02-3. Database

Firebase

21-1 파이어스토어 데이터베이스

파이어스토어 사용 설정

- 파이어베이스는 **파이어스토어 데이터베이스**Firestore Database와 **실시간 데이터베이스**Realtime Database 이렇게 2가지 클라우드를 기반으로 한 데이터베이스를 제공

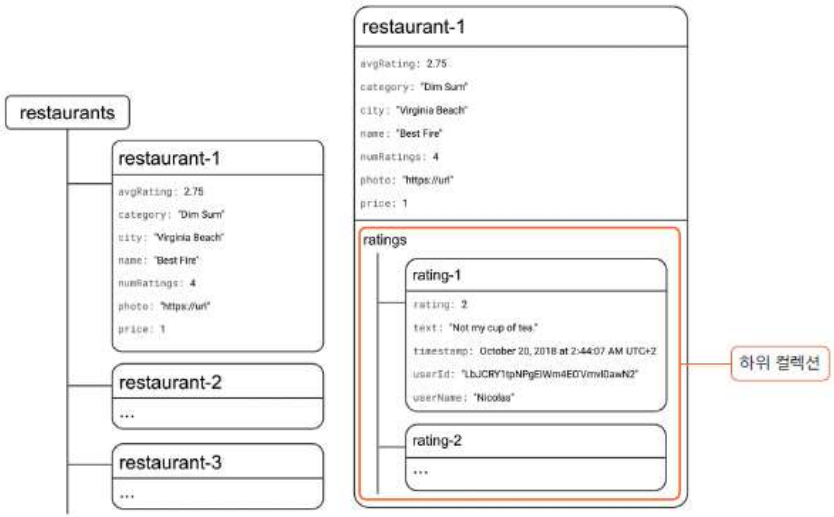
• 파이어스토어 사용 등록

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

21-1 파이어스토어 데이터베이스

파이어스토어 데이터 모델

- 파이어스토어는 NoSQL 데이터베이스
- 컬렉션으로 정리되는 문서에 데이터가 저장
- 문서에는 키-값 쌍의 데이터가 저장되며 모든 문서는 컬렉션에 저장



21-1 파이어스토어 데이터베이스

파이어스토어 보안 규칙

- 보안 규칙은 콘솔의 [규칙] 탭에서 설정
- match와 allow 구문을 조합해서 작성
- match 구문으로 데이터베이스 문서를 식별하고 allow 구문으로 접근 권한을 작성

• 모든 문서의 읽기/쓰기 거부

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {          // 모든 문서
      allow read, write: if false;  // 읽기/쓰기 거부
    }
  }
}
```

• 모든 문서의 읽기/쓰기 허용

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

21-1 파이어스토어 데이터베이스

• 인증된 사용자에게만 모든 문서의 읽기/쓰기 허용

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

인증된 사용자에게만 읽기/쓰기 허용

• 자신의 데이터만 읽기/쓰기 허용

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read, update, delete: if request.auth.uid == userId;
      allow create:  if request.auth.uid != null;
    }
  }
}
```

자신의 데이터만 읽기, 수정, 삭제 허용

인증된 사용자에게만 문서 생성 허용

21-1 파이어스토어 데이터베이스

• 문서에 저장된 데이터 활용

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /cities/{city} {
      allow read: if resource.data.visibility == 'public';
    }
  }
}
```

문서의 visibility값이 public일 때만 읽기 허용

• 전달받은 데이터 활용

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {{
      allow update: if request.resource.data.population > 0
                    && request.resource.data.name == resource.data.name;
    }
  }
}
```

전달받은 데이터가 0 이상일 때만 population 데이터 수정 허용.
단, name 데이터는 수정할 수 없음

21-1 파이어스토어 데이터베이스

데이터 저장하기

- FirebaseFirestore 객체로 컬렉션을 선택하고 문서를 추가하거나 가져오는 작업을 합니다.

• 파이어스토어 객체 얻기

```
var db: FirebaseFirestore = FirebaseFirestore.getInstance()
```

- add() 함수로 데이터 저장하기
  - 데이터를 저장하려면 먼저 컬렉션을 선택하고 문서 작업을 하는 CollectionReference 객체를 얻어야 합니다.
  - CollectionReference 객체의 add(), set(), get() 등의 함수로 문서 작업

21-1 파이어스토어 데이터베이스

• add() 함수로 데이터 저장

```
val user = mapOf(
    "name" to "kkang",
    "email" to "a@a.com",
    "avg" to 10
)
val colRef: CollectionReference = db.collection("users")
val docRef: Task<DocumentReference> = colRef.add(user)
docRef.addOnSuccessListener { documentReference ->
    Log.d("kkang", "DocumentSnapshot added with ID: ${documentReference.id}")
}                                                    성공 콜백
docRef.addOnFailureListener { e ->
    Log.w("kkang", "Error adding document", e)
}                                                    실패 콜백
```

• add() 함수로 데이터 저장

```
db.collection("users")
    .add(user)
    .addOnSuccessListener { documentReference ->
        Log.d("kkang", "DocumentSnapshot added with ID: ${documentReference.id}")
    }
    .addOnFailureListener { e ->
        Log.w("kkang", "Error adding document", e)
    }
```

21-1 파이어스토어 데이터베이스

- 객체 저장하기
  - Users 컬렉션에 User 클래스의 객체를 저장

• 객체 저장하기

```
class User(val name: String, val email: String, val avg: Int,
    @JvmField val isAdmin: Boolean, val isTop: Boolean)
val user = User("kim", "kim@a.com", 20, true, true)
db.collection("users")
    .add(user)
```

21-1 파이어스토어 데이터베이스

- set() 함수로 데이터 저장하기
  - set() 함수는 신규 데이터뿐만 아니라 기존의 데이터를 변경할 때도 사용
  - add() 함수는 CollectionReference 객체에서 제공하므로 문서를 추가할 때 식별자가 자동으로 지정
  - set() 함수는 DocumentReference 객체에서 제공하므로 document() 함수로 작업 대상 문서를 먼저 지정

• set() 함수로 데이터 저장

```
val user = User("lee", "lee@a.com", 30)
db.collection("users")
    .document("ID01")
    .set(user)
```

▶ 실행 결과

21-1 파이어스토어 데이터베이스

데이터 업데이트와 삭제

- update() 함수로 데이터 업데이트하기
- set() 함수는 문서 전체를 덮어 쓰기
- 기존 문서의 특정 필드값만 업데이트하려면 update() 함수를 이용

• 특정 필드값만 업데이트

```
db.collection("users")
    .document("ID01")
    .update("email", "lee@b.com")
```

• 여러 필드값 업데이트

```
db.collection("users")
    .document("ID01")
    .update(mapOf(
        "name" to "lee01",
        "email" to "lee@c.com"
    ))
```

21-1 파이어스토어 데이터베이스

- delete() 함수로 데이터 삭제하기
- 특정 필드값을 삭제하려면 update() 함수로 필드값을 지정하고 Field Value.delete() 함수를 호출
- 문서 전체를 삭제하려면 document() 함수로 문서를 지정하고 delete() 함수를 호출

• 특정 필드값 삭제

```
db.collection("users")
    .document("ID01")
    .update(mapOf(
        "avg" to FieldValue.delete()
    ))
```

• 문서 전체 삭제

```
db.collection("users")
    .document("ID01")
    .delete()
```

21-1 파이어스토어 데이터베이스

데이터 불러오기

- get() 함수로 컬렉션의 전체 문서 가져오기

• 전체 문서 가져오기

```
db.collection("users")
    .get()
    .addOnSuccessListener { result ->
        for (document in result) {
            Log.d("kkang", "${document.id} => ${document.data}")
        }
    }
    .addOnFailureListener { exception ->
        Log.d("kkang", "Error getting documents: ", exception)
    }
```

21-1 파이어스토어 데이터베이스

- get() 함수로 단일 문서 가져오기

• 단일 문서 가져오기

```
val docRef = db.collection("users").document("ID01")
docRef.get()
    .addOnSuccessListener { document ->
        if (document != null) {
            Log.d("kkang", "DocumentSnapshot data: ${document.data}")
        } else {
            Log.d("kkang", "No such document")
        }
    }
    .addOnFailureListener { exception ->
        Log.d("kkang", "get failed with ", exception)
    }
```

• 문서를 객체에 담기

```
class User{
    var name: String? = null
    var email: String? = null
    var avg: Int = 0
}
val docRef = db.collection("users").document("ID01")
docRef.get().addOnSuccessListener { documentSnapshot ->
    val selectUser = documentSnapshot.toObject(User::class.java)
    Log.d("kkang", "name: ${selectUser?.name}")
}
```

21-1 파이어스토어 데이터베이스

- whereXXX() 함수로 조건 설정
- whereXXX() 함수로 조건을 지정한 Query 객체를 만들고, 그 Query 객체의 get() 함수를 호출
- whereEqualTo(), whereGreaterThan(), whereIn(), whereArrayContains(), whereLessThan(), whereNotEqualTo(), whereNotIn() 등을 제공

• 조건에 맞는 문서 가져오기

```
db.collection("users")
    .whereEqualTo("name", "lee")
    .get()
    .addOnSuccessListener { documents ->
        for (document in documents) {
            Log.d("kkang", "${document.id} => ${document.data}")
        }
    }
    .addOnFailureListener { exception ->
        Log.w("kkang", "Error getting documents: ", exception)
    }
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare