

LAB 7-1 – SỬ DỤNG FIREBASE FIRESTORE

MỤC TIÊU

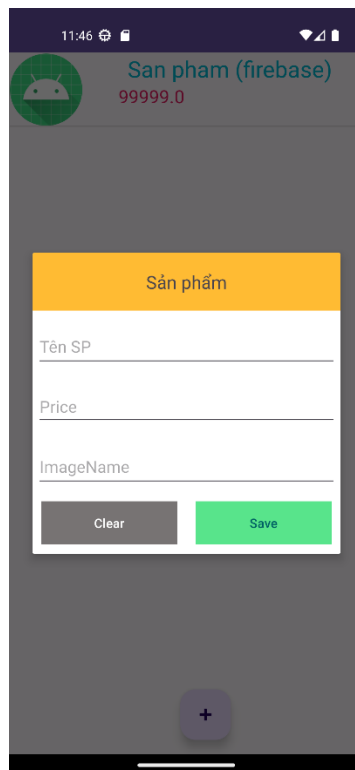
Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Biết cách setup một project Firebase Cloud Firestore
- ✓ Hiểu rõ về CRUD của Firebase Cloud Firestore.

NỘI DUNG :

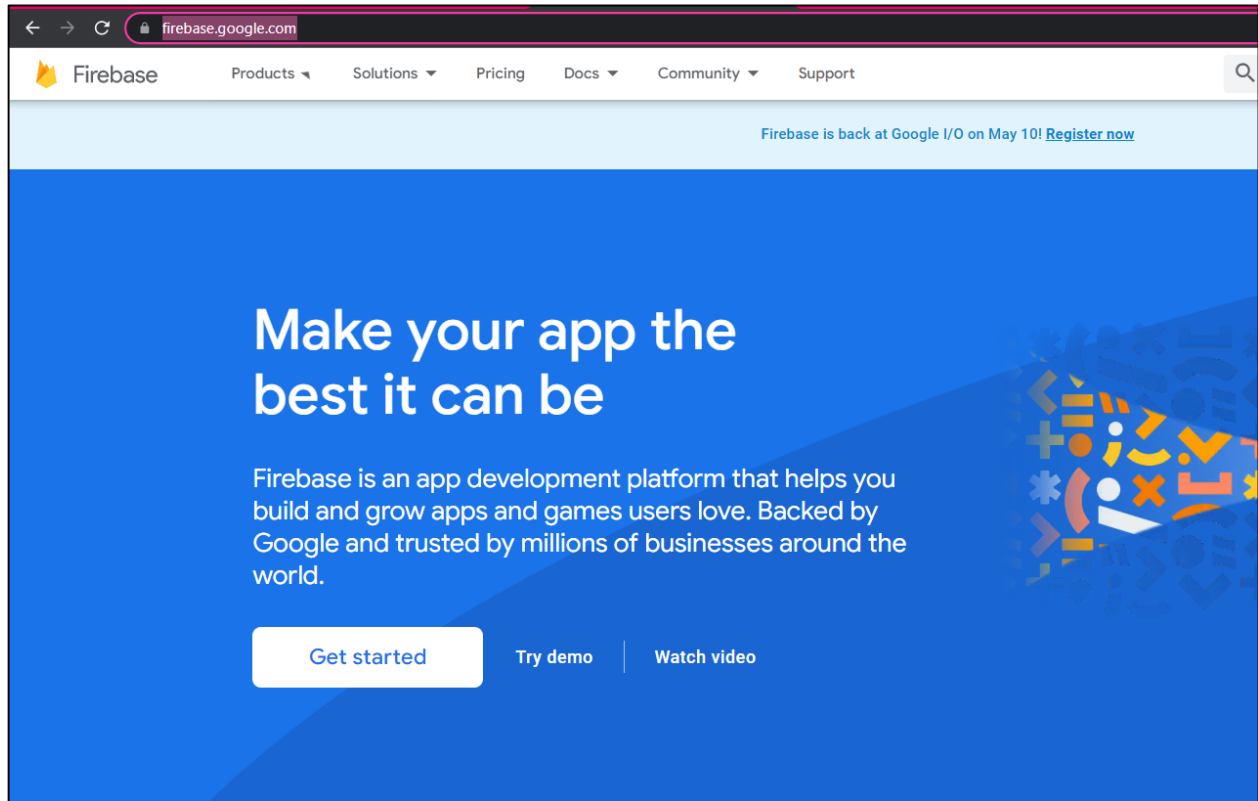
Tạo một ứng dụng **QLSP** với database bằng **Firebase Cloud Firestore**

Sử dụng lại các layout từ Project Lab06 để thực hiện ☺

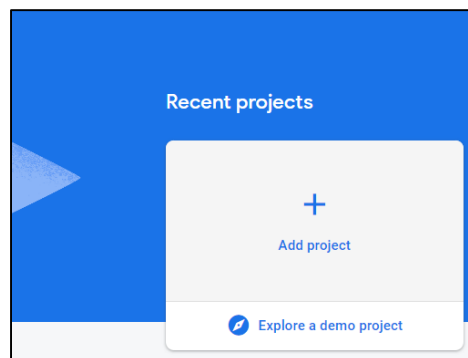


HƯỚNG DẪN P1- Setup Project Firebase Cloud Firestore

Bước 1: Truy cập vào <https://firebase.google.com/> đăng nhập tài khoản. Sau đó chọn **Get started**



Chọn **Add project**



Đặt tên project và nhấn **Continue**

× Create a project (Step 1 of 3)

Let's start with a name for your project[?]

Project name

DemoQLSPFirebase

[Select parent resource](#)

[Continue](#)

Tiếp tục chọn **Continue**

× Create a project (Step 2 of 3)

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

A/B testing[?]

User segmentation & targeting across Firebase products[?]

Crash-free users[?]

Event-based Cloud Functions triggers[?]

Free unlimited reporting[?]

☒ **Enable Google Analytics for this project**
Recommended


[Previous](#) [Continue](#)

Chọn **Default Account for Firebase** và nhấn **Create Project**

× Create a project (Step 3 of 3)

Configure Google Analytics

Choose or create a Google Analytics account ⓘ

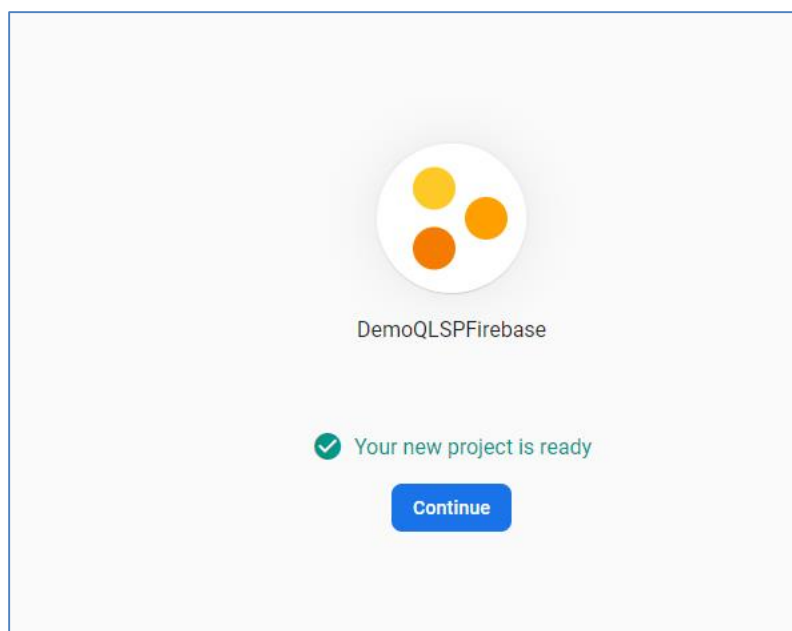
 Default Account for Firebase

Automatically create a new property in this account ✎

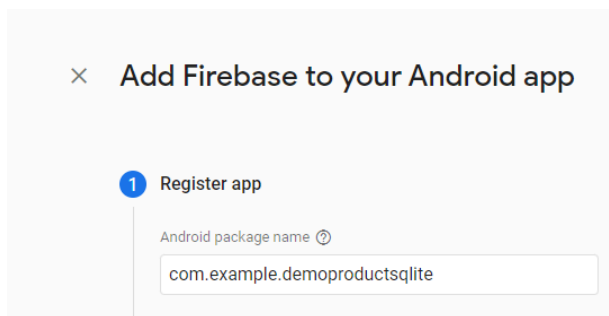
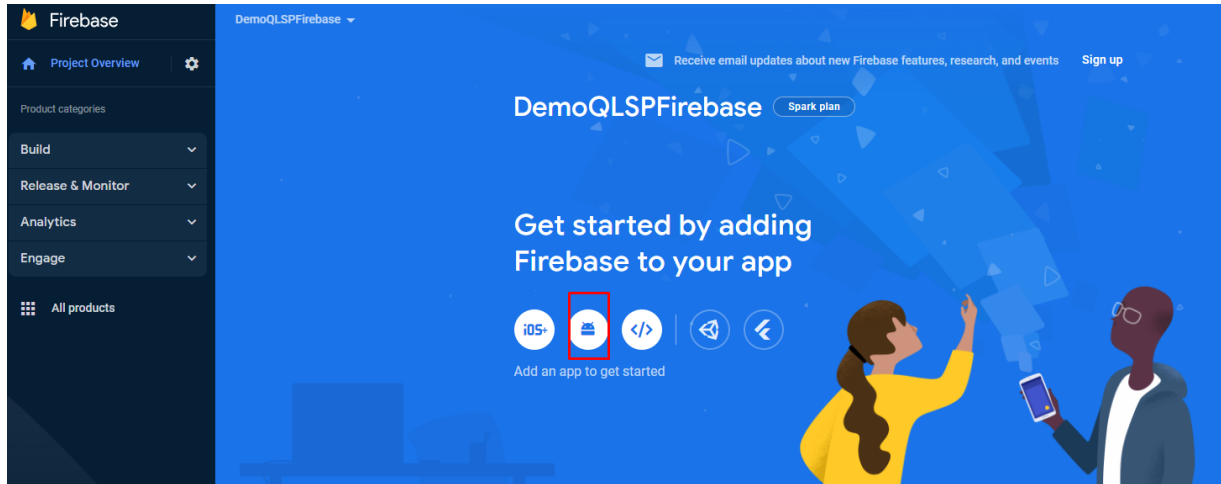
Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more](#) ⓘ

[Previous](#)[Create project](#)

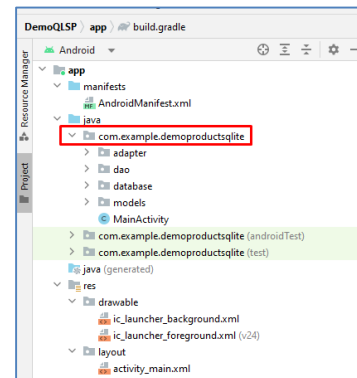
Sau khi xong khi Firebase tạo xong Project, ta chọn **Continue** để bắt đầu sử dụng



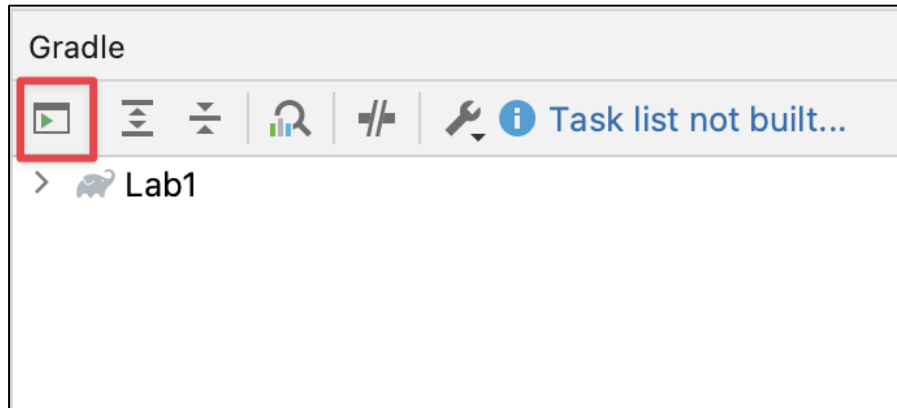
Chọn vào biểu tượng Android để bắt đầu setup



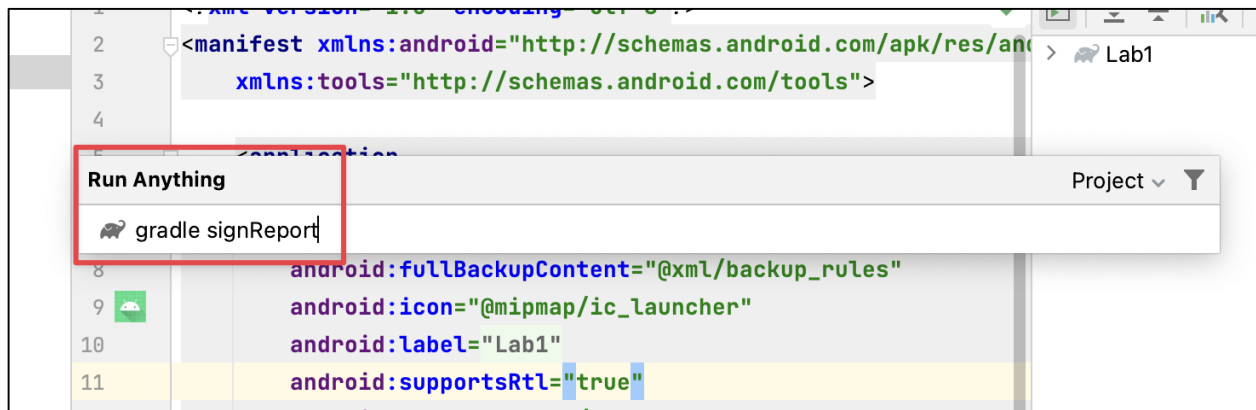
- Điền tên package của app



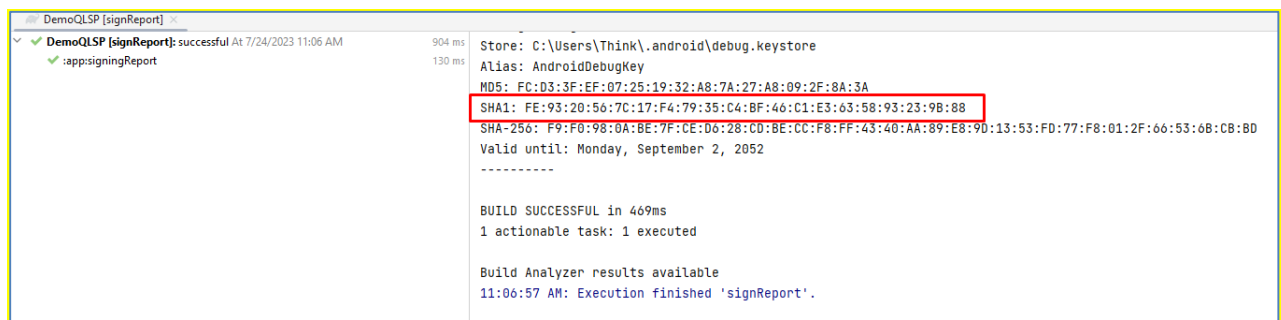
Chọn biểu tượng **Execute Gradle Task**



Gõ vào ô tìm kiếm **signReport** và nhấn enter



Sau đó sẽ có cửa sổ Terminal hiện lên copy SHA-1



Paste SHA-1 vào ô như hình bên dưới, sau đó nhấn tiếp **Register app**

× Add Firebase to your Android app

1 Register app

Android package name ⓘ
com.example.demoproductsqlite

App nickname (optional) ⓘ
DemoLab07Firebase

Debug signing certificate SHA-1 (optional) ⓘ
FE:93:20:56:7C:17:F4:79:35:C4:BF:...

ⓘ Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

Bước 2: Click vào nút “Download google-services.json” để tải file config về máy

2 Download and then add config file

Instructions for Android Studio below | [Unity](#) [C++](#)

Download google-services.json

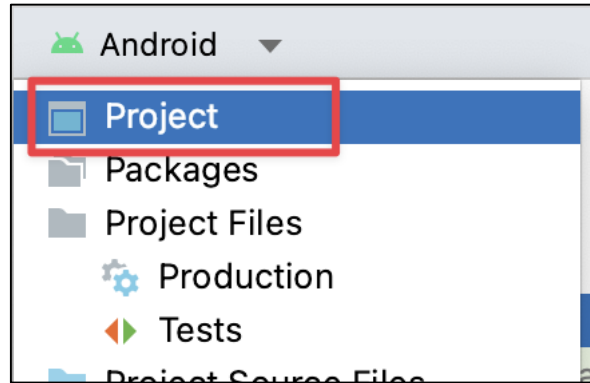
Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.

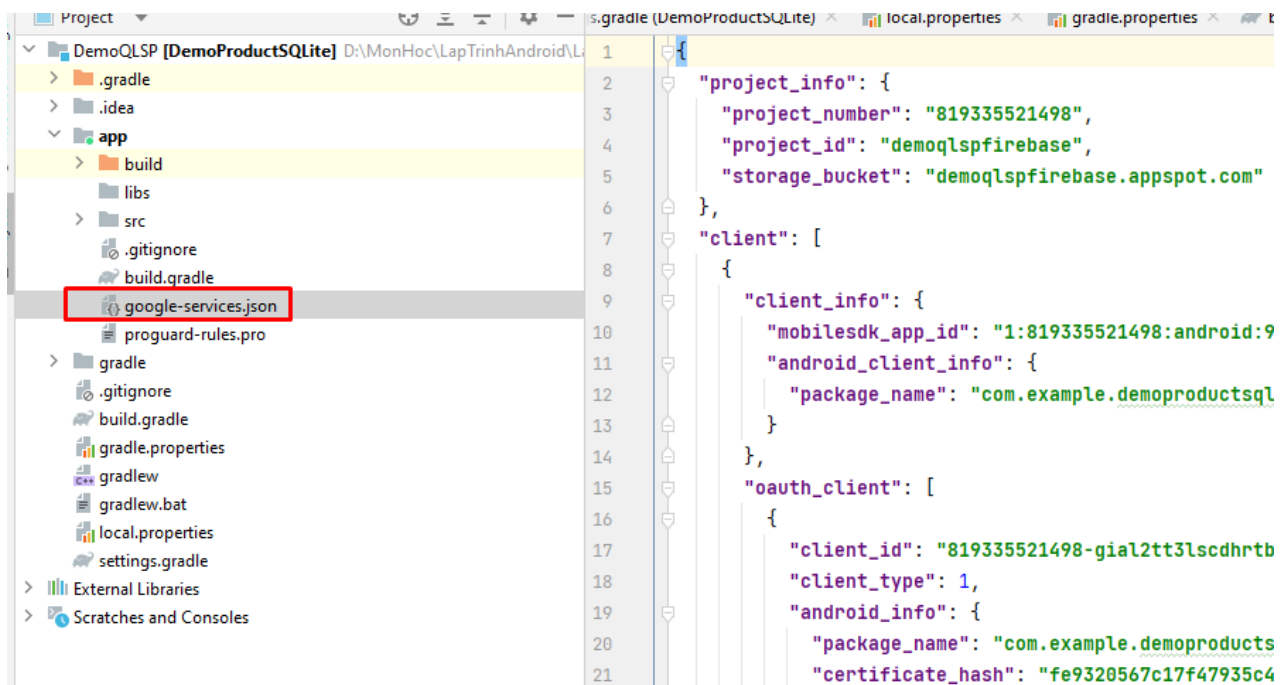
google-services.json

Next

Paste file config vừa tải vào project Android bằng cách thay đổi cấu trúc view của project từ **Android** sang **Project**



Copy file config vừa tải vào thư mục **app**



Bước 3: Thêm các thông số, thư viện cần thiết

3 Add Firebase SDK

Instructions for Gradle | [Unity](#) [C++](#)

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

Add the plugin as a buildscript dependency to your project-level `build.gradle` file:

Root-level (project-level) Gradle file (`<project>/build.gradle`):

```

buildscript {
    repositories {
        // Make sure that you have the following two repositories
        google() // Google's Maven repository
        mavenCentral() // Maven Central repository
    }
    dependencies {
        ...
        // Add the dependency for the Google services Gradle plugin
        classpath 'com.google.gms:google-services:4.3.15'
    }
}

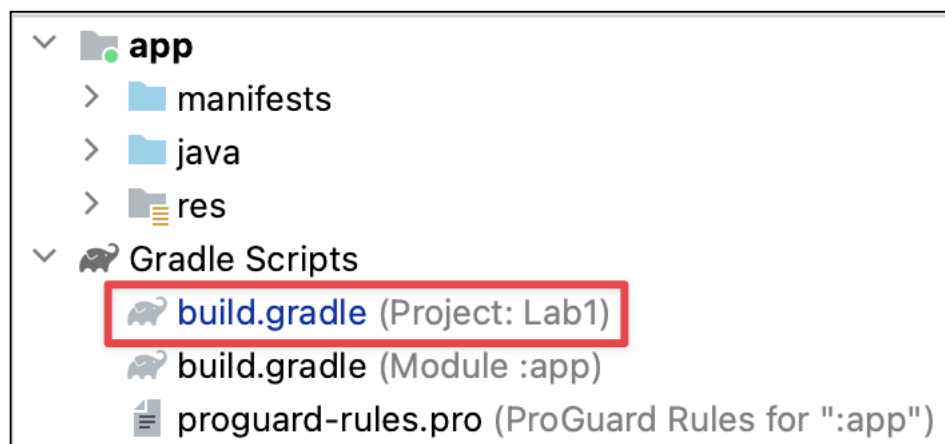
allprojects {
    ...
    repositories {
        // Make sure that you have the following two repositories
        google() // Google's Maven repository
        mavenCentral() // Maven Central repository
    }
}

```

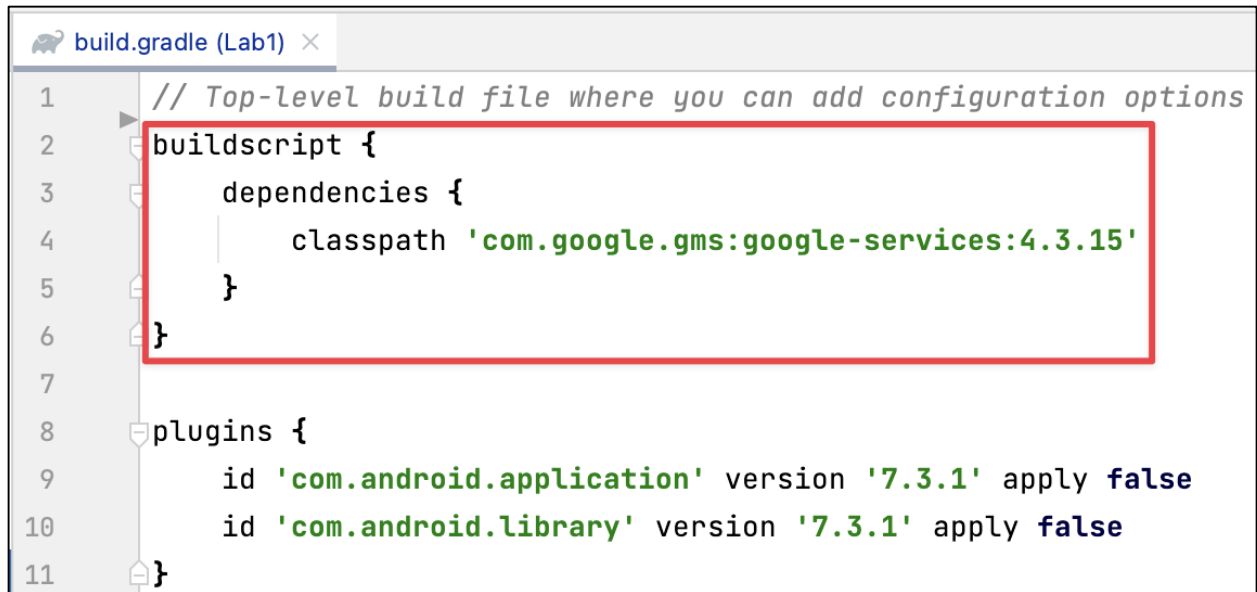
2. Then, in your module (app-level) `build.gradle` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

☐ Kotlin
 ☒ Java

Trong **build.gradle (Module: Project)**

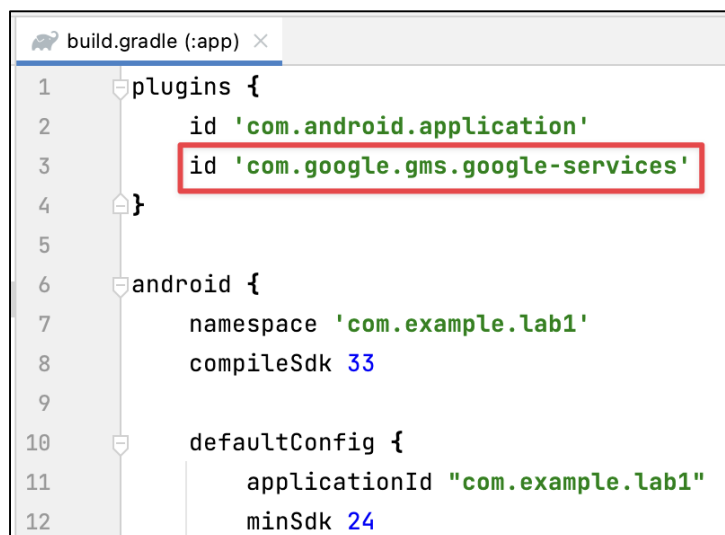
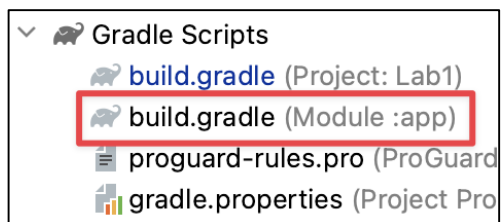


Thêm đoạn code sau (copy ở bước 3 trên Firebase)



```
1 // Top-level build file where you can add configuration options
2 buildscript {
3     dependencies {
4         classpath 'com.google.gms:google-services:4.3.15'
5     }
6 }
7
8 plugins {
9     id 'com.android.application' version '7.3.1' apply false
10    id 'com.android.library' version '7.3.1' apply false
11 }
```

Tiếp theo trong file **build.gradle (Module)**, phần plugins thêm đoạn code:



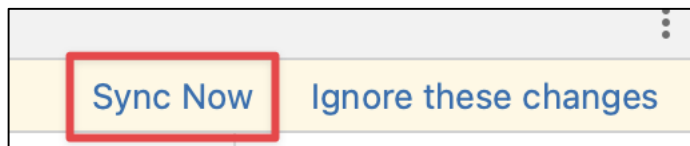
```
1 plugins {
2     id 'com.android.application'
3     id 'com.google.gms.google-services'
4 }
5
6 android {
7     namespace 'com.example.lab1'
8     compileSdk 33
9
10    defaultConfig {
11        applicationId "com.example.lab1"
12        minSdk 24
13    }
14 }
```

Tại **dependencies** trong **build.gradle (Module)** thêm thư viện:

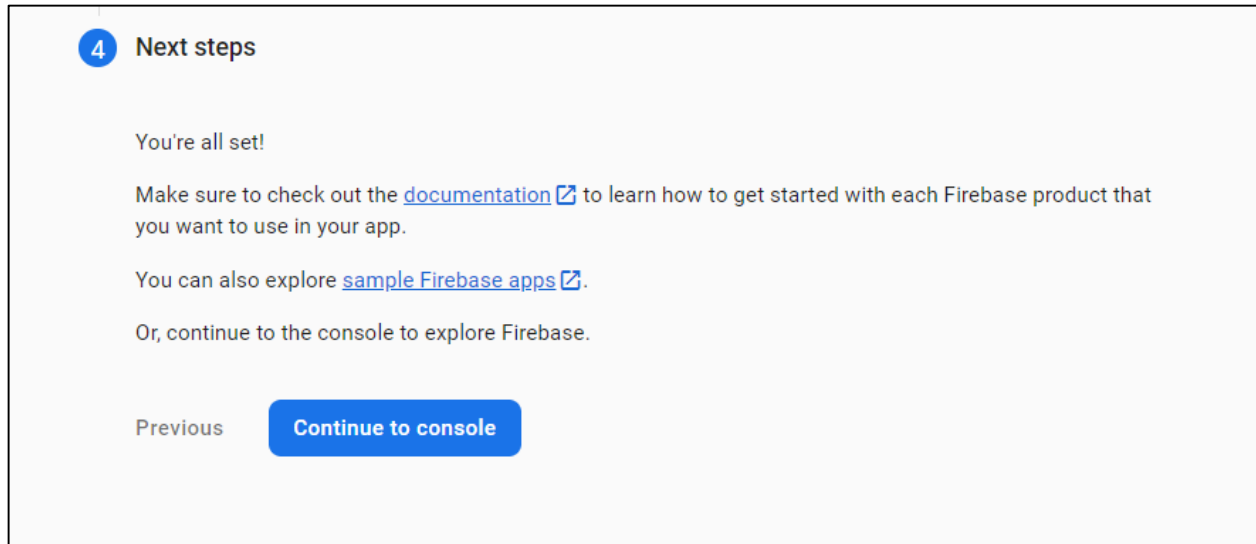
```
dependencies {  
  
    implementation 'androidx.appcompat:appcompat:1.6.1'  
    implementation 'com.google.android.material:material:1.8.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'  
  
    //Firebase  
    implementation platform('com.google.firebase:firebase-bom:31.2.3')  
    implementation 'com.google.firebase:firebase-analytics'  
    //Cloud Firestore  
    implementation 'com.google.firebase:firebase-firestore'  
}
```

```
implementation("com.google.firebase:firebase-firestore")
```

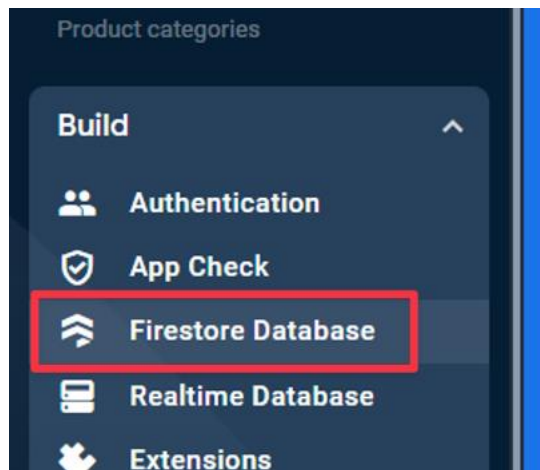
Sau đó nhấn **Sync Now**



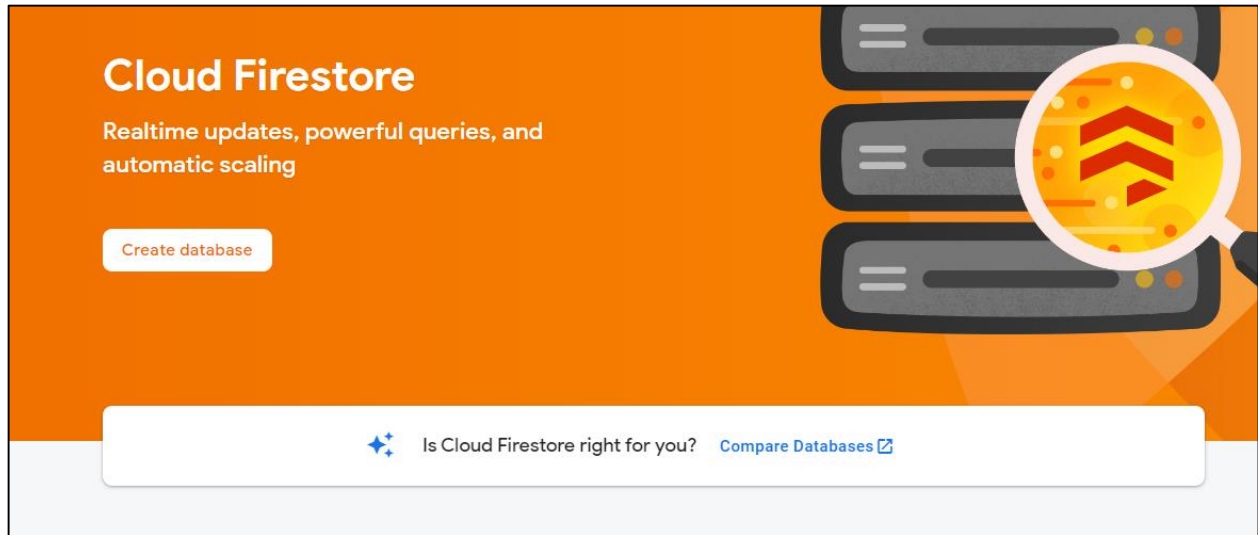
Trên Firebase nhấn **Next** và tiếp tục nhấn **Continue to console**



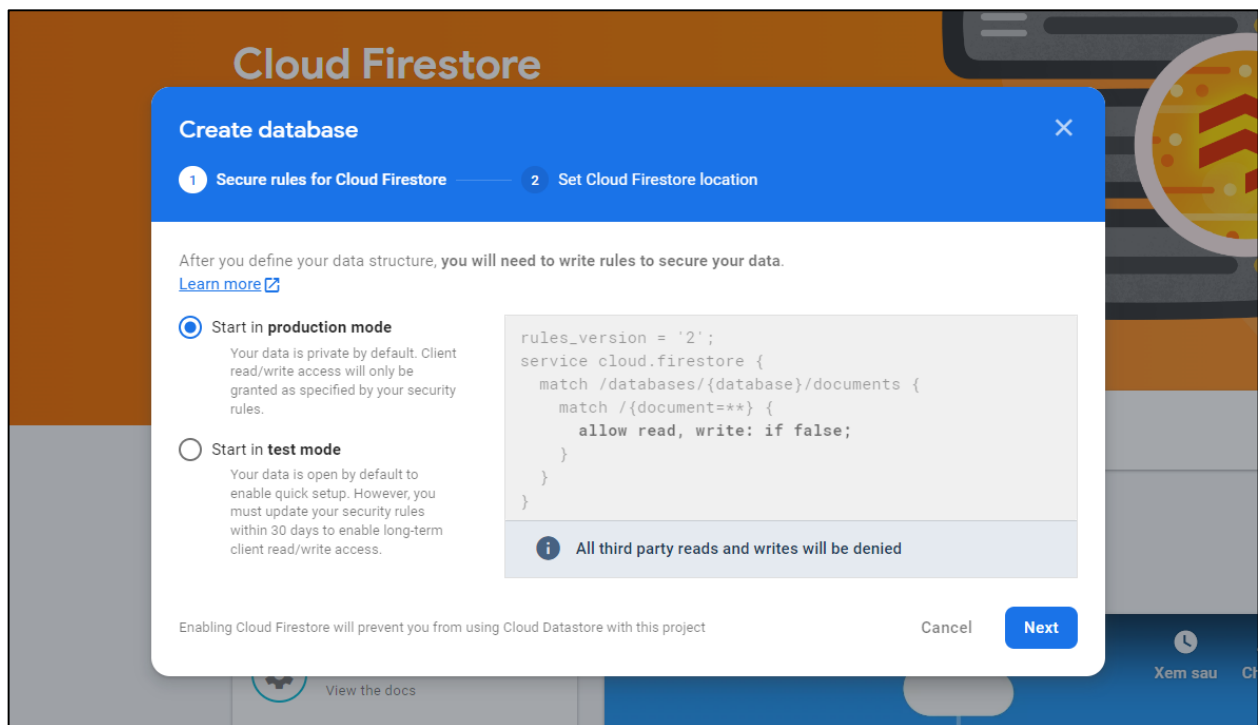
Bước 4: Ở giao diện **Console Firebase** trong mục **Build** chọn **Firestore Database**



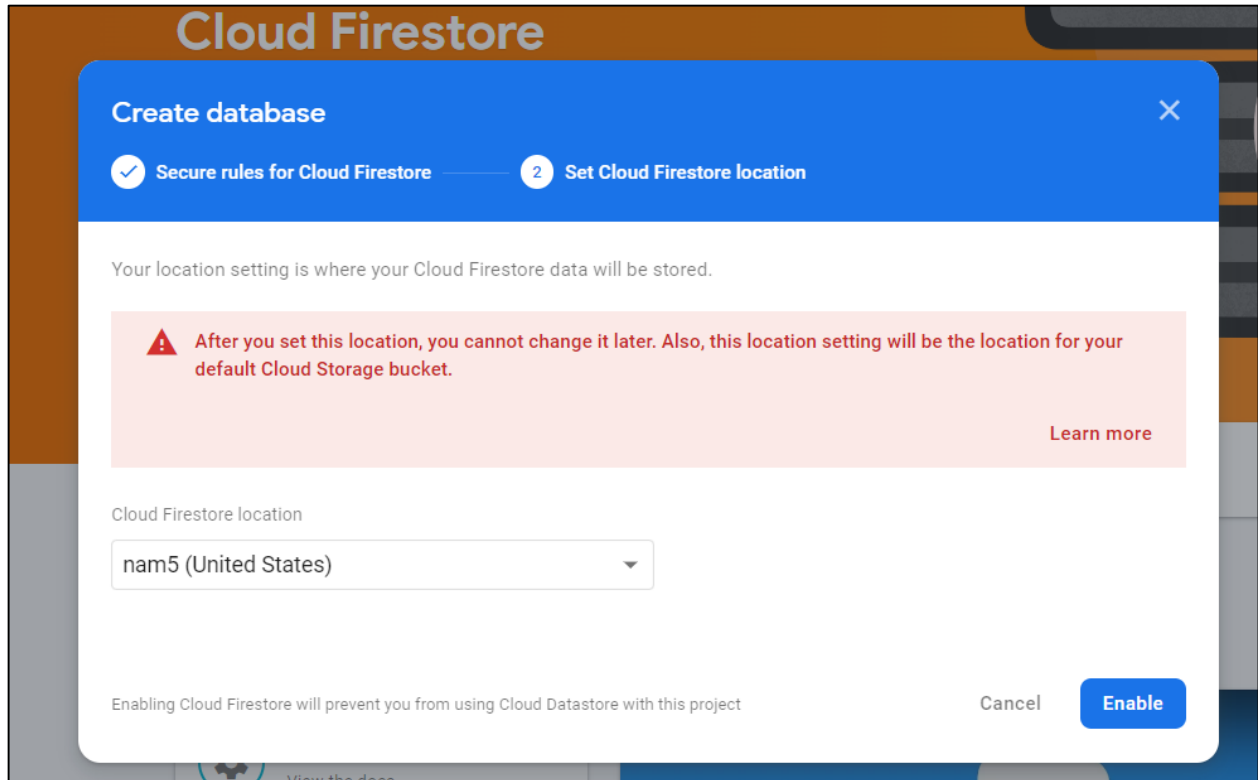
Sau đó chọn **Create database**



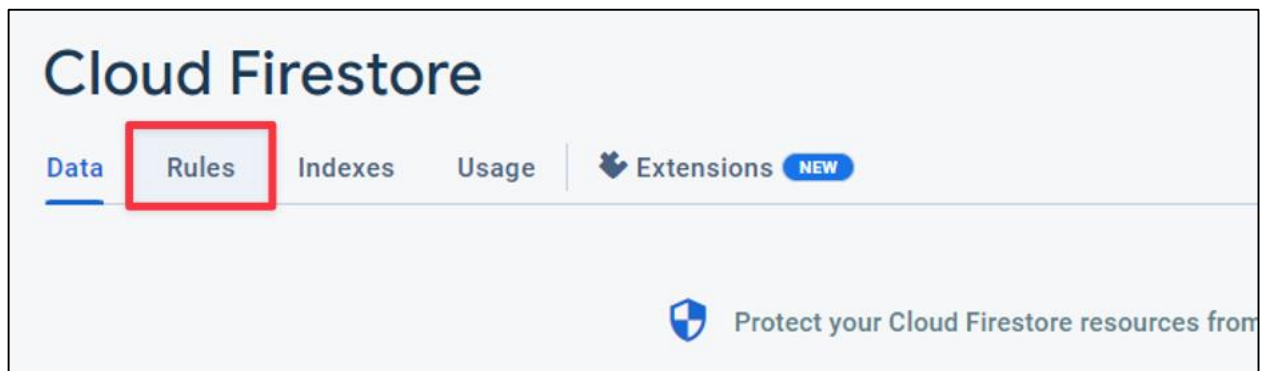
Chọn **production mode** rồi nhấn **Next**



Nhấn chọn location sau đó nhấn **Enable**



Sau đó tại Firestore bạn chọn qua tab **Rules**



Sửa lại nội dung như sau:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if true;
    }
  }
}
```

Ta đã cấu hình xong project firebase với **Firestore database**

PHẦN 2: Thực hiện chức năng thêm sản phẩm bằng cách sử dụng Firestore Database

Bước 1: Với mong muốn tạo **id** của product một cách ngẫu nhiên và tự động, ta thay đổi kiểu dữ liệu của thuộc tính **id** trong model product từ **int** thành **String**

```
public class Product {  
    2 usages  
    private String id;  
    3 usages  
    private String name;  
    3 usages  
    private String image;  
    3 usages  
    private float price;  
}
```

Thay đổi **constructor**, **get**, **set** của model **Todo**, kiểm tra và chỉnh sửa một số đoạn code cho phù hợp sau khi thay đổi kiểu dữ liệu của thuộc tính **id**

Ta sẽ thêm một hàm mới đặt tên là **convertHashMap** trong model **Todo**, hàm này có nhiệm vụ *xử lý dữ liệu thao tác trên Firestore Database*

```
public HashMap<String, Object> convertHashMap() {  
    HashMap<String, Object> work = new HashMap<>();  
    work.put("id", id);  
    work.put("title", title);  
    work.put("content", content);  
    work.put("date", date);  
    work.put("type", type);  
    work.put("status", status);  
    return work;  
}
```

Bước 2: Kết nối với Firebase và tạo tên cho Collection

Giả sử mình có **ProductFirebaseDAO** sẽ cung cấp các hàm cần thiết

```
public class ProductFireBaseDAO {
    FirebaseFirestore db;
    Context context;
    public ProductFireBaseDAO(Context context)
    {
        //kết nối với DB hiện tại
        db = FirebaseFirestore.getInstance();
        this.context = context;
    }
    public void Insert(Product p) {
        // Add a new document with a generated ID
        p.setId(UUID.randomUUID().toString());
        HashMap<String, Object> mapproduct = p.convertHashMap();

        db.collection("Product").document(p.getId())
            .set(mapproduct)
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {
                    Toast.makeText(context, "Thêm mới sp thành công!",
                        Toast.LENGTH_SHORT).show();
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(context, "Thêm mới sp thất bại!",
                        Toast.LENGTH_SHORT).show();
                }
            });
    }

    public void Delete(String productId) {

        db.collection("Product").document(productId)
            .delete()
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {
                    Toast.makeText(context, "Xóa sp thành công!",
                        Toast.LENGTH_SHORT).show();
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(context, " Xóa sp thất bại!",
                        Toast.LENGTH_SHORT).show();
                }
            });
    }
}
```

Có thể sử dụng **ProductFireBaseDAO** ở **Adapter** của **product**

```
ProductFireBaseDAO productFireBaseDao = new ProductFireBaseDAO( context: MainActivity.this);
ProductAdapter adapter = new ProductAdapter(productFireBaseDao, new ArrayList<>());
// lắng nghe thay đổi dữ liệu trên Product ở Firebase
adapter.listenProductFirestore(FirebaseFirestore.getInstance());
```

Và thay đổi Adapter tương ứng

```
public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ViewHolder> {

    11 usages
    private List<Product> listProduct;

    3 usages
    private ProductFireBaseDAO productFireBaseDao;

    1 usage
    public ProductAdapter(ProductFireBaseDAO productFireBaseDao, List<Product> listProduct ) {
        this.listProduct = listProduct;
        this.productFireBaseDao = productFireBaseDao;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {...}

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {...}

    @Override
    public int getItemCount() { return listProduct.size(); }

    1 usage
    public void deleteItem(int pos) {...}

    1 usage
    public void insertItem(Product p) {...}

    1 usage
    public void listenProductFirestore(FirebaseFirestore db) {...}

    4 usages
```

Để thêm mới 1 sản phẩm

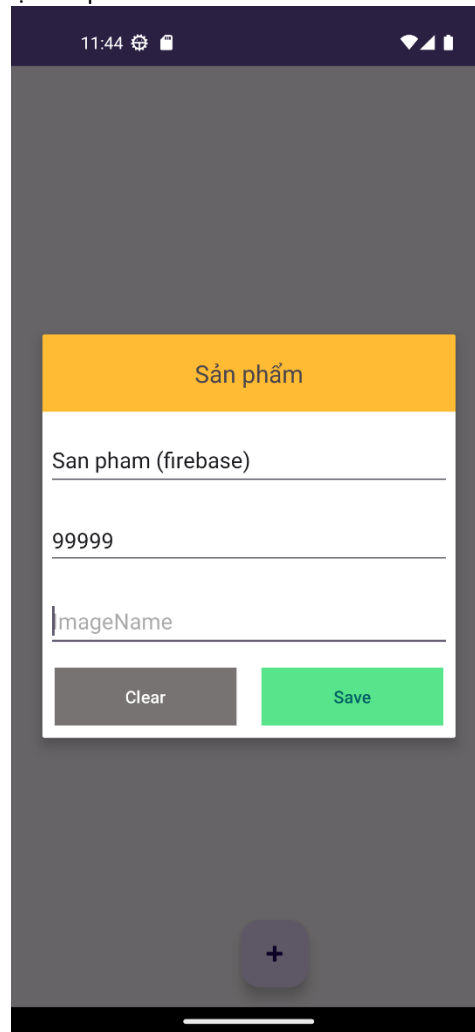
- Thay đổi đoạn code khi Save ở Dialog

```
viewDialog.findViewById(R.id.btnDialogSaveProduct).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Product p = new Product( txtName.getText().toString(),
                                txtImage.getText().toString(),
                                Float.parseFloat(txtprice.getText().toString()));
        adapter.insertItem(p);
        alert.dismiss();
    }
});
```

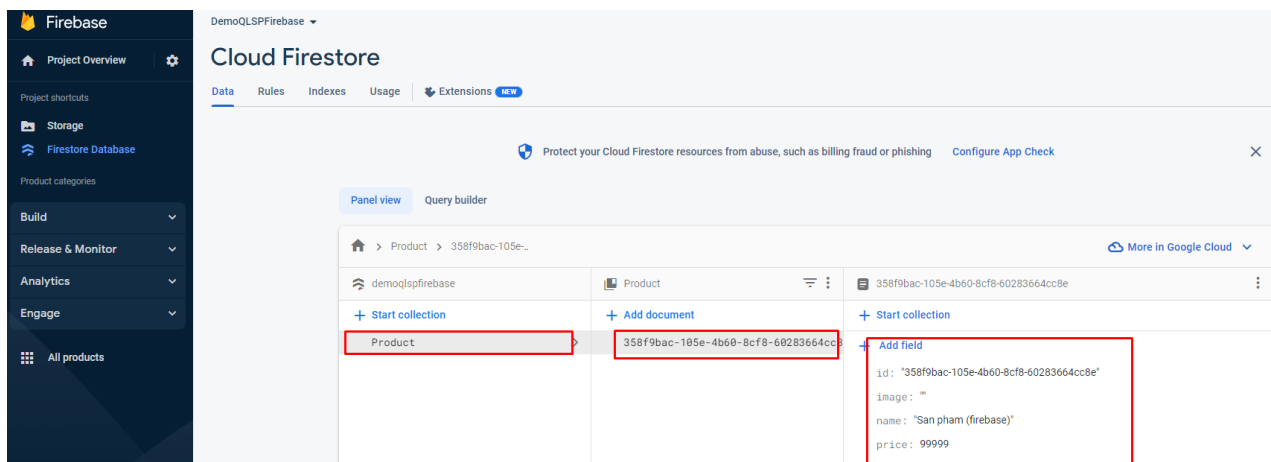
- Thực hiện việc insertItem tại Adapter

```
9 usages
public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ViewHolder> {
    1 usage
    public void deleteItem(int pos) {...}
    1 usage
    public void insertItem(Product p) {
        // listProduct.add(p);
        productFireBaseDao.Insert(p);
    }
    11 usages
```

Chạy lại project và thực hiện thêm một sản phẩm mới



Sau khi thêm thành công, ta tiến hành kiểm tra **FirebaseFirestore** trên Firebase, ta được kết quả như hình



PHẦN 3: Thực hiện chức năng lắng nghe khi dữ liệu thay đổi trên FirebaseFirestore

Tạo một hàm mới tên **ListenFirestore**, trong hàm này sẽ lắng nghe sự thay đổi dữ liệu trên FirebaseFirestore (Thêm, Sửa, Xóa). Có thể đặt hàm này ở **Adapter**

```
public void listenProductFirestore(FirebaseFirestore db) {
    db.collection("Product").addSnapshotListener(new
    EventListener<QuerySnapshot>() {
        @Override
        public void onEvent(@Nullable QuerySnapshot snapshots, @Nullable
        FirebaseFirestoreException e) {
            if (e != null) {
                Log.w("FireBase", "listen:error", e);
                return;
            }
            for (DocumentChange dc : snapshots.getDocumentChanges()) {
                Product product =
                dc.getDocument().toObject(Product.class);
                switch (dc.getType()) {
                    case ADDED:
                        listProduct.add(product);
                        notifyItemInserted(listProduct.size() - 1);
                        break;

                    case MODIFIED:
                        if (dc.getOldIndex() == dc.getNewIndex()) //nếu
                        vị trí đối tượng tương đồng với vị trí mới
                        {
                            listProduct.set(dc.getOldIndex(), product);
                            notifyItemChanged(dc.getOldIndex());
                        } else //khác vị trí sẽ xóa đối tượng ở danh
                        sách và thêm lại
                        {
                            listProduct.remove(dc.getOldIndex());
                            listProduct.add(dc.getNewIndex(), product);
                            notifyItemMoved(dc.getOldIndex(),
                            dc.getNewIndex());
                        }
                        break;
                    case REMOVED:
                        listProduct.remove(dc.getOldIndex());
                        notifyItemRemoved(dc.getOldIndex());
                        break;
                }
            }
        }
    });
}
```

Gọi hàm **ListenFirestore()** trong **onCreate()**

```
4 usages
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ProductFireBaseDAO productFireBaseDao = new ProductFireBaseDAO(context: MainActivity.this);
        ProductAdapter adapter = new ProductAdapter(productFireBaseDao, new ArrayList<>());
        // lắng nghe thay đổi dữ liệu trên Product ở Firebase
        adapter.listenProductFirestore(FirebaseFirestore.getInstance());
    }
}
```

Kể từ bây giờ, khi bạn thực hiện thay đổi dữ liệu trên **Firestore**, dữ liệu bên dưới ứng dụng sẽ tự động cập nhật

PHẦN 4: Thực hiện chức năng CẬP NHẬT và XÓA Sản phẩm

--- Hết ---