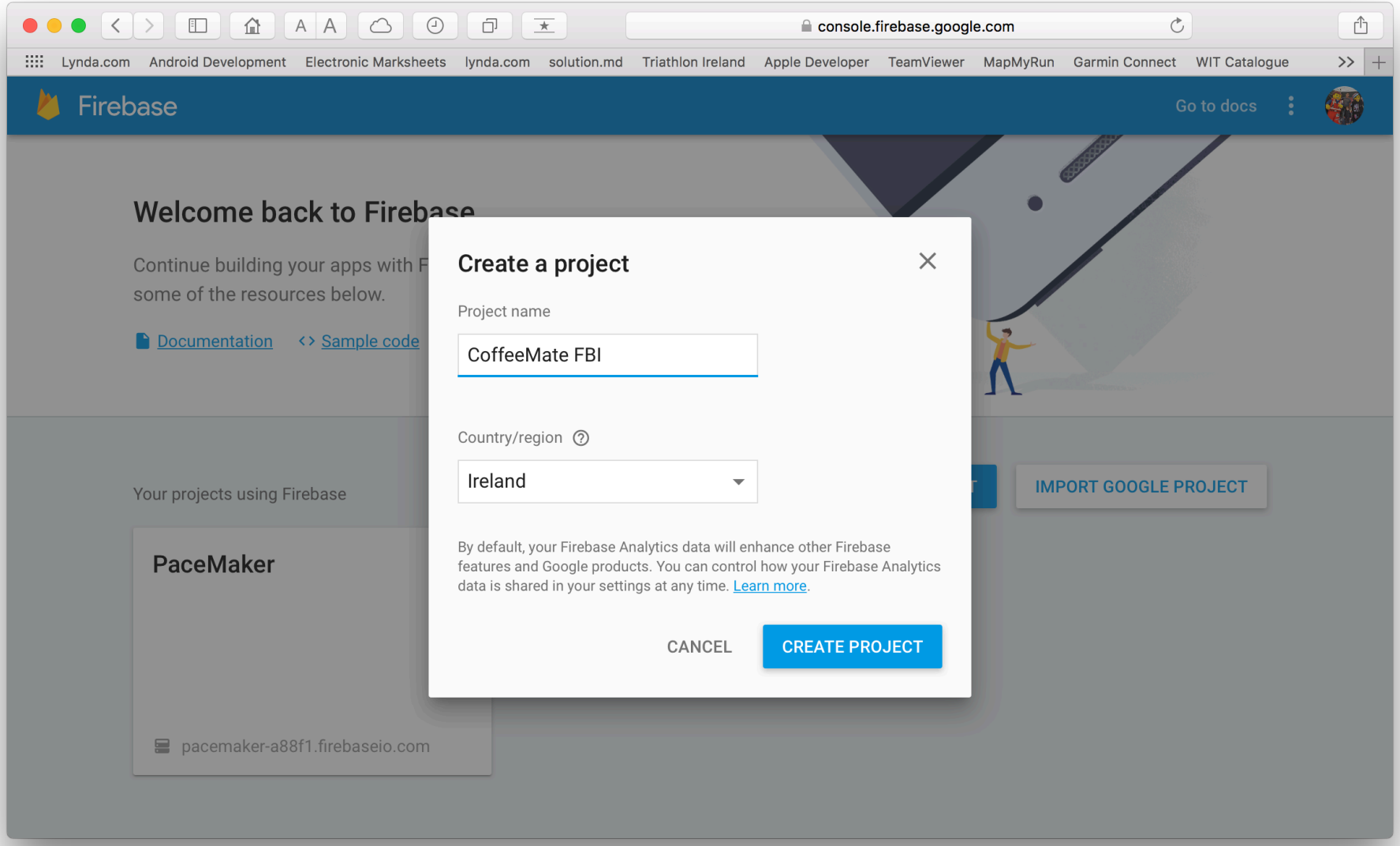
CoffeeMateFBI 1.0

Setup
&
Code
Highlights

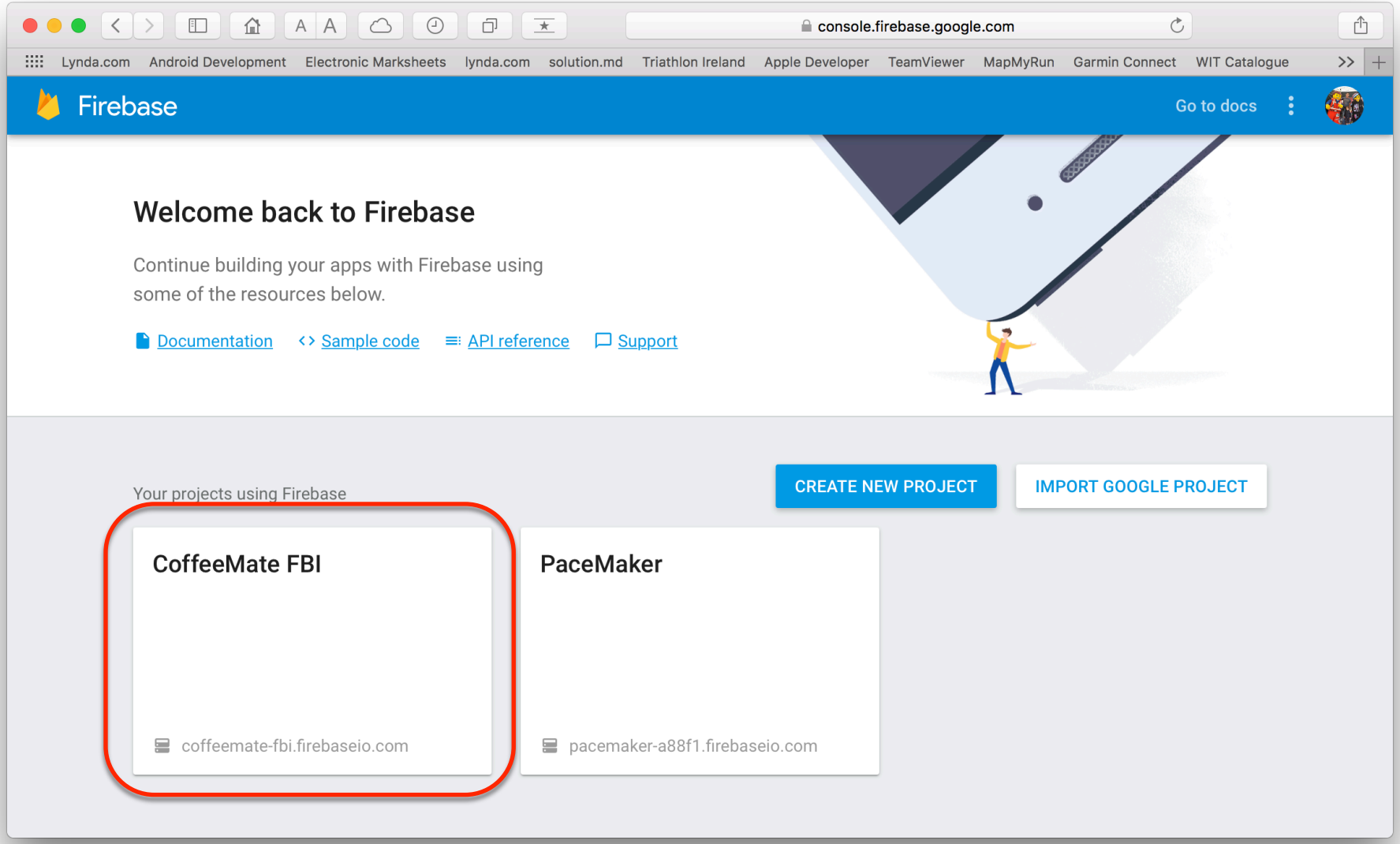
1. Create a Firebase Project *



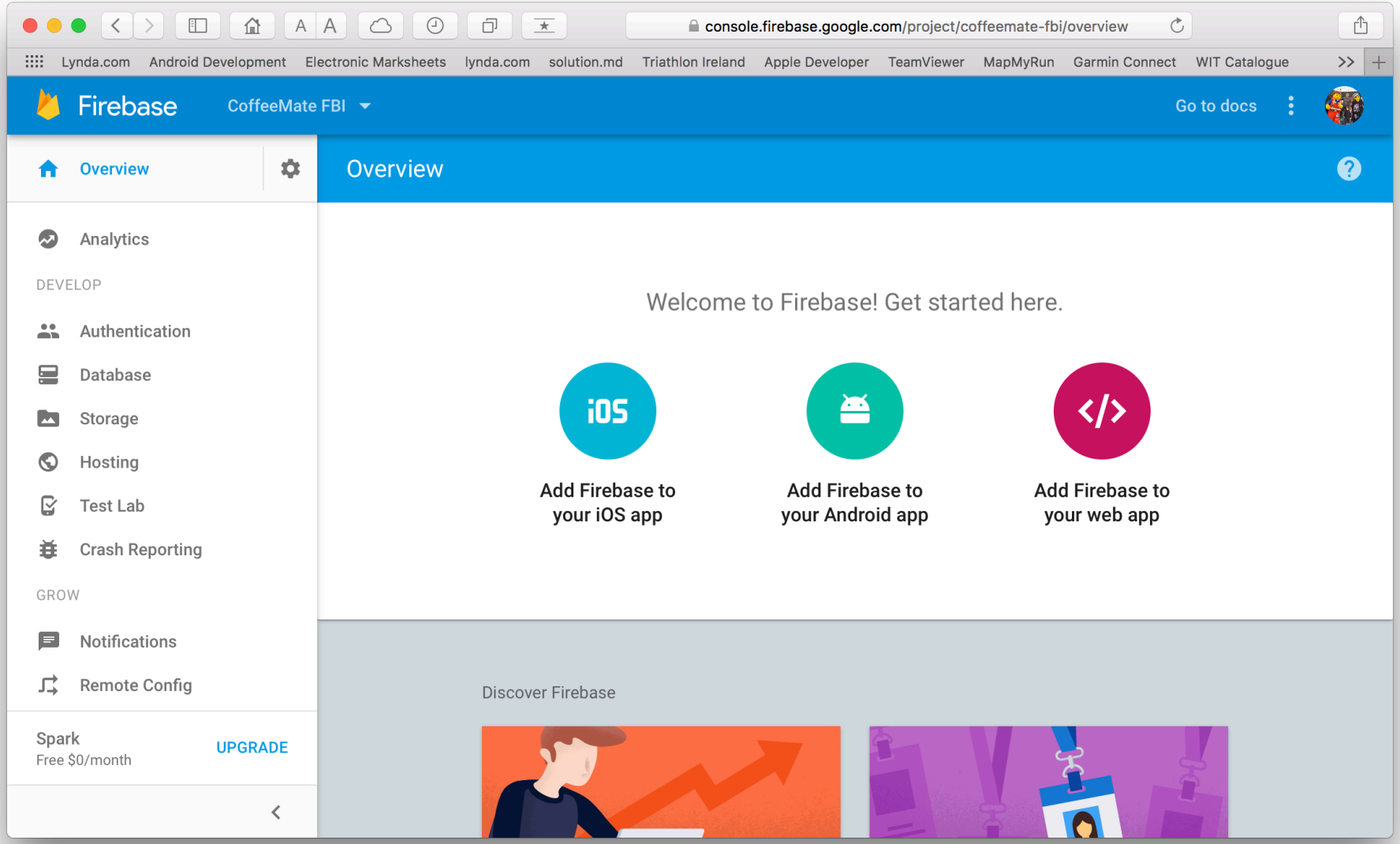
1. Create a Firebase Project



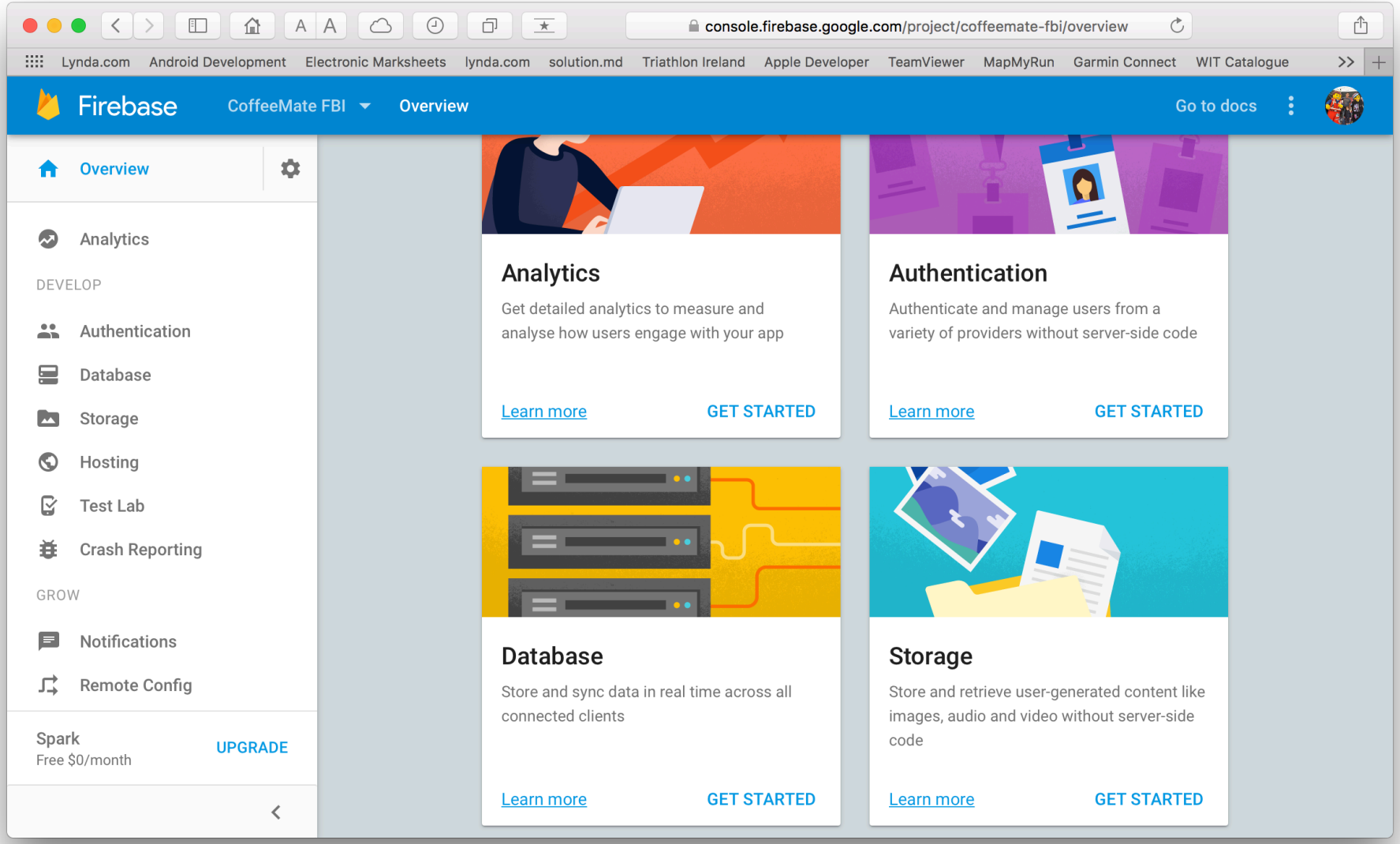
1. Create a Firebase Project *



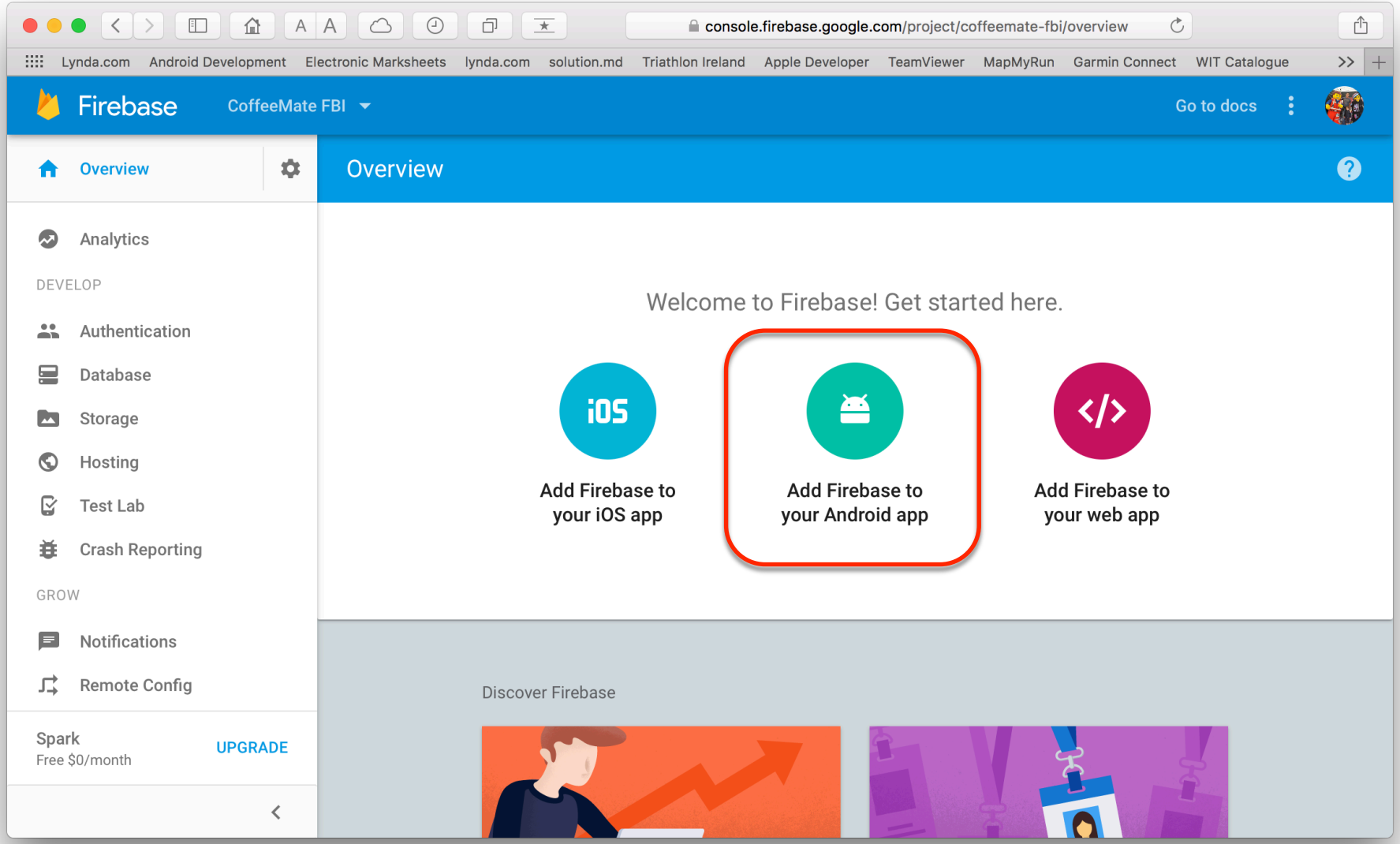
1. Create a Firebase Project - Overview



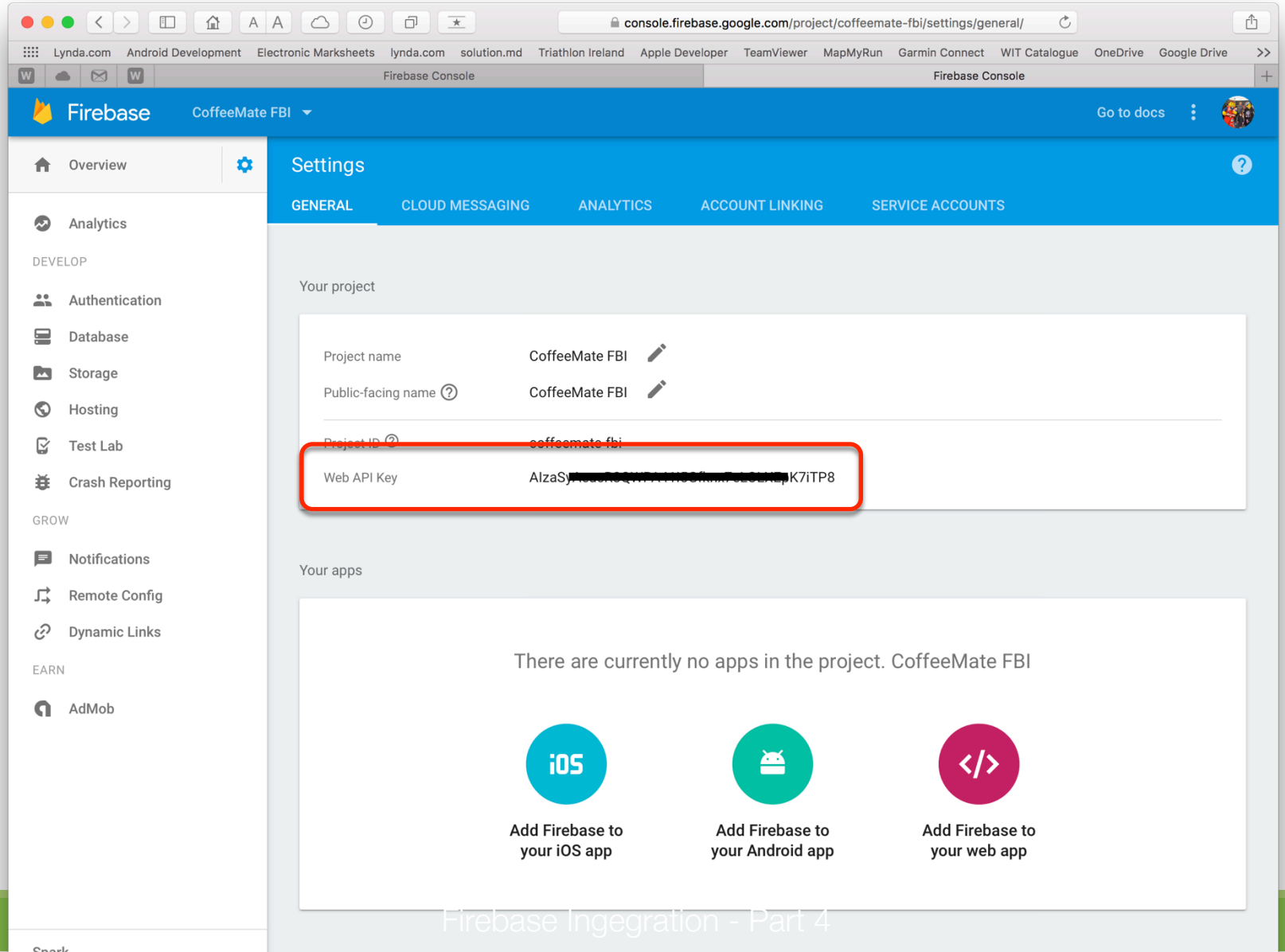
1. Create a Firebase Project - Overview



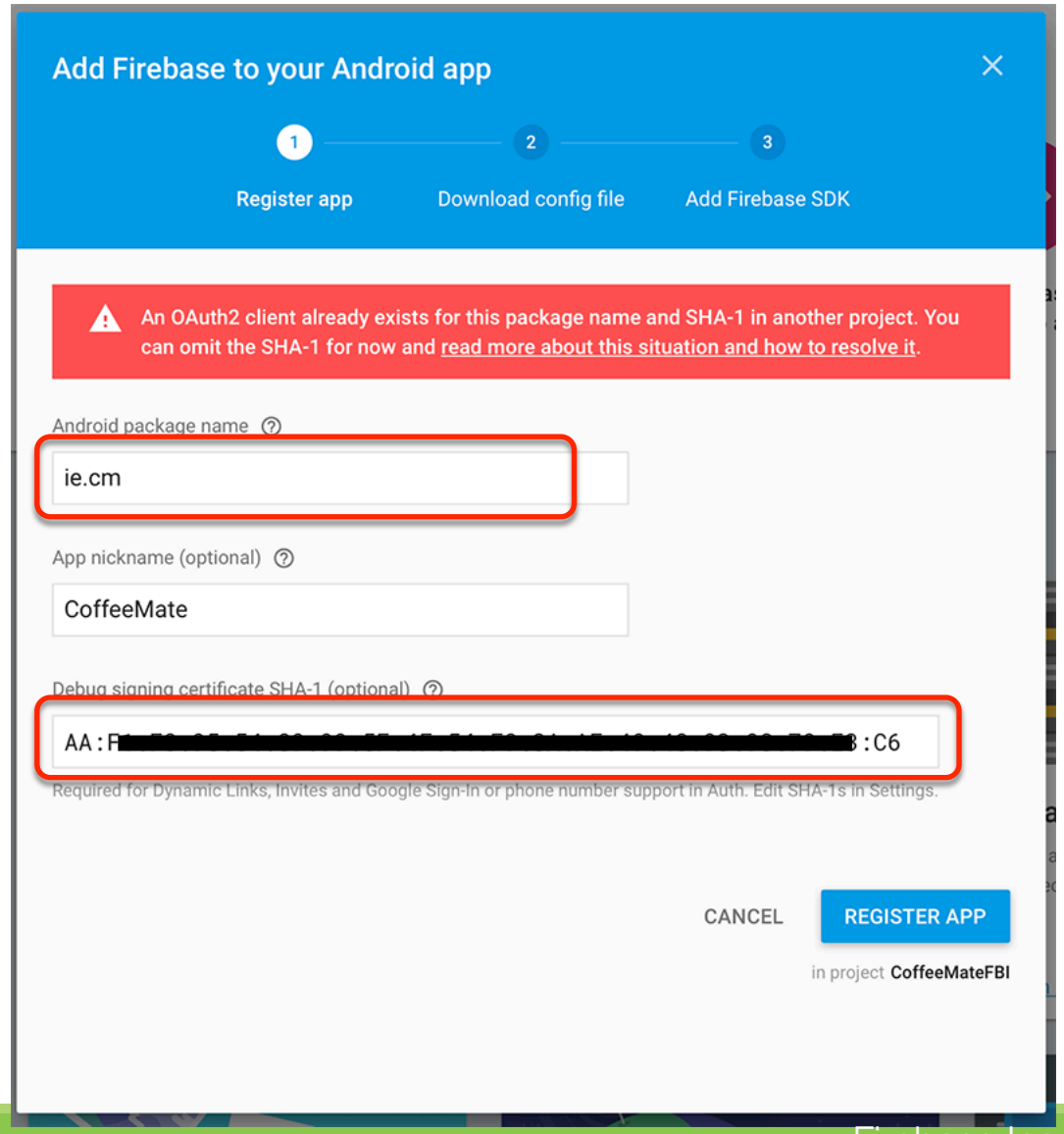
2. Add Firebase to your Android App *

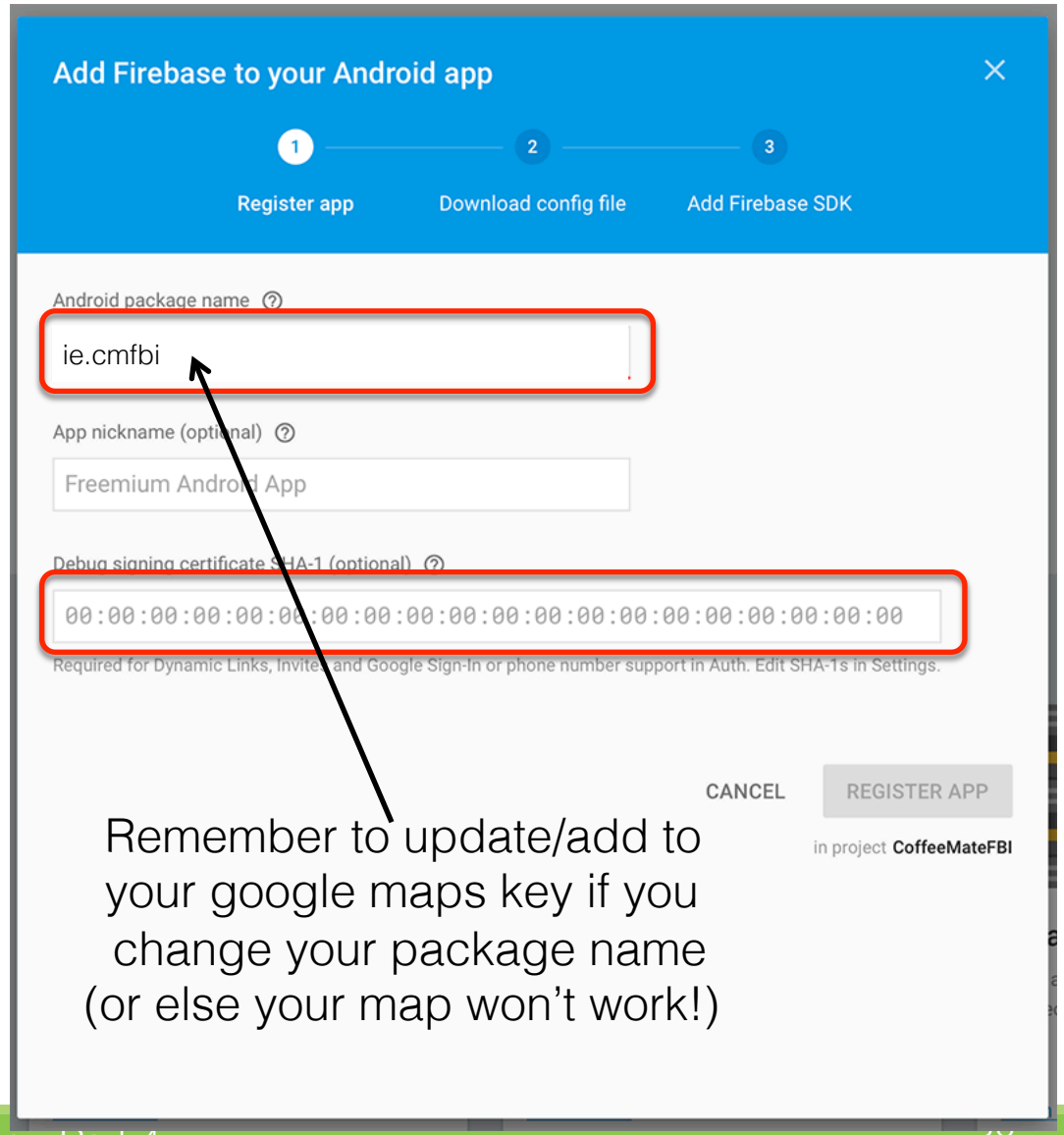


2. Add Firebase to your Android App – Web Key *



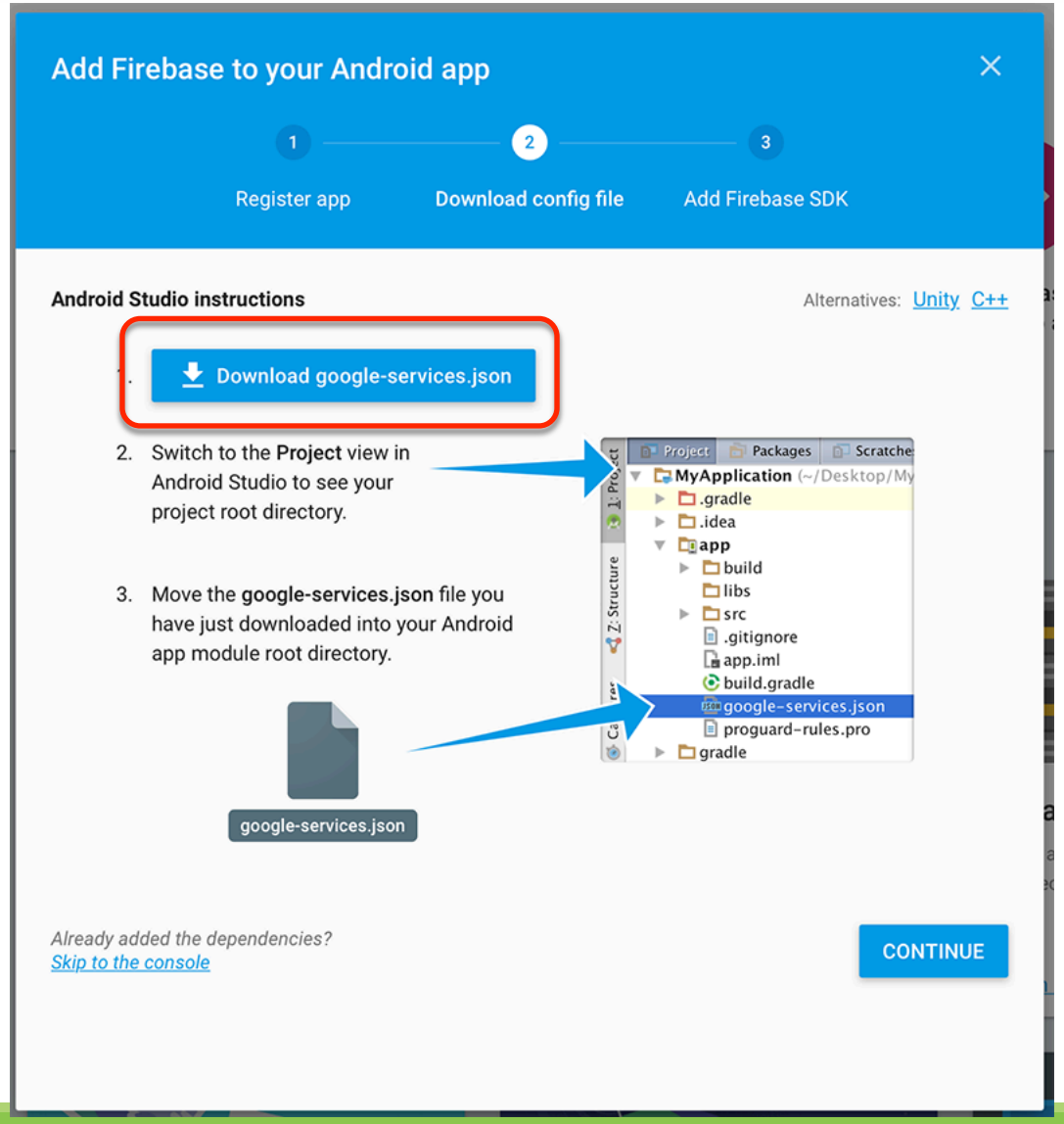
2. Add Firebase to your Android App *



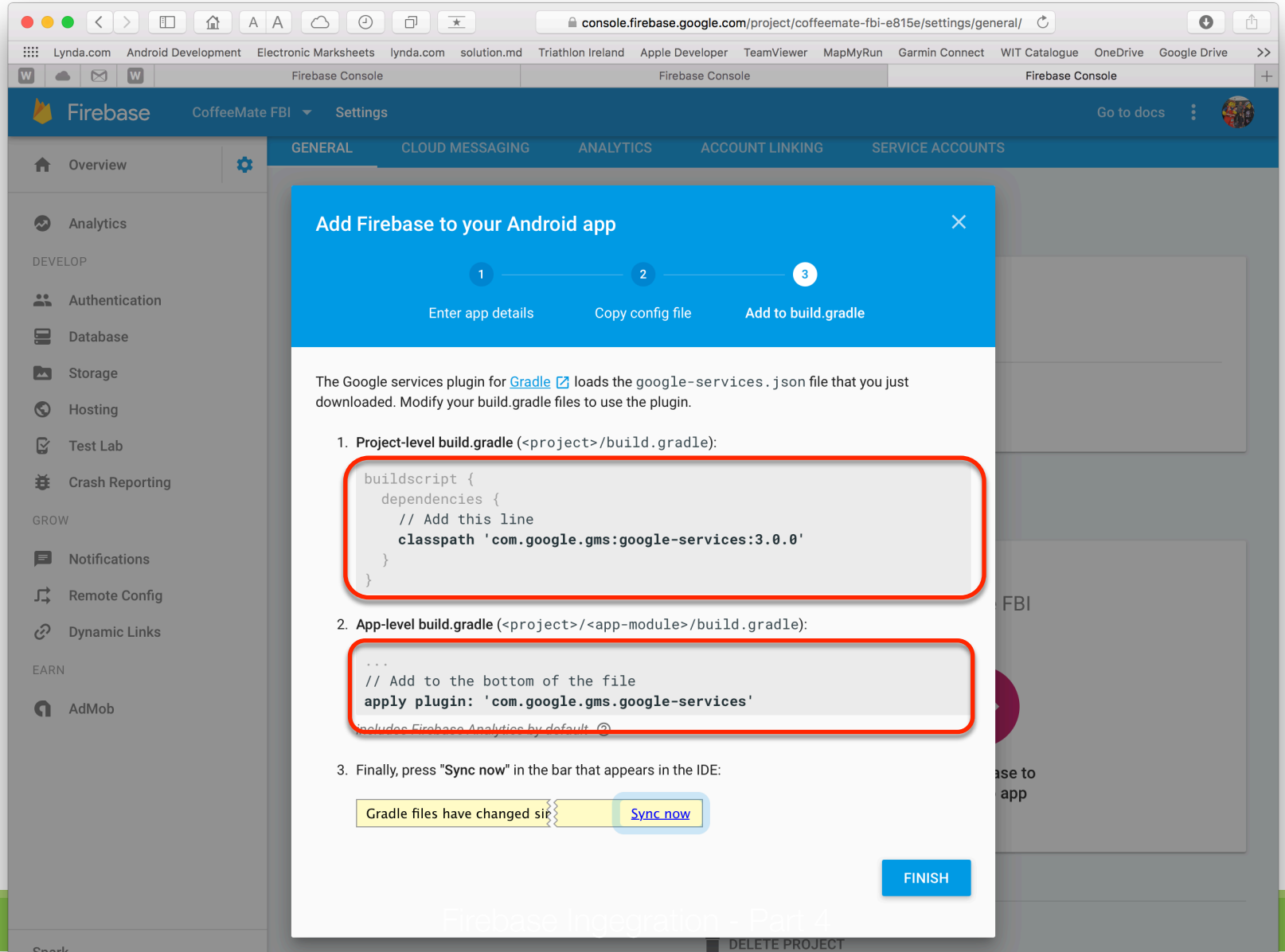


Remember to update/add to your google maps key if you change your package name (or else your map won't work!)

2. Add Firebase to your Android App *



2. Add Firebase to your Android App *



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.firebase:firebase-core:11.0.2'
    compile 'com.google.firebase:firebase-auth:11.0.2'
    compile 'com.google.firebase: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";
    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);
    }
```

- Our Firebase db reference.
- Binding to our db instance.

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;
}
```

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

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

    return query;
}
```

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

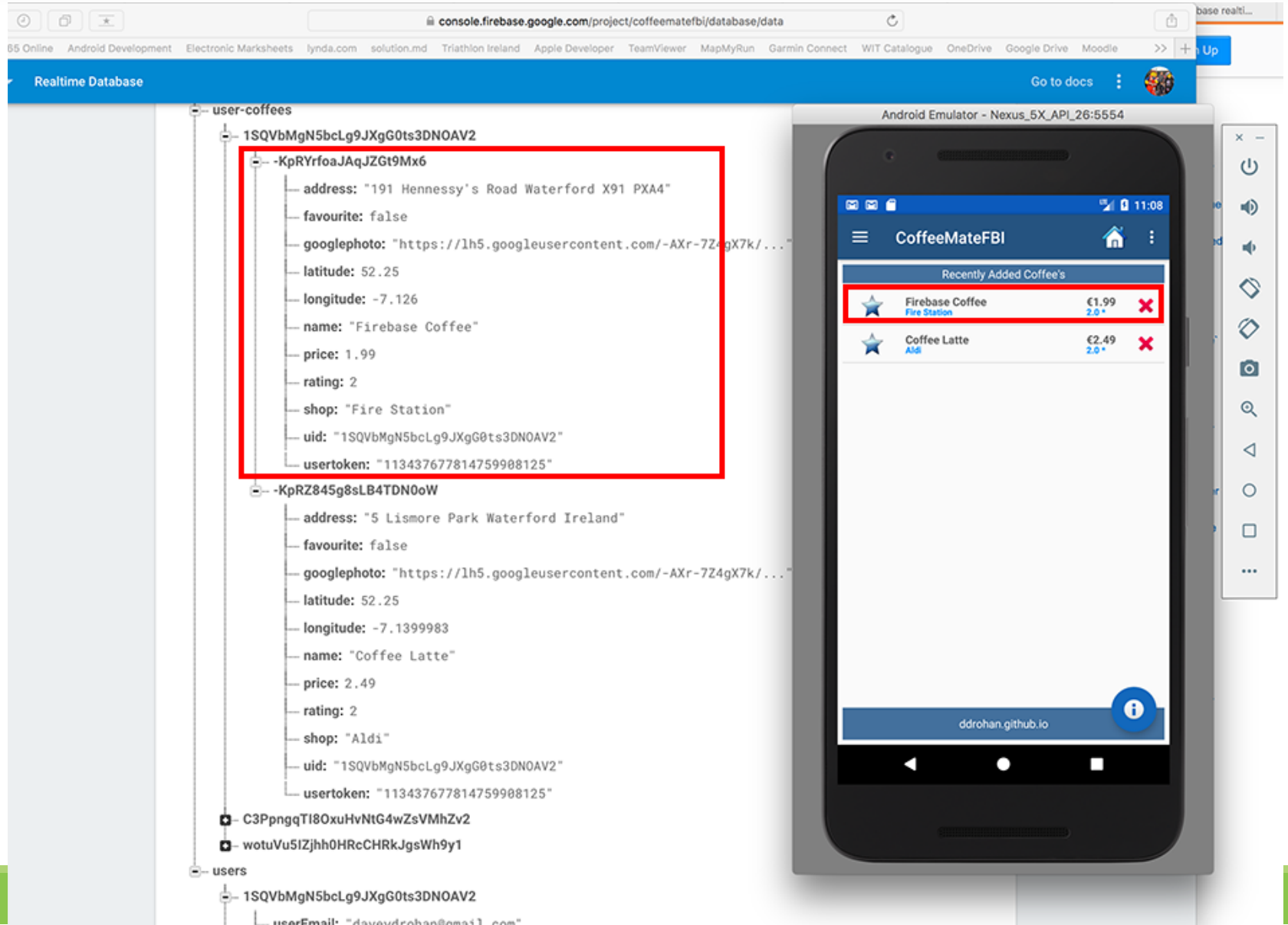
```
public void getACoffee(final String coffeeKey)
{
    mFirebaseDatabase.child("user-coffees").child(mFBUserId).child(coffeeKey)
            .addValueEventListener(
                new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    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) *

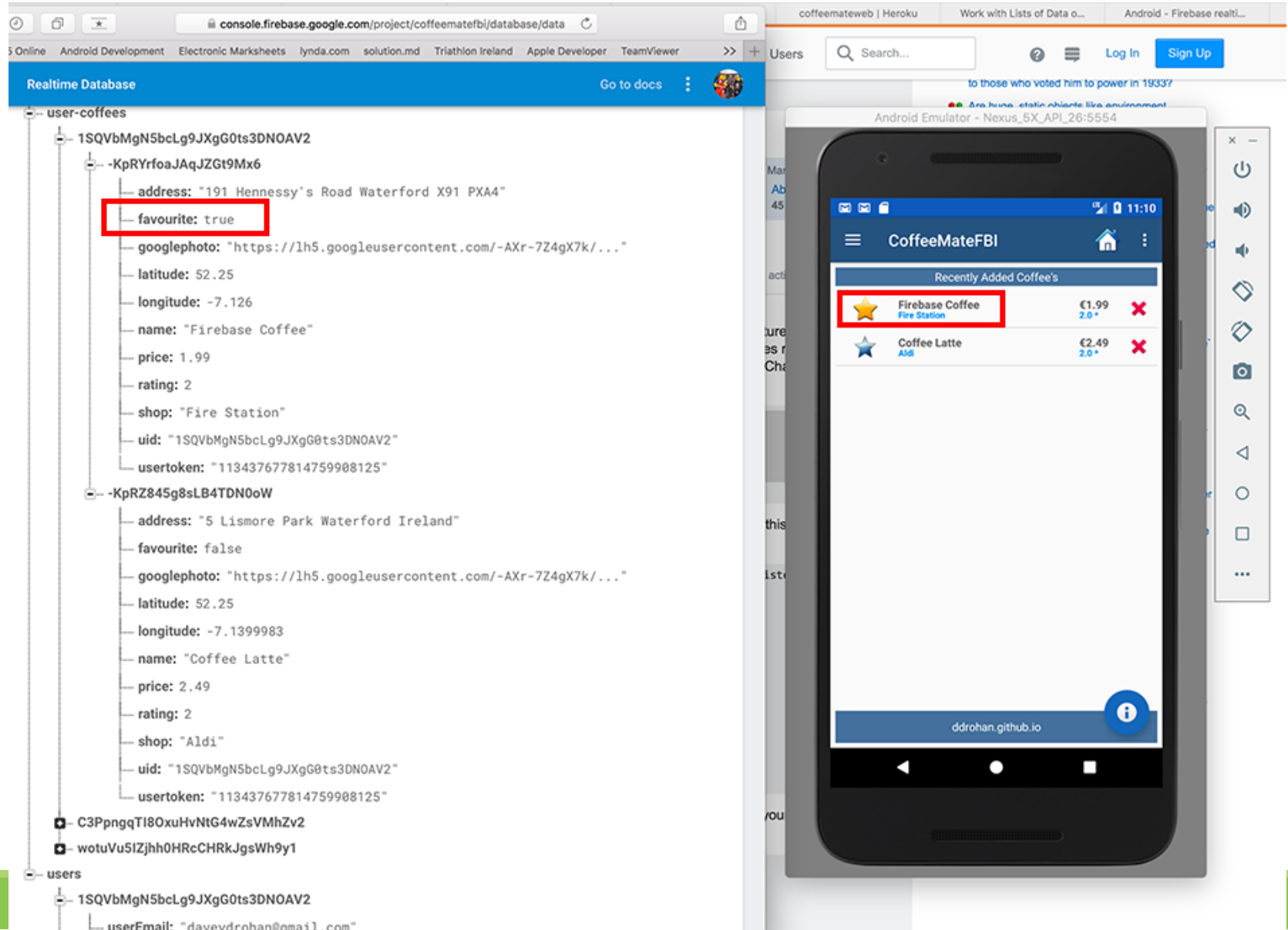
```
@Override
public void onResume() {
    super.onResume();

    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)



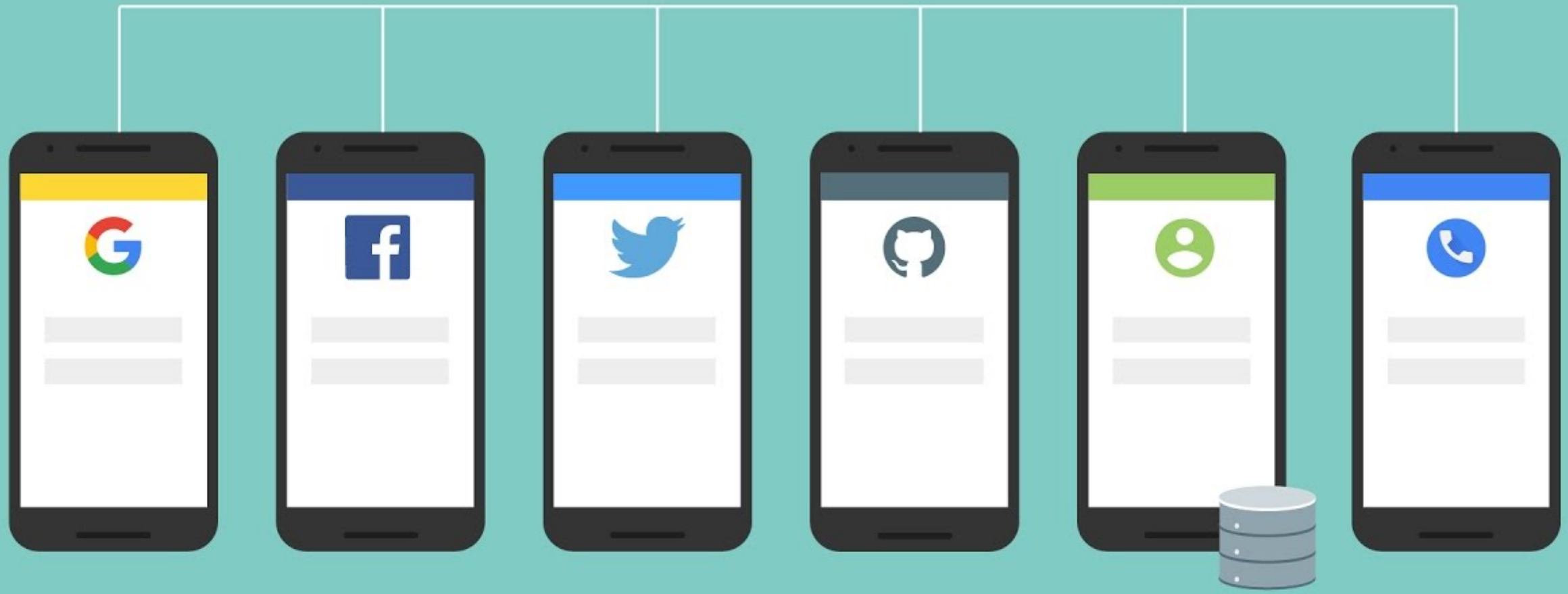
Firebase Console (actual data)



Pricing

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

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

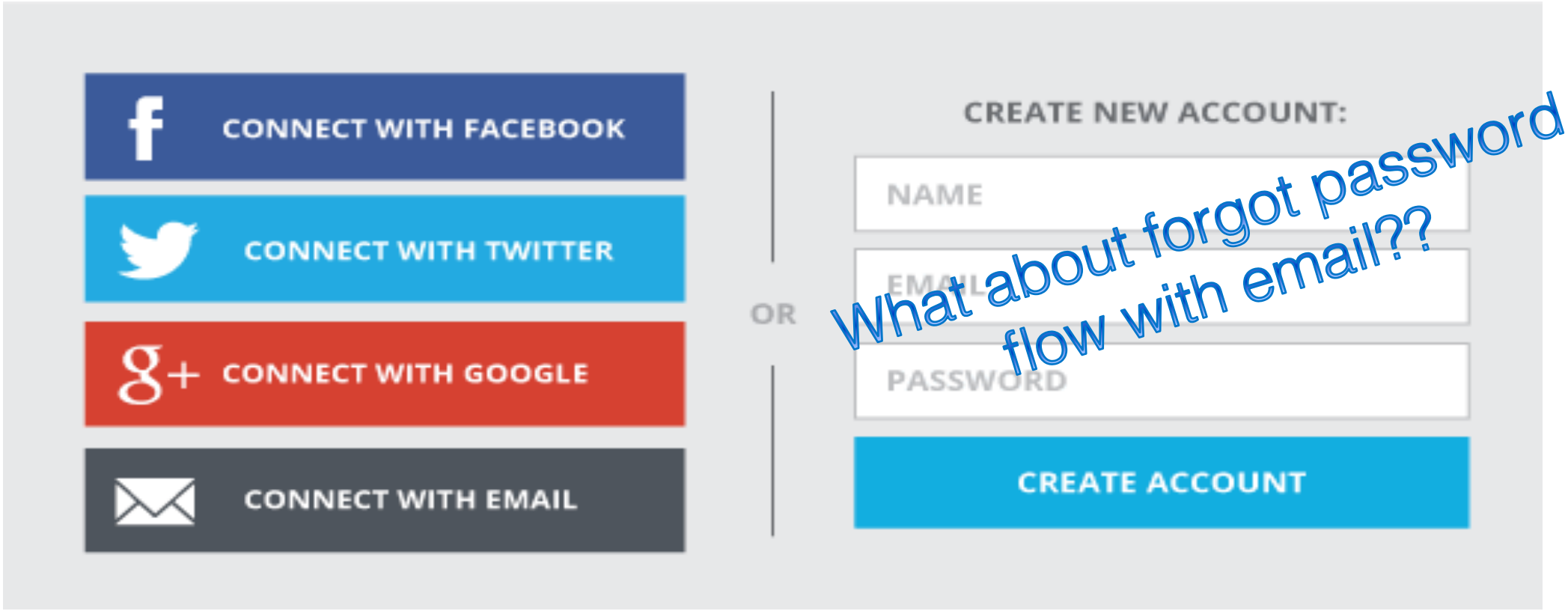


All goodness bundled as one…

Authentication

Firebase Authentication *

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



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

Firebase Authentication

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

Firebase Authentication

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

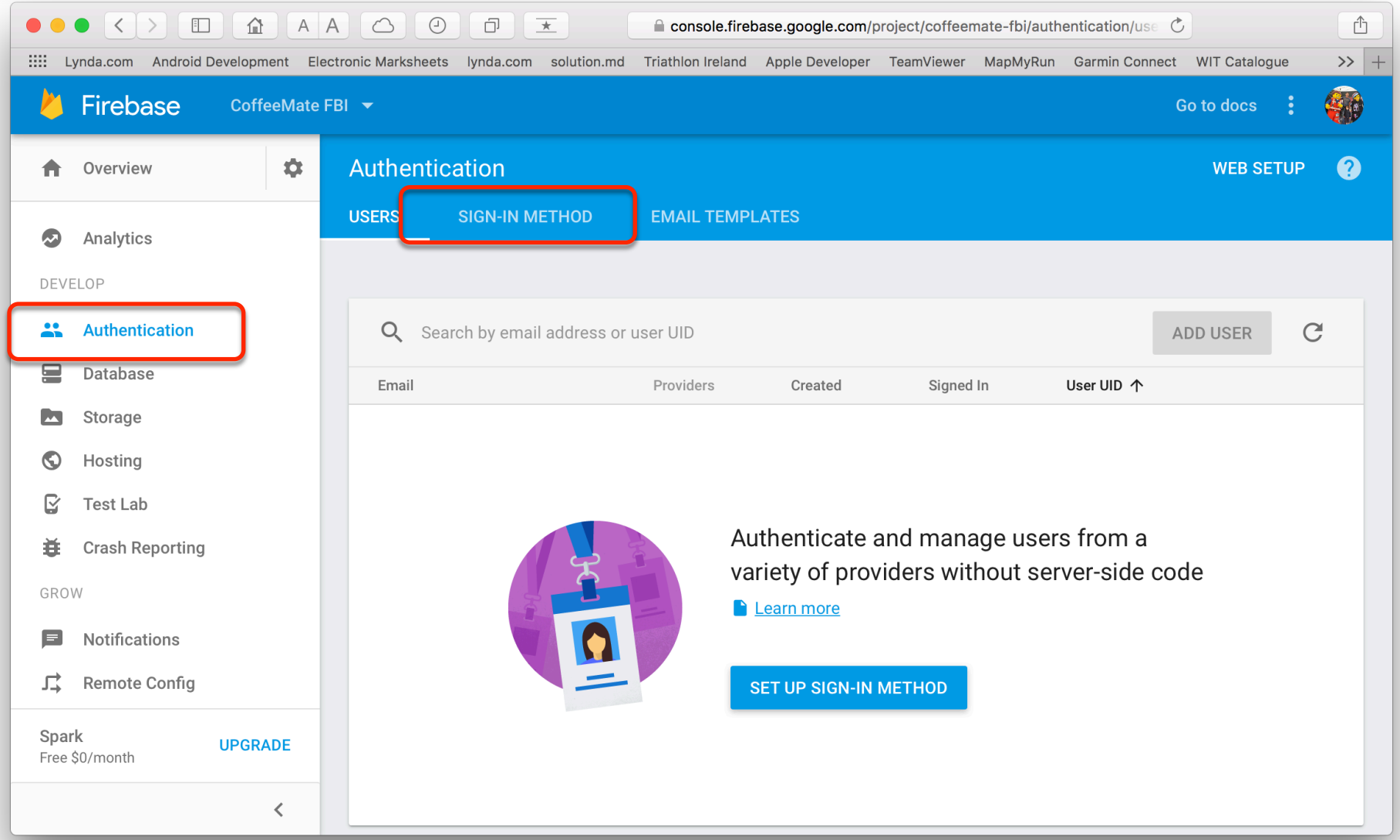
Firebase Authentication

Quick DEMO…

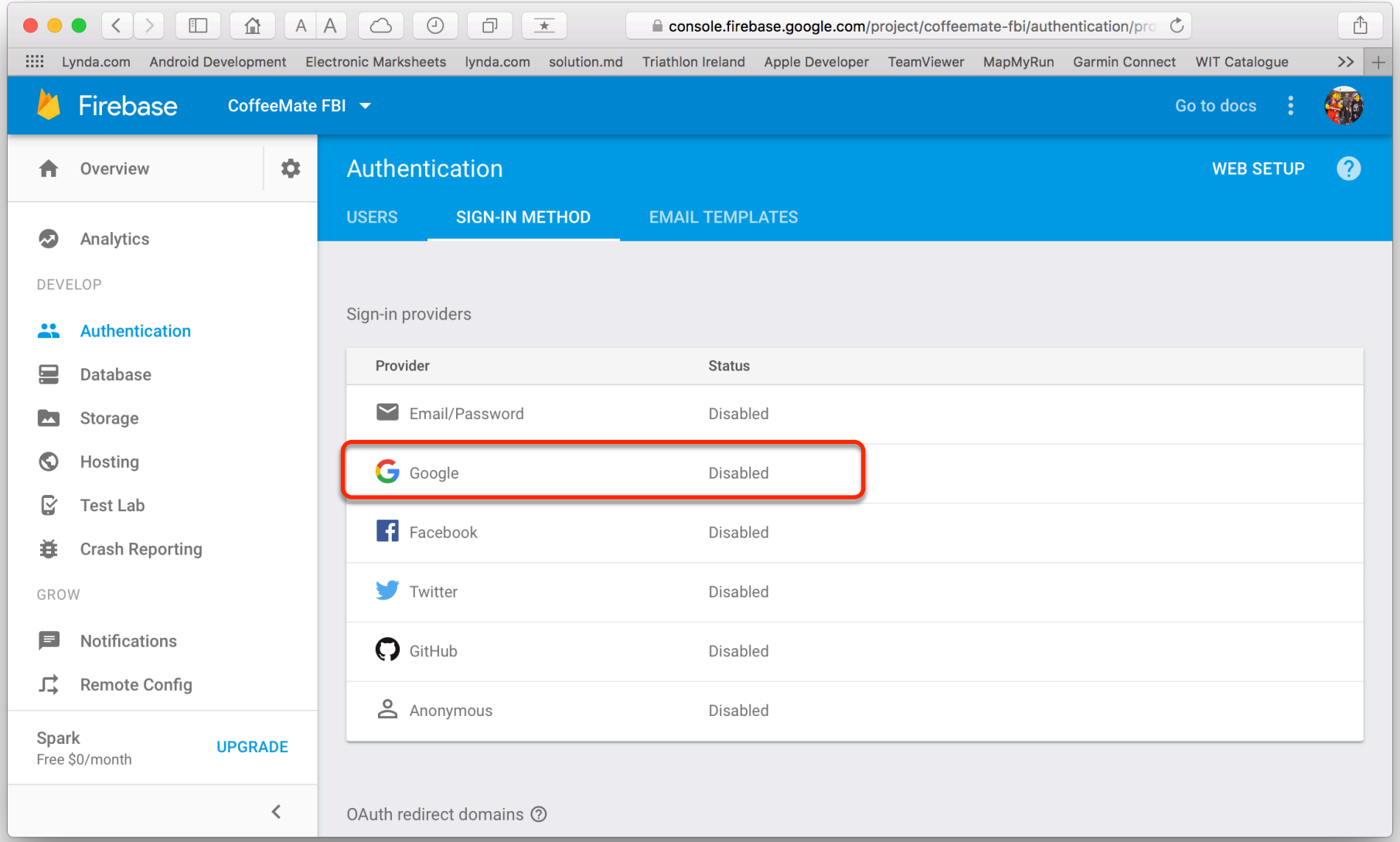
CoffeeMateFBI 1.0

Setup
&
Code
Highlights

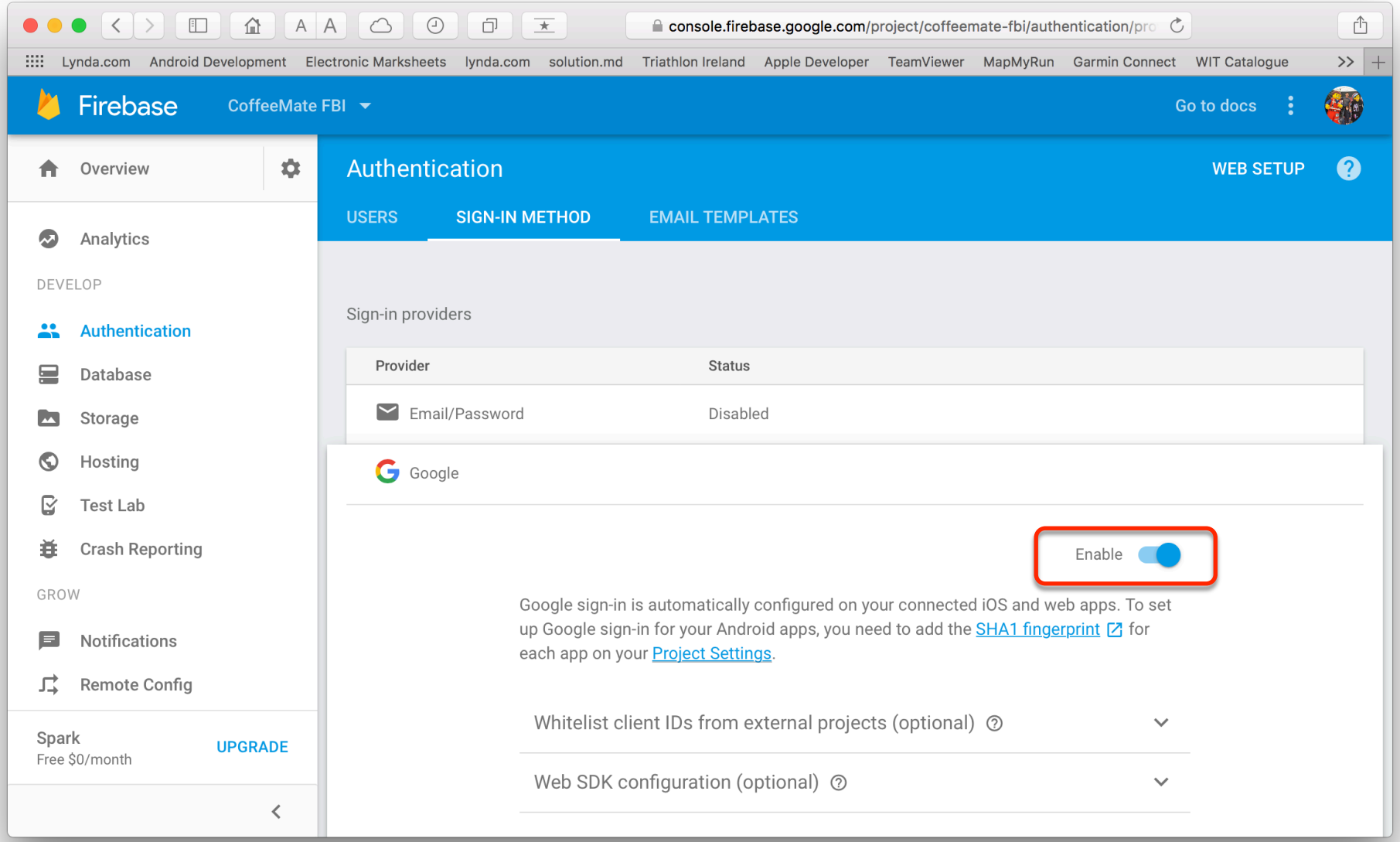
1. Setup your Sign-In Method *



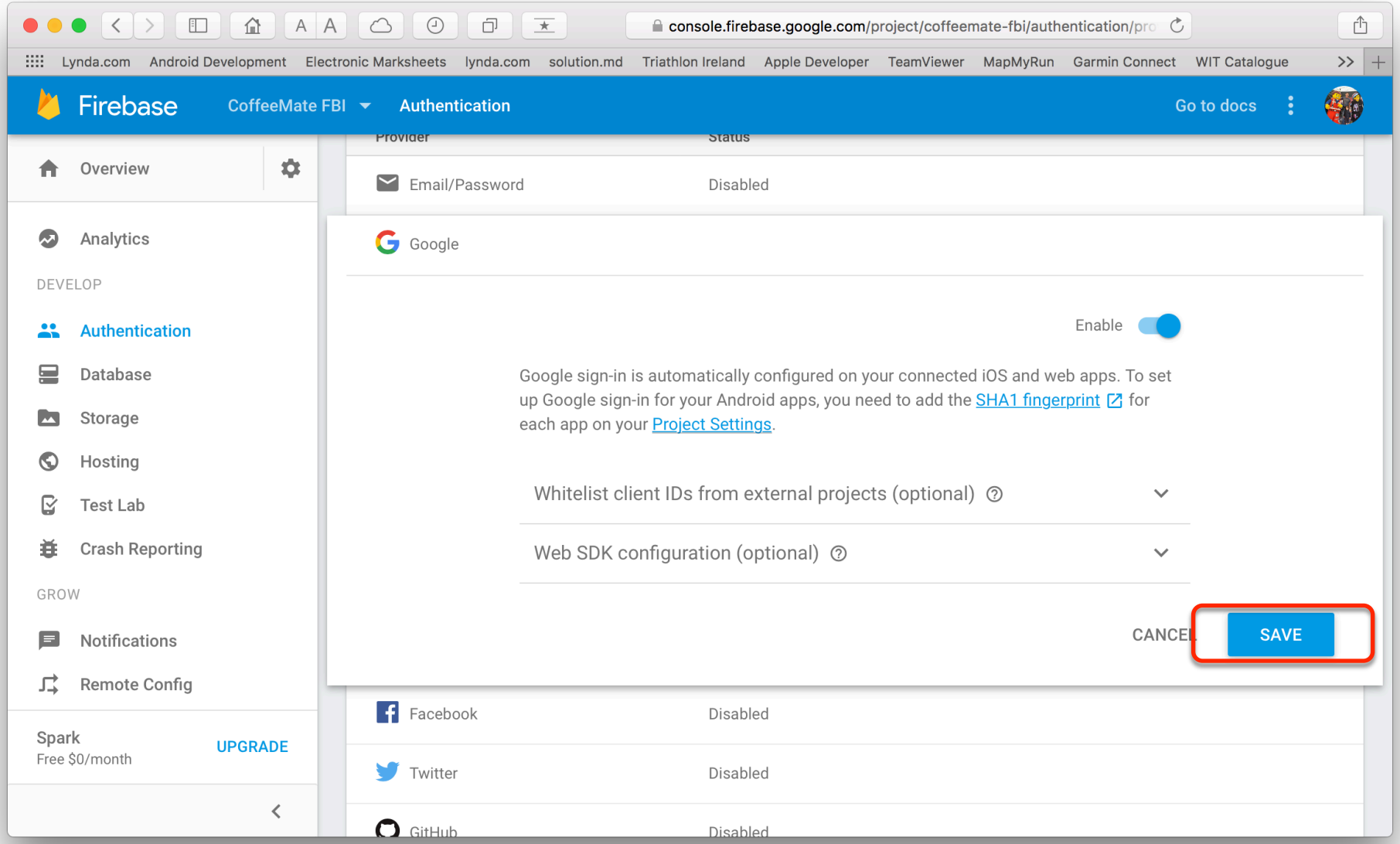
1. Setup your Sign-In Method *



1. Setup your Sign-In Method *



1. Setup your Sign-In Method *



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());
}
```

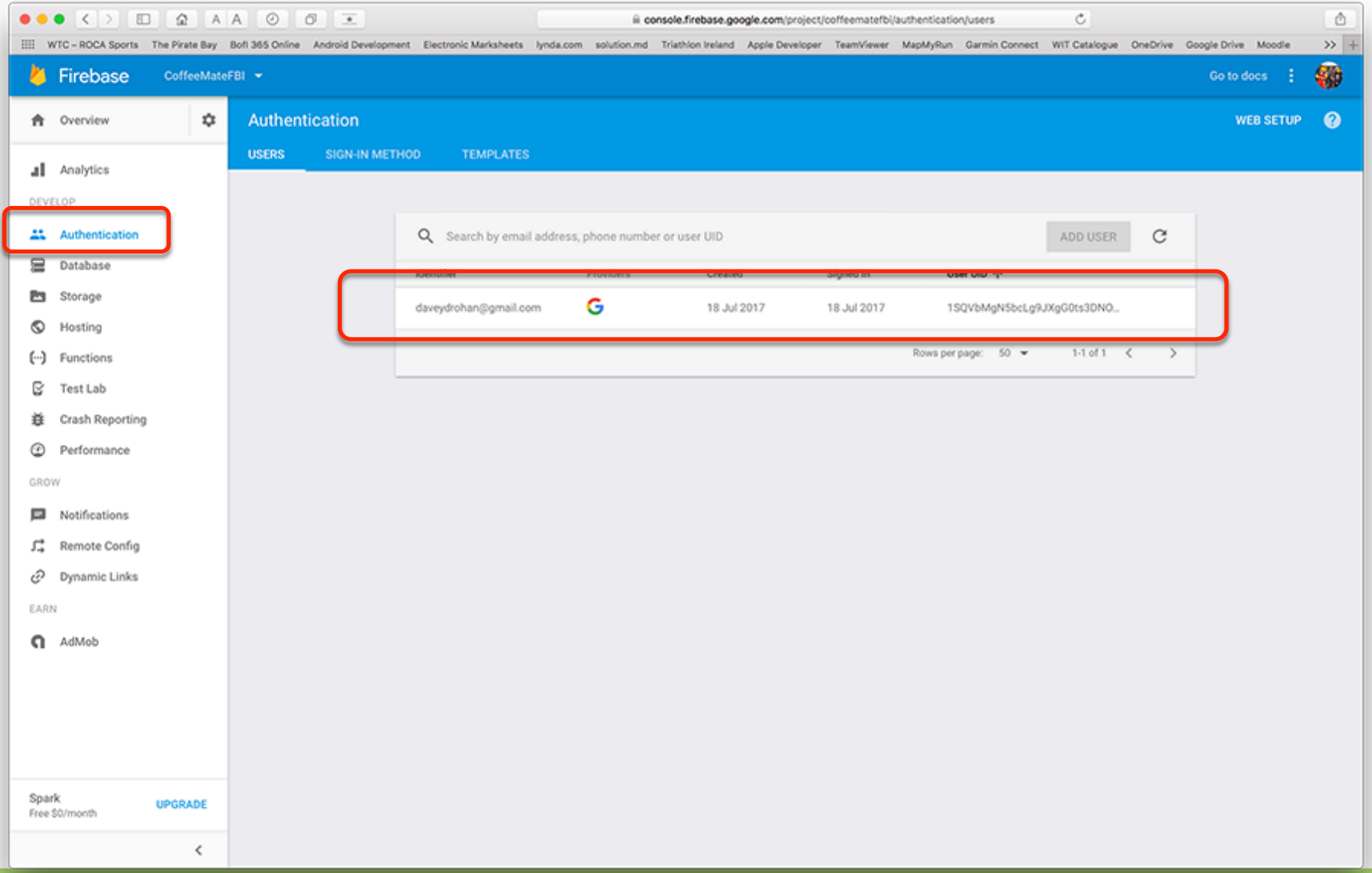
2. Introduce Authentication Flow *

```
//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();
                }
            }
    );
}
```

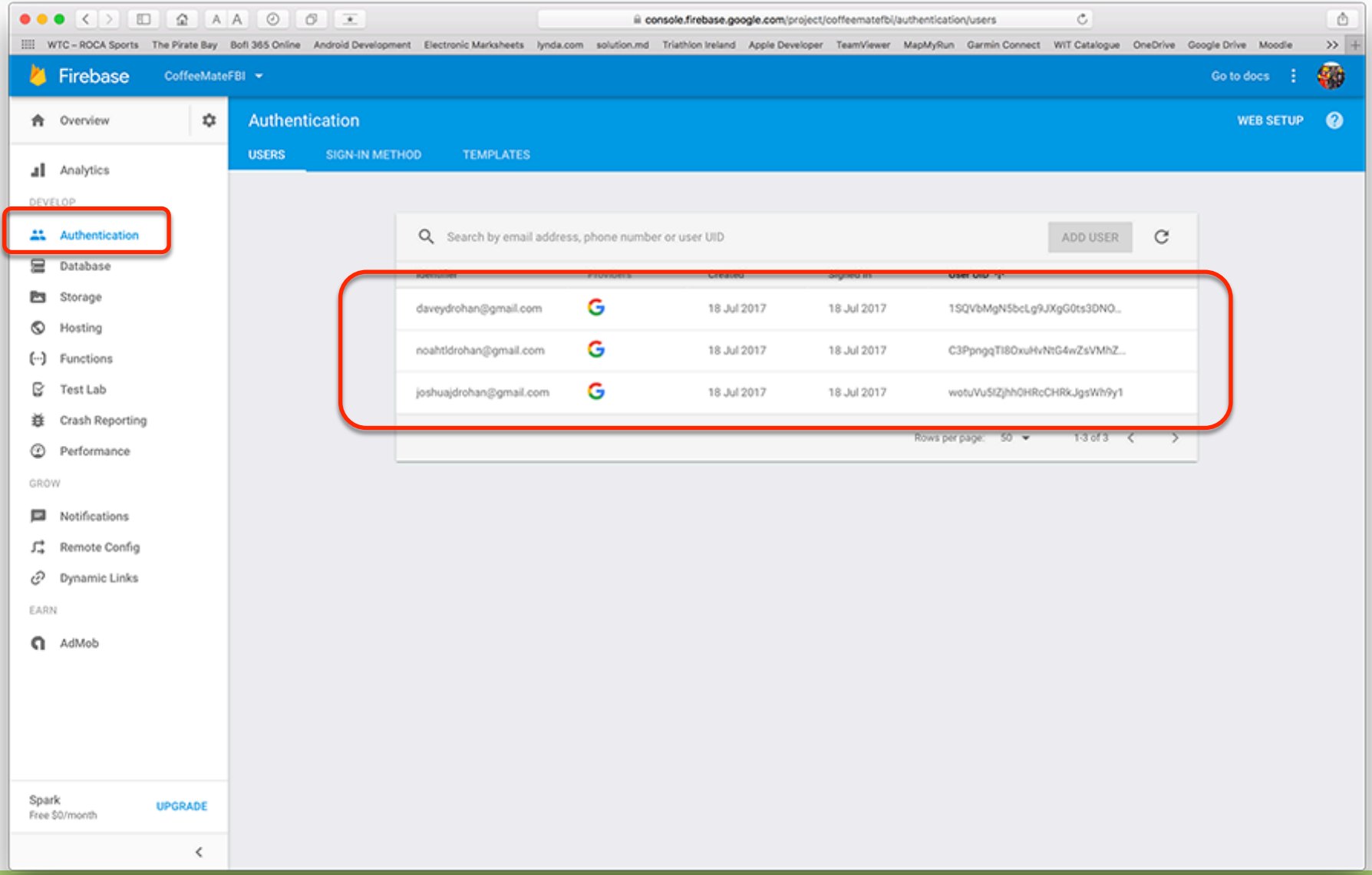
2. Introduce Authentication Flow *

```
@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.mFirebaseDatabase.child("users")
                                          .child(app.mFirebaseUser.getUid())
                                          .setValue(newUser);
    }
    app.mFBDBManager.mFBUserId = app.mFirebaseUser.getUid();

    startHomeScreen();
}
```

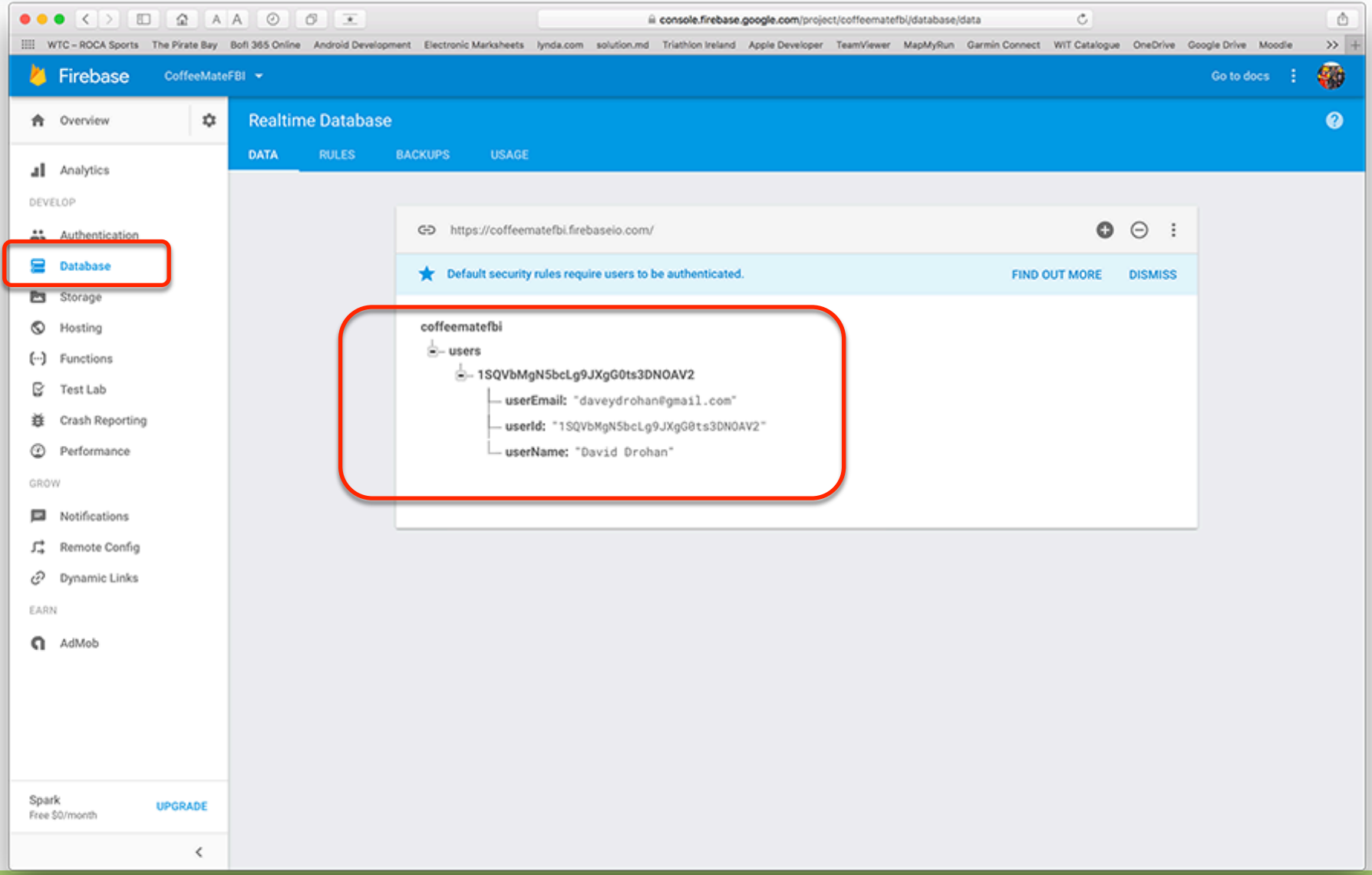
Firebase Console – Authenticated Users *



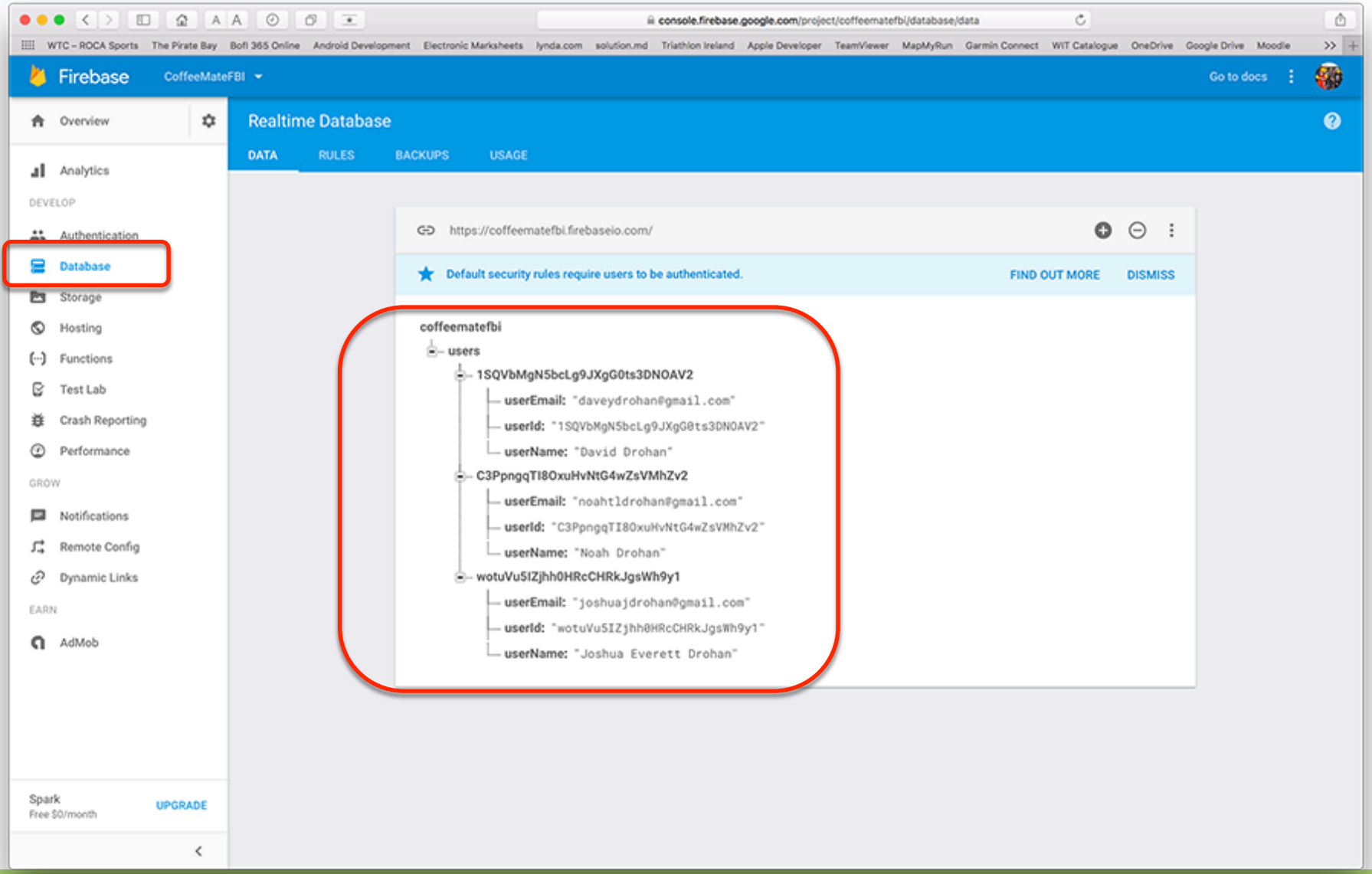
Firebase Console – Authenticated Users *

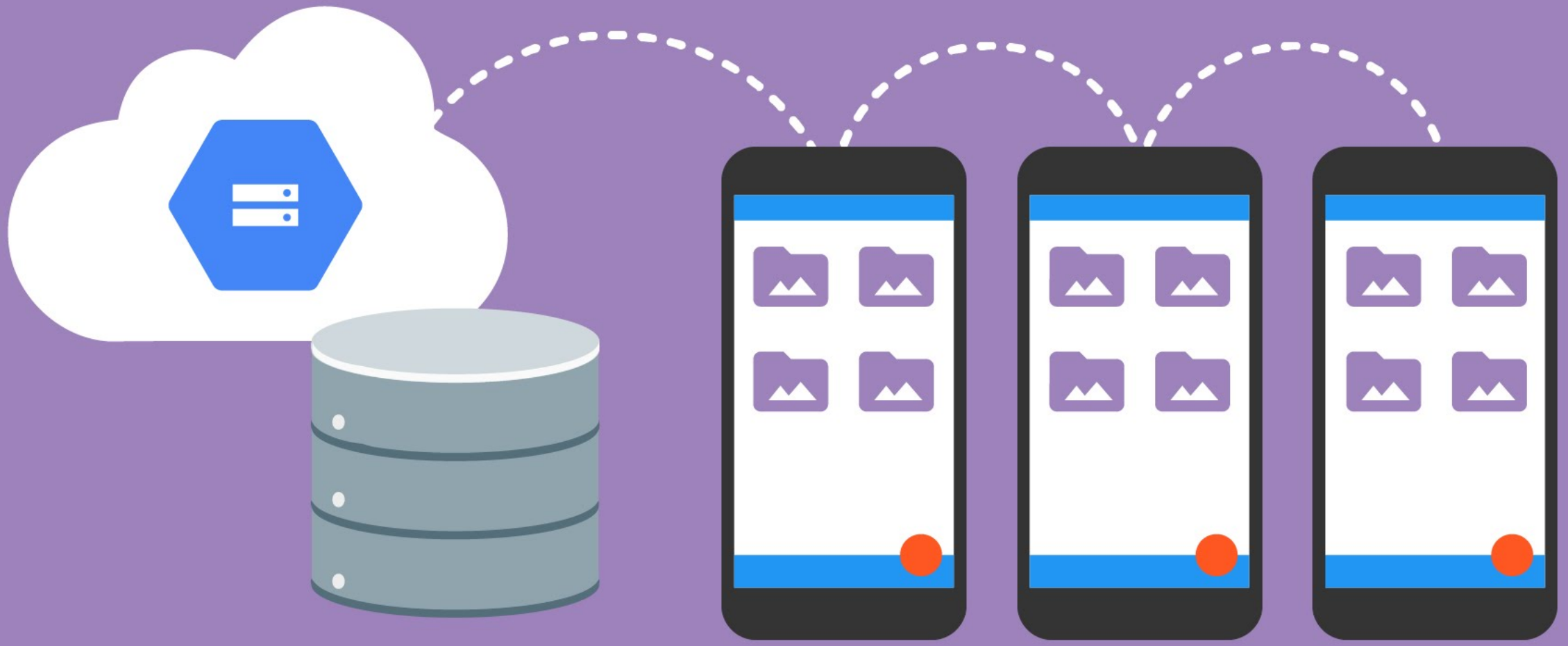


Firebase Console – User Data in db *



Firebase Console – User Data in db *





Storage

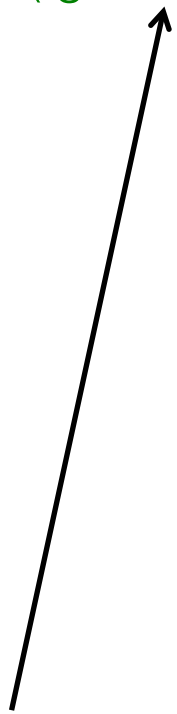
Sync files and folders seamlessly…

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, OnFailuerListener

Download Files From Firebase

- Download to a local file
  - File localFile = File.createTempFile('mefile','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(bytes[] 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;
    }
  }
}
```

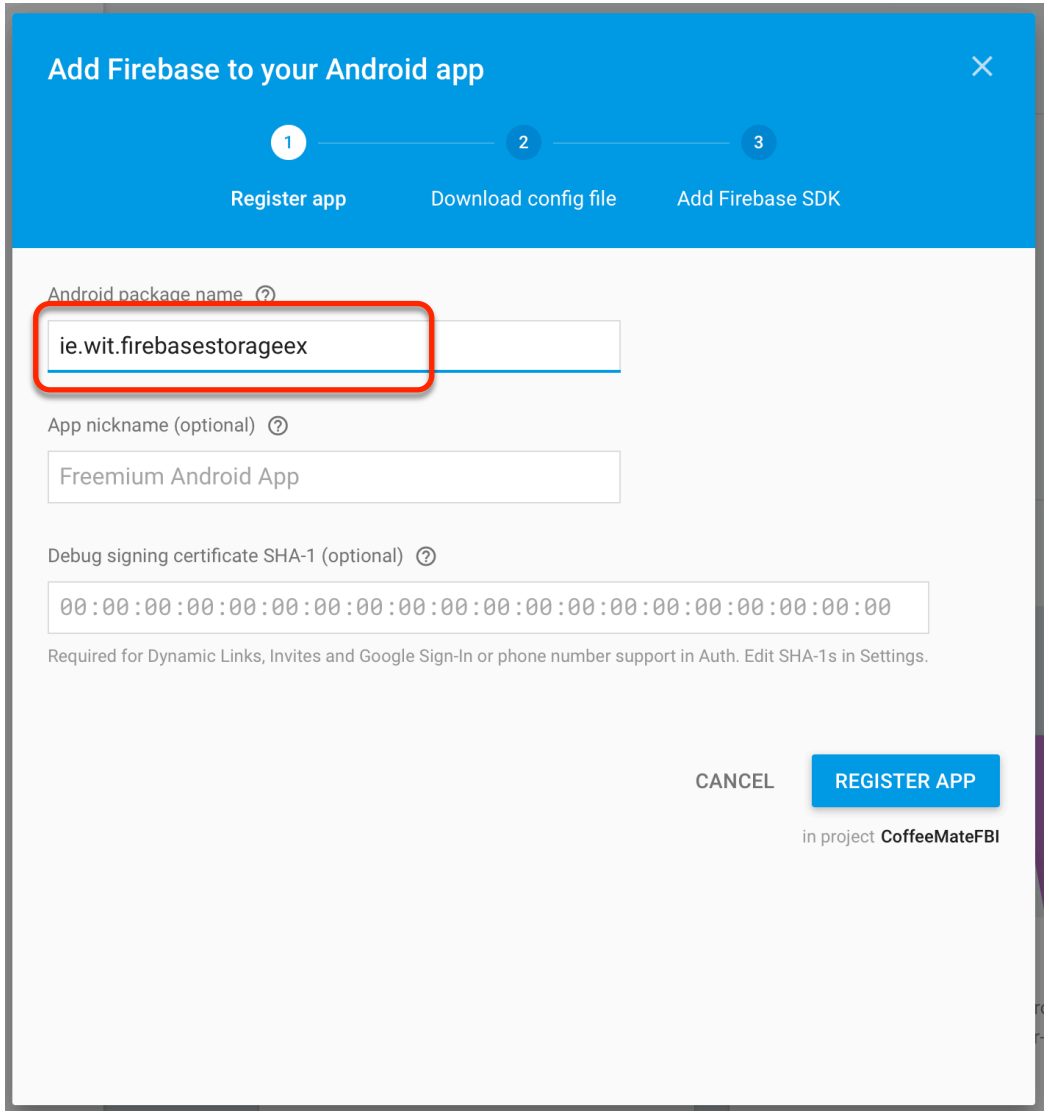
BEFORE

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

AFTER

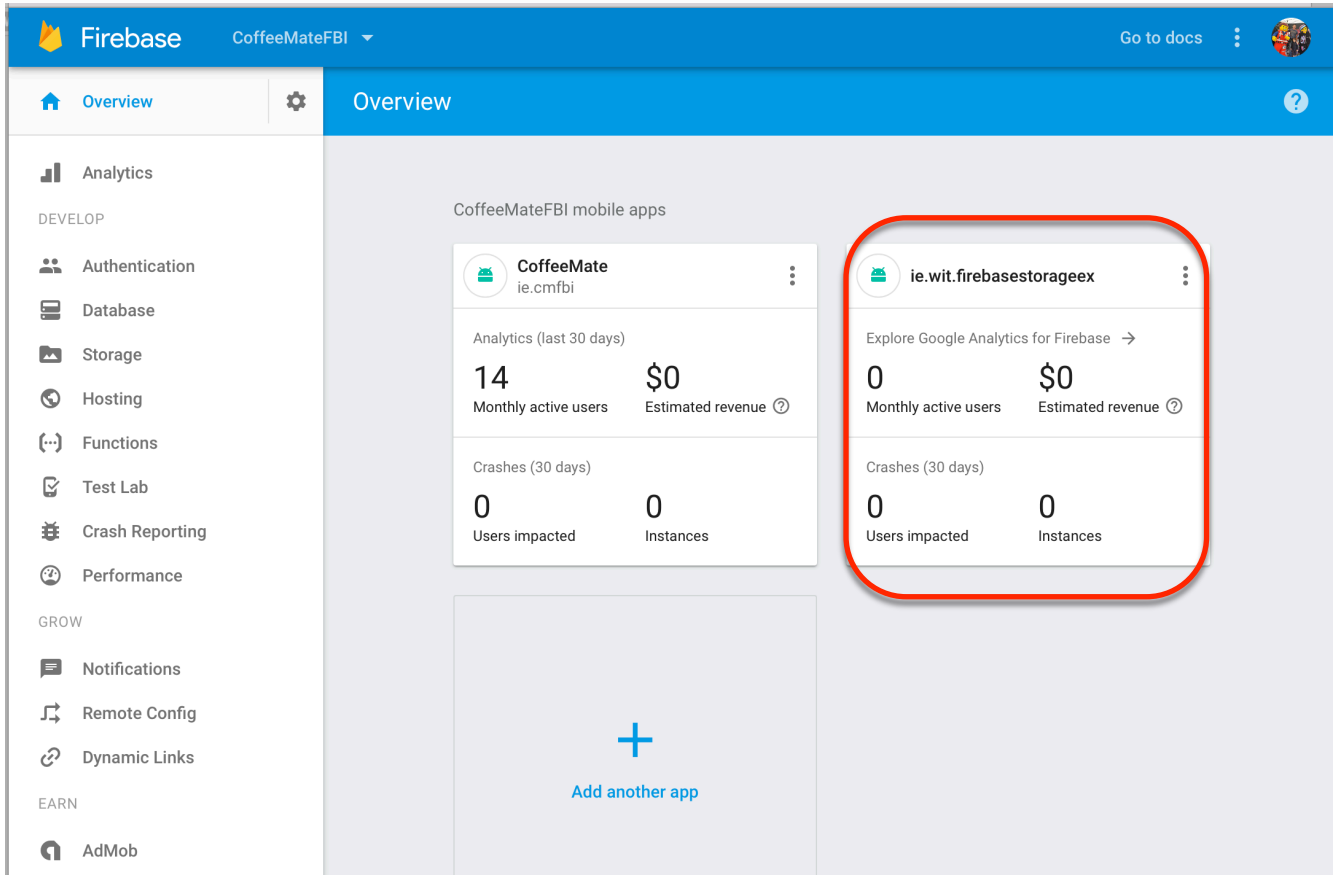
FirebaseStorageEx App - Setup *

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

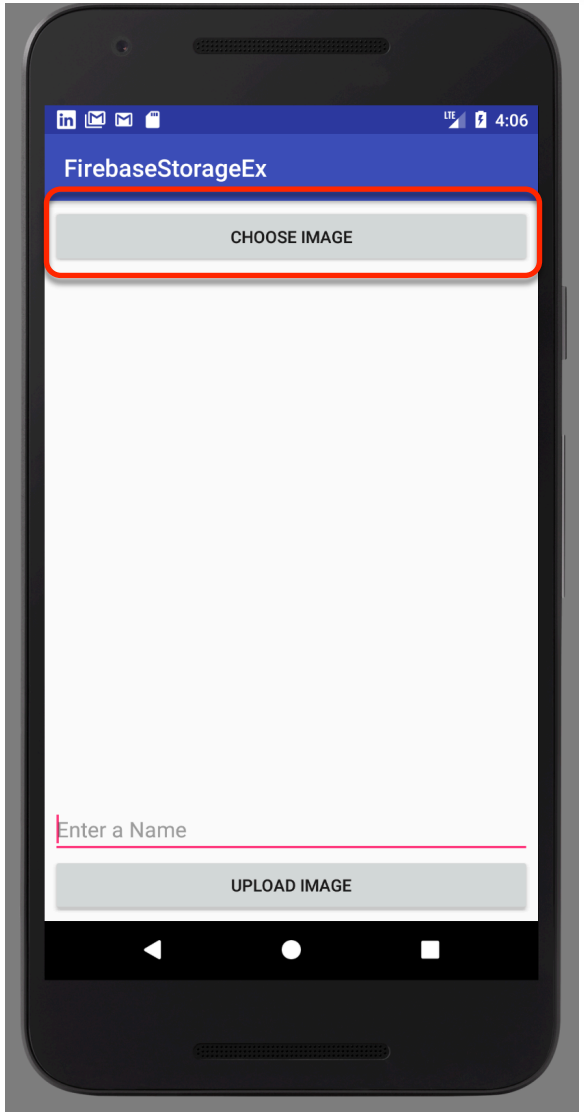


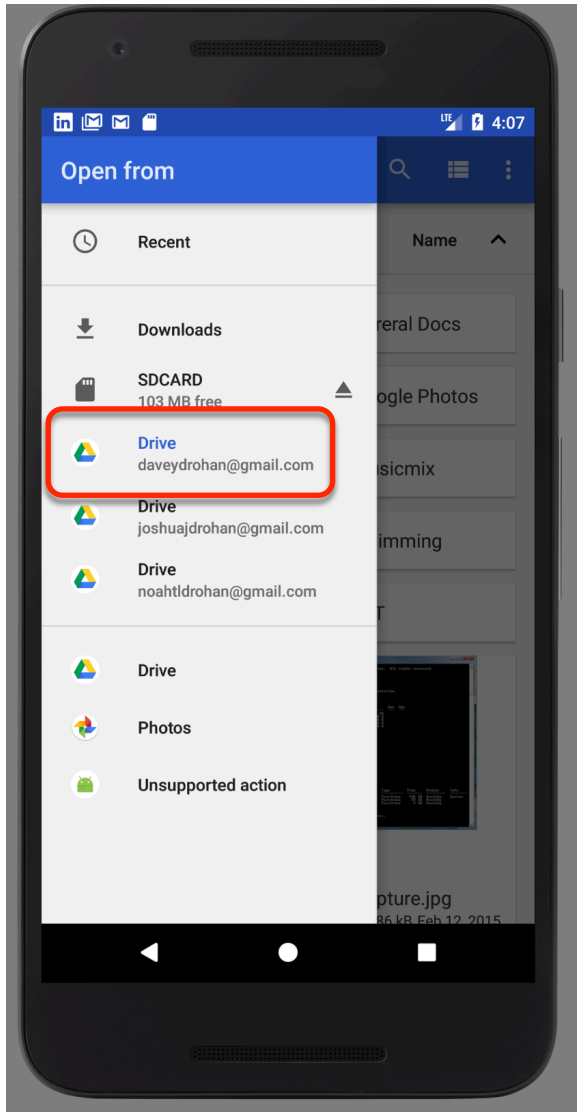
FirebaseStorageEx App - Setup *

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

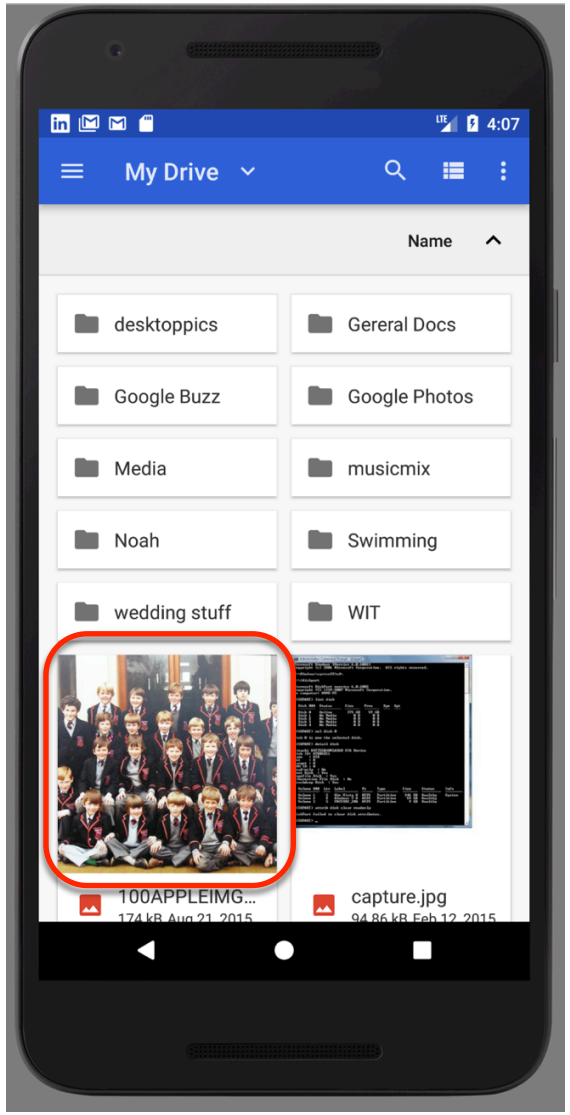


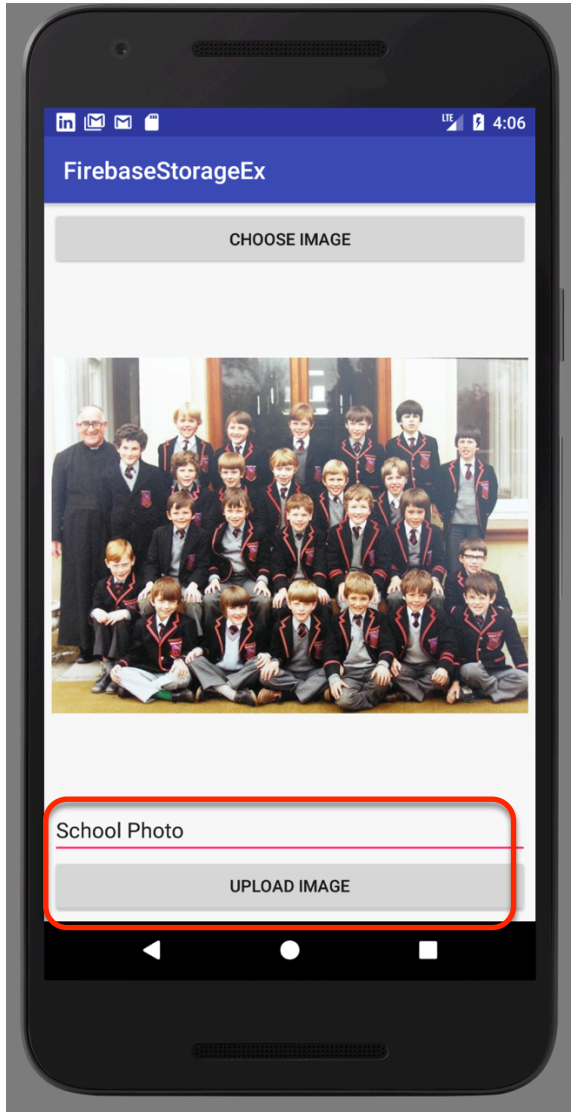
FirebaseStorageEx App - Upload *



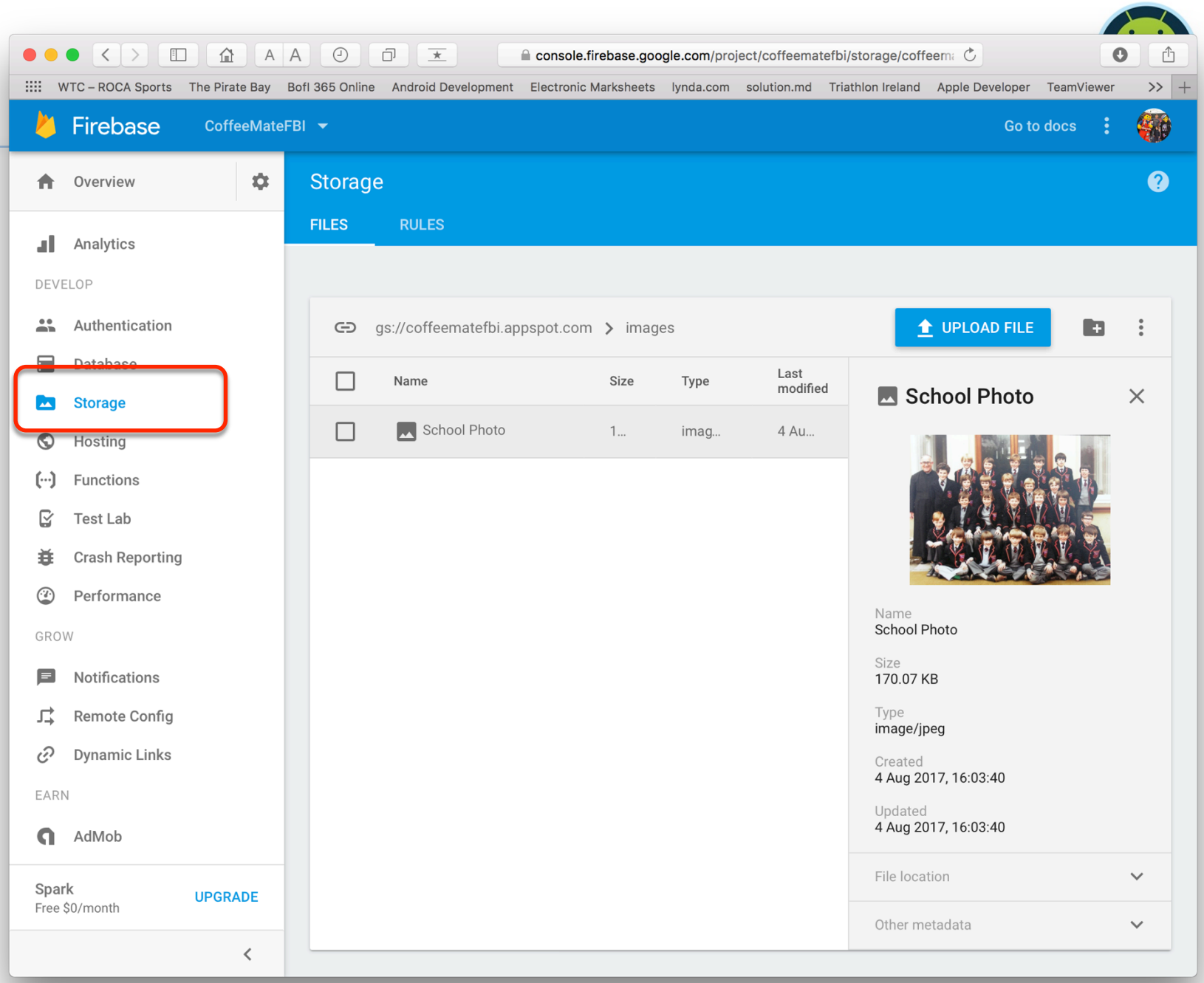


FirebaseStorageEx App - Upload *

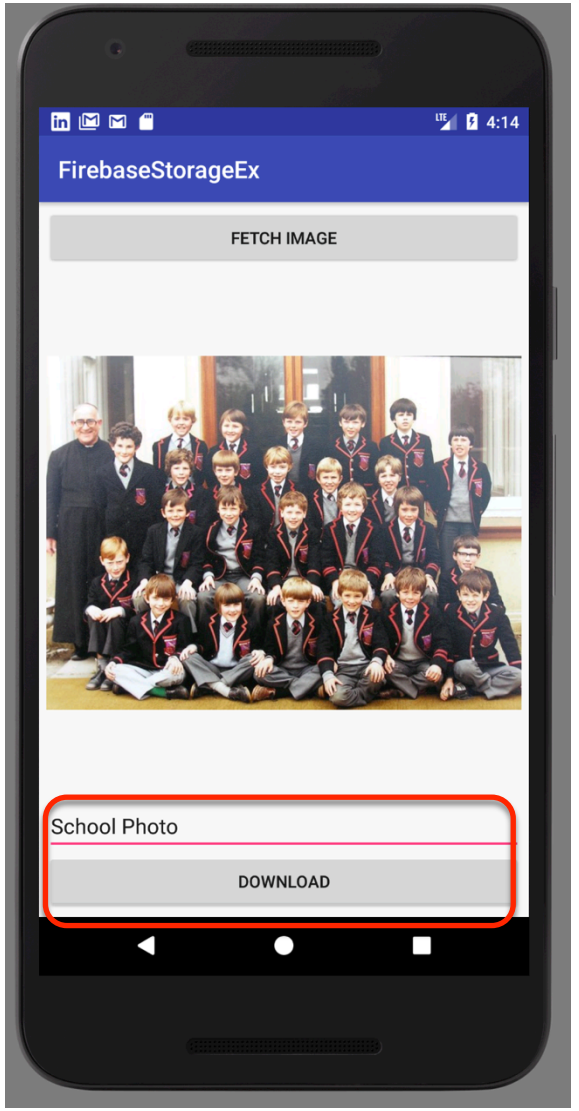




FirebaseStorageEx App – Upload



FirebaseStorageEx App - Download *



FirebaseStorageEx App – uploadImage( )*

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

FirebaseStorageEx App - fetchImage( ) *

```
private void fetchImage() {
    FirebaseStorage storage=FirebaseStorage.getInstance();
    // Points to the root reference
    StorageReference storageRef = storage.getReferenceFromUrl("gs://coffeematefbi.appspot.com");
    // Points to "images" Directory
    StorageReference imagesRef = storageRef.child("images");
    // Points to "images/'filename'.jpg"
    // Note that you can use variables to create child values
    String fileName = editTextName.getText().toString();
    StorageReference fileNameRef = imagesRef.child(fileName);

    // File path is "images/'filename'.jpg"
    String path = fileNameRef.getPath();
    // File name is "'filename'.jpg"
    String name = fileNameRef.getName();
    // Points to "images"
    imagesRef = fileNameRef.getParent();

    Glide.with(this /* context */)
            .using(new FirebaseImageLoader())
            .load(fileNameRef)
            .into(imageView);
}
```

FirebaseStorageEx App - downloadImage( ) *

```
private void downloadImage() {

    FirebaseStorage storage=FirebaseStorage.getInstance();
    // Create a storage reference from our app
    StorageReference storageRef = storage.getReferenceFromUrl("gs://coffeematefbi.appspot.com");

    storageRef.child("images/"+editTextName.getText().toString()).getBytes(Long.MAX_VALUE)
            .addOnSuccessListener(new OnSuccessListener<byte[]>() {
        @Override
        public void onSuccess(byte[] bytes) {
            // Use the bytes to display the image
            String path=Environment.getExternalStorageDirectory()+"/"+editTextName.getText().toString();
            try {
                FileOutputStream fos=new FileOutputStream(path);
                fos.write(bytes);
                fos.close();
                Toast.makeText(ViewDownloadActivity.this, "Success!!!", Toast.LENGTH_SHORT).show();

            }  catch (IOException e) {
                e.printStackTrace();
                Toast.makeText(ViewDownloadActivity.this, e.toString(), Toast.LENGTH_SHORT).show();
            }
            pd.dismiss();
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
            // Handle any errors
            pd.dismiss();
            Toast.makeText(ViewDownloadActivity.this, exception.toString()+"!!!", Toast.LENGTH_SHORT).show();
        }
    });

}
```

Pricing

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