

# 앱 개발 입문

성균관대학교 컬처애펀테크놀로지융합전공 하계 부트캠프

6일차 - 2022. 08. 03 (수)

## 강의 슬라이드 링크

[https://github.com/kunny/skku-bootcamp-2022-summer/blob/main/\\_slides/day6.pdf](https://github.com/kunny/skku-bootcamp-2022-summer/blob/main/_slides/day6.pdf)

# 오늘 강의에서 다룰 내용

- 데이터베이스를 알아봅니다.
- 데이터베이스 처리를 도와주는 패키지를 설치합니다.
- Dart에서 비동기 작업을 처리하는 방법을 알아봅니다.
- 데이터베이스를 사용하여 노트 데이터를 처리하도록 앱을 수정합니다.
- 앱 분석 및 최적화에 사용하는 도구인 Firebase를 소개합니다.
- 노트 앱에 Firebase를 적용합니다.

# 데이터베이스 알아보기

## 데이터베이스란?

- 논리적으로 연관된 하나 이상의 자료의 모음입니다.
- 자료를 구조화된 형식으로 저장하므로, 검색과 갱신을 효율적으로 수행할 수 있습니다.
- 안드로이드와 iOS는 간단한 데이터베이스를 처리할 수 있는 SQLite가 내장되어 있습니다.

# 왜 데이터베이스를 사용하나요? (파일 vs. 데이터베이스)

todo.txt

(제목 없음)

부트캠프 앱 개발  
강의 신청하기

2022.07.02

shopping.txt

장보기 목록

- 🍌 양파 1망
- 🥬 양배추 1통
- 🍊 귤 1박스
- 🍷 손질 닭고기 1팩
- 🍖 삼겹살 1팩
- 🍜 우동면 1팩
- (4입)
- 🥛 우유 2팩
- 🍞 식빵 1개

022.08.01

dev\_todo.txt

만들어야 할 것들

- 파이어베이스 프로젝트
- 애드몹 계정
- 구글 플레이 개발자 계정
- 애플 개발자 계정

2022.07.24

# 왜 데이터베이스를 사용하나요? (파일 vs. 데이터베이스)

todo.txt

(제목 없음)

부트캠프 앱 개발  
강의 신청하기

2022.07.02

새 데이터를 추가하려면?

특정 날짜 이후에 생성된  
메모만 찾으려면?

shopping.txt

장보기 목록

- 🍌 양파 1망
- 🥬 양배추 1통
- 🍊 귤 1박스
- 🐔 손질 닭고기 1팩
- 🥓 삼겹살 1팩
- 🍜 우동면 1팩
- (4입)
- 🥛 우유 2팩
- 🍞 식빵 1개

2022.08.01

dev\_todo.txt

만들어야 할 것들

- 파이어베이스 프로젝트
- 애드몹 계정
- 구글 플레이 개발자 계정
- 애플 개발자 계정

2022.07.24

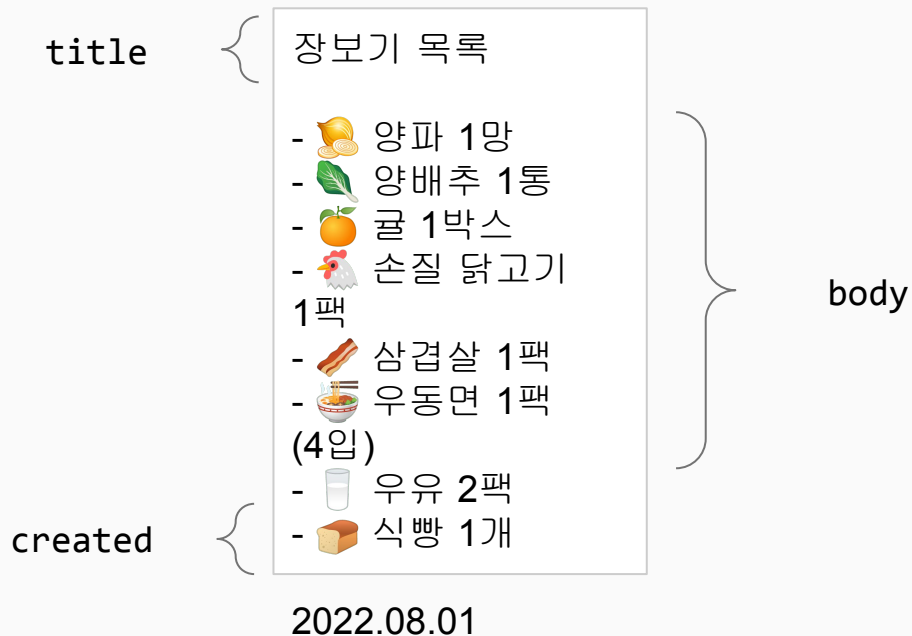
특정 문자열을 포함하는  
메모만 검색하려면?

# 관계형 데이터베이스

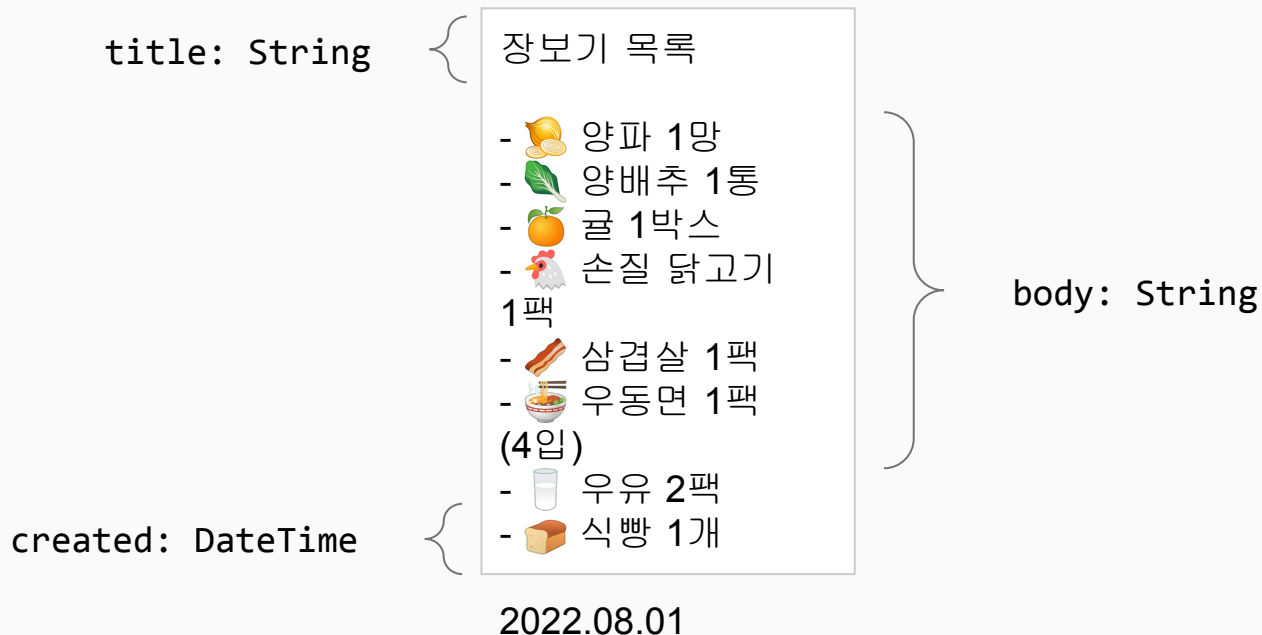
- 자료가 구성되는 형태에 따라 다양한 종류의 데이터베이스가 있으며, **관계형(Relational) 데이터베이스**를 가장 많이 사용합니다.
- 관계형 데이터베이스에서 자료는 아래 구성요소로 구성됩니다.
  - 열 (Column): 자료를 구성하는 세부 구성요소로, 필드라 부르기도 합니다. (예: 연락처 자료인 경우 이름, 전화번호 등이 해당)
  - 행 (Row): 자료를 구성하는 열(Column)이 모여 이룬 의미있는 자료로, 레코드라 부르기도 합니다.
  - 고유 키 (Primary Key): 각 레코드를 고유하게 식별할 수 있는 필드를 의미합니다.
  - 테이블: 같은 특징을 갖는 레코드의 집합을 의미합니다. (예: 연락처 목록)



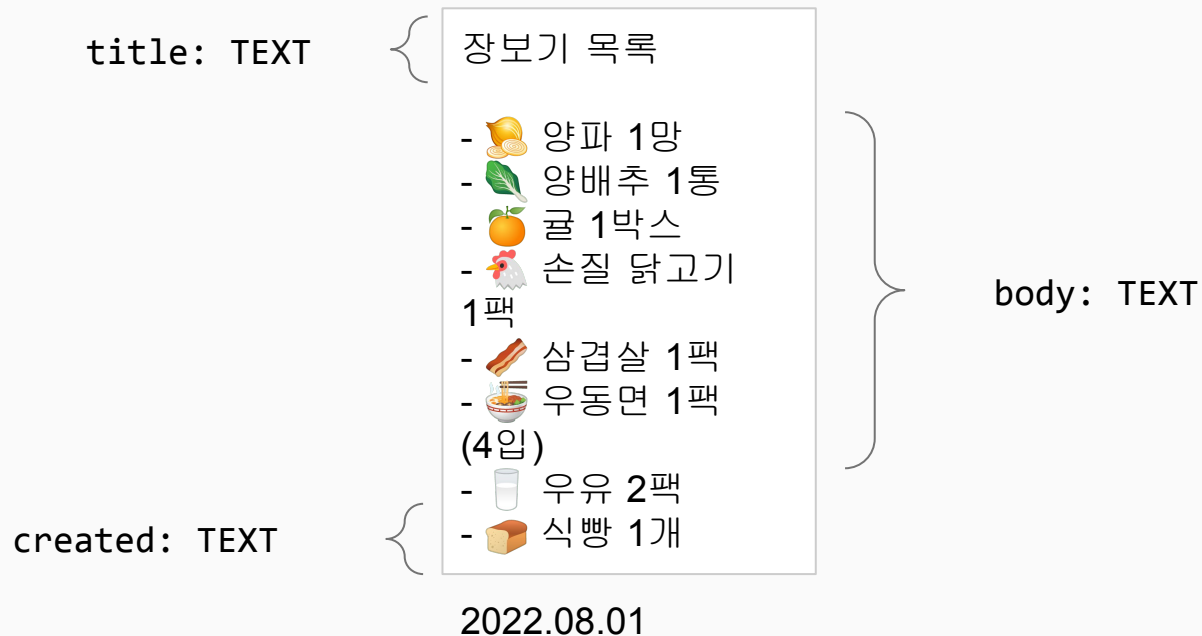
# 왜 데이터베이스를 사용하나요? (파일 vs. 데이터베이스)



# 왜 데이터베이스를 사용하나요? (파일 vs. 데이터베이스)



# 왜 데이터베이스를 사용하나요? (파일 vs. 데이터베이스)



# 왜 데이터베이스를 사용하나요? (파일 vs. 데이터베이스)

id: INTEGER (AUTO-GENERATED)

title: TEXT

장보기 목록

- 🍌 양파 1망
- 🥬 양배추 1통
- 🍊 귤 1박스
- 🐔 손질 닭고기 1팩

- 🥓 삼겹살 1팩
- 🍜 우동면 1팩 (4입)

- 🥛 우유 2팩
- 🍞 식빵 1개

body: TEXT

created: TEXT

2022.08.01

## 왜 데이터베이스를 사용하나요? (파일 vs. 데이터베이스)

id: INTEGER (AUTO-GENERATED)

title: TEXT

장보기 목록

body: TEXT

created: TEXT

2022.08.01

Record

# 왜 데이터베이스를 사용하나요? (파일 vs. 데이터베이스)

Primary Key

id	title	body	created
1		부트캠프 앱 개발 강의..	2022.07.02
2	장보기 목록	- 🍷 양파 1망..	2022.08.01
...	...	...	...

Record

Record

Table

# 데이터베이스 관리하기

- SQL(Structured Query Language)를 사용하여 데이터베이스를 생성하거나 테이블 내 데이터를 조작합니다.
- 데이터 정의 언어 (Data Definition Language)
  - CREATE, ALTER, DROP 등
- 데이터 조작 언어 (Data Manipulation Language)
  - SELECT, INSERT, DELETE, Update 등
- 데이터 제어 언어 (Data Control Language)
  - GRANT, REVOKE, COMMIT, ROLLBACK 등

# 데이터베이스 관리하기

## 데이터 정의 언어 예시 (테이블 생성)

```
CREATE TABLE notes(  
  id INTEGER PRIMARY KEY AUTOINCREMENT,  
  title TEXT,  
  body TEXT NOT NULL,  
  color INTEGER NOT NULL  
)
```



# 데이터베이스 관리하기

## 데이터 조작 언어 예시

# 모든 노트 반환

```
SELECT * FROM notes;
```

# 노트 제목에 '장보기' 문자열을 포함하는 노트 반환

```
SELECT * FROM notes WHERE title LIKE '%장보기%';
```

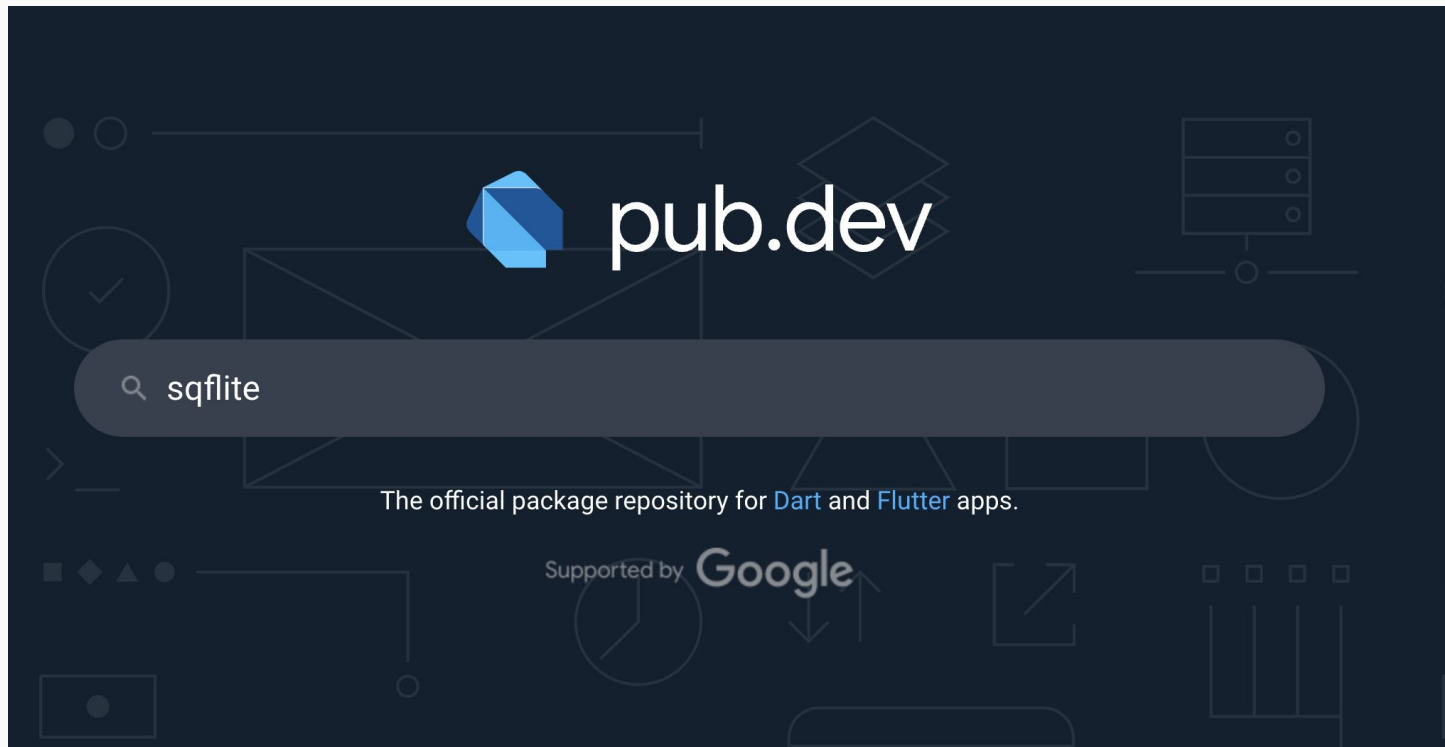
# 데이터베이스 처리를 도와주는 패키지 설치하기

## 데이터베이스 처리를 도와주는 패키지 설치하기

- 안드로이드와 iOS 모두 SQLite 데이터베이스를 지원하지만, 이를 사용하려면 각 플랫폼에 특화된 코드가 필요합니다. (안드로이드 : Java/Kotlin, iOS: Objective-C/Swift)
- **sqflite** 패키지는 각 플랫폼에서 데이터베이스를 사용할 때 필요한 코드 포함하고 있습니다. 따라서, 이를 사용하면 Flutter 단에서 Dart 코드만으로 데이터베이스를 처리할 수 있습니다.

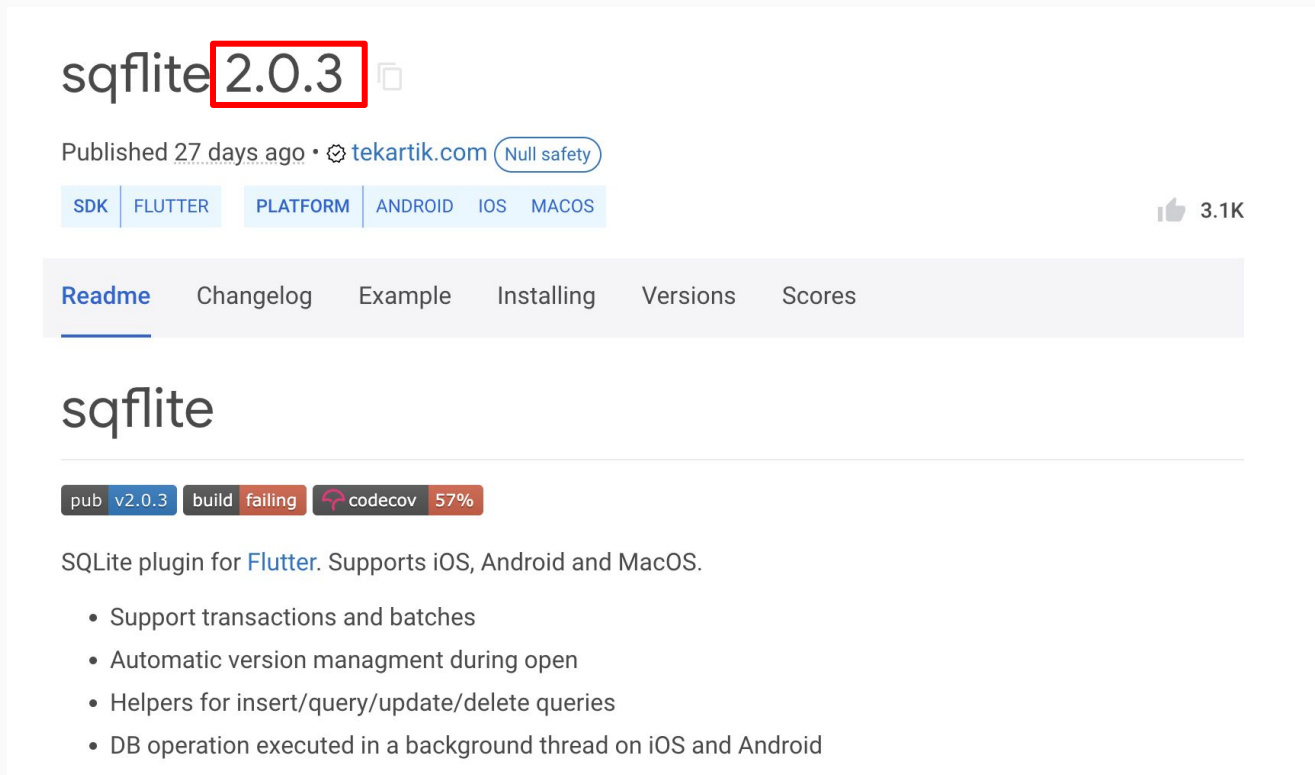
# 데이터베이스 처리를 도와주는 패키지 설치하기

pub.dev에서 **sqlite**를 검색합니다.



# 데이터베이스 처리를 도와주는 패키지 설치하기

최신 버전을 확인합니다. (아래 스크린샷 기준 2.0.3)



The screenshot shows the pub.dev page for the sqflite package. The package name 'sqflite' is followed by the version '2.0.3', which is highlighted with a red box. Below the name, it says 'Published 27 days ago' and 'tekartik.com' with a 'Null safety' badge. There are tabs for 'SDK', 'FLUTTER', 'PLATFORM', 'ANDROID', 'IOS', and 'MACOS'. The 'FLUTTER' tab is selected. To the right, there is a thumbs up icon and '3.1K'. Below the tabs, there are links for 'Readme', 'Changelog', 'Example', 'Installing', 'Versions', and 'Scores'. The 'Readme' link is underlined. Below the links, the package name 'sqflite' is repeated. Underneath, there are status bars for 'pub v2.0.3', 'build failing', and 'codecov 57%'. The description reads 'SQLite plugin for Flutter. Supports iOS, Android and MacOS.' followed by a bulleted list of features.

sqflite 2.0.3

Published 27 days ago • [tekartik.com](#) Null safety

[SDK](#) [FLUTTER](#) [PLATFORM](#) [ANDROID](#) [IOS](#) [MACOS](#) 👍 3.1K

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#)

## sqflite

pub v2.0.3 build failing codecov 57%

SQLite plugin for [Flutter](#). Supports iOS, Android and MacOS.

- Support transactions and batches
- Automatic version management during open
- Helpers for insert/query/update/delete queries
- DB operation executed in a background thread on iOS and Android

# 데이터베이스 처리를 도와주는 패키지 설치하기

pubspec.yaml 파일을 연 후, dependencies 섹션에 sqflite 패키지를 추가합니다.

(주의: dependencies 섹션에 패키지를 추가할 때 인덴트로 스페이스 2칸을 추가해야 합니다)

```
dependencies:  
  flutter:  
    sdk: flutter  
  sqflite: ^2.0.3
```

↑ 인덴트로 스페이스 2칸을 넣습니다. (적거나 많으면 오류 발생)

## 데이터베이스 처리를 도와주는 패키지 설치하기

상단의 Pub get 버튼을 눌러 sqflite 패키지를 프로젝트에 설치합니다.



# 비동기 작업 처리하기



# 동기(synchronous) vs. 비동기(asynchronous) 작업

- 동기 작업

- 현재 진행중인 작업이 완전히 완료되기 전 까지는 다음 작업이 실행되지 않습니다.
- 예: 실행 순서가 중요한 연산 (계산 등)

- 비동기 작업

- 현재 진행중인 작업이 완전히 완료되지 않았더라도 다른 작업을 실행할 수 있습니다.
- 예)
  - 네트워크 요청을 통해 데이터를 받아야하는 작업
  - 데이터베이스 읽기/쓰기 연산을 포함하는 작업

# Future

- 특정 값을 반환하는 비동기 작업을 표현할 때 사용합니다.
- 작업 실행 상태에 따라 다음과 같이 나뉩니다.
  - Uncompleted: 작업이 실행 중이며 아직 완료되지 않은 상태입니다.
  - Completed with a value: 작업이 정상적으로 완료되어 결과값을 반환한 상태입니다.
  - Completed with an error: 작업을 실행하던 도중 오류가 발생하여 작업을 완료하지 못한 상태입니다.

# Future 사용하기

```
Future<void> foo() {  
    // 2초 뒤에 인자로 전달받은 함수 블록을 실행하는 Future를 생성합니다.  
    return Future.delayed(const Duration(seconds: 2), () {  
        print("Called after 2 seconds");  
    });  
}  
  
void main() {  
    foo();  
    print("Waiting the future to be completed");  
}
```

## Output

Waiting the future to be completed  
Called after 2 seconds

2초 지연

# Future 반환값 사용하기

```
Future<String> foo() {  
    // 2초 뒤에 "Called after 2 seconds" 문자열을 반환하는 Future를 생성합니다.  
    return Future.delayed(const Duration(seconds: 2), () {  
        return "Called after 2 seconds";  
    });  
}  
  
void main() {  
    foo().then((value) {  
        print(value);  
    });  
    print("Waiting the future to be completed");  
}
```

## Output

Waiting the future to be completed  
Called after 2 seconds

2초 지연

## async & await

- await
  - 비동기 작업이 완료될 때까지 다음 작업을 실행하지 않고 대기합니다.
- async
  - await을 사용하는 함수에 붙여줍니다. 함수가 비동기 작업을 포함하고 있다는 것을 의미합니다.

# async/await 사용하기

```
Future<String> foo() {  
    // 2초 뒤에 "Called after 2 seconds" 문자열을 반환하는 Future를 생성합니다.  
    return Future.delayed(const Duration(seconds: 2), () {  
        return "Called after 2 seconds";  
    });  
}  
  
void main() async {  
    print(await foo());  
    print("Waiting the future to be completed");  
}
```

## Output

Called after 2 seconds

Waiting the future to be completed

2초 지연

데이터베이스를 사용하도록  
노트 앱 수정하기

## 데이터베이스를 사용하도록 노트 앱 수정하기

- 노트 데이터를 데이터베이스를 통해 처리할 수 있도록 **Note** 클래스와 **NoteService** 클래스를 수정합니다.
- 변경된 **NoteService** 인터페이스에 맞게 노트 목록, 보기, 수정 페이지를 수정합니다.



# 데이터베이스를 사용하도록 노트 앱 수정하기

다음과 같이 노트 테이블을 정의합니다.

필드 이름	자료형	비고
id	INTEGER	Primary key, 새 데이터가 입력되면 자동으로 새로운 ID 부여
title	TEXT	노트 제목
body	TEXT NOT NULL	노트 본문
color	INTEGER NOT NULL	노트 색상 (Color.value)

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class Note {  
  
  static const tableName = 'notes';  
  
  static const columnId = 'id';  
  
  static const columnTitle = 'title';  
  
  static const columnBody = 'body';  
  
  static const columnColor = 'color';  
  
  final int? id;  
  
  Note(  
    this.body, {  
    this.id,  
    this.title = '',  
    this.color = colorDefault,  
  });  
  
  ...  
}
```

note.dart

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class Note {  
  ...  
  
  Note.fromRow(Map<String, dynamic> row)  
    : this(  
      row[columnBody],  
      id: row[columnId],  
      title: row[columnTitle],  
      color: Color(row[columnColor]),  
    );  
  
  Map<String, dynamic> toRow() {  
    return {  
      columnTitle: title,  
      columnBody: body,  
      columnColor: color.value,  
    };  
  }  
}
```

note.dart

# 데이터베이스를 사용하도록 노트 앱 수정하기

## 장보기 목록

- 🍅 양파 1망
- 🥬 양배추 1통
- 🍊 귤 1박스
- 🐔 손질 닭고기 1팩
- 🥓 삼겹살 1팩
- 🍜 우동면 1팩 (4입)
- 🥛 우유 2팩
- 🍞 식빵 1개

Note 클래스

Note.color = fromRow()

fromRow()

레코드

id	title	body	color
1	장보기 목록	- 🍅 양파 1망..	4293981379

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class NoteService {  
  static const _databaseName = 'notes.db';  
  
  static const _databaseVersion = 1;  
  
  Database? _database;  
  
  ...  
}
```

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class NoteService {  
  ...  
  Future<Database> _getDatabase() async {  
    if (_database == null) {  
      _database = await openDatabase(  
        _databaseName,  
        version: _databaseVersion,  
        onCreate: (db, version) {  
          const sql = '''  
CREATE TABLE ${Note.tableName} (  
  ${Note.columnId} INTEGER PRIMARY KEY AUTOINCREMENT,  
  ${Note.columnTitle} TEXT,  
  ${Note.columnBody} TEXT NOT NULL,  
  ${Note.columnColor} INTEGER NOT NULL  
);  
''';  
          return db.execute(sql);  
        },  
      );  
    }  
    return _database!;  
  }  
}
```

note\_service.dart

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class NoteService {  
  ...  
  Future<void> addNote(Note note) async {  
    final db = await _getDatabase();  
    await db.insert(Note.tableName, note.toRow());  
  }  
  
  Future<void> deleteNote(int id) async {  
    final db = await _getDatabase();  
    await db.delete(  
      Note.tableName,  
      where: '${Note.columnId} = ?',  
      whereArgs: [id],  
    );  
  }  
  ...  
}
```

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class NoteService {  
  ...  
  Future<Note> getNote(int id) async {  
    final db = await _getDatabase();  
    final rows = await db.query(  
      Note.tableName,  
      where: '${Note.columnId} = ?',  
      whereArgs: [id],  
    );  
    return Note.fromRow(rows.single);  
  }  
  
  Future<List<Note>> listNotes() async {  
    final db = await _getDatabase();  
    final rows = await db.query(Note.tableName);  
    return rows.map((row) => Note.fromRow(row)).toList();  
  }  
  ...  
}
```

note\_service.dart



# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class NoteService {  
  ...  
  Future<void> updateNote(int id, Note note) async {  
    final db = await _getDatabase();  
    await db.update(  
      Note.tableName,  
      note.toRow(),  
      where: '${Note.columnId} = ?',  
      whereArgs: [id],  
    );  
  }  
  ...  
}
```

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class _NoteListPageState extends State<NoteListPage> {  
  ...  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      ...  
      body: FutureBuilder<List<Note>>(  
        future: noteService().listNotes(),  
        builder: (context, snap) {  
          if (snap.connectionState == ConnectionState.waiting) {  
            return const Center(  
              child: CircularProgressIndicator(),  
            );  
          }  
  
          if (snap.hasError) {  
            return const Center(  
              child: Text('오류가 발생했습니다.'),  
            );  
          }  
  
          final notes = snap.requireData;  
          return _buildCards(notes);  
        },  
      ),  
    );  
  }  
}
```

note\_list\_page.dart

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
Widget _buildCards(List<Note> notes) {  
  return _showAsGrid  
    ? GridView.builder(  
      padding:  
        const EdgeInsets.symmetric(horizontal: 12.0, vertical: 16.0),  
      itemBuilder: (context, index) => _buildCard(notes[index]),  
      itemCount: notes.length,  
      gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(  
        crossAxisCount: 2,  
        childAspectRatio: 1,  
      ),  
    )  
    : ListView.builder(  
      padding:  
        const EdgeInsets.symmetric(horizontal: 12.0, vertical: 16.0),  
      itemCount: notes.length,  
      itemBuilder: (context, index) {  
        return SizedBox(  
          height: 160,  
          child: _buildCard(notes[index]),  
        );  
      },  
    );  
}
```

## 데이터베이스를 사용하도록 노트 앱 수정하기

```
Widget _buildCard(Note note) {  
  return InkWell(  
    onTap: () {  
      Navigator.pushNamed(  
        context,  
        NoteViewPage.routeName,  
        arguments: note.id,  
      ).then((value) {  
        setState(() {});  
      });  
    },  
    ...  
  );  
}
```

note\_list\_page.dart

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class NoteViewPage extends StatefulWidget {  
  static const routeName = '/view';  
  
  final int id;  
  
  const NoteViewPage(this.id, {Key? key}) : super(key: key);  
  
  @override  
  State createState() => _NoteViewPageState();  
}
```

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class _NoteViewState extends State<NoteViewPage> {  
  @override  
  Widget build(BuildContext context) {  
    return FutureBuilder<Note>(  
      future: noteService().getNote(widget.id),  
      builder: (context, snap) {  
        if (snap.connectionState == ConnectionState.waiting) {  
          return const Center(  
            child: CircularProgressIndicator(),  
          );  
        }  
  
        if (snap.hasError) {  
          return Scaffold(  
            appBar: AppBar(),  
            body: const Center(  
              child: Text('오류가 발생했습니다'),  
            ),  
          );  
        }  
  
        final note = snap.requireData;  
        return Scaffold( ... );  
      },  
    );  
  }  
}
```

note\_view\_page.dart

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class NoteEditPage extends StatefulWidget {  
  static const routeName = '/edit';  
  
  final int? id;  
  
  const NoteEditPage(this.id, {Key? key}): super(key: key);  
  
  @override  
  State createState() => _NoteEditPageState();  
}
```

# 데이터베이스를 사용하도록 노트 앱 수정하기

```
class _NoteEditPageState extends State<NoteEditPage> {  
  ...  
  @override  
  void initState() {  
    super.initState();  
    final noteId = widget.id;  
    if (noteId != null) {  
      noteService().getNote(noteId).then((note) {  
        titleController.text = note.title;  
        bodyController.text = note.body;  
        setState(() {  
          color = note.color;  
        });  
      });  
    }  
  }  
  ...  
}
```

note\_edit\_page.dart



## Sticky Notes

**About Flutter**

Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google

**(제목 없음)**

부트캠프 신청하기

**(제목 없음)**

목요일까지 관리비 납부하기

**만들어야 할 것들**

- 파이어베이스 프로젝트
- 애드몹 계정
- 구글 플레이 개발자 계정
- 애플 개발자 계정



# Firebase 소개

**Firestore is**  
**Google's app**  
**development platform**

**Our mission is to**  
**help app developers succeed**



- 사용자가 앱을 어떻게, 얼마나, 어디에서 유입되어 사용하는지 파악할 수 있습니다.
- 앱에서 발생하는 수익 (인앱결제 및 광고수익)을 한눈에 파악할 수 있습니다.
- 비슷한 성향을 가진 사용자를 그룹으로 나누어 분류할 수 있습니다.



# Firestore A/B Testing

- 앱의 구현/동작에 따른 사용자의 반응을 실험할 수 있는 제품입니다.
- 사용 예
  - 어떤 색상/모양의 버튼이 사용자의 관심을 많이 끄는지 알고 싶을 때
  - 새로운 광고를 추가했을 때, 사용자에게 부정적인 영향이 있는지 확인하고 싶을 때



# Firebase Remote Config

- 앱을 다시 배포하지 않아도 앱의 동작을 실시간으로 변경할 수 있습니다.
- 사용 예
  - 특정 기간 중에만 게임 내 주어지는 보상의 양을 조정하고 싶을 때
  - 국가별로 사용 가능한 메뉴를 조정하고 싶을 때



# Firebase Cloud Messaging

- 푸시 메시지를 전송할 수 있는 제품입니다.
- 안드로이드, iOS, 웹 앱을 대상으로 푸시 메시지를 전송할 수 있습니다.



# Firebase CLI 설치




# Firebase CLI (Command Line Interface)

- Firebase 프로젝트와 관련된 다양한 작업을 할 수 있는 명령줄(command line) 도구입니다.
- Flutter 애플리케이션에 Firebase 프로젝트를 자동으로 연결할 때 사용할 도구인 **flutterfire\_cli** 를 사용하려면 Firebase CLI를 미리 설치해야 합니다.

# Firebase CLI 설치하기 (Windows)

- Firebase CLI를 구동할 때 필요한 추가 도구인 nvm-windows를 설치해야 합니다.
- [이 링크](#)를 클릭하여 nvm-windows 설치 파일을 받을 수 있는 페이지로 이동합니다.

**1.1.9** Latest Compare

 coreybutler released this Dec 11, 2021 · 17 commits to master since this release  

NOTICE: If you downloaded this version before Dec 15, 2021 and are affected by issue [#706](#), please re-download. This fix was not present in the initial binary and has since been corrected.

### What's Changed

- Code signed release.
- Fix broken link to nodejs versions by [@crhstianramirez](#) in [#679](#)
- Fix readme links. by [@veleek](#) in [#689](#)
- docs: update address of china mirror by [@hyj1991](#) in [#692](#)
- Use Windows defined env vars in all paths in README by [@gdziadkiewicz](#) in [#699](#)

### New Contributors

- [@crhstianramirez](#) made their first contribution in [#679](#)
- [@veleek](#) made their first contribution in [#689](#)
- [@hyj1991](#) made their first contribution in [#692](#)
- [@gdziadkiewicz](#) made their first contribution in [#699](#)
- [@ajyong](#) sponsored the code signing certificate

Full Changelog: [1.1.8...1.1.9](#)

### UPDATE 4/27/22

Added nvm-setup.exe after initial release to support winget efforts.

# Firebase CLI 설치하기 (Windows)

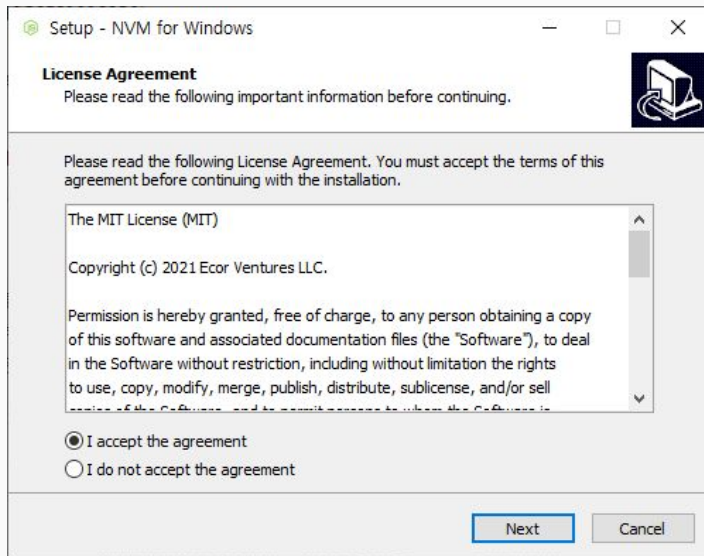
설치 파일 (nvm-setup.exe)을 다운로드 합니다.

## ▼ Assets 9

 <a href="#">nvm-noinstall.zip</a>	3.64 MB	Dec 16, 2021
 <a href="#">nvm-noinstall.zip.checksum.txt</a>	34 Bytes	Dec 16, 2021
 <a href="#">nvm-setup.exe</a>	4.64 MB	Apr 28, 2022
 <a href="#">nvm-setup.zip</a>	4.14 MB	Dec 16, 2021
 <a href="#">nvm-setup.zip.checksum.txt</a>	34 Bytes	Dec 16, 2021
 <a href="#">nvm-update.zip</a>	3.45 MB	Dec 16, 2021
 <a href="#">nvm-update.zip.checksum.txt</a>	34 Bytes	Dec 16, 2021
 <a href="#">Source code</a> (zip)		Dec 11, 2021
 <a href="#">Source code</a> (tar.gz)		Dec 11, 2021

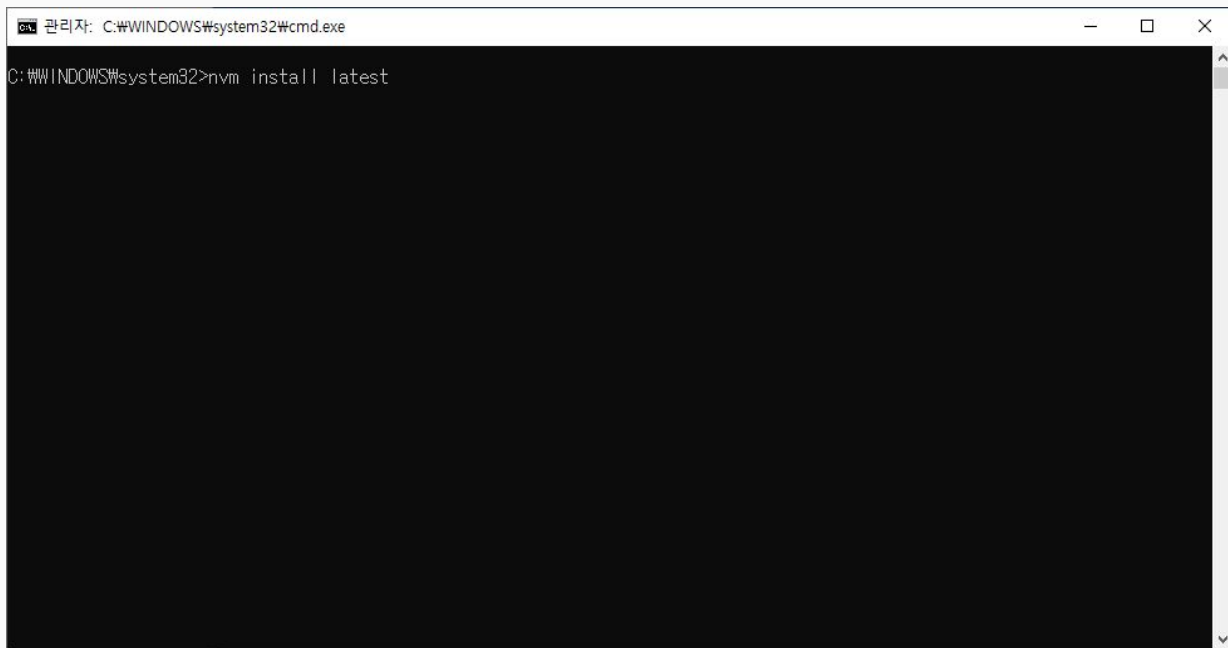
# Firestore CLI 설치하기 (Windows)

설치 마법사가 실행되면, 기본 옵션 그대로 설치를 완료합니다.



# Firestore CLI 설치하기 (Windows)

**관리자 모드**로 명령 프롬프트를 실행한 후, **`nvm install latest`** 명령어를 실행하여 Node.js 런타임을 설치합니다. (Firestore CLI 실행에 필요한 구성요소)



```
관리자: C:\WINDOWS\system32\cmd.exe
C:\WINDOWS\system32>nvm install latest
```

# Firebase CLI 설치하기 (Windows)

설치가 완료되면 설치된 버전이 화면이 표시됩니다. `nvm use [설치된 버전]` 명령을 실행하여 설치한 버전을 활성화합니다. (예: 설치한 버전이 18.7.0인 경우 `nvm use 18.7.0` 실행)



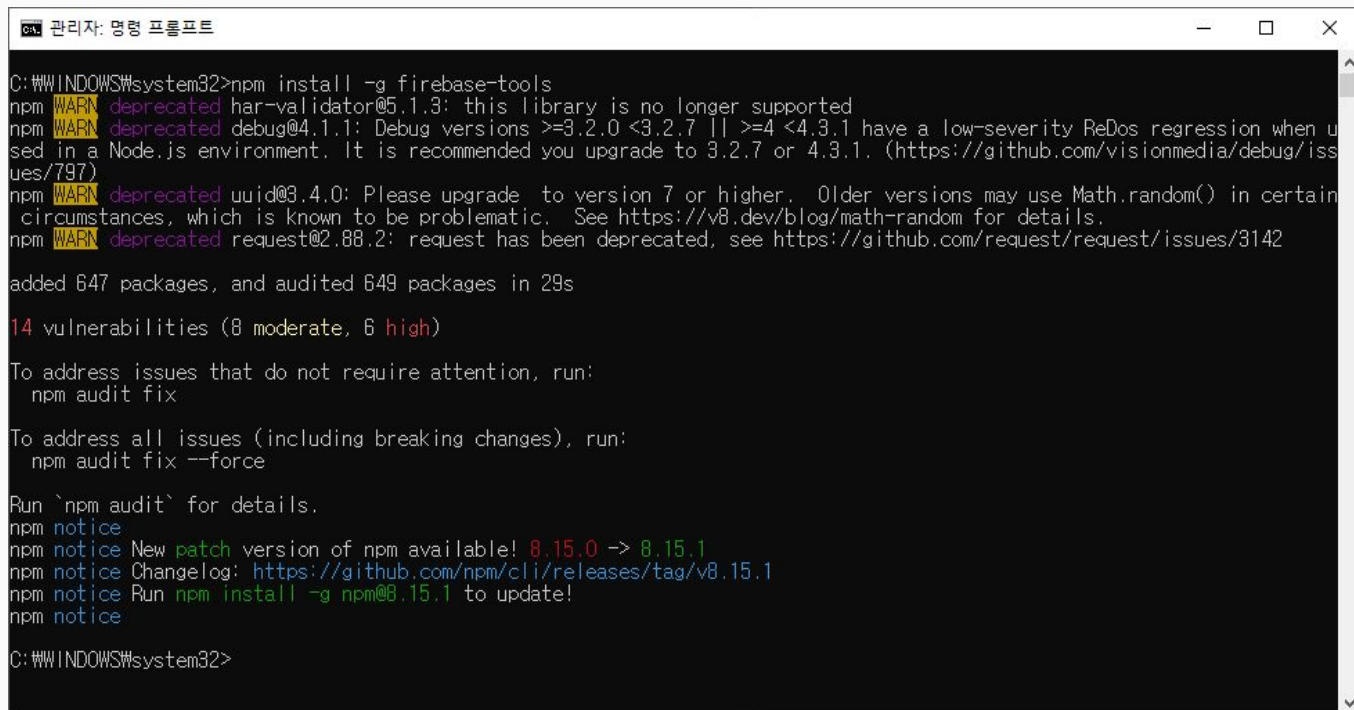
```
관리자: 명령 프롬프트
Microsoft Windows [Version 10.0.19044.1826]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>nvm use 18.7.0
Now using node v18.7.0 (64-bit)

C:\WINDOWS\system32>
```

# Firebase CLI 설치하기 (Windows)

`npm install -g firebase-tools` 명령을 실행하여 Firebase CLI를 설치합니다.



```
C:\WINDOWS\system32>npm install -g firebase-tools
npm WARN deprecated har-validator@5.1.3: this library is no longer supported
npm WARN deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.3.1 have a low-severity ReDos regression when used in a Node.js environment. It is recommended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/debug/issues/797)
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 647 packages, and audited 649 packages in 29s

14 vulnerabilities (8 moderate, 6 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New patch version of npm available! 8.15.0 -> 8.15.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.15.1
npm notice Run npm install -g npm@8.15.1 to update!
npm notice
C:\WINDOWS\system32>
```



## Firebase CLI 설치하기 (macOS)

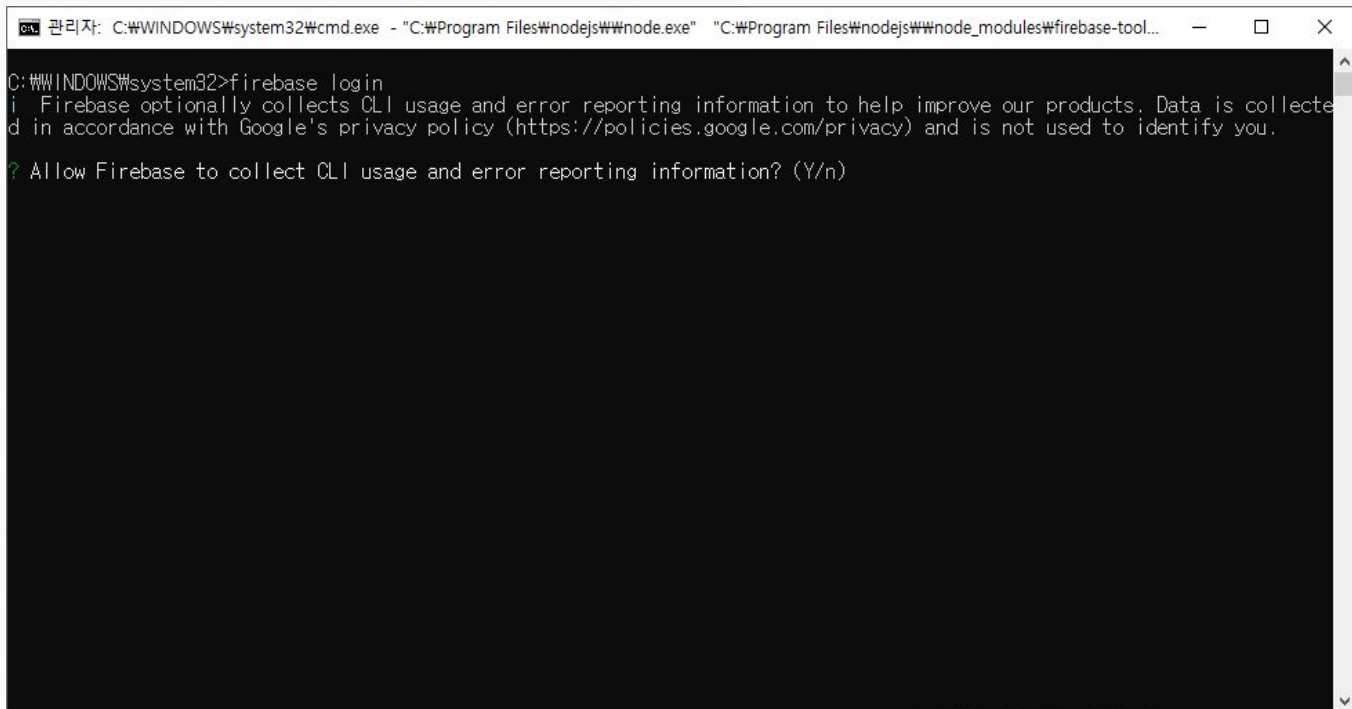
터미널을 연 후, 아래 명령어를 실행하여 Firebase CLI를 설치합니다.

```
curl -sL https://firebase.tools | bash
```

# Firestore CLI 구성 및 flutterfire\_cli 설치

# Firebase CLI 구성 및 flutterfire\_cli 설치

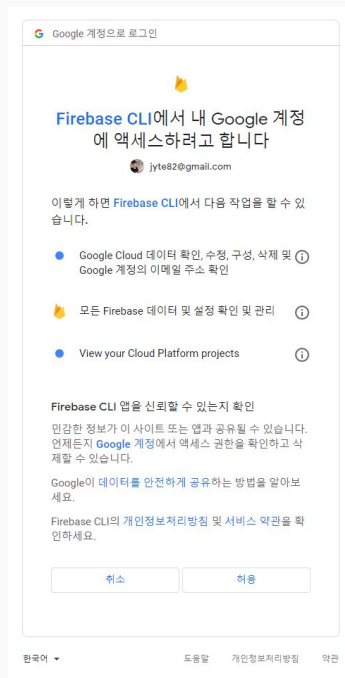
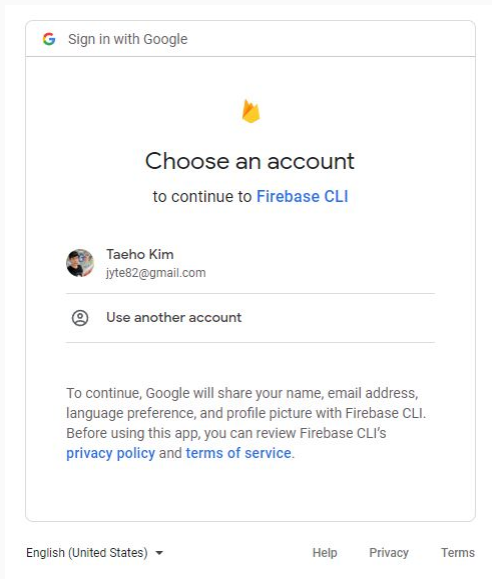
명령 프롬프트/터미널을 실행한 후, **firebase login** 명령을 실행합니다.



```
관리자: C:\WINDOWS\system32\cmd.exe - "C:\Program Files\nodejs\node.exe" "C:\Program Files\nodejs\node_modules\firebase-tool...
C:\WINDOWS\system32>firebase login
i Firebase optionally collects CLI usage and error reporting information to help improve our products. Data is collected
in accordance with Google's privacy policy (https://policies.google.com/privacy) and is not used to identify you.
? Allow Firebase to collect CLI usage and error reporting information? (Y/n)
```

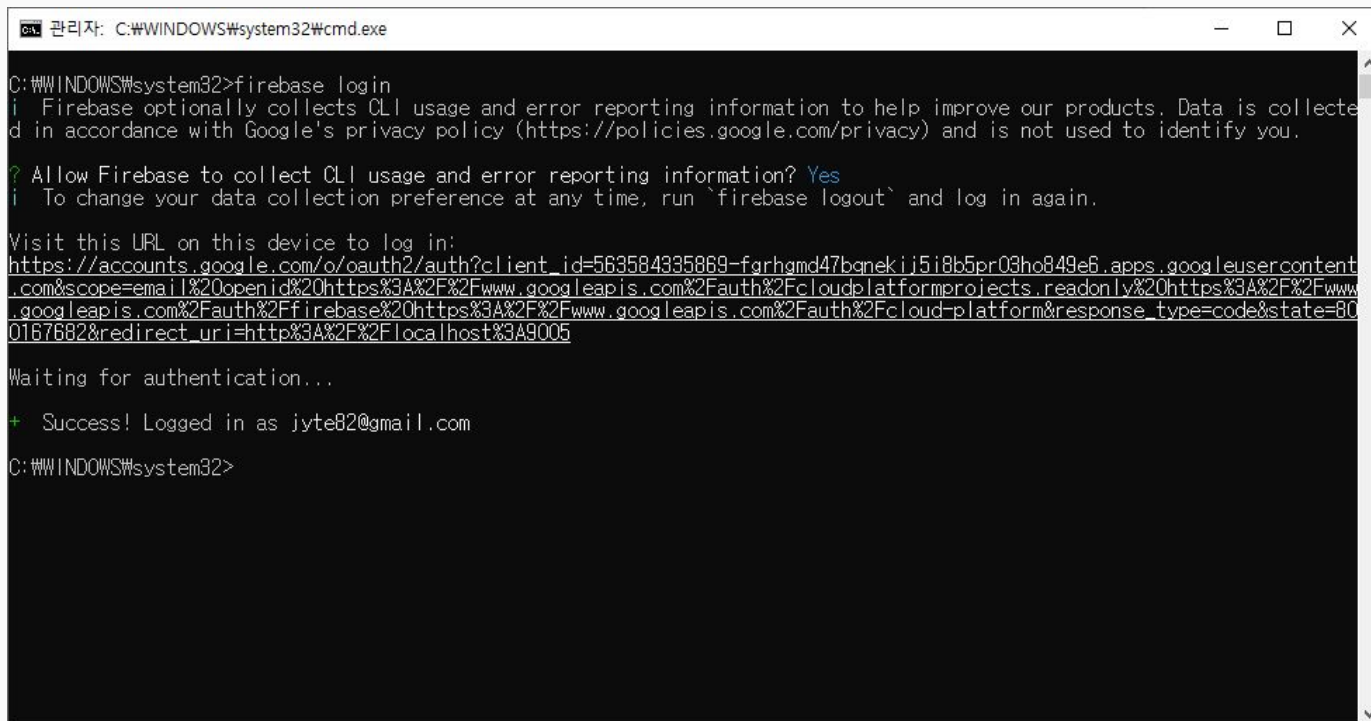
# Firebase CLI 구성 및 flutterfire\_cli 설치

웹 브라우저가 표시되며 로그인 화면이 표시됩니다. Firebase 프로젝트를 관리할 구글 계정으로 로그인한 후, Firebase CLI에서 계정과 연결된 파이어베이스 데이터에 접근할 수 있도록 권한을 부여합니다.



# Firebase CLI 구성 및 flutterfire\_cli 설치

로그인이 완료되면 아래와 같이 로그인 성공 메시지가 표시됩니다.



```
C:\WINDOWS\system32\cmd.exe

C:\WINDOWS\system32>firebase login
i  Firebase optionally collects CLI usage and error reporting information to help improve our products. Data is collected in accordance with Google's privacy policy (https://policies.google.com/privacy) and is not used to identify you.

? Allow Firebase to collect CLI usage and error reporting information? Yes
i  To change your data collection preference at any time, run `firebase logout` and log in again.

Visit this URL on this device to log in:
https://accounts.google.com/o/oauth2/auth?client_id=563584335869-fgrhmd47bqnekij5i8b5pr03ho849e6.apps.googleusercontent.com&scope=email%20openid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform&response_type=code&state=8001b7682&redirect_uri=http%3A%2F%2Flocalhost%3A9005

Waiting for authentication...

+ Success! Logged in as jyte82@gmail.com

C:\WINDOWS\system32>
```

## Firebase CLI 구성 및 flutterfire\_cli 설치

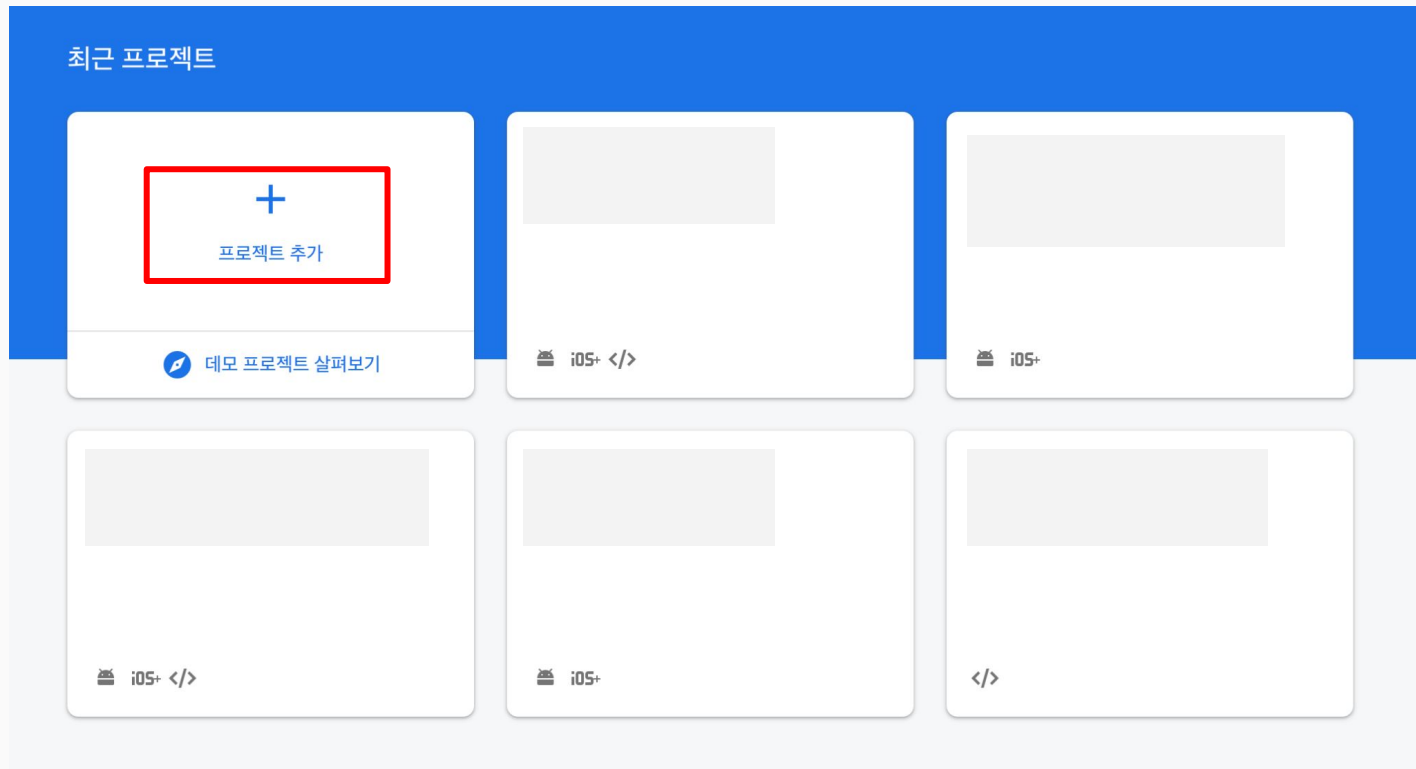
다음 명령어를 실행하여 Flutter 앱에 파이어베이스 프로젝트를 자동으로 연결해주는 도구인 **flutterfire\_cli**를 설치합니다.

```
dart pub global activate flutterfire_cli
```

# Firebase 프로젝트 생성하기

# Firebase 프로젝트 생성하기

[console.firebase.google.com](https://console.firebase.google.com)에 접속한 후, 프로젝트 추가 버튼을 누릅니다.





# Firebase 프로젝트 생성하기

프로젝트 이름을 지정합니다. (예: androidhuman-sticky-notes)

× 프로젝트 만들기(1/3단계)

프로젝트 이름을 지정하여 시작하  
기<sup>②</sup>

프로젝트 이름 입력

my-awesome-project-id

계속



# Firebase 프로젝트 생성하기



Google 애널리틱스를 사용 설정합니다.



## Firebase 프로젝트를 위한 Google 애널리틱스



무제한 무료 분석 솔루션인 Google 애널리틱스를 사용하면 Firebase 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 

 장애가 발생하지 않은 사용자 

 Firebase 제품 전반에서 사용자 세분화 및 타  
겟팅 

 이벤트 기반 Cloud Functions 트리거 

 제한 없는 무료 보고 

☒ 이 프로젝트에서 Google 애널리틱스 사용 설정  
권장

[이전](#)


[계속](#)

# Firebase 프로젝트 생성하기

Google 애널리틱스 계정을 선택하거나 새 계정을 생성한 후, 프로젝트 만들기 버튼을 누릅니다.

## Google 애널리틱스 구성

Google 애널리틱스 계정 선택 또는 만들기 ?



새 계정 만들기

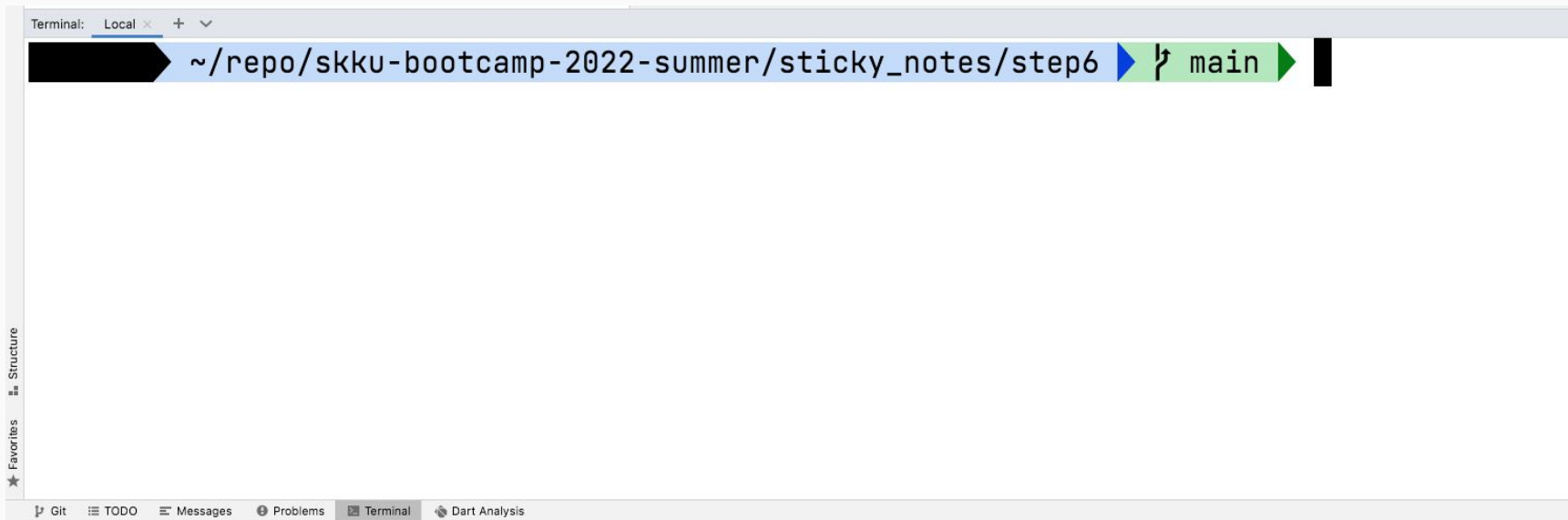
Google 애널리틱스 속성이 생성되고 Firebase 프로젝트에 연결됩니다. 이 연결을 통해 제품  
base로 내보낸 데이터에는 Firebase 서비스 약관이 적용되지만 Google 애널리  
이 적용됩니다. [자세히 알아보기](#)

프로젝트 만들기

# 노트 앱 프로젝트에 Firebase 설정하기

# 노트 앱 프로젝트에 Firebase 설정하기

안드로이드 스튜디오 하단의 Terminal 탭을 클릭하여 터미널을 실행합니다.



# 노트 앱 프로젝트에 Firebase 설정하기

`flutter pub add firebase_core` 명령어를 실행하여 프로젝트에 파이어베이스 패키지를 추가합니다.

```
$ flutter pub add firebase_core
Resolving dependencies...
  async 2.8.2 (2.9.0 available)
  characters 1.2.0 (1.2.1 available)
  clock 1.1.0 (1.1.1 available)
  fake_async 1.3.0 (1.3.1 available)
  matcher 0.12.11 (0.12.12 available)
  material_color_utilities 0.1.4 (0.1.5 available)
  meta 1.7.0 (1.8.0 available)
  path 1.8.1 (1.8.2 available)
  source_span 1.8.2 (1.9.1 available)
  string_scanner 1.1.0 (1.1.1 available)
  term_glyph 1.2.0 (1.2.1 available)
  test_api 0.4.9 (0.4.12 available)
Got dependencies!
```

# 노트 앱 프로젝트에 Firebase 설정하기

[pubspec.yaml](#) 의 dependencies 섹션에 firebase\_core 패키지가 추가되었는지 확인합니다.

```
dependencies:  
  firebase_core: ^1.20.0  
  flutter:  
    sdk: flutter  
  sqflite: ^2.0.3
```

# 노트 앱 프로젝트에 Firebase 설정하기

다시 터미널 탭을 열고, **flutterfire configure** 명령어를 실행하여 파이어베이스 프로젝트를 연결을 시작합니다. 표시되는 프로젝트 중 연결할 프로젝트를 선택합니다.

```
i Found 9 Firebase projects.  
? Select a Firebase project to configure your Flutter application with ›  
  androidhuman-blog (androidhuman)  
  androidhuman-cubewhere (Cubewhere)  
  androidhuman-gc (androidhuman-gc)  
  androidhuman-my-photo (androidhuman-my-photo)  
  androidhuman-mycloud (myCloud)  
  androidhuman-myplaces (myPlaces)  
  androidhuman-newtown3 (Newtown 3)  
  androidhuman-pb (Pregnancy benefits)  
› androidhuman-sticky-notes (androidhuman-sticky-notes)  
  tesla-price-calculator (Tesla Price Calculator)  
  <create a new project>
```



# 노트 앱 프로젝트에 Firebase 설정하기

연결할 플랫폼을 선택합니다. 안드로이드와 iOS를 선택한 후 다음 단계로 진행합니다.

? Which platforms should your configuration support (use arrow keys & space to `select`)? ›

✓ android

✓ ios

macos

web

# 노트 앱 프로젝트에 Firebase 설정하기

Flutter 프로젝트에 있는 안드로이드/iOS 앱 설정을 기반으로 Firebase 프로젝트에 앱이 추가되고, 이 설정 정보가 담긴 파일 (**firebase\_options.dart**)이 프로젝트에 추가됩니다.

```
i Firebase android app com.androidhuman.skku.stickynotes registered.  
i Firebase ios app com.androidhuman.skku.stickynotes registered.
```

Firebase configuration file lib/firebase\_options.dart generated successfully with the following Firebase apps:

Platform	Firebase App Id
android	1:356563551621:android:049331c3dfea914d51b0ec
ios	1:356563551621:ios:9757ebd3cd9e5d7e51b0ec

Learn more about using this file and next steps from the documentation:  
> <https://firebase.google.com/docs/flutter/setup>

# 노트 앱 프로젝트에 Firebase 설정하기

Firebase 서비스를 사용하려면 앱 시작 시점에 초기화가 필요합니다.

**Firestore.initializeApp()**을 호출하여 초기화를 해 줍니다.

```
void main() async {  
  WidgetsFlutterBinding.ensureInitialized();  
  await Firestore.initializeApp(options: DefaultFirestoreOptions.currentPlatform);  
  runApp(MyApp());  
}
```

main.dart

## 완성된 코드

[https://github.com/kunny/skku-bootcamp-2022-summer/tree/main/sticky\\_notes/step5](https://github.com/kunny/skku-bootcamp-2022-summer/tree/main/sticky_notes/step5)