

Firebase Realtime Database

Firestore 소개

- Firebase 원격 클라우드 서비스
 - 모바일, 웹 개발자를 위한 여러가지 서비스 제공
 - Analytics : 사용자 이벤트 중심의 분석 솔루션
 - Realtime Database : 실시간 데이터베이스와 백엔드 서비스 제공
 - Storage : 이미지, 동영상 및 기타 대용량 파일 저장 기능
 - Firebase Cloud Messaging(FCM) : 푸시 메시지 서비스
 - Authentication : 인증관련 서비스
 - 기타 다양한 서비스 제공
- Firebase 실시간 데이터베이스
 - 클라우드 호스팅 데이터베이스
 - NoSQL 데이터베이스
 - 데이터는 JSON으로 저장되며 모든 클라이언트에 실시간으로 동기화됨
 - Tree-like map 구조

Firebase 가격 책정

- <https://firebase.google.com/pricing?authuser=0>
 - 구글계정 로그인 상태

요금제

무료 사용 할당량으로 시작하고
소진 후 사용한 만큼만 지불하세요.

제품	무료 Spark 요금제 시작하기가 충분한 넉넉한 한도	총량제 Blaze 요금제 규모별 월 가격 계산 ✓ Spark 요금제의 무료 사용 혜택 포함*
A/B Testing	무료	무료
Analytics	무료	무료
앱 배포	무료	무료
앱 색인 생성	무료	무료

Google Cloud

BigQuery ?
기타 IaaS ?

요금제 선택
Flame 요금제를 찾고 계신가요?

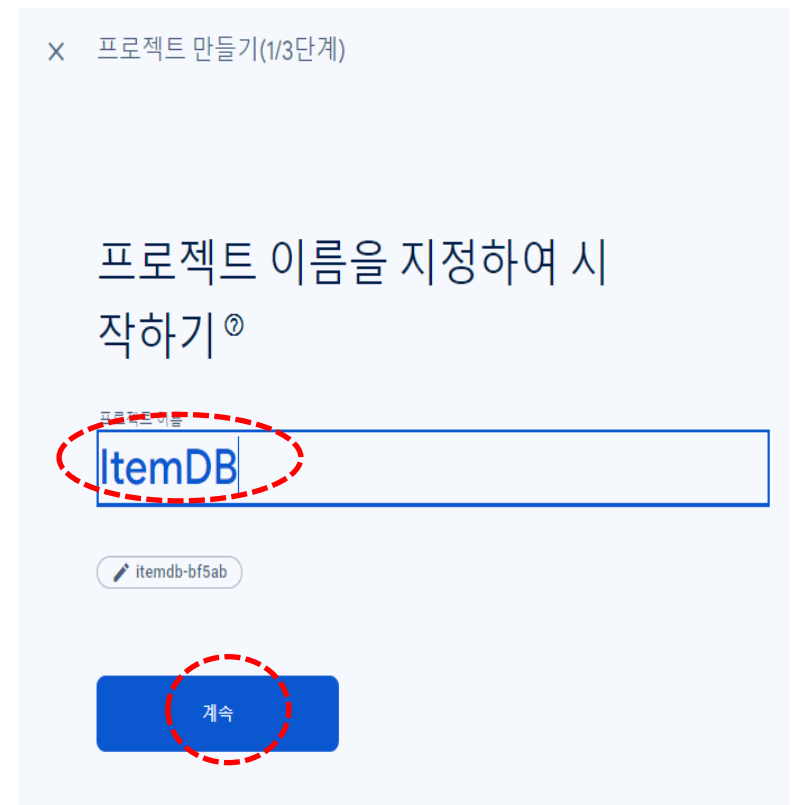
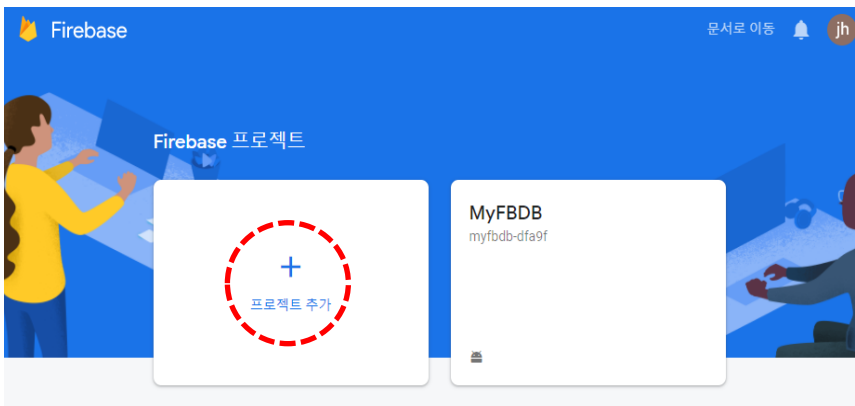
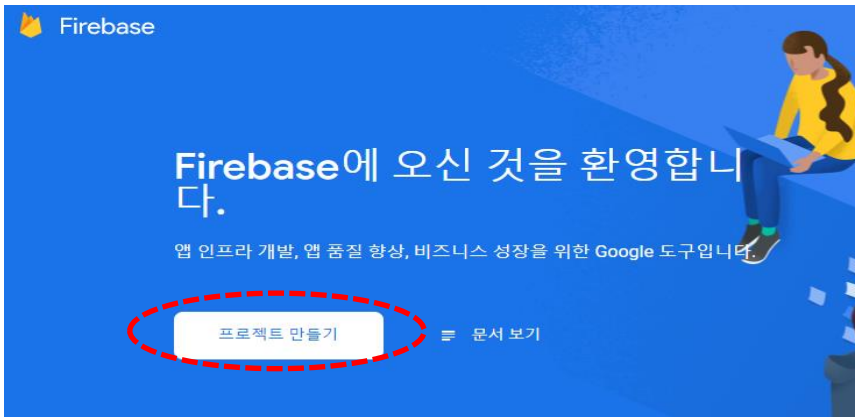
무료
Spark 요금제
지금 시작

총량제
Blaze 요금제
요금제 선택

*Blaze 요금제의 무료 사용량은 매일 계산됩니다. 세부정보는 Cloud Functions, Firebase ML, 전화 인증, Test Lab에 따라 약간 다릅니다.
자세한 정보는 [FAQ](#)를 참조하거나 결제 이해를 위한 [문서](#)를 확인하세요.

Firebase 프로젝트 생성

- <https://console.firebase.google.com/>
 - 구글계정 로그인 상태
 - 안드로이드 앱을 미리 만들어 놓고 진행할 것 (앱의 패키지 정보 필요)













Firestore 프로젝트 생성

× 프로젝트 만들기(2/3단계)

Firestore 프로젝트를 위한 Google 애널리틱스

무제한 무료 분석 솔루션인 Google 애널리틱스를 사용하면 Firestore Crashlytics, 클라우드 메시징, 인앱 메시징, 원격 구성, A/B 테스트, Cloud Functions에서 타겟팅, 보고 등을 이용할 수 있습니다.

Google 애널리틱스를 통해 다음 기능을 이용할 수 있습니다.

-  A/B 테스트 
-  Firestore 제품 전반에서 사용자 세분화  및 타겟팅
-  장애가 발생하지 않은 사용자 
-  이벤트 기반 Cloud Functions 트리거 
-  제한 없는 무료 보고 


☒ 이 프로젝트에서 Google 애널리틱스 사용 설정 권장



이전


계속

× 프로젝트 만들기(3/3단계)

Google 애널리틱스 구성

Google 애널리틱스 계정 선택 또는 만들기 

 Default Account for Firestore 

이 계정에서 자동으로 새 속성 만들기 

프로젝트를 만들면 선택한 Google 애널리틱스 계정에 새 Google 애널리틱스 속성이 생성되고 Firestore 프로젝트에 연결됩니다. 이 연결을 통해 제품 간에 데이터 흐름이 활성화됩니다. Google 애널리틱스 속성에서 Firestore로 내보낸 데이터에는 Firestore 서비스 약관이 적용되지만 Google 애널리틱스로 가져온 Firestore 데이터에는 Google 애널리틱스 서비스 약관이 적용됩니다. [자세히 알아보기](#)

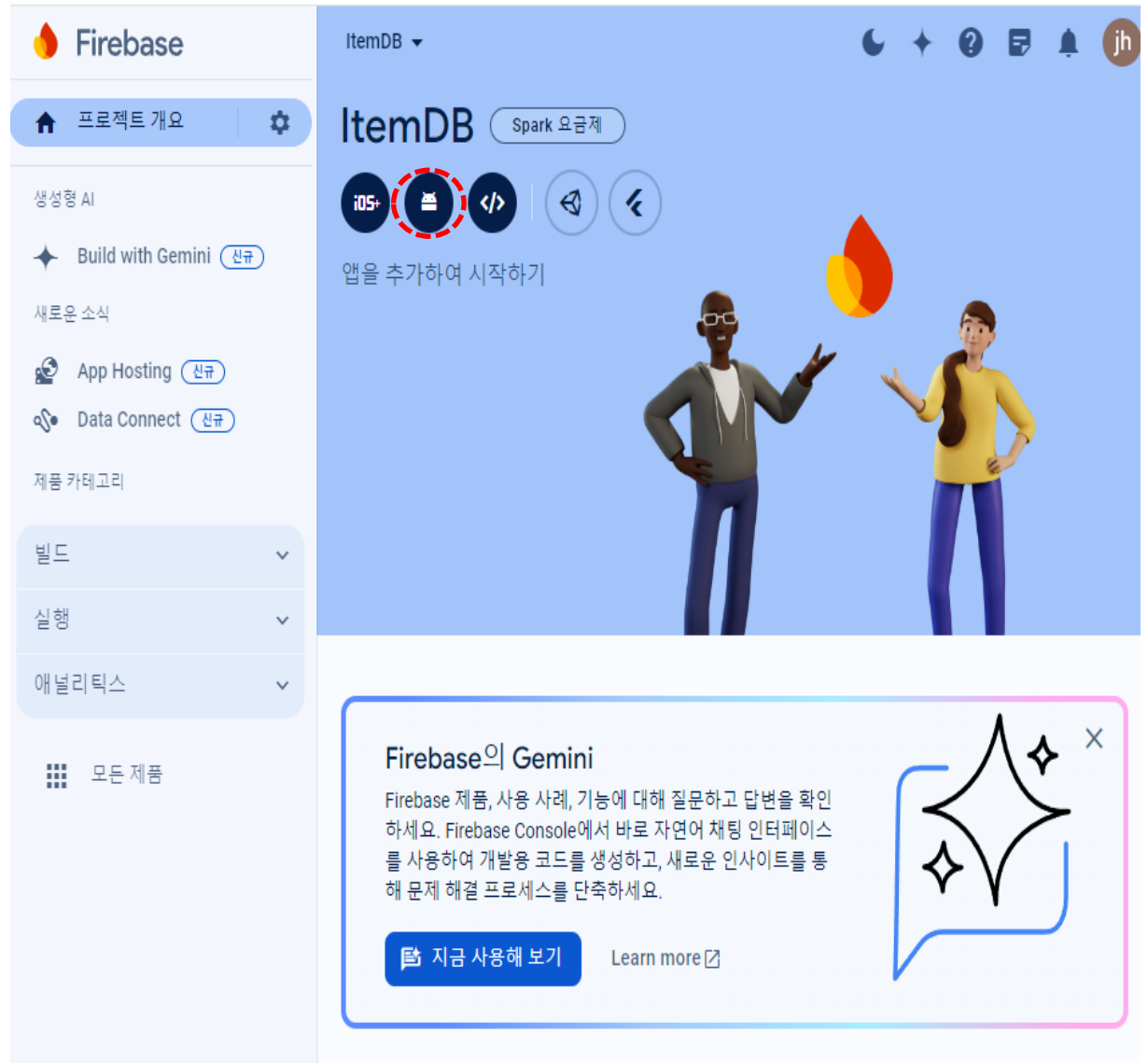
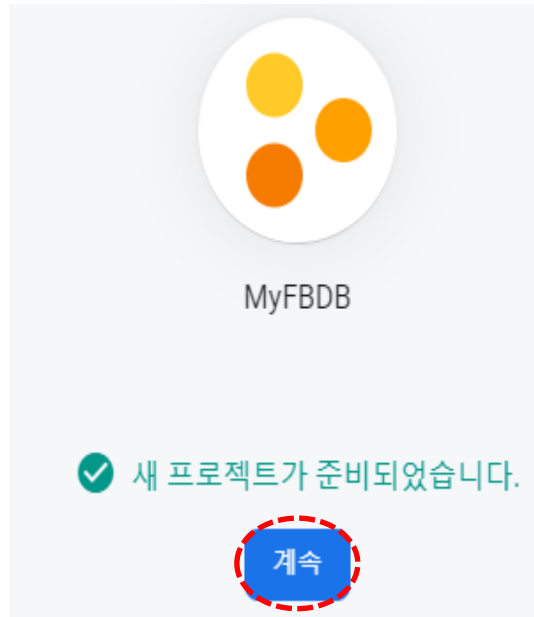
이전

프로젝트 만들기



프로젝트 생성 중...
MyFDBB

Firestore 프로젝트 생성



Firestore 프로젝트 생성

- 앱 등록 : Android 앱의 패키지 이름 추가

Android 앱에 Firebase 추가

1 앱 등록

Android 패키지 이름 ⓘ
com.example.myfbdbapp

앱 닉네임 (선택사항) ⓘ
내 Android 앱

디버그 서명 인증서 SHA-1(선택사항) ⓘ
00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:!

인증에서 동적 링크, Google 로그인, 전화번호를 지원하는 데 필요합니다. 설정에서 SHA-1을 수정하세요.

앱 등록

2 구성 파일 다운로드

3 Firebase SDK 추가

4 다음 단계

Firebase 프로젝트 생성

- 구성 파일 다운로드

× Android 앱에 Firebase 추가

✓ 앱 등록
Android 패키지 이름: com.example.myfdbapp

2 구성 파일 다운로드 Android 스튜디오에 대한 안내(아래 참조) | [Unity](#) [C++](#)

google-services.json 다운로드

Android 스튜디오에서 프로젝트 보기로 전환하여 프로젝트 루트 디렉터리를 표시하세요.

방금 다운로드한 google-services.json 파일을 Android 앱 모듈 루트 디렉터리로 이동하세요.

google-services.json

다음

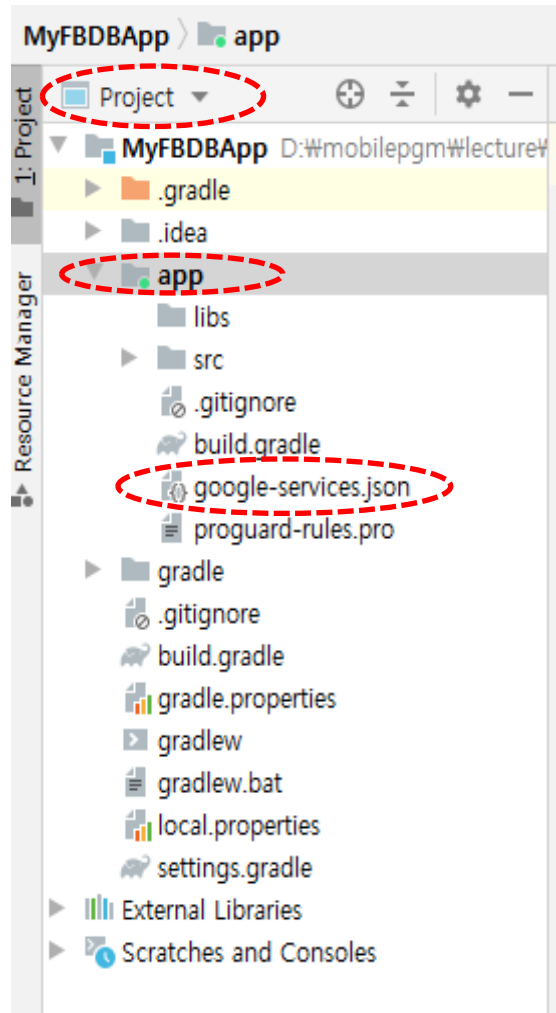
3 Firebase SDK 추가

4 다음 단계

google-services.json

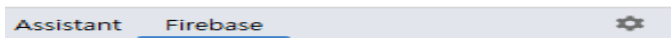
Firestore 프로젝트 생성

- 다운로드된 google-services.json 파일을 앱의 루트 디렉토리에 복사



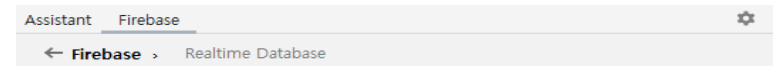
Firestore 프로젝트 생성

- Firestore SDK 추가 (Firestore Assistant)
 - Android studio >> Tools >> Firestore



- ▶ **Analytics**
Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)
- ▶ **Authentication**
Sign in and manage users with ease using popular login providers like Google, Facebook, and others. You can even use a custom authentication system. [More info](#)
- ▶ **Realtime Database**
Store and sync data with this cloud-hosted NoSQL database. Data is synced across all clients in realtime and remains available when your app goes offline. [More info](#)
- ▶ **Cloud Firestore**
Store and sync your app data with a flexible, scalable NoSQL cloud database. [More info](#)
- ▶ **Vertex AI for Firestore**
Build AI-powered mobile and web apps and features with the Gemini API using Vertex AI for Firestore. [More info](#)
- ▶ **Cloud Storage for Firestore**
Store and retrieve large files like images, audio, and video without writing server-side code. [More info](#)
- ▶ **Cloud Functions for Firestore**
Automatically run backend code in response to events triggered by Firestore features and HTTPS requests. [More info](#)

- ▶ **Realtime Database**
Store and sync data with this cloud-hosted NoSQL database. Data is synced across all clients in realtime and remains available when your app goes offline. [More info](#)
 - ▶ [Get started with Realtime Database](#)
 - ▶ [Get started with Realtime Database \[Java\]](#)



Get started with Realtime Database [KOTLIN]

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronizes in real time.

[Launch in browser](#)

1 Connect your app to Firebase

✓ Connected

2 Add the Realtime Database to your app

[Add the Realtime Database SDK to your app](#)

NOTE: After adding the SDK, here are some other helpful configurations to consider:

- **Do you want an easier way to manage library versions?**
You can use the [Firebase Android BoM](#) to manage your Firebase library versions and compatible library versions.

To use the Realtime Database, you need to create a database instance in the [Firebase console](#).

3 Configure Realtime Database Rules

The Realtime Database provides a declarative rules language that allows you to define how data should be indexed, and when your data can be read from and written to.

By default, read and write access to your database is restricted so only authenticated users can access the database. [Without setting up Authentication, you can configure your rules for public access. This does not mean you should allow public access to your database, so be sure to restrict your database again when you set up Authentication.](#)

4 Write to your database

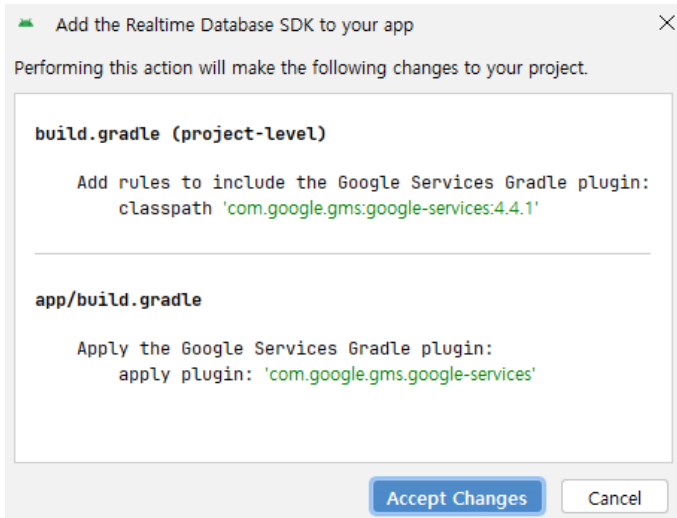
Retrieve an instance of your database using [Firebase.database](#) and reference the location [you want to write to](#).

To get a reference to a database other than a [us-central1](#) default database, you must pass the [us-central1](#) default database, you can call [database\(\)](#) without arguments. [Learn more about database references.](#)

```
// Write a message to the database
val database = Firebase.database
val myRef = database.getReference("message")
```

Firebase 프로젝트 생성

- Firebase SDK 추가 (Firebase Assistant)



Get started with Realtime Database

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronizes across all clients.

[Launch in browser](#)

- 1 Connect your app to Firebase

✓ Connected

- 2 Add the Realtime Database to your app

Add the Realtime Database SDK to your app

NOTE: After adding the SDK, here are some other helpful configurations to consider:

- **Do you want an easier way to manage library versions?**
You can use the [Firebase Android BoM](#) to manage your Firebase library versions and compatible library versions.

To use the Realtime Database, you need to create a database instance in the [Firebase console](#).

* Project의 build.gradle

```
plugins { this: PluginDependenciesSpecScope
    alias(libs.plugins.android.application) apply false
    alias(libs.plugins.jetbrains.kotlin.android) apply false
    alias(libs.plugins.google.gms.google.services) apply false
}
```

* Module의 build.gradle

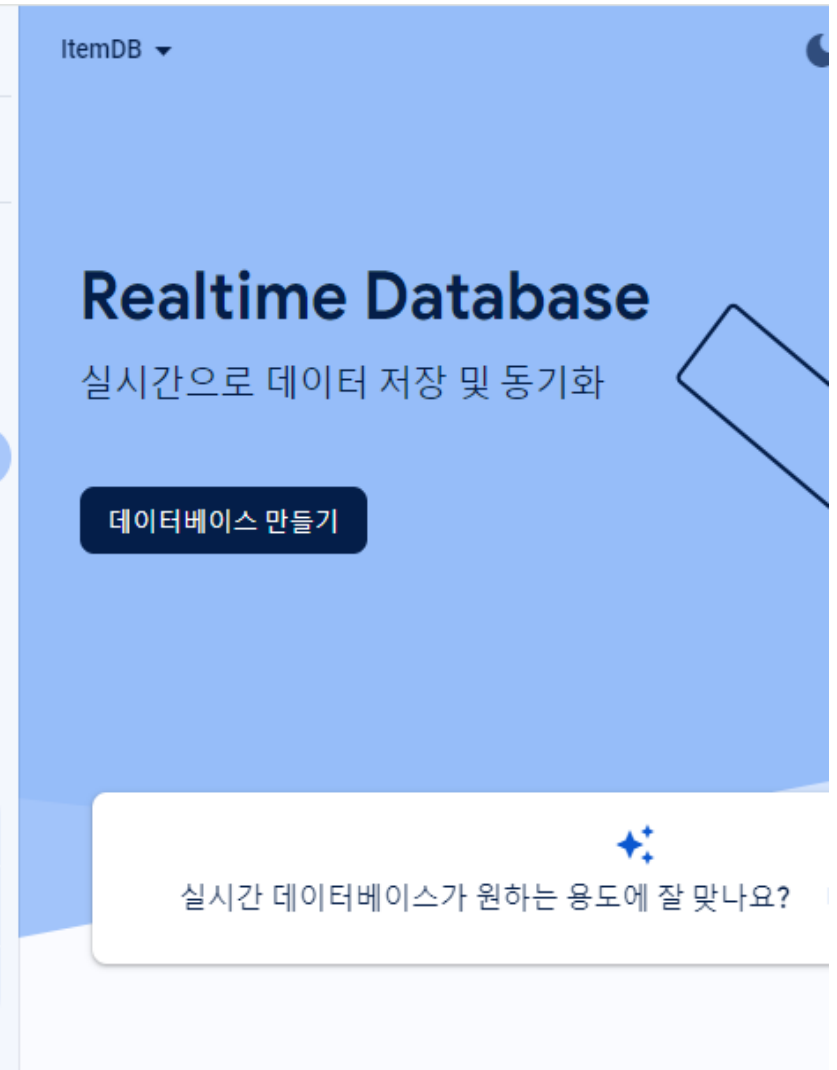
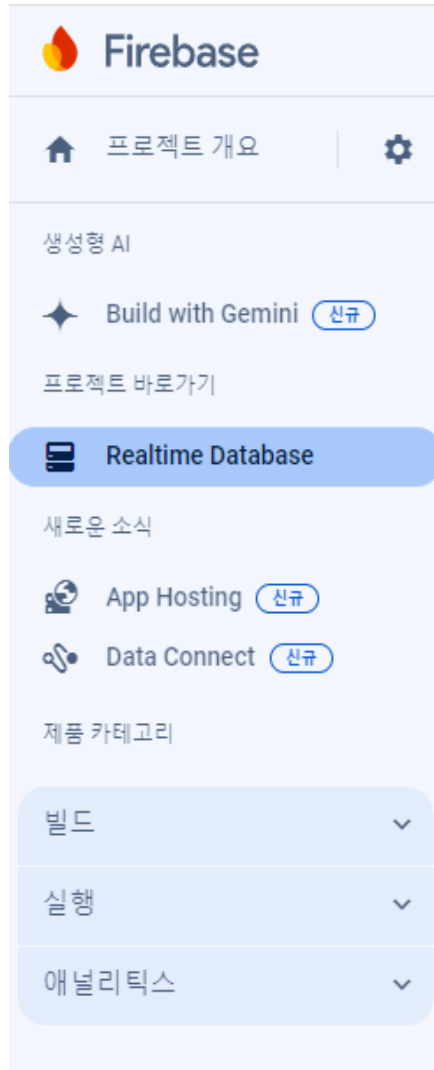
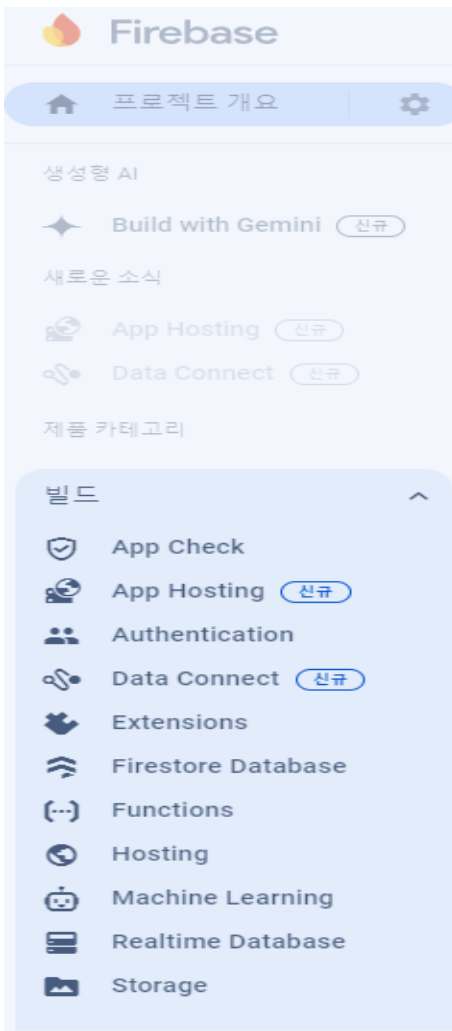
```
plugins { this: PluginDependenciesSpecScope
    alias(libs.plugins.android.application)
    alias(libs.plugins.jetbrains.kotlin.android)
    alias(libs.plugins.google.gms.google.services)
}
```

```
dependencies {
```

```
    implementation(libs.firebase.database)
```

Firestore 프로젝트 생성

- 데이터베이스 만들기 (<https://console.firebase.google.com/>)



Firebase 프로젝트 생성

- 데이터베이스 설정
 - 데이터베이스 옵션

데이터베이스 설정

1 데이터베이스 옵션 — 2 보안 규칙

위치 설정은 실시간 데이터베이스 데이터가 저장되는 위치입니다.

실시간 데이터베이스 위치

미국(us-central1)

취소 다음

- 보안규칙

데이터베이스 설정

1 데이터베이스 옵션 — 2 보안 규칙

데이터 구조를 정의한 후 데이터의 보안을 강화하는 규칙을 작성해야 합니다.

[자세히 알아보기](#)

☒ 잠금 모드로 시작

데이터는 기본적으로 비공개됩니다. 클라이언트 읽기/쓰기 액세스 권한은 보안 규칙에서 지정한 대로만 부여됩니다.

☐ 테스트 모드에서 시작

빠른 설정을 위해 기본적으로 데이터가 공개됩니다. 하지만 30일 내에 보안 규칙을 업데이트하여 장기적 클라이언트 읽기/쓰기 액세스 권한을 사용 설정해야 합니다.

```
{
  "rules": {
    ".read": false,
    ".write": false
  }
}
```

모든 제3자 읽기 및 쓰기가 거부됩니다.

취소 사용 설정

Firebase 프로젝트 생성

- 데이터베이스 설정

Realtime Database

데이터 규칙 백업 사용량 Extensions

결제 사기나 피싱과 같은 악용으로부터 Realtime Database 리소스를 보호하세요. [앱 체크 구성](#)

<https://itemdb-bf5ab-default-rtdb.firebaseio.com>

<https://itemdb-bf5ab-default-rtdb.firebaseio.com/> null

Realtime Database

데이터 규칙 백업 사용량 Extensions

게시되지 않은 변경사항 **게시** 삭제 [규칙 플레이그라운드](#)

★ 기본 보안 규칙이 액세스할 수 없도록 잠겨 있습니다. [자세히 알아보기](#) [닫기](#)

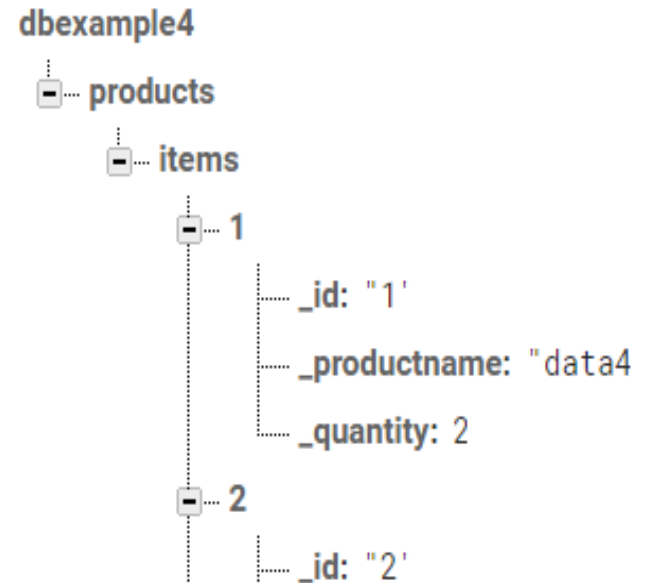
```
1 {
2   "rules": {
3     ".read": true,
4     ".write": true
5   }
6 }
```

앱과 연동

- 데이터베이스 참조 객체 생성

```
var mDatabase = Firebase.database.getReference("products")  
  
var rDatabase = mDatabase.child("items");
```

- 데이터는 key/value 형태로 저장
 - Key는 데이터 객체의 이름을 나타내는 스트링
 - Value는 다양한 타입이 가능
 - Boolean, long, double, list, map



Firestore 데이터 입력

고객 릴레이션

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500

```
var database = Firebase.database
var table = database.getReference("SalesDB/Customers")
```

```
var apple = table.child("apple")
apple.child("customerID").setValue("apple")
apple.child("customerName").setValue("정소화")
apple.child("customerAge").setValue(20)
apple.child("customerGrade").setValue("gold")
apple.child("customerJob").setValue("학생")
apple.child("customerPoint").setValue(1000)
```

```
var banana = database.getReference("SalesDB/Customers/banana")
banana.child("customerID").setValue("banana")
banana.child("customerName").setValue("김선우")
banana.child("customerAge").setValue(25)
banana.child("customerGrade").setValue("vip")
banana.child("customerJob").setValue("간호사")
banana.child("customerPoint").setValue(2500)
```

mydbapp-fc41f

SalesDB

Customer

apple

customerAge: 20
customerGrade: "gold"
customerID: "apple"
customerJob: "학생"
customerName: "정소화"
customerPoint: 1000

banana

customerAge: 25
customerGrade: "vip"
customerID: "banana"
customerJob: "간호사"
customerName: "김선우"
customerPoint: 2500

Firestore 데이터 매핑

- 키 자동 생성

```
var carrot = table.push()  
carrot.child("customerID").setValue("carrot")  
carrot.child("customerName").setValue("고영석")  
carrot.child("customerAge").setValue(28)  
carrot.child("customerGrade").setValue("gold")  
carrot.child("customerJob").setValue("교사")  
carrot.child("customerPoint").setValue(4500)
```

고객 릴레이션

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500

<https://mydbapp-fc41f.firebaseio.com/SalesDB>

[mydbapp-fc41f](#) > [SalesDB](#)

SalesDB

Customer

-Kjh6m6S96MHLqO55r-8

- customerAge: 28
- customerGrade: "gold"
- customerID: "carrot"
- customerJob: "교사"
- customerName: "고영석"
- customerPoint: 4500

+ apple

+ banana

Firestore 데이터 매핑

- 객체 삽입

```
var customer = Customer( "apple" , "정소화" , 20 , "gold" , "학생" , 1000 )
```

```
var table = database.getReference( "SalesDB/Customer" )
```

```
var apple = table.child( "apple" ).setValue( customer )
```

- data 클래스에는 디폴트 생성자가 존재해야 함

Firebase 질의

- SQL 구문과 같이 Query를 사용하여 처리 가능함

```
//SELECT * FROM CUSTOMER WHERE CUSTOMERID='banana' ;  
val query = table.orderByChild("customerID").equalTo("banana")  
val query = table.orderByKey().equalTo("banana")
```

- ValueEventListener 객체 생성
 - 데이터 변경에 대한 이벤트 수신
 - 해당 쿼리문에 맞는 데이터들을 자동으로 반환

```
val query = Firebase.database  
                .reference.child("SalesDB/Customer").limitToLast(50)  
  
query.addListenerForSingleValueEvent(listener)
```

ValueEventListener

- 데이터 변경 사항을 수신하기 위해 사용하는 리스너 인터페이스
- 주요 메서드
 - onDataChange(snapshot:DataSnapshot)
 - 지정된 위치의 데이터가 변경될 때 호출
 - DataSnapshot 객체를 통해 데이터 읽기
 - onCancelled(error:DatabaseError)
 - 데이터 읽기 작업이 취소되었을 때 호출
 - 권한문제나 네트워크 오류로 인한 작업 실패
- 수신 방법
 - ref.addValueEventListener : 실시간 업데이트 되는 데이터 지속적으로 수신
 - ref.addListenerForSingleValueEvent : 단일 조회가 필요할 때 사용

ValueEventListener

```
val table: DatabaseReference = FirebaseDatabase.getInstance().getReference("items")
```

```
val valueEventListener = object : ValueEventListener {  
    override fun onDataChange(snapshot: DataSnapshot) {  
        val items = mutableListOf<Item>()  
        for (itemSnapshot in snapshot.children) {  
            val item = itemSnapshot.getValue(Item::class.java)  
            item?.let { items.add(it) }  
        }  
    }  
  
    override fun onCancelled(error: DatabaseError) {  
        Log.w("Firebase", "loadPost:onCancelled", error.toException())  
    }  
}
```

```
table.addValueEventListener(valueEventListener)
```

```
table.removeEventListener(valueEventListener)
```

```
}
```

callbackFlow

- 비동기적인 이벤트를 Flow로 변환하는데 사용되는 라이브러리
 - 콜백 기반의 비동기 API를 편리하게 사용하여 데이터를 수신하고, 이를 Flow로 변환하여 데이터 처리 가능
 - trySend 함수
 - 데이터 전달
 - awaitClose 함수
 - Flow가 닫힐 때 호출

```
fun getAllItems(): Flow<List<Item>> = callbackFlow {  
    val listener = object : ValueEventListener {  
        override fun onDataChange(snapshot: DataSnapshot) {  
            val itemList = mutableListOf<Item>()  
            for (itemSnapshot in snapshot.children) {  
                val item = itemSnapshot.getValue(Item::class.java)  
                item?.let { itemList.add(it) }  
            }  
            trySend(itemList)  
        }  
    }  
    override fun onCancelled(error: DatabaseError) {  
        close(error.toException())  
    }  
}  
table.addValueEventListener(listener)  
awaitClose {  
    table.removeEventListener(listener)  
}  
}
```

수고하셨습니다.