



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

Lập trình Android

Bài 31: *Google Firebase*

Phòng LT & Mạng

<http://csc.edu.vn/lap-trinh-va-csdl>





Nội dung

1. Giới thiệu Firebase
2. Các dịch vụ của Firebase
3. Cấu hình và sử dụng Firebase Console
4. Firebase Authentication
5. Firebase Cloud Firestore
6. Firebase Cloud Storage



Giới thiệu Firebase

- **Firestore** là một dịch vụ hệ thống backend (Backend as a Service - BaaS) được cung cấp bởi Google
- Firestore giúp rút ngắn thời gian phát triển các ứng dụng di động, web
- Hỗ trợ Android, iOS, C++, Web apps, REST API, Unity...
- Được sử dụng bởi The New York Times, Shazam, Duolingo, Alibaba.com, Trivago...



Các dịch vụ của Firebase

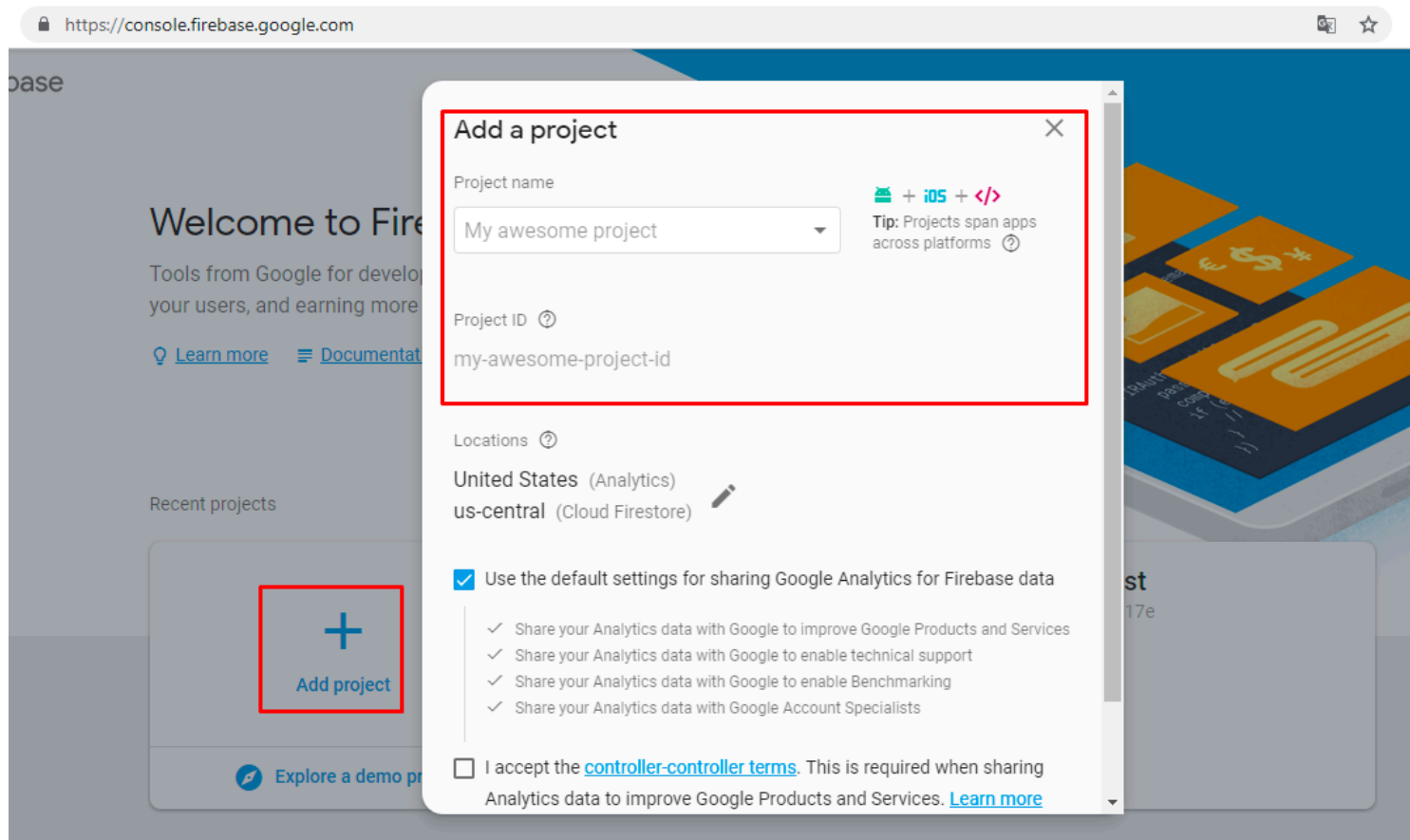
- Các dịch vụ của **Firebase** được chia thành 3 mục chính:
 - **Develop** – giúp xây dựng các ứng dụng, gồm các thành phần tiêu biểu:
 - **Cloud Firestore** – Lưu và đồng bộ CSDL trên hạ tầng cloud
 - **Authentication** – Quản lý người dùng, hỗ trợ nhiều phương thức xác thực: email/password, Google, Facebook, SMS...
 - **Cloud Storage** – Lưu trữ và chia sẻ tập tin
 - **Quality** – giúp nâng cao chất lượng ứng dụng, gồm các thành phần:
 - **Crashlytics** – theo dõi crash trên ứng dụng, thông báo theo thời gian thực
 - **Performance Monitoring** – theo dõi và chuẩn đoán hiệu năng của ứng dụng
 - **Test Lab** – chạy thử nghiệm trên thiết bị di động ảo/vật lý của Google, đặt trên cloud
 - **Grow** – gồm các thành phần tiêu biểu:
 - **Cloud Messaging** – gửi tin nhắn và thông báo (notifications), đã được trình bày tại bài Notifications
 - **AdMob** – quảng cáo tích hợp trong ứng dụng di động Android, iOS
 - **Google Analytics** – theo dõi và phân tích chi tiết cách thức người dùng ứng dụng





Cấu hình và sử dụng Firebase Console

- **Firebase Console:** <https://console.firebase.google.com>
- Đăng nhập với tài khoản Google → Tạo project mới → Tải về file `google-services.json` và copy vào thư mục module của project Android:





Cấu hình và sử dụng Firebase Console (2)

❑ Cấu hình **build.gradle** của **project**:

```
buildscript {  
  
    repositories {  
        // đảm bảo có Google's Maven repository  
        google()  
    }  
  
    dependencies {  
        // ...  
        // thêm Google Services plugin  
        classpath 'com.google.gms:google-services:4.3.1'  
    }  
}  
  
allprojects {  
    // ...  
    repositories {  
        // thêm Google's Maven repository  
        google()  
        // ...  
    }  
}
```

❑ Cấu hình **build.gradle** của **module**:

```
apply plugin: 'com.android.application'  
  
android {  
    // ...  
}  
  
// thêm Google Play services Gradle plugin  
apply plugin: 'com.google.gms.google-services'
```




❑ Khởi tạo **FirebaseApp** (có thể khởi tạo trong class Application hoặc Activity):

```
FirebaseApp.initializeApp(context);
```



Firebase Authentication

- Hầu hết các ứng dụng đều cần đến tính năng xác thực người dùng
- **Firebase Authentication** cung cấp dịch vụ backend, cho phép xác thực người dùng ứng dụng
- Hỗ trợ nhiều hình thức xác thực: Email/Password; Số điện thoại (SMS); Các dịch vụ bên thứ 3: Google, Facebook, Twitter
- Cần cấu hình hình thức xác thực trên **Firebase Console**
 - Develop → Authentication → Sign-in method → chọn enable hình thức đăng nhập mà ứng dụng hỗ trợ

Authentication	
Users	Sign-in method
Sign-in providers	
Provider	Status
 Email/Password	Enabled
 Phone	Disabled
 Google	Disabled





Firebase Authentication (2)

- Minh họa sau chỉ dẫn các bước xác thực dựa trên **email/password**:
 - Cấu hình **build.gradle** của **module**:
- implementation 'com.google.firebase:firebase-auth:19.0.0'
- Tạo đối tượng **FirebaseAuth** trong phương thức **onCreate()** của Activity:

// Khai báo đối tượng FirebaseAuth

```
private FirebaseAuth mAuth;
```

// ...

// Tạo đối tượng Firebase Auth

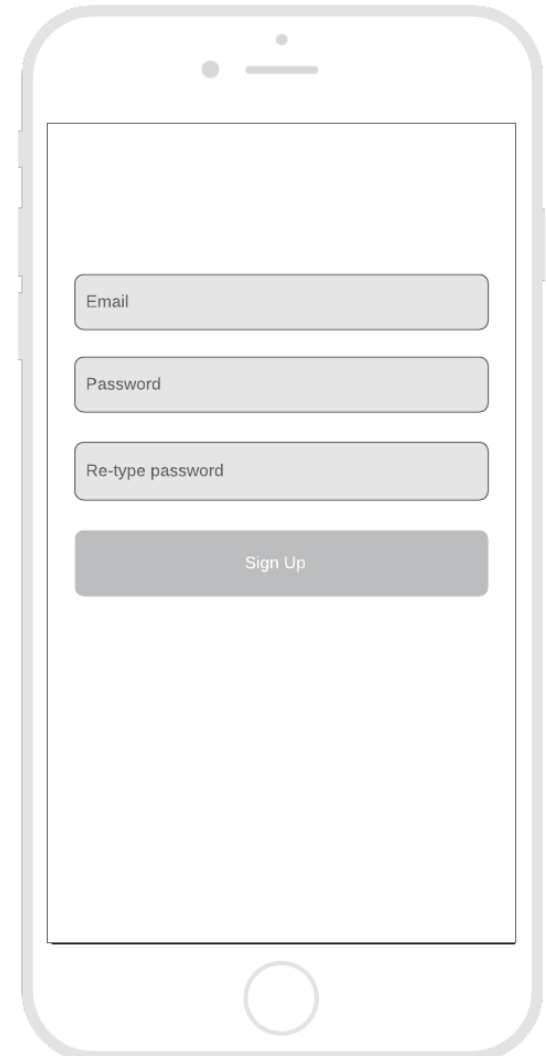
```
mAuth = FirebaseAuth.getInstance();
```




Firebase Authentication (3)

- (1) Tạo tài khoản sử dụng email/password: đăng ký tài khoản với phương thức `createUserWithEmailAndPassword()`:

```
mAuth.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                // Đăng nhập thành công, cập nhật UI với thông tin của người dùng
                Log.d(TAG, "createUserWithEmail:success");
                FirebaseUser user = mAuth.getCurrentUser();
                updateUI(user);
            } else {
                // Đăng nhập lỗi, hiển thị thông báo đến người dùng
                Log.w(TAG, "createUserWithEmail:failure", task.getException());
                Toast.makeText(EmailPasswordActivity.this, "Authentication failed.",
                    Toast.LENGTH_SHORT).show();
                updateUI(null);
            }
        }
    });
```



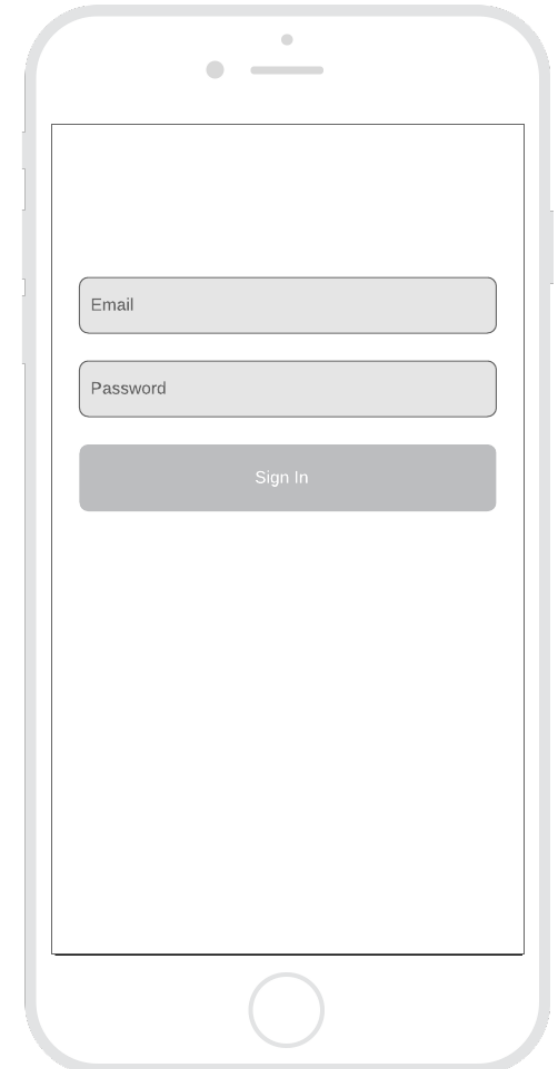


Firebase Authentication (4)

- (2) Đăng nhập sử dụng email/password: sử dụng phương thức `signInWithEmailAndPassword()`:

```
mAuth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                // đăng nhập thành công, cập nhật UI với thông tin của người dùng
                Log.d(TAG, "signInWithEmail:success");
                FirebaseUser user = mAuth.getCurrentUser();
                updateUI(user);
            } else {
                // nếu đăng nhập thất bại, thông báo lỗi đến người dùng
                Log.w(TAG, "signInWithEmail:failure", task.getException());
                Toast.makeText(EmailPasswordActivity.this, "Authentication failed.",
                    Toast.LENGTH_SHORT).show();
                updateUI(null);
            }

            // ...
        }
    });
```





Firebase Authentication (5)

- **(3) Ghi nhớ đăng nhập:** Firebase hỗ trợ tính năng ghi nhớ đăng nhập, nếu người dùng đã đăng nhập rồi, có thể kiểm tra theo cách sau:

- Trong phương thức **onStart()**, kiểm tra user đã đăng nhập chưa (khác **null**) và cập nhật UI:

@Override

```
public void onStart() {  
    super.onStart();  
    FirebaseAuth currentUser = mAuth.getCurrentUser();  
    updateUI(currentUser);  
}
```

- **(4) Đăng xuất:**

```
FirebaseAuth.getInstance().signOut();
```

- Quản lý danh sách user tại Firebase Console: Develop → Authentication → tab Users

Search by email address, phone number, or user UID					Add user	↺	⋮
Identifier	Providers	Created	Signed In	User UID ↑			
minh@le.com	✉	Sep 1, 2019	Sep 1, 2019	IX4umJjgcQWd1HABoqZaber4Qk52			
					Rows per page: 50 ▼	1-1 of 1	< >



Firestore Cloud Firestore – Dữ liệu


- **Cloud Firestore** - Lưu và đồng bộ CSDL NoSQL trên hạ tầng cloud
- Là CSDL hướng tài liệu, có cấu trúc lưu trữ tương tự dữ liệu JSON
- Khai báo sử dụng **Firestore** trong **build.gradle** của **module**:
implementation 'com.google.firebase:firebase-firestore:21.0.0'
- Khởi tạo đối tượng Firestore (trong Activity/Fragment):

```
FirebaseFirestore db = FirebaseFirestore.getInstance();
```



Firestore Cloud Firestore – Dữ liệu

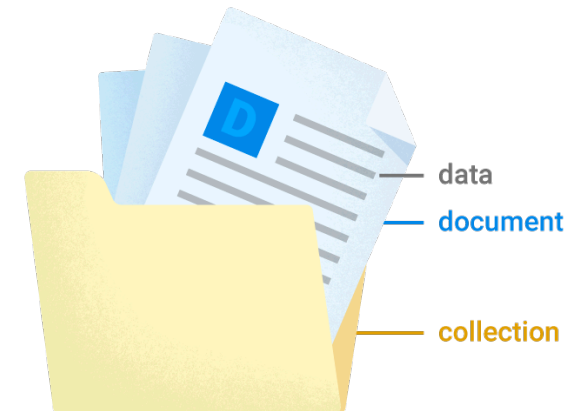
- Đơn vị lưu trữ cơ bản của **Firestore** là **Document**
 - Mỗi **document** chứa các **fields**, mỗi **field** chứa một cặp **key-value**
 - Mỗi **document** được xác định bởi tên (**name**)
 - Một **field** có thể chứa dữ liệu: Boolean, số, chuỗi, tọa độ địa lý, dữ liệu nhị phân, timestamp

 alovelace


```
first : "Ada"
```

```
last  : "Lovelace"
```

```
born  : 1815
```



- Một field bên trong document, có thể gồm các field con, gọi là **map**

 alovelace

```
name :
```

```
  first : "Ada"
```


```
  last  : "Lovelace"
```


```
born  : 1815
```



Firestore Cloud Firestore - Dữ liệu (2)

- ❑ **Document** sống bên trong **Collection**, hay một **Collection** chứa các **Document**


 users

 alovelace

first : "Ada"

last : "Lovelace"

born : 1815


 aturing


first : "Alan"

last : "Turing"


born : 1912


- ❑ Bên trong một **document** có thể chứa **collection**, gọi là **sub-collection**

 rooms

 roomA


name : "my chat room"

 messages


 message1

from : "alex"

msg : "Hello World!"

 message2

...

 roomB

...

Lưu ý: Một collection không chứa các collections khác
Một document không chứa các document khác



Firestore Cloud Firestore – Ghi dữ liệu

- Tạo mới hoặc ghi đè một **document**, bên trong một **collection**:

```
Map<String, Object> city = new HashMap<>(); // mỗi data của document là một thành phần của kiểu dữ liệu Map
city.put("name", "Los Angeles");
city.put("state", "CA");
city.put("country", "USA");
```

```
db.collection("cities").document("LA").set(city) // gọi hàm set() để thiết lập data cho document
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            Log.d(TAG, "DocumentSnapshot successfully written!");
        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Log.w(TAG, "Error writing document", e);
        }
    });
```

🏠 > cities > LA		
📁 fcm-test-dbb0b	📁 cities	📁 LA
+ Start collection	+ Add document	+ Start collection
cities >	LA >	+ Add field
		country: "USA"
		name: "Los Angeles"
		state: "CA"



Firestore Cloud Firestore – Ghi dữ liệu (2)

- Cập nhật một **field** của **document**, hoặc tạo **document** nếu nó chưa tồn tại:

```
Map<String, Object> data = new HashMap<>();  
data.put("capital", true);
```

```
db.collection("cities").document("BJ").set(data, SetOptions.merge());
```

🏠 > cities > BJ		
⌵ fcm-test-dbb0b	📁 cities ⌵ ⋮	📁 BJ ⌵ ⋮
+ Start collection	+ Add document	+ Start collection
cities >	BJ > LA	+ Add field capital: true



Firestore Cloud Firestore – Ghi dữ liệu (3)

- Thêm một đối tượng (**Java object**) vào **document**. Thay vì dùng kiểu dữ liệu **Map** để gom nhóm các data, sử dụng một đối tượng **Model** để nắm giữ thông tin của **document** đó:

```
public class City {  
    public String name;  
    public String state;  
    public String country;  
    public boolean capital;  
    public long population;  
    public List<String> regions;  
  
    public City() {  
  
    }  
}
```

fcm-test-dbb0b	cities	LA
+ Start collection	+ Add document	+ Start collection
cities >	BJ	+ Add field
	LA >	capital: false
		country: "USA"
		name: "Los Angeles"
		population: 5000000
		▼ regions
		0 "west_coast"
		1 "sorcal"
		state: "CA"

```
public City(String name, String state, String country, boolean capital, long population, List<String> regions) {  
    // ...  
}  
}
```

```
City city = new City("Los Angeles", "CA", "USA", false, 5000000L, Arrays.asList("west_coast", "sorcal"));  
db.collection("cities").document("LA").set(city);
```





Firestore Cloud Firestore – Ghi dữ liệu (4)

- Có thể chỉ định rõ **document ID** khi tạo mới một **document**:

```
db.collection("cities").document("new-city-id").set(data);
```

- hoặc để **Firestore** tự động phát sinh **ID** cho **document**:

```
Map<String, Object> data = new HashMap<>();  
data.put("name", "Tokyo");  
data.put("country", "Japan");
```

```
db.collection("cities")  
    .add(data)  
    .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {  
        @Override  
        public void onSuccess(DocumentReference documentReference) {  
            Log.d(TAG, "DocumentSnapshot written with ID: " + documentReference.getId());  
        }  
    })  
    .addOnFailureListener(new OnFailureListener() {  
        @Override  
        public void onFailure(@NonNull Exception e) {  
            Log.w(TAG, "Error adding document", e);  
        }  
    });
```





Firestore Cloud Firestore – Ghi dữ liệu (5)

- Đọc một **document** và cập nhật một vài **fields** của **document** đó:

```
DocumentReference washingtonRef = db.collection("cities").document("DC");
```

```
// cập nhật field "capital" của document "DC"
```

```
washingtonRef
```

```
.update("capital", true)
.addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        Log.d(TAG, "DocumentSnapshot successfully updated!");
    }
})
.addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Log.w(TAG, "Error updating document", e);
    }
});
```



Firestore Cloud Firestore – Xoá dữ liệu

- Để xoá một **document**, dùng phương thức **delete()**:

```
db.collection("cities").document("DC")
    .delete()
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            Log.d(TAG, "DocumentSnapshot successfully deleted!");
        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Log.w(TAG, "Error deleting document", e);
        }
    });
```



Firestore Cloud Firestore – Xóa dữ liệu (2)

- Để xóa một vài field từ document, dùng phương thức `FieldValue.delete()`:

```
DocumentReference docRef = db.collection("cities").document("BJ");
```

```
// Remove the 'capital' field from the document
```

```
Map<String, Object> updates = new HashMap<>();
```

```
updates.put("capital", FieldValue.delete());
```

```
docRef.update(updates).addOnCompleteListener(new OnCompleteListener<Void>() {
```

```
    // ...
```

```
    // ...
```

- Để xóa một collection, cần thực hiện xóa trước tất cả các document bên trong collection đó
- Xóa một collection không được khuyến khích thực hiện từ Android client, thay vào đó nên sử dụng Firebase CLI



Firestore Cloud Firestore – Đọc dữ liệu

- Để đọc một **document**, sử dụng phương thức **get()**, dữ liệu trả về dưới dạng **HashMap**:

```
DocumentReference docRef = db.collection("cities").document("SF");
docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task.isSuccessful()) {
            DocumentSnapshot document = task.getResult();
            if (document.exists()) {
                Log.d(TAG, "DocumentSnapshot data: " + document.getData());
            } else {
                Log.d(TAG, "No such document");
            }
        } else {
            Log.d(TAG, "get failed with ", task.getException());
        }
    }
});
```

- Có thể đọc và trả về đối tượng Java:

```
DocumentReference docRef = db.collection("cities").document("BJ");
docRef.get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
    @Override
    public void onSuccess(DocumentSnapshot documentSnapshot) {
        City city = documentSnapshot.toObject(City.class);
    }
});
```



Firestore Cloud Firestore – Đọc dữ liệu (2)

- Đọc tất cả các **documents** bên trong một **collection**:

```
db.collection("cities")
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    Log.d(TAG, document.getId() + " => " + document.getData());
                }
            } else {
                Log.d(TAG, "Error getting documents: ", task.getException());
            }
        }
    });
```



Firestore Cloud Firestore – Đọc dữ liệu (3)

- Đọc tất cả các documents bên trong một collection, sử dụng phương thức **where()** để lọc dữ liệu:

```
db.collection("cities")
    .whereEqualTo("capital", true)
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    Log.d(TAG, document.getId() + " => " + document.getData());
                }
            } else {
                Log.d(TAG, "Error getting documents: ", task.getException());
            }
        }
    });
```

- Một số phương thức **where()** khác: **whereLessThan()**, **whereGreaterThan()**, **whereGreaterThanOrEqualTo()**, **whereLessThanOrEqualTo()**, **whereArrayContains()** ...



Firestore Cloud Firestore – Đọc dữ liệu (4)

- Có thể đọc dữ liệu có sắp xếp tăng dần và hạn chế số lượng → VD: lấy 3 thành phố đầu trong danh sách được sắp xếp theo thứ tự tên tăng dần:

```
citiesRef.orderBy("name").limit(3);
```

- Hoặc đọc dữ liệu có sự sắp xếp giảm dần và hạn chế số lượng → VD: lấy 3 thành phố đầu trong danh sách sắp xếp theo tên giảm dần:

```
citiesRef.orderBy("name", Direction.DESCENDING).limit(3);
```

- Có thể kết hợp sắp xếp theo nhiều fields:

```
citiesRef.orderBy("state").orderBy("population", Direction.DESCENDING);
```

- Có thể kết hợp các phương thức where(), orderBy() và limit():

```
citiesRef.whereGreaterThan("population", 100000).orderBy("population").limit(2);
```

- **Chú ý:** where() và orderBy() phải thực hiện trên cùng một field

```
citiesRef.whereGreaterThan("population", 100000).orderBy("population");
```



```
citiesRef.whereGreaterThan("population", 100000).orderBy("country");
```





Firebase Cloud Storage

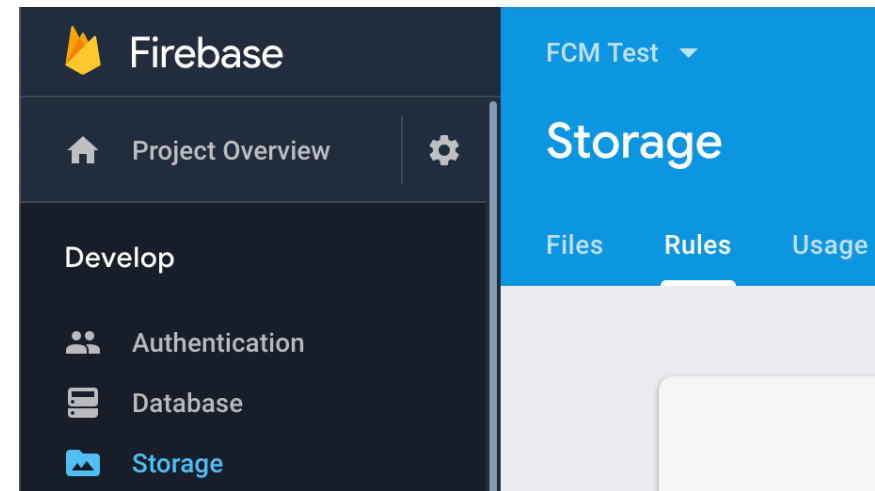
- Để sử dụng **Cloud Storage**, cần vào Firebase Console → Develop → Storage → Get started
- Mặc định, chỉ có thể upload, download file vào **Cloud Storage** nếu đã đăng nhập (authentication). Có thể cấu hình các luật upload, download trong tab **Rules**:
 - Rules chỉ cho phép upload, download nếu user đã đăng nhập:

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

- Rules luôn luôn cho phép upload, download

(không yêu cầu đăng nhập):

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```





Firebase Cloud Storage

- Cấu hình sử dụng **Cloud Storage** trong **build.gradle** của **module**:

```
implementation 'com.google.firebase:firebase-storage:19.0.0'
```

- Khai báo đối tượng **Cloud Storage**:

```
FirebaseStorage storage = FirebaseStorage.getInstance();
```

- Tạo tham chiếu đến storage từ app:

- Tham chiếu đến storage:

```
StorageReference storageRef = storage.getReference();
```

- Tham chiếu đến file “mountains.jpg”:

```
StorageReference mountainsRef = storageRef.child("mountains.jpg");
```

- Tham chiếu đến file “images/mountains.jpg”:

```
StorageReference mountainImagesRef = storageRef.child("images/mountains.jpg");
```



Firestore Cloud Storage (2)

- Upload file từ thiết bị lên **Cloud Storage**:

```
Uri file = Uri.fromFile(new File("path/to/images/rivers.jpg"));
StorageReference riversRef = storageRef.child("images/"+file.getLastPathSegment());
UploadTask uploadTask = riversRef.putFile(file);
// Đăng ký listener để nhận kết quả quá trình upload file
uploadTask.addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception exception) {
        // Xử lý khi upload file thất bại
    }
}).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
        // Xử lý khi upload thành công
    }
});
```





Firestore Cloud Storage (3)

- Download file từ Cloud Storage về thiết bị:

```
islandRef = storageRef.child("images/island.jpg");
```

```
File localFile = File.createTempFile("images", "jpg");
```

```
islandRef.getFile(localFile).addOnSuccessListener(new  
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {  
    @Override  
    public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {  
        // Xử lý nếu tải file về thành công  
    }  
}).addOnFailureListener(new OnFailureListener() {  
    @Override  
    public void onFailure(@NonNull Exception exception) {  
        // Xử lý nếu có lỗi xảy ra  
    }  
});
```



Firestore Cloud Storage (4)

- Các file được upload lên Cloud Storage có thể được tìm thấy tại: Firebase Console → Develop → Files:

Storage

Files Rules Usage

gs://fcm-test-dbb0b.appspot.com > images

Upload file

+

⋮

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	1009928407	196....	image/jpeg	Sep 2, 2019
<input type="checkbox"/>	bia_bandatgiabaonhieu_full.jpg	67.4...	image/jpeg	Sep 2, 2019
<input type="checkbox"/>	dung_lua_chon_an_nhan.jpg	42.1...	image/jpeg	Sep 2, 2019
<input type="checkbox"/>	IMG_1567418074901	195....	image/jpeg	Sep 2, 2019
<input type="checkbox"/>	nha_gia_kim.jpg	23.3...	image/jpeg	Sep 2, 2019

dung_lua_chon_an_n... ×

Name
[dung_lua_chon_an_nhan.jpg](#) ↗

Size
43,121 bytes

Type
image/jpeg



