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

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…

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

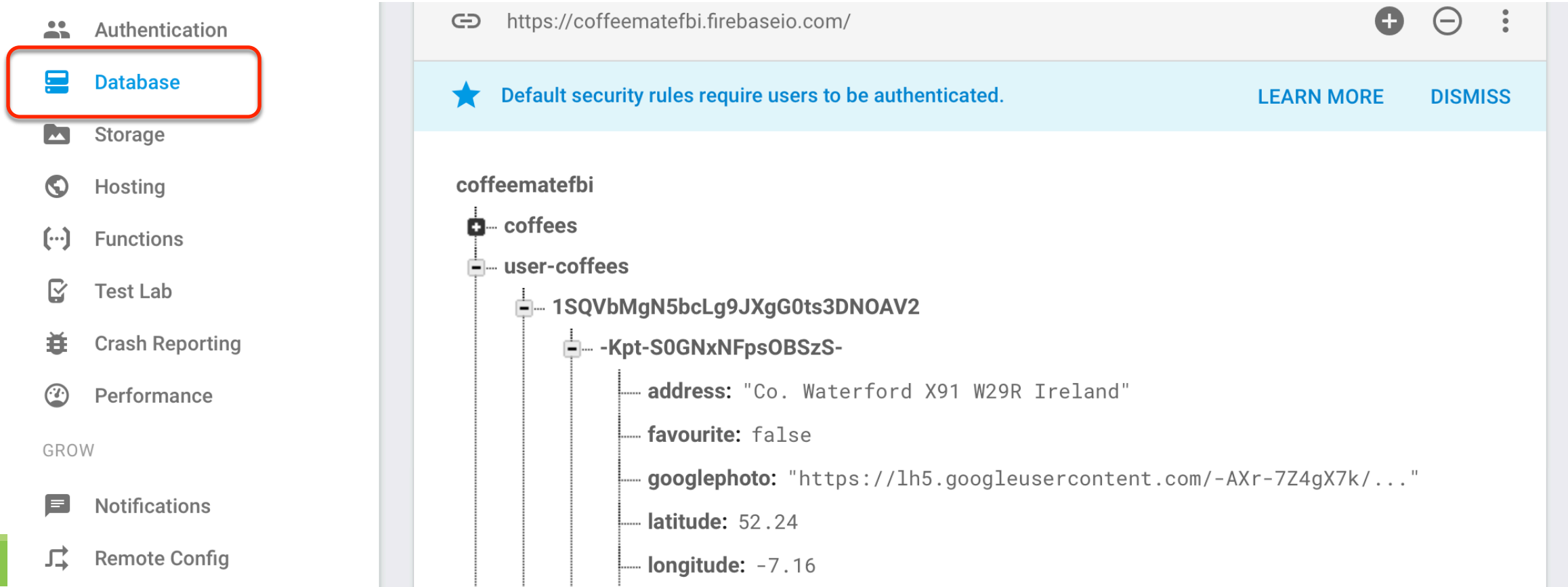
- Get database reference (your base node)
  - DatabaseReference mDBRef = FirebaseDatabase.*getInstance*().getReference();
- Point it to the right JSON node
  - mDBRef = mDBRef.child("mydb").child("table1");
- Set the value at the pointed node
  - mDBRef.child('row1').setValue(myPOJO);
- Or push the value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

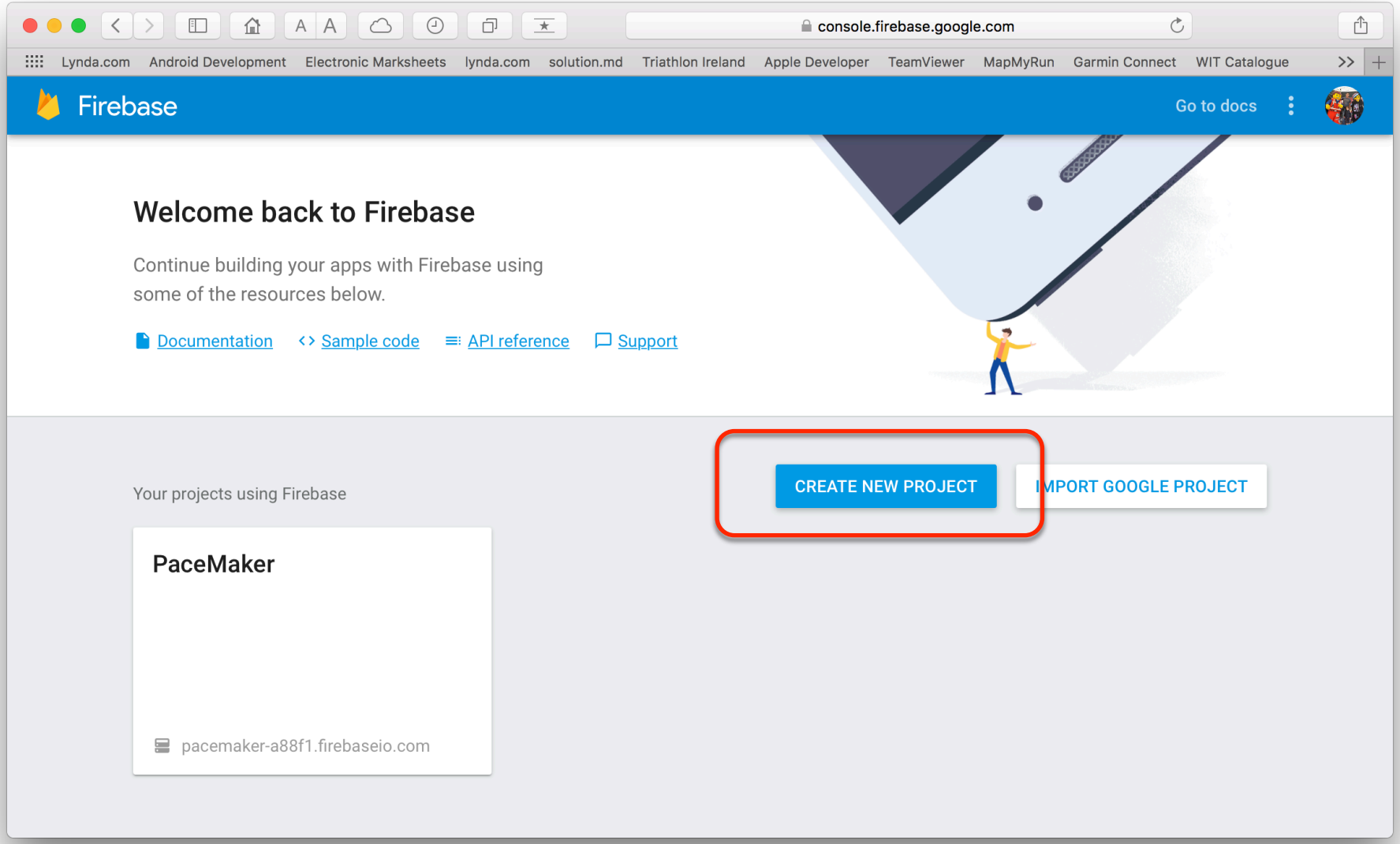
Firebase Realtime Database

Quick DEMO…

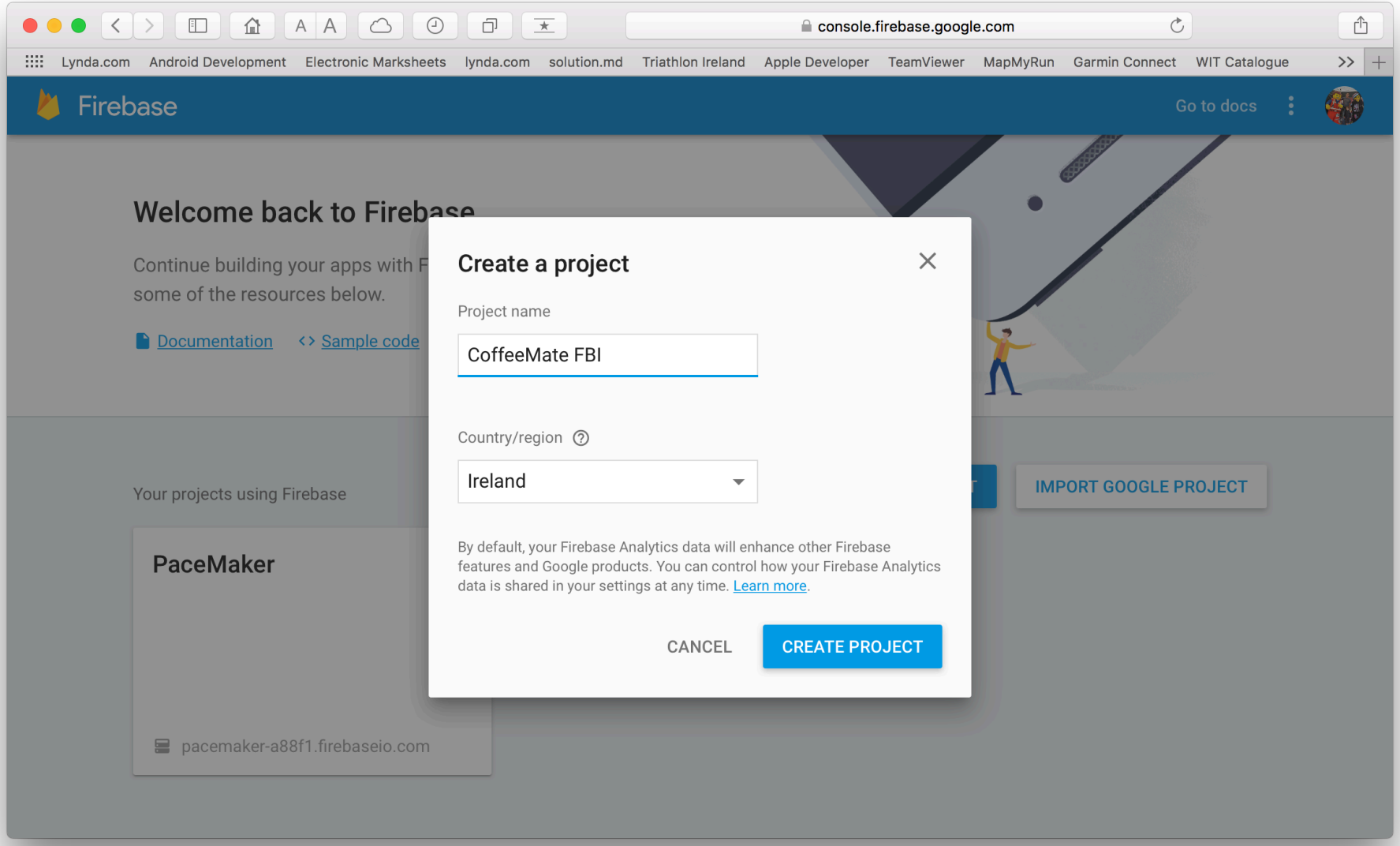
CoffeeMateFBI 1.0

Setup
&
Code
Highlights

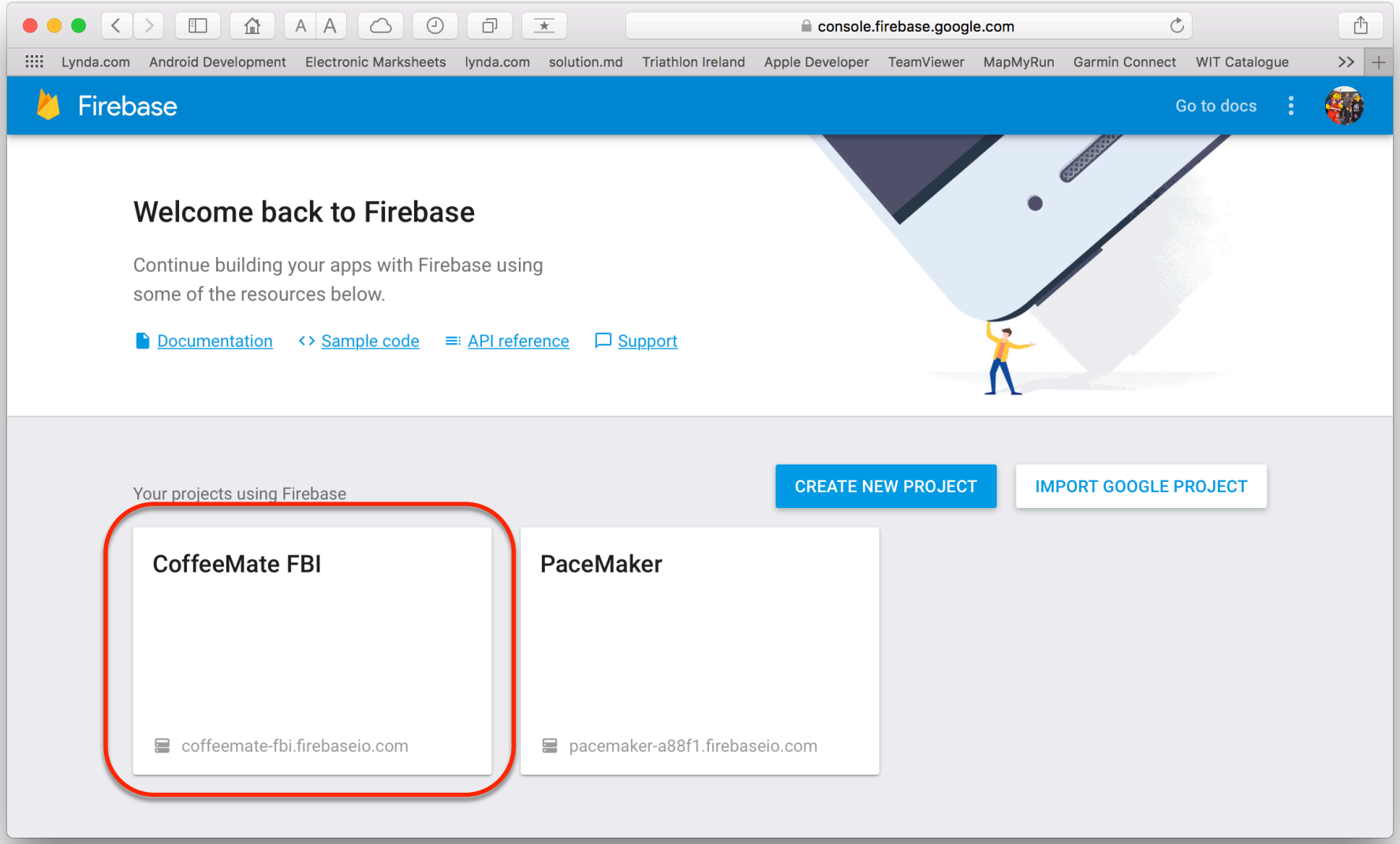
1. Create a Firebase Project *



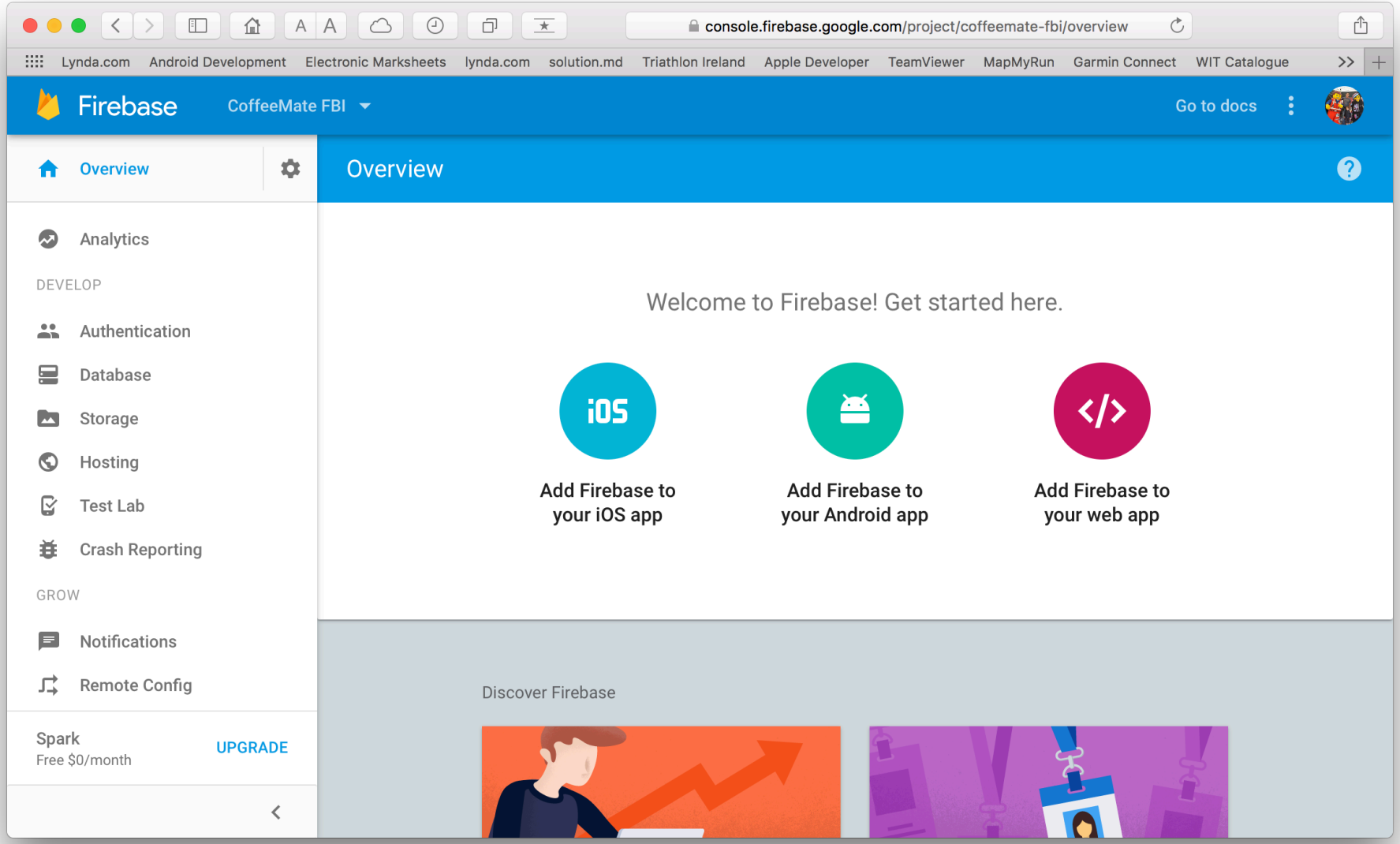
1. Create a Firebase Project



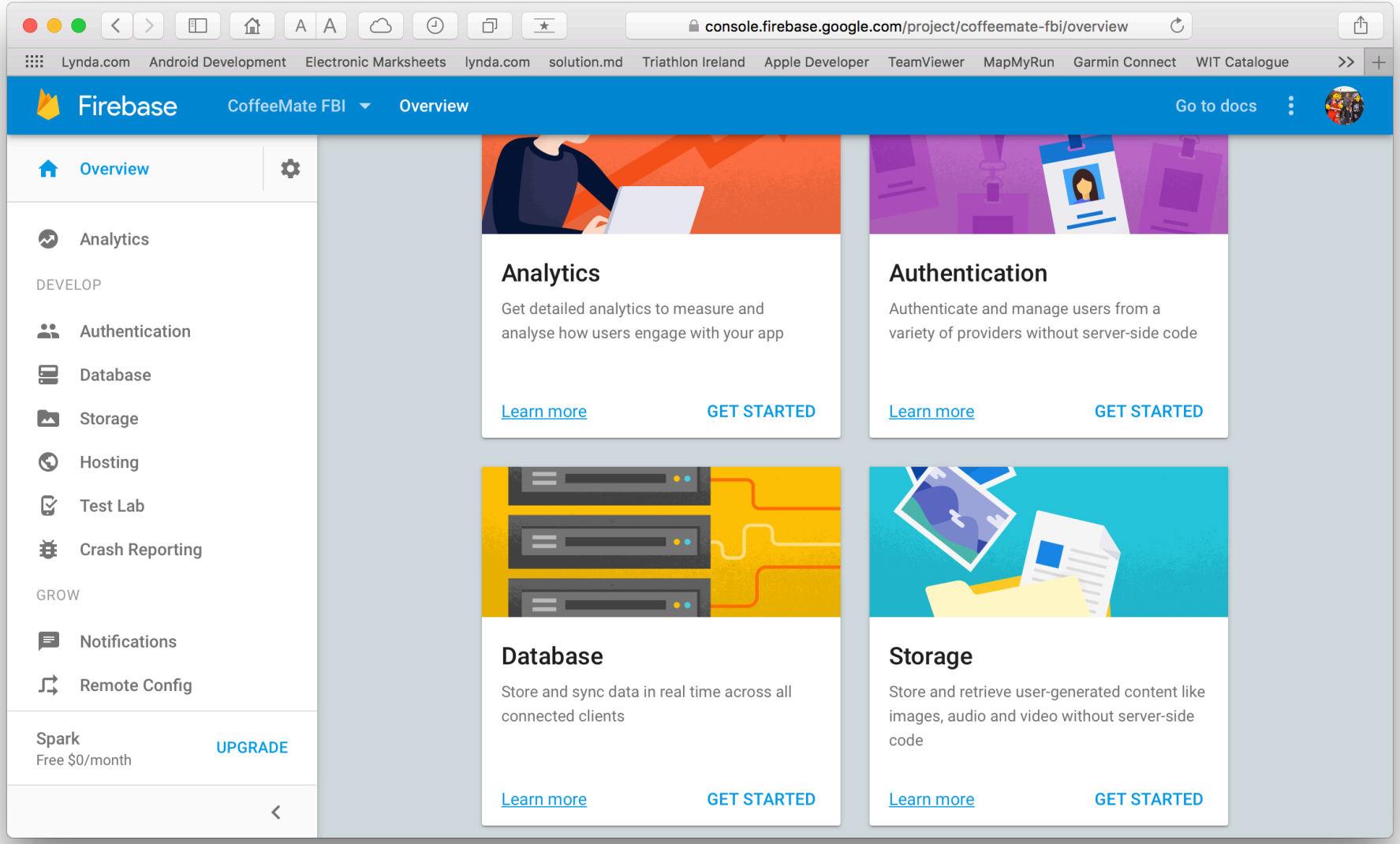
1. Create a Firebase Project *



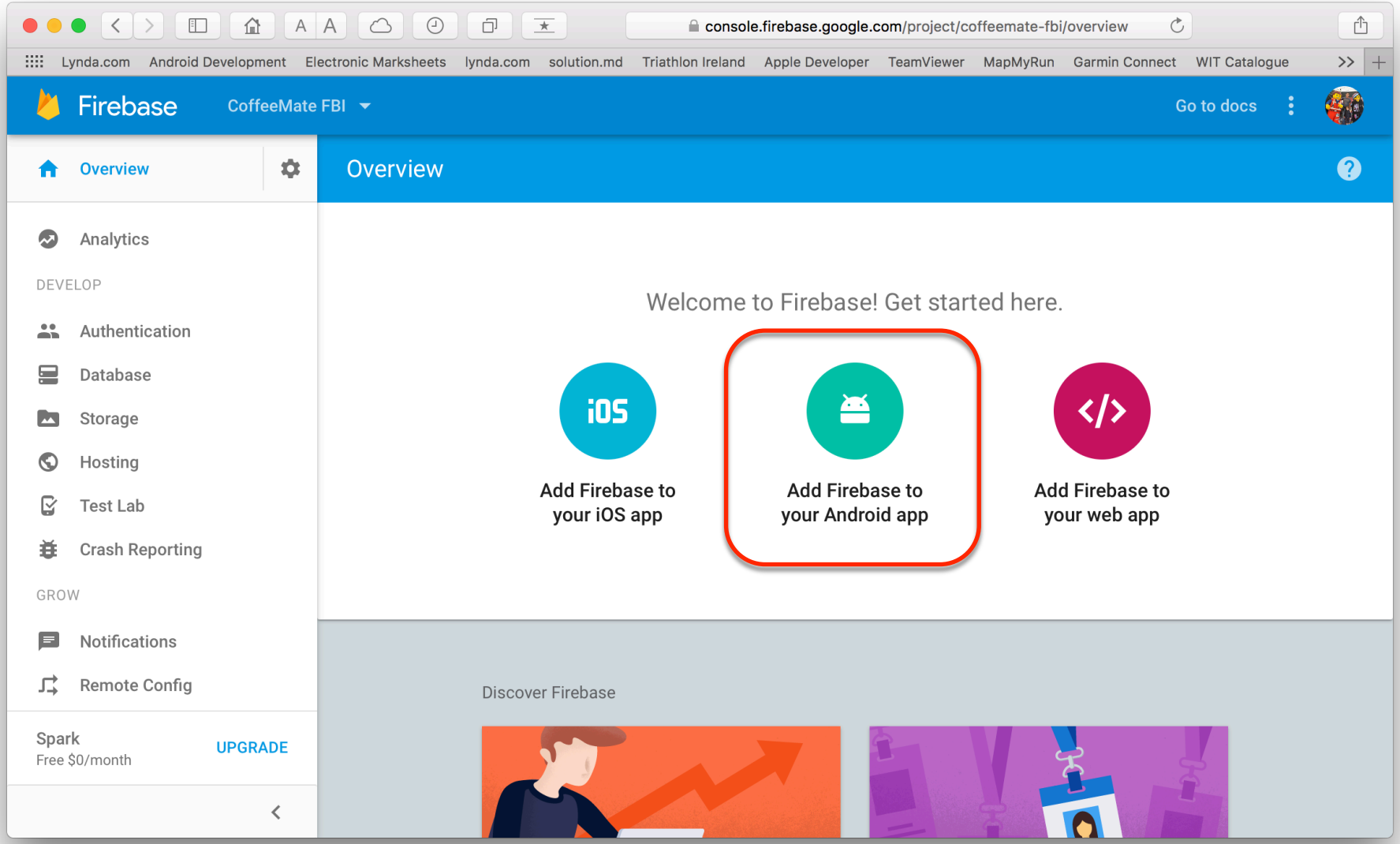
1. Create a Firebase Project - Overview



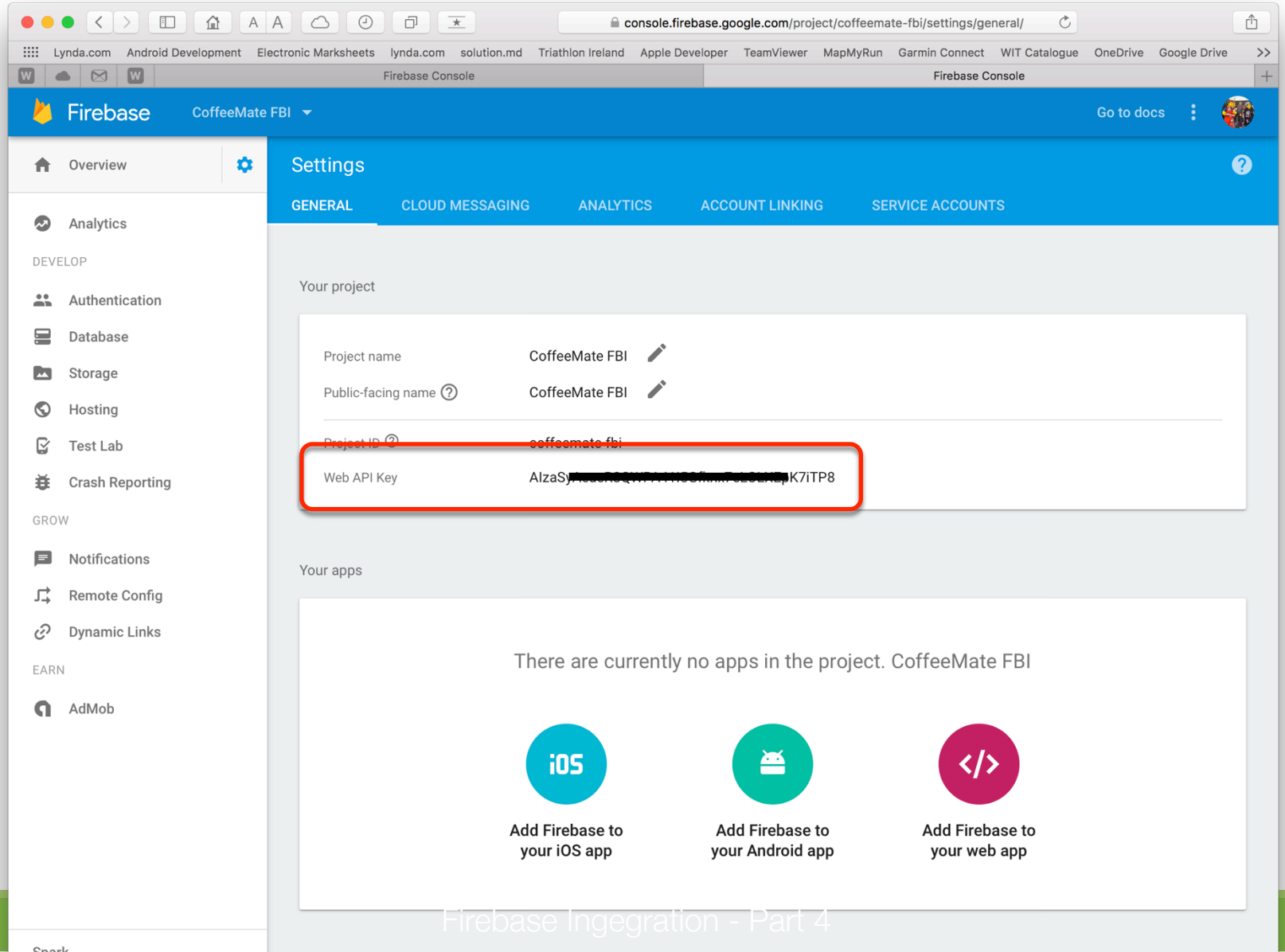
1. Create a Firebase Project - Overview



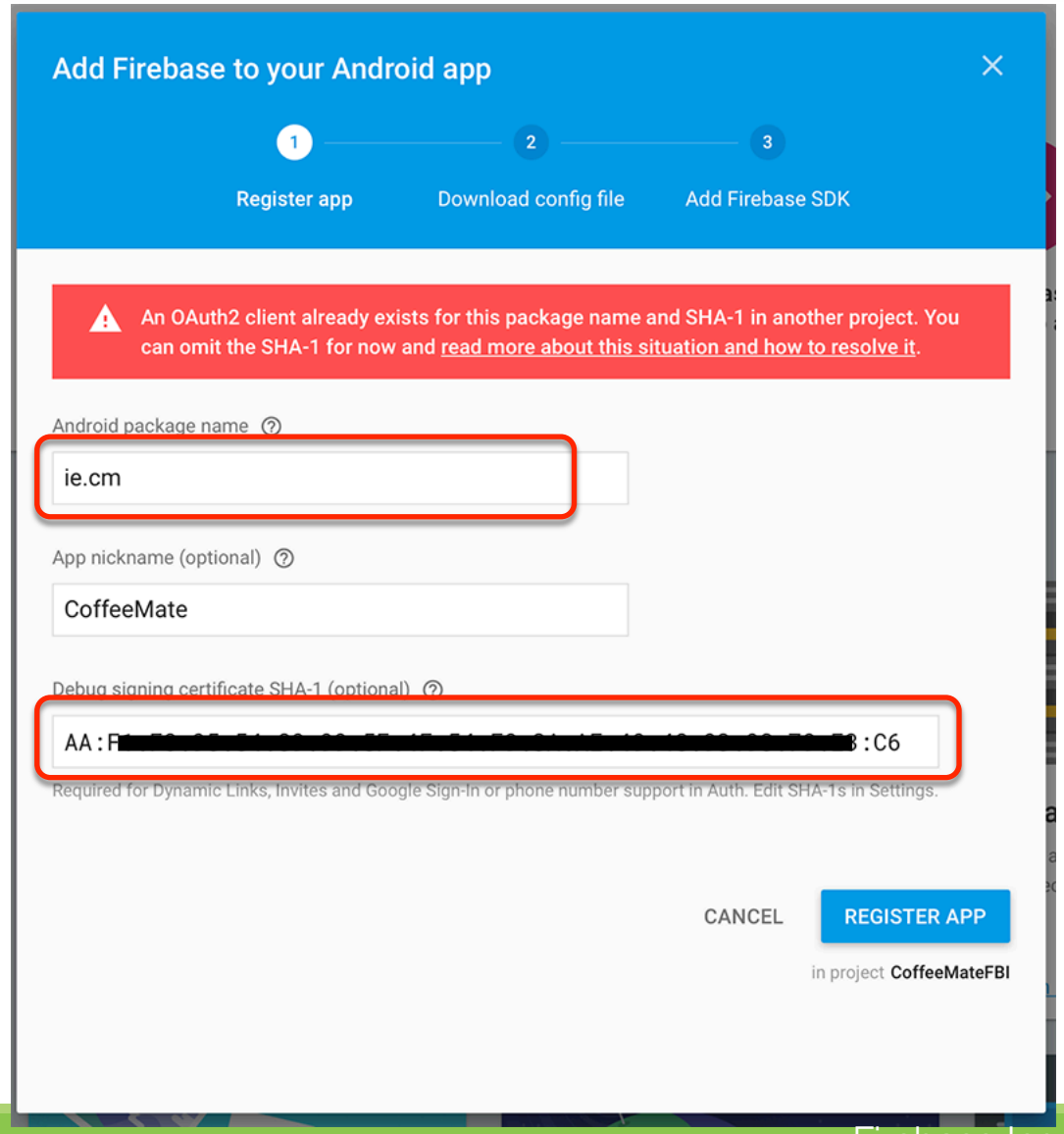
2. Add Firebase to your Android App *

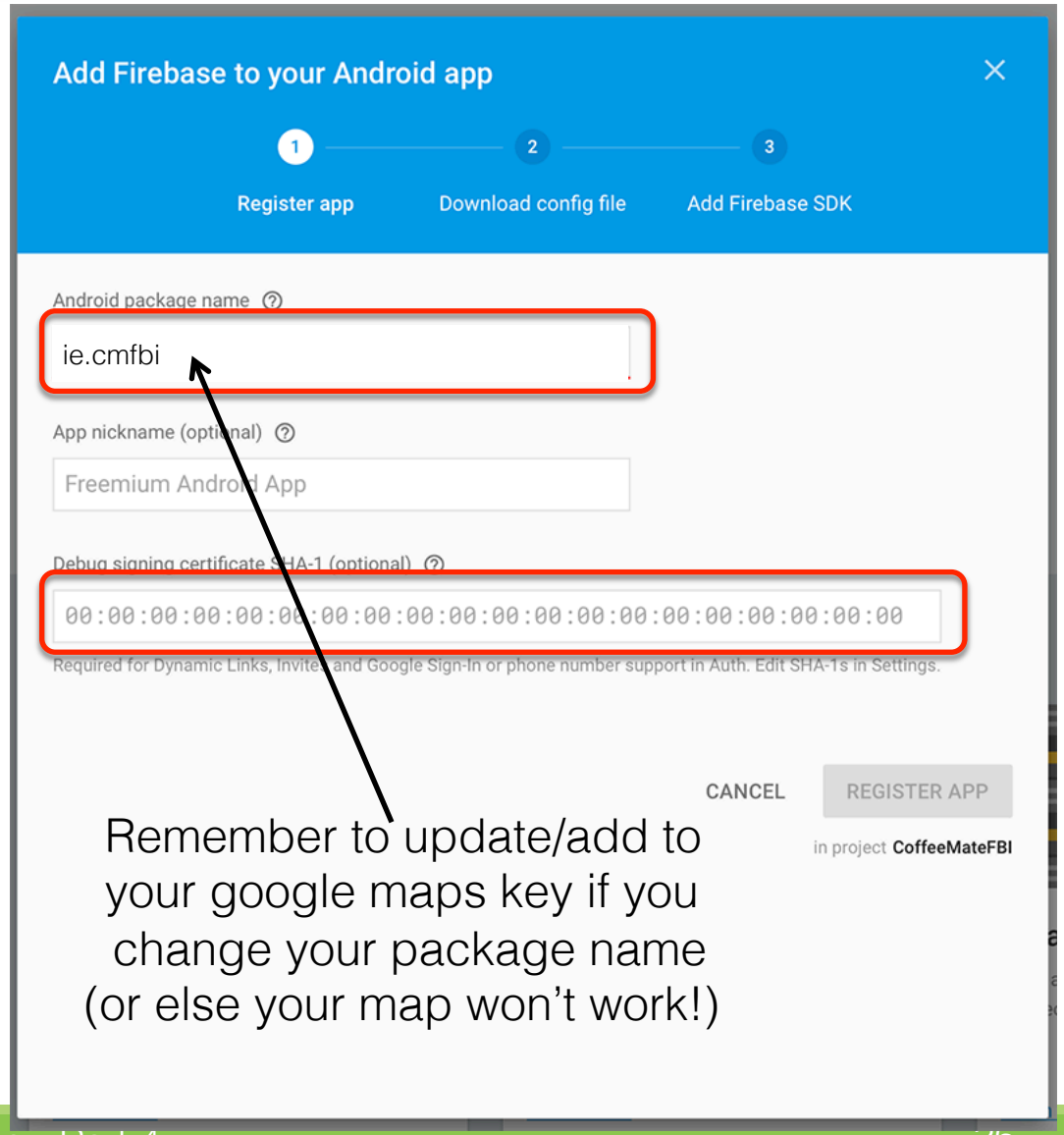


2. Add Firebase to your Android App – Web Key *



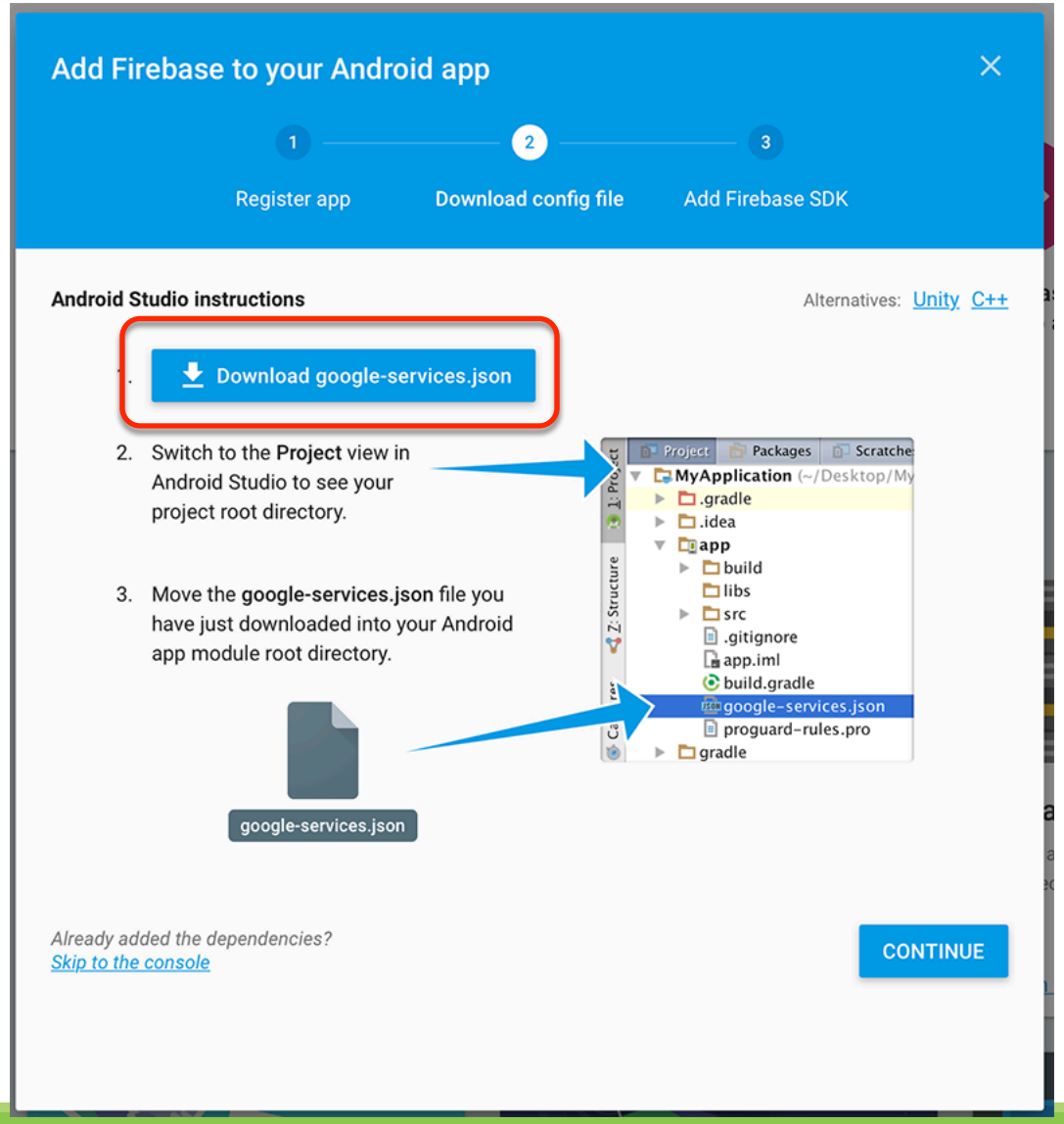
2. Add Firebase to your Android App *



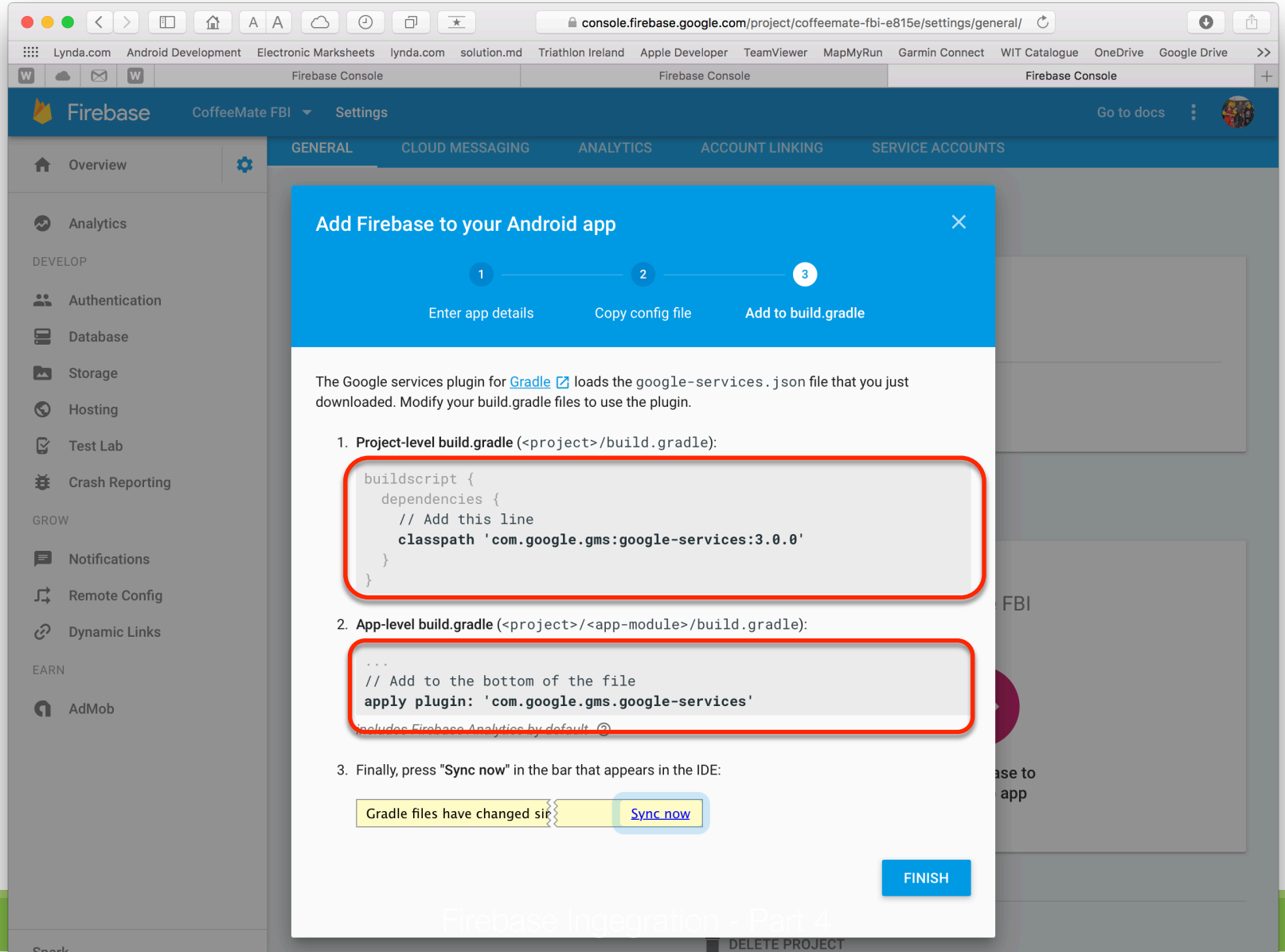


Remember to update/add to your google maps key if you change your package name (or else your map won't work!)

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

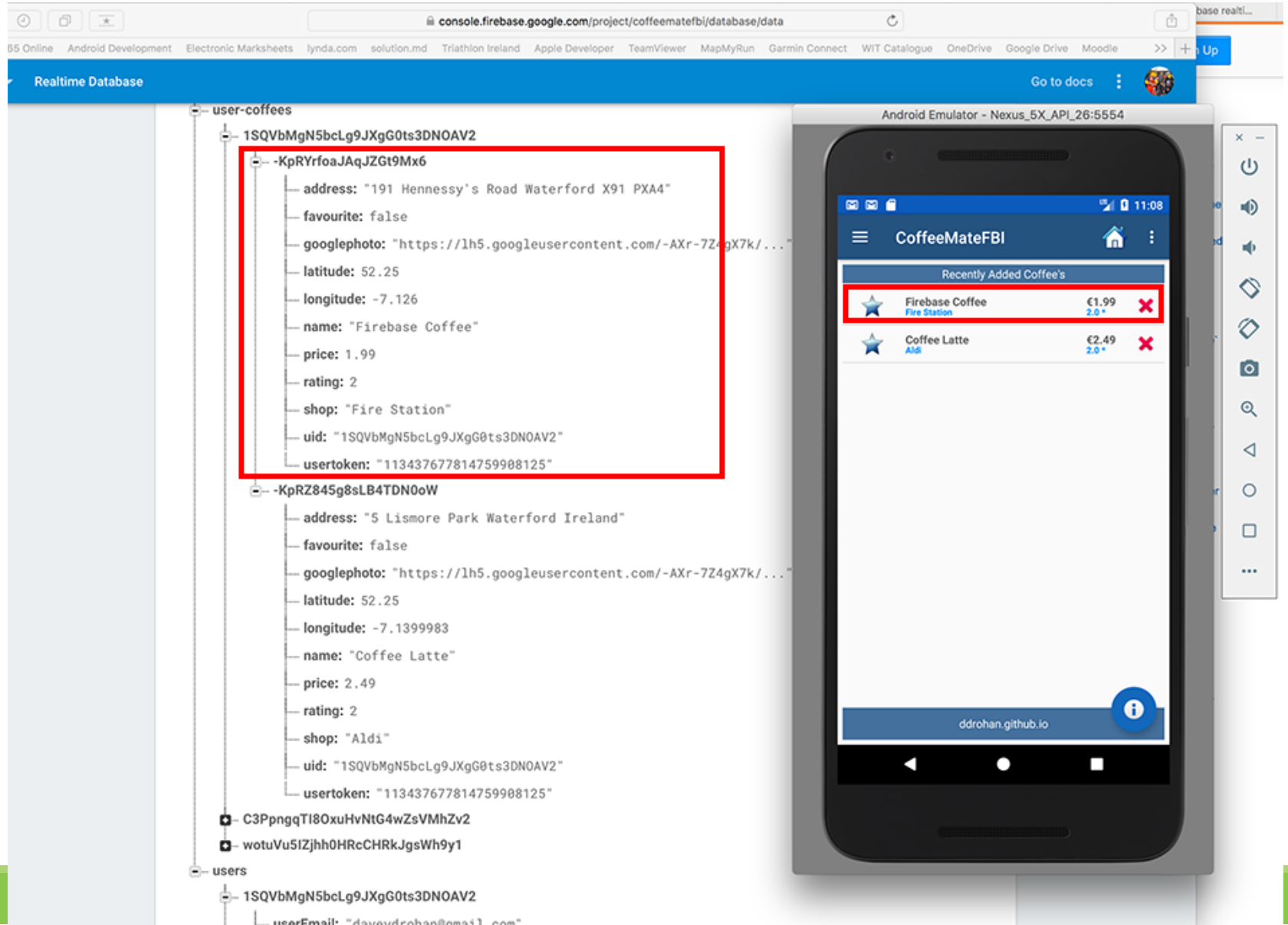
```
public void getACoffee(final String coffeeKey)
{
    mFirebaseDatabase.child("user-coffees").child(mFBUserId).child(coffeeKey)
            .addValueEventListener(
                new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot)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

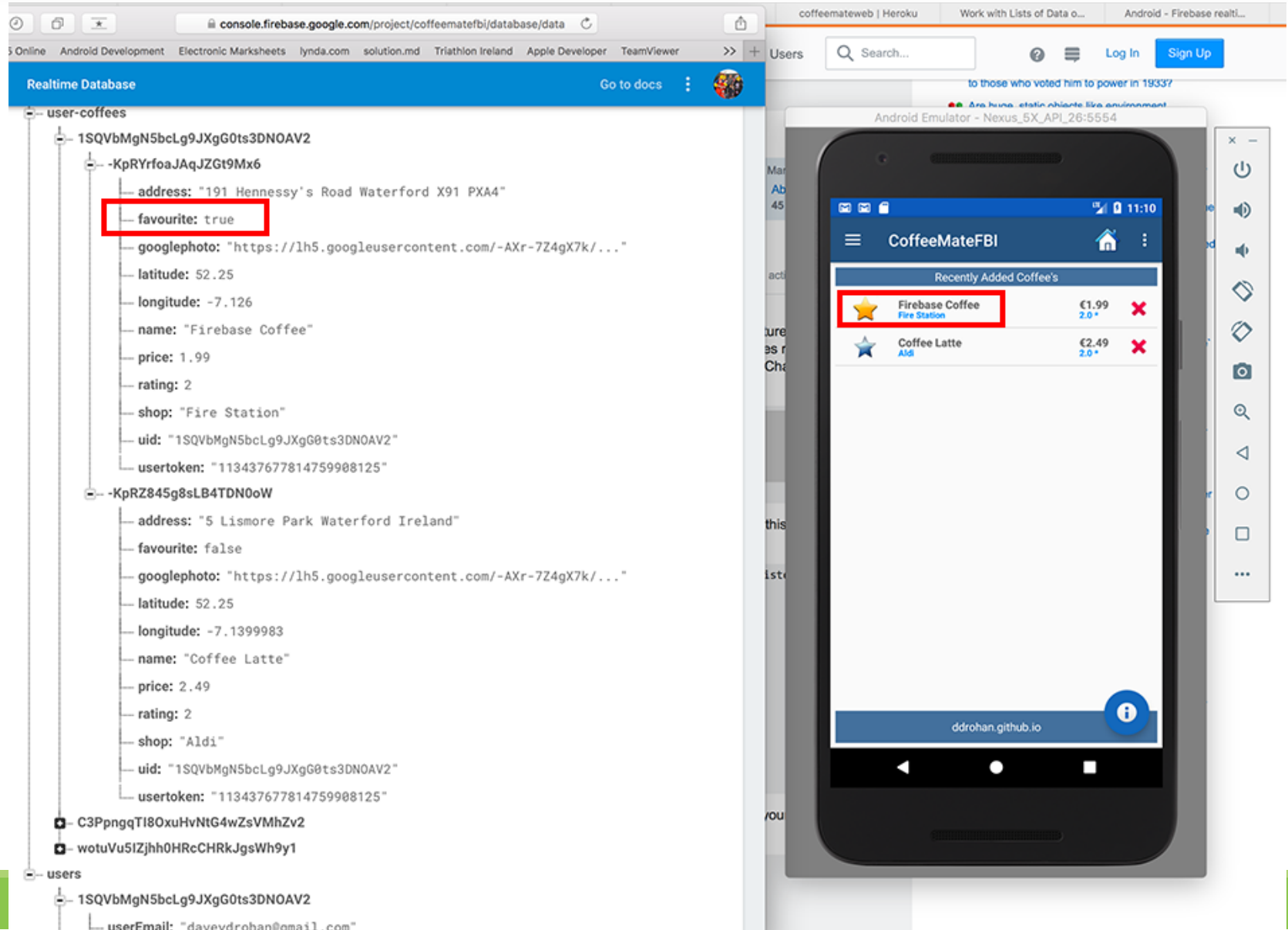
```
@Override
public void onResume() {
    super.onResume();

    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	✗	✗	✓


Thanks!

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
- Demo app available at https://goo.gl/WBP5fR

Questions?