

앱 개발 입문

성균관대학교 컬처애펬크놀로지융합전공 하계 부트캠프

6일차 - 2023. 07. 07 (금)

강의 슬라이드 링크

https://github.com/kunny/skku-bootcamp-2023-summer/blob/main/_slides/day6.pdf

오늘 강의에서 다룰 내용

- 앱 분석 및 최적화에 사용하는 도구인 Firebase를 소개합니다.
- 노트 앱에 Firebase를 적용합니다.

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.



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

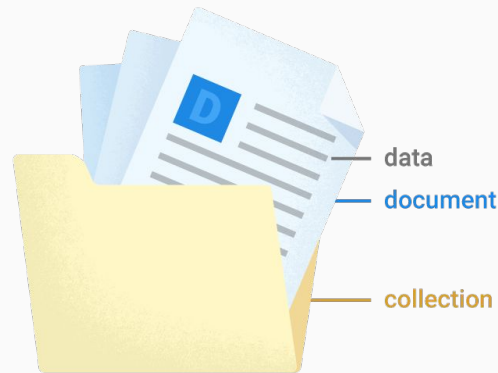


Firestore A/B Testing

- 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

Cloud Firestore

- NoSQL 기반 데이터베이스로, 데이터 저장 및 실시간 동기화를 지원합니다.
 - collection - document 기반 구조로 이루어져 있습니다.
- Firebase의 다른 제품과 연동하여 기능을 확장할 수 있습니다.





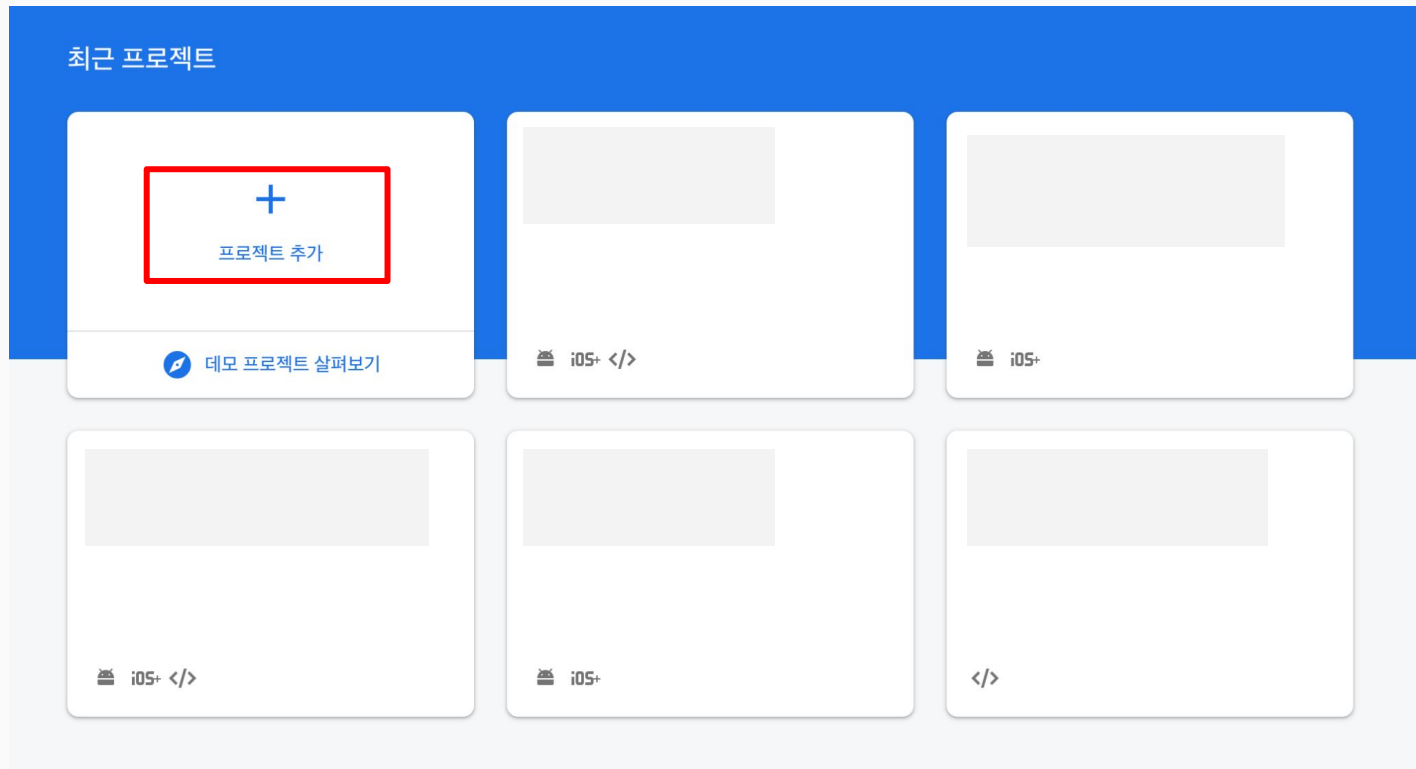
Firebase Cloud Messaging

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

Firebase 프로젝트 생성하기

Firebase 프로젝트 생성하기

console.firebase.google.com에 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^②

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 메시지, 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

Firestore 설정하기

Firestore 설정하기

파이어베이스 콘솔에 접속한 후, 왼쪽 메뉴에서 Firestore Database를 선택한 다음
데이터베이스 만들기 버튼을 누릅니다.

Cloud Firestore

실시간 업데이트, 강력한 쿼리, 자동 확장

데이터베이스 만들기



Firestore 설정하기

보안 규칙 설정 단계에서 **테스트 모드에서 시작** 버튼을 누릅니다.

데이터베이스 만들기

1 Cloud Firestore의 보안 규칙

2 Cloud Firestore 위치 설정

데이터 구조를 정의한 후 데이터에 보안을 적용하는 규칙을 작성해야 합니다.
[자세히 알아보기](#)

☐ 프로덕션 모드에서 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 이내에 보안 규칙을 업데이트하여 장기적 클라이언트 읽기/쓰기 액세스 권한을 사용 설정해야 합니다.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
        request.time < timestamp.date(2023, 8, 6);
    }
  }
}
```

!

테스트 모드의 기본 보안 규칙에서는 데이터베이스 참조 사용자는 누구나 향후 30일 동안 데이터베이스의 모든 데이터를 보고 수정하며 삭제할 수 있습니다.

Cloud Firestore를 사용 설정하면 이 프로젝트에서 Cloud Datastore를 사용할 수 없습니다.

취소

다음

Firestore 설정하기


Cloud Firestore 위치로 **nam5(United States)** 를 선택한 후, 사용 설정을 눌러 데이터베이스를 만듭니다.


데이터베이스 만들기

✓ Cloud Firestore의 보안 규칙

2 Cloud Firestore 위치 설정

위치 설정은 Cloud Firestore 데이터가 저장되는 위치를 결정합니다.

 이 위치를 설정한 후에는 나중에 변경할 수 없습니다. 또한 설정한 위치가 기본 Cloud Storage 버킷의 위치가 됩니다.

자세히 알아보기 

Cloud Firestore 위치

nam5 (United States) ▼

Cloud Firestore를 사용 설정하면 이 프로젝트에서 Cloud Datastore를 사용할 수 없습니다.

취소

사용 설정

Firestore 프로젝트에 안드로이드 앱 등록하기

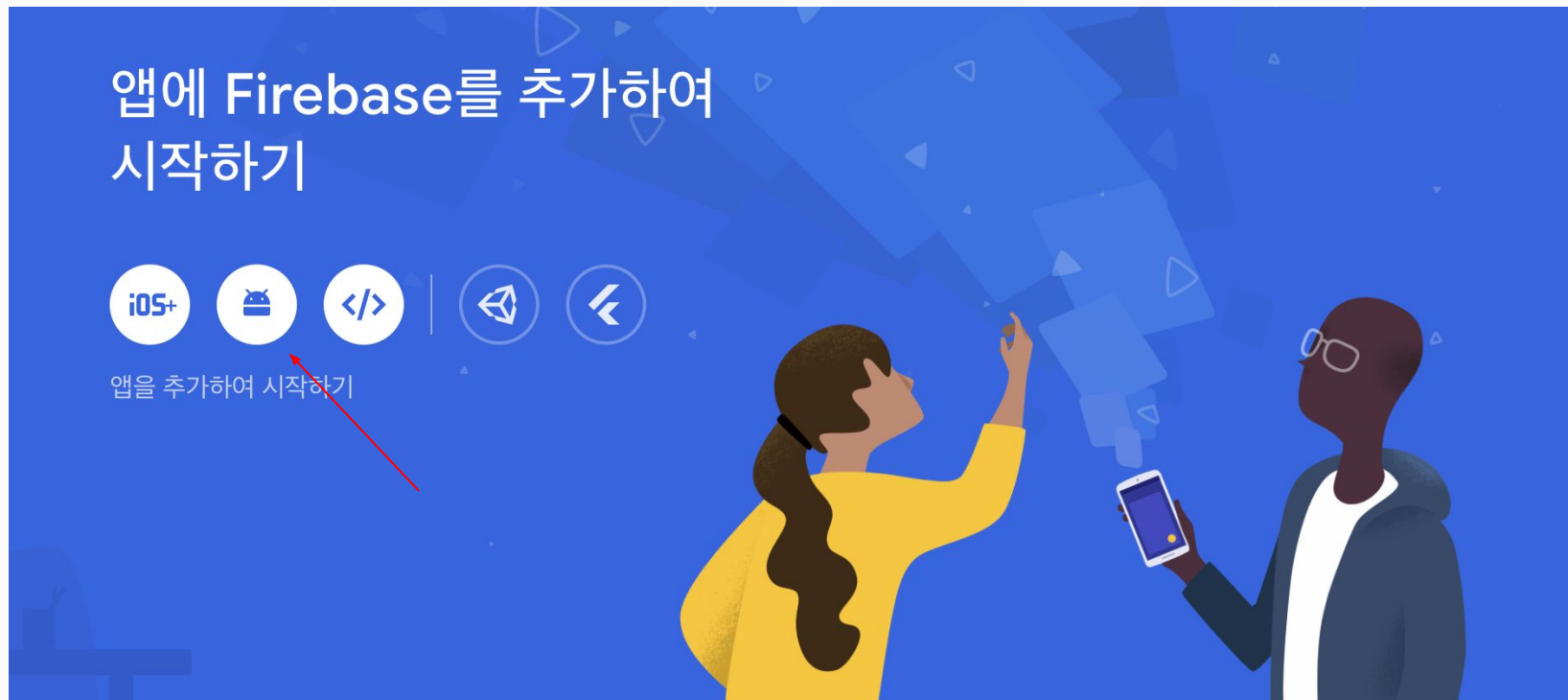
노트 앱 프로젝트에 Firebase 설정하기

android/app/build.gradle 파일 내 android > defaultConfig > applicationId 속성을 확인합니다.

```
android {  
    ...  
  
    defaultConfig {  
        applicationId "com.androidhuman.stickynotes"  
        minSdkVersion flutter.minSdkVersion  
        targetSdkVersion flutter.targetSdkVersion  
        versionCode flutterVersionCode.toInteger()  
        versionName flutterVersionName  
    }  
}
```

Firebase 프로젝트에 안드로이드 앱 등록하기

[파이어베이스 콘솔](#)에 접속한 후, 안드로이드 아이콘을 누릅니다.



Firebase 프로젝트에 안드로이드 앱 등록하기

Android 패키지 이름에 앞 단계에서 확인한 applicationId 값을 입력한 후, **앱 등록** 버튼을 누릅니다.

1 애플 등기

Android 패키지 이름 (?)

com.company.appname

앱 닉네임 (선택사항) (?)

내 Android 앱

디버그 서명 인증서 SHA-1(선택사항) ②

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:(

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

Firebase 프로젝트에 안드로이드 앱 등록하기

파이어베이스 프로젝트 구성 정보를 담고 있는 `google-services.json` 파일을 다운로드 한 후, `android/app` 폴더 아래에 넣어줍니다.

2 구성 파일 다운로드 후 추가

Android 스튜디오에 대한 안내(아래 참조) | [Unity](#) [C++](#)

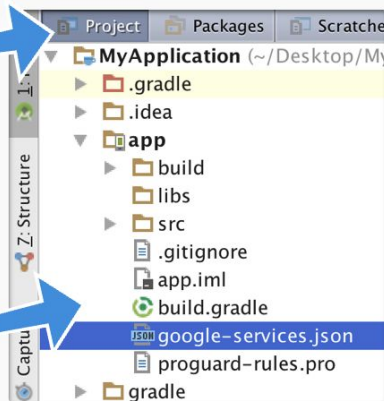
↓ `google-services.json` 다운로드

Android 스튜디오에서 프로젝트 뷰로 전환하여 프로젝트 루트 디렉터리를 표시합니다.

다운로드한 `google-services.json` 파일을 모듈(앱 수준) 루트 디렉터리로 이동합니다.



`google-services.json`



노트 앱 프로젝트에 Firebase 설정하기

노트 앱 프로젝트에 Firebase 설정하기

pubspec.yaml의 dependencies 섹션에 Firestore 구현에 필요한 라이브러리를 추가해줍니다.

```
name: sticky_notes
description: A new Flutter project.
publish_to: 'none'
version: 1.0.0+1

...

dependencies:
  flutter:
    sdk: flutter
  cloud_firestore: ^4.8.2
  firebase_core: ^2.14.0

...
```

노트 앱 프로젝트에 Firebase 설정하기

상단의 Pub get 버튼을 눌러 패키지를 프로젝트에 설치합니다.



노트 앱 프로젝트에 Firebase 설정하기

android/build.gradle 의 buildscript > dependencies 섹션에 google-services 라이브러리를 추가합니다.

```
buildscript {  
    dependencies {  
        classpath 'com.android.tools.build:gradle:7.3.0'  
        classpath 'com.google.gms:google-services:4.3.14'  
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"  
    }  
}  
  
allprojects {  
    ...  
}
```

노트 앱 프로젝트에 Firebase 설정하기

android/app/build.gradle 의 android 섹션 상단에 google-services 플러그인을 사용하도록 설정하고, Firebase를 원활히 사용할 수 있도록 minSdkVersion을 변경합니다.

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
apply plugin: 'kotlin-android'
apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"

android {
    defaultConfig {
        minSdkVersion 21
        ...
    }
}
```

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

비동기 작업 처리하기

동기(synchronous) vs. 비동기(asynchronous) 작업

- 동기 작업

- 현재 진행중인 작업이 완전히 완료되기 전까지는 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

파이어베이스를 사용하도록
노트 클래스 수정하기

파이어베이스를 사용하도록 노트 클래스 수정하기

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

파이어베이스를 사용하도록 노트 클래스 수정하기

```
class Note {  
  ...  
  
  Note.fromData(dynamic data)  
    : this(  
      data['body'],  
      title: data['title'],  
      color: Color(data['color']),  
    );  
  
  Map<String, dynamic> toData() {  
    return {  
      'title': title,  
      'body': body,  
      'color': color.value,  
    };  
  }  
}
```

note.dart

파이어베이스를 사용하도록 노트 클래스 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 = toData()

fromData()

Document data

title	body	color
장보기 목록	- 🍌 양파 1망..	4293981379

노트 목록 화면 수정하기

노트 목록 화면 수정하기

```
class _NoteListPageState extends State<NoteListPage> {  
  bool _showAsGrid = true;  
  
  late Future<List<DocumentSnapshot>> _listNotes;  
  
  @override  
  void initState() {  
    super.initState();  
    _loadData();  
  }  
  
  void _loadData() {  
    _listNotes = noteService().listNotes();  
  }  
}
```

노트 목록 화면 수정하기

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

note_list_page.dart

노트 목록 화면 수정하기

```
class _NoteListPageState extends State<NoteListPage> {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      ...  
      floatingActionButton: FloatingActionButton(  
        tooltip: '새 노트',  
        child: const Icon(Icons.add),  
        onPressed: () {  
          Navigator.pushNamed(context, NoteEditPage.routeName).then((value) {  
            setState(() {  
              _loadData();  
            });  
          });  
        },  
      ),  
    );  
  }  
}
```

note_list_page.dart

노트 목록 화면 수정하기

```
class _NoteListPageState extends State<NoteListPage> {  
  Widget _buildCard(String id, Note note) {  
    return InkWell(  
      onTap: () {  
        Navigator.pushNamed(  
          context,  
          NoteViewPage.routeName,  
          arguments: id,  
        ).then((value) {  
          setState(() {  
            _loadData();  
          });  
        });  
      },  
      ...  
    );  
  }  
}
```

노트 목록 화면 수정하기

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


노트 편집 화면 수정하기

노트 편집 화면 수정하기

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

노트 편집 화면 수정하기

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

note_edit_page.dart

노트 편집 화면 수정하기

```
class _NoteEditPageState extends State<NoteEditPage> {  
  ...  
  void _saveNote() {  
    if (_bodyController.text.isNotEmpty) {  
      ...  
  
      final noteId = widget.id;  
      Future<void> future;  
      if (noteId != null) {  
        future = noteService().updateNote(noteId, note);  
      } else {  
        future = noteService().addNote(note);  
      }  
  
      future.then((result) {  
        Navigator.pop(context);  
      });  
    } else { ... }  
  }  
}
```

note_edit_page.dart

노트 편집 화면 수정하기

```
class MyApp extends StatelessWidget {  
  const MyApp({super.key});  
  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      ...  
      routes: {  
        NoteEditPage.routeName: (context) {  
          final args = ModalRoute.of(context)!.settings.arguments;  
          final id = args != null ? args as String : null;  
          return NoteEditPage(id);  
        },  
        ...  
      },  
    );  
  }  
}
```

main.dart

노트 보기 화면 수정하기

노트 보기 화면 수정하기

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

노트 보기 화면 수정하기

```
class _NoteViewState extends State<NoteViewPage> {  
  late Future<DocumentSnapshot> _note;  
  
  @override  
  void initState() {  
    super.initState();  
    _loadData();  
  }  
  
  void _loadData() {  
    _note = noteService().getNote(widget.id);  
  }  
}
```


노트 보기 화면 수정하기

```
class _NoteViewState extends State<NoteViewPage> {  
  ...  
  void _edit(String id) {  
    Navigator.pushNamed(  
      context,  
      NoteEditPage.routeName,  
      arguments: id,  
    ).then((value) {  
      setState(() {  
        _loadData();  
      });  
    });  
  }  
}
```

노트 보기 화면 수정하기

```
class _NoteViewPageState extends State<NoteViewPage> {  
  ...  
  void _confirmDelete(String id) {  
    showDialog(context: context, builder: (context) {  
      return AlertDialog(  
        ...  
        actions: [  
          TextButton(  
            child: const Text('아니오'),  
            onPressed: () {  
              Navigator.pop(context);  
            },  
          ),  
          TextButton(  
            child: const Text('예'),  
            onPressed: () {  
              noteService().deleteNote(id).then((result) {  
                Navigator.popUntil(context, (route) => route.isFirst);  
              });  
            },  
          ),  
        ],  
      );  
    });  
  }  
}
```

노트 보기 화면 수정하기

```
class _NoteViewPageState extends State<NoteViewPage> {
  ...
  @override
  Widget build(BuildContext context) {
    return FutureBuilder<DocumentSnapshot>({
      future: _note,
      builder: (context, snap) {
        Widget title;
        Widget body;

        if (snap.connectionState == ConnectionState.waiting) {
          title = const Text('불러오는 중...');
          body = const CircularProgressIndicator();
        } else if (snap.hasError) {
          title = const Text('오류');
          body = const Text('정보를 불러오지 못했습니다.');
```

노트 보기 화면 수정하기

```
class _NoteViewPageState extends State<NoteViewPage> {  
  ...  
  @override  
  Widget build(BuildContext context) {  
    return FutureBuilder<DocumentSnapshot>(  
      future: _note,  
      builder: (context, snap) {  
        (앞 슬라이드에서 이어짐)  
        return Scaffold(  
          appBar: AppBar(  
            title: title,  
            actions: [  
              IconButton(  
                icon: const Icon(Icons.edit),  
                tooltip: '편집',  
                onPressed: () => _edit(widget.id),  
              ),  
              IconButton(  
                icon: const Icon(Icons.delete),  
                tooltip: '삭제',  
                onPressed: () => _confirmDelete(widget.id),  
              ),  
            ],  
          ),  
          body: body,  
        );  
      },  
    );  
  }  
}
```

note_edit_page.dart

노트 보기 화면 수정하기

```
class MyApp extends StatelessWidget {  
  const MyApp({super.key});  
  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      ...  
      routes: {  
        NoteViewPage.routeName: (context) {  
          final args = ModalRoute.of(context)!.settings.arguments;  
          final id = args != null ? args as String : null;  
          return NoteViewPage(id);  
        },  
        ...  
      },  
    );  
  }  
}
```

1:50

LTE



Sticky Notes

테스트 노트

테스트 노트입니다.



패널 뷰

쿼리 빌더



>

notes

>

yFU5jp0kvcvz8...



Google Cloud의 추가 기능



androidhuman-sticky-notes



notes



yFU5jp0kvcvz8LD75n6Q



+ 컬렉션 시작

+ 문서 추가

+ 컬렉션 시작

notes



yFU5jp0kvcvz8LD75n6Q



+ 필드 추가

body: "테스트 노트입니다."

color: 4294965700

title: "테스트 노트"

완성된 코드

https://github.com/kunny/skku-bootcamp-2023-summer/tree/main/sticky_notes/step4