배달 앱(패캠마트)

2 Firebase 알아보기
Firebase Cloud Firestore

Firebase Cloud Firestore

NoSQL

1. Cloud Firestore
2. MongoDB & Realm

- Documents : A document is a lightweight record that contains fields, which map to values. Each document is identified by a name.
- Collections: Documents live in collections, which are simply containers for documents.
- **여기서 알 수 있는것 !**
  - **컬렉션안에는 도큐먼트가, 도큐먼트 안에 또 컬렉션이 포함될 수 있음.**

출처: https://firebase.google.com/docs/firestore/data-model

Firebase Cloud Firestore

Realtime Data vs Firestore Data Model

Realtime Database	Cloud Firestore
Stores data as one large JSON tree. • Simple data is very easy to store. • Complex, hierarchical data is harder to organize at scale.	**Stores data as collections of documents.** • Simple data is easy to store in documents, which are very similar to **JSON**. • Complex, hierarchical data is easier to organize at scale, using sub-collections within documents. • Requires less denormalization and data flattening.

출처: https://firebase.google.com/docs/firestore/rtdb-vs-firestore

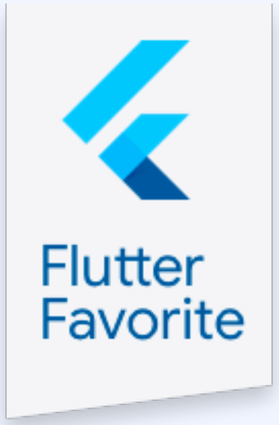
데이터 타입

1.Null values
2.Boolean values
3.NaN values
4.Integer and floating-point values, sorted in numerical order
5.Date values
6.Text string values
7.Byte values
8.Cloud Firestore references
9.Geographical point values
10.Array values
11.Map values

Firebase Cloud Firestore

설치 필요 패키지

1. https://pub.dev/packages/firebase_core
2. https://pub.dev/packages/cloud_firestore



Document 추가하기

1.add()
- Document id가 Random한 고유키를 기반으로 데이터 생성

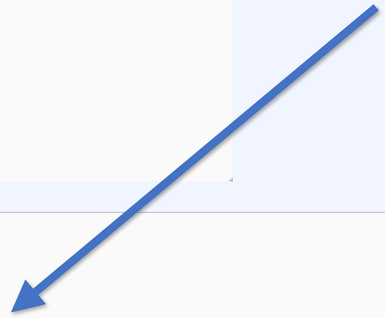
2.set()
- Document의 id(고유이름)을 지정할 수 있음.

Firebase Cloud Firestore

Document 추가하기 - add()

```
final docData = {
 "stringType": "Hello world!",
 "booleanType": true,
 "numberType": 3.14159265,
 "dateType": Timestamp.now(),
 "listType": [1, 2, 3],
 "nullType": null,
 "geo": const GeoPoint(36.15, 128.15),
};
```

Documents 고유 키를 중복되지 않는 랜덤한 키로 자동 생성



```
await db.collection("types").add(docData);
```

Firebase Cloud Firestore

Document 추가하기 - set()

```
final flutter = <String, String>{
  "sdk": "stable",
  "dart-version": "3",
  "platform": "mobile,web"
};
```

문서 명을 flutter로 지정하고 flutter 문서에 데이터를 삽입

```
db.collection("fast-campus").doc("flutter")
    .set(flutter)
    .onError((e, _) ⇒ print("Error writing document: $e"));
```

데이터 읽기(read) - get()

```
final docRef = db.collection("campus").doc("flutter");   가져올 데이터의 Document 를 지정
docRef.get().then(   ref를 통해 get() 메소드를 사용하여 데이터를 가져온다.
  (DocumentSnapshot doc) {
    final data = doc.data() as Map<String, dynamic>;   데이터타입 casting
  },
  onError: (e) => print("Error getting document: $e"),
);
```

Firebase Cloud Firestore

데이터 읽기(read) - 실시간 업데이트 대응

```
final docRef = db.collection("fast-campus").doc("flutter");
docRef.snapshots().listen(
  (event) ⇒ print("current data: ${event.data()}"),
  onError: (error) ⇒ print("Listen failed: $error"),
);
```

StreamBuilder 위젯 활용

데이터 업데이트 (Update) - update()

```
final flutterRef = db.collection("capmus").doc("flutter");

flutterRef.update({"dart-version": "3.0"}).then(
    (value) ⇒ print("DocumentSnapshot successfully updated!"),
    onError: (e) ⇒ print("Error updating document $e"));
```

도큐먼트 삭제 - delete()

```
db.collection("fast-campus").doc("chore").delete().then(
  (doc) => print("Document deleted"),
  onError: (e) => print("Error updating document $e"),
);
```

Document 완전 삭제
삭제 전 사용자에게 삭제 안내 제공 필요

Firebase Cloud Firestore

도큐먼트 삭제 - 도큐먼트 내 필드 삭제

```
final docRef = db.collection("fast-campus").doc("flutter");
final updates = <String, dynamic>{
  "dart2": FieldValue.delete(),
};
docRef.update(updates);
```

컬렉션 삭제

중요
컬렉션 삭제는 사용자에게 제공하는 행위는 권장하지 않음