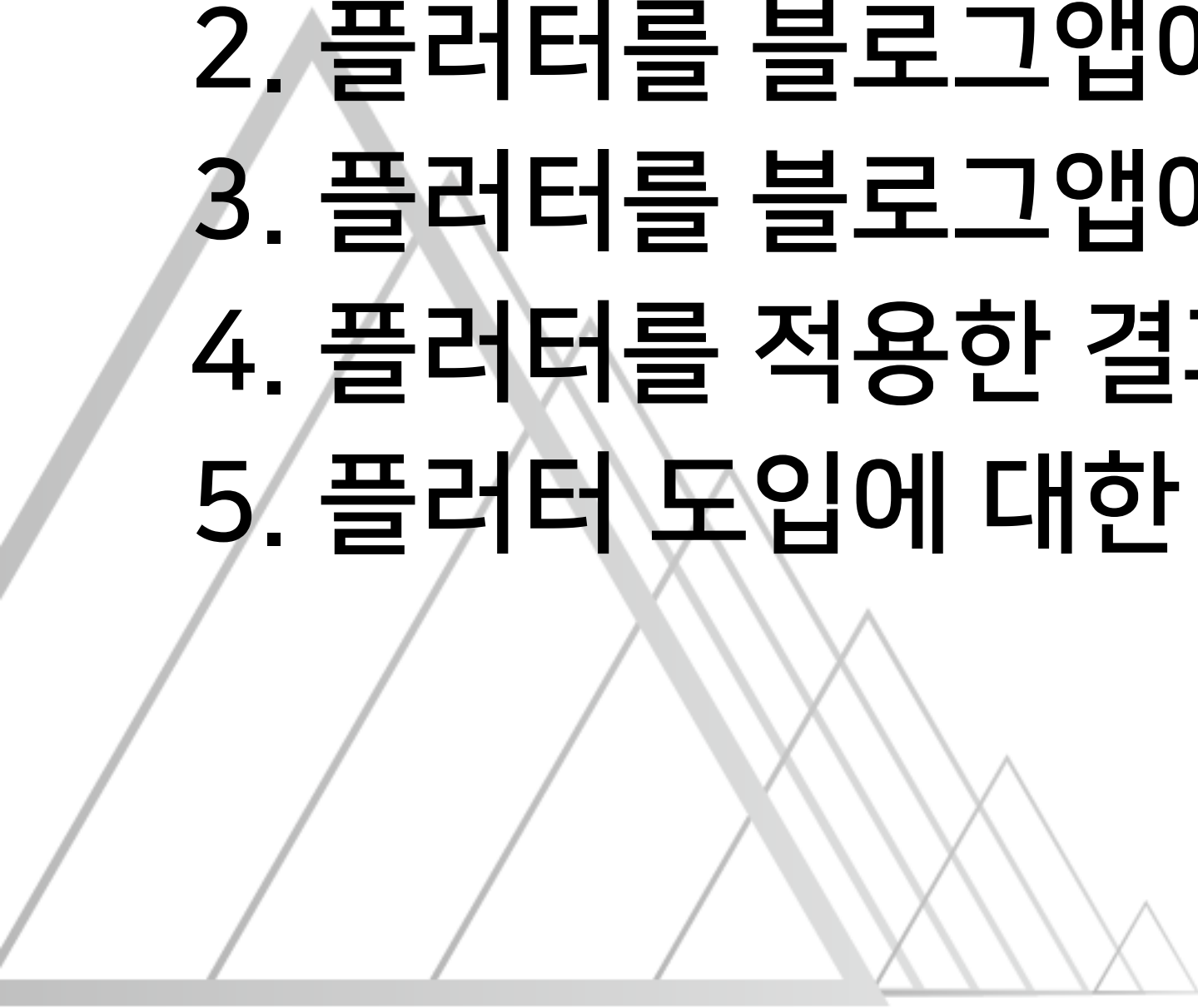


네이티브로 개발 된 앱을 플러터로 바꾸고 있습니다.

(네이버 블로그앱에 플러터 도입 1년의 과정)

CONTENTS

1. 플러터를 도입하게 된 배경
2. 플러터를 블로그앱에 적용한 방법
3. 플러터를 블로그앱에 적용한 과정
4. 플러터를 적용한 결과
5. 플러터 도입에 대한 회고



1. 블로그앱에 플러터를 도입하게 된 배경

1.1 블로그앱과 서비스 현황

블로그는 오래된 서비스, 유지보수 비용이 증가

- 네이버 블로그앱은 2009년에 출시한 오래 된 서비스.
- 오래된 화면이나 기능 등, 관련하여 유지보수를 해야하는 상황이 지속적으로 발생.



1.1 블로그앱과 서비스 현황

블로그는 오래된 서비스, 유지보수 비용이 증가

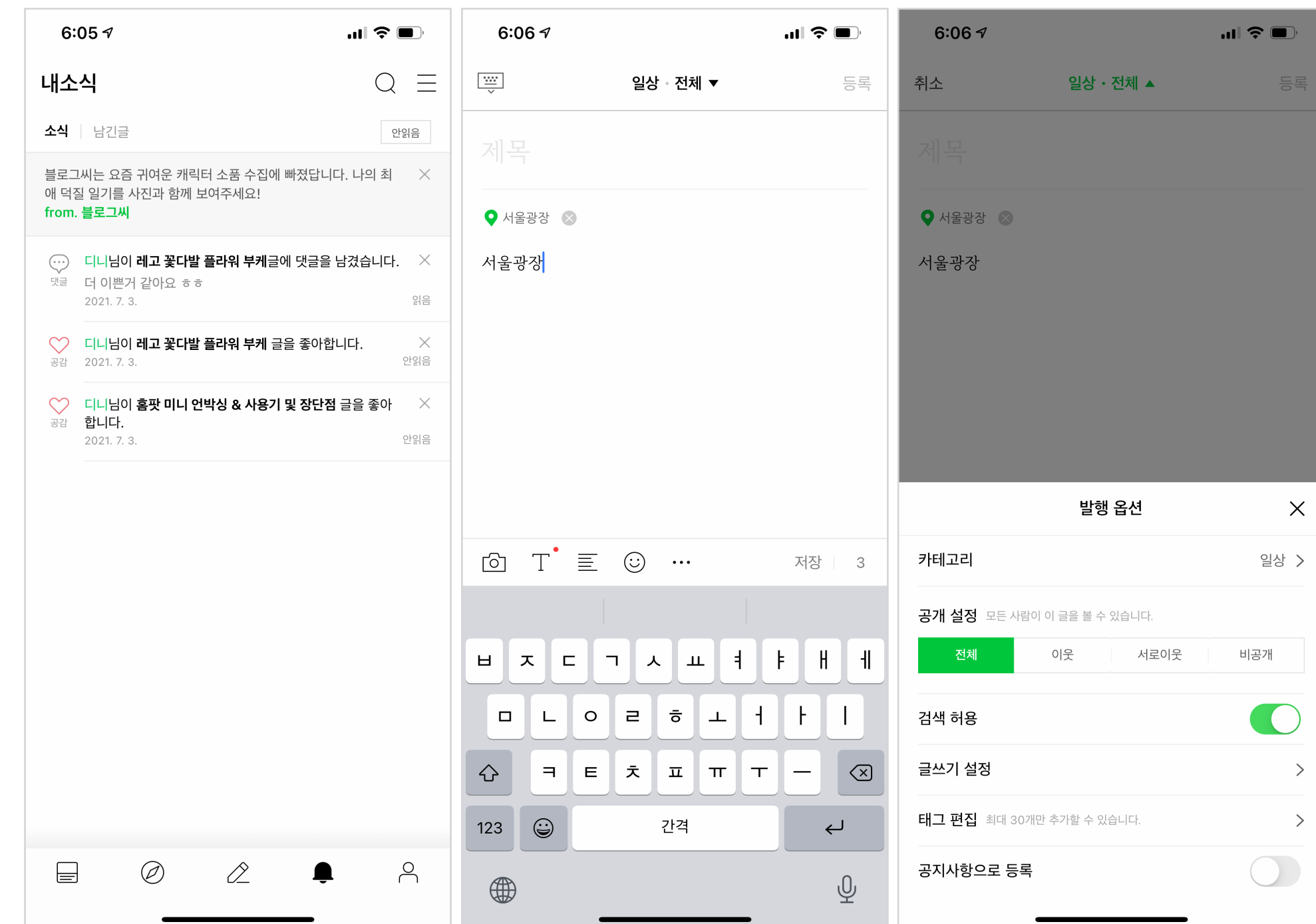
- 네이버 블로그앱은 2009년에 출시한 오래 된 서비스.
- 오래된 화면이나 기능 등, 관련하여 유지보수를 해야하는 상황이 지속적으로 발생.

```
1 //
2 // NBUIPickerView.h
3 // NaverBlog
4 //
5 // Created by      on 09. 05. 25.
6 // Copyright 2009 NHN. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10
11 // picker select
12 typedef enum{
13     PICKER_SELECT_CATEGORY = 0,
14     PICKER_SELECT_BUDDYGROUP,
15     PICKER_SELECT_IMAGE_SIZE,
16     PICKER_SELECT_EVENTCATEGORY
```

2009년 코드?..



최소 2011~2013년 사이 화면 (추정..)

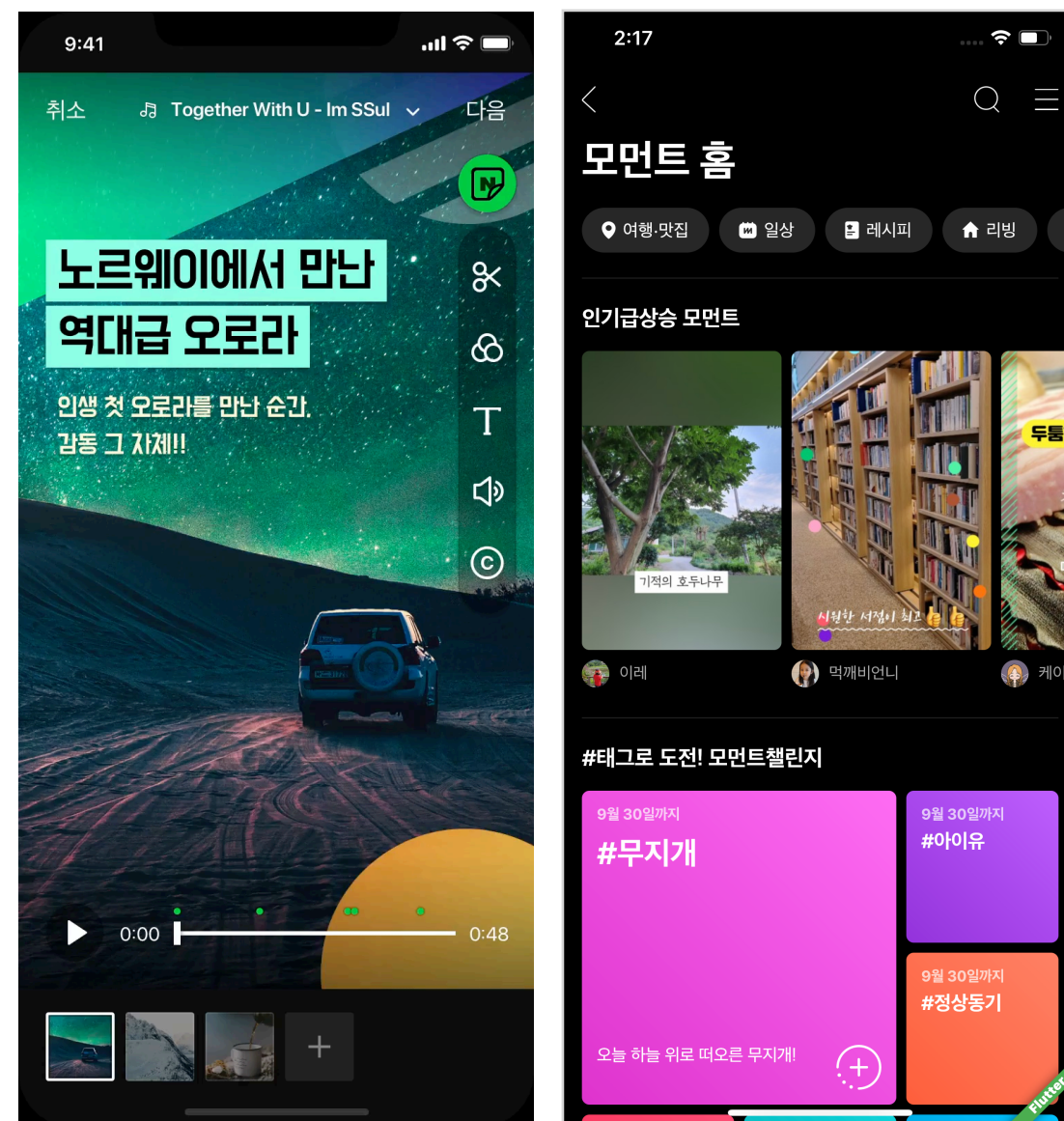


현재 블로그앱 화면

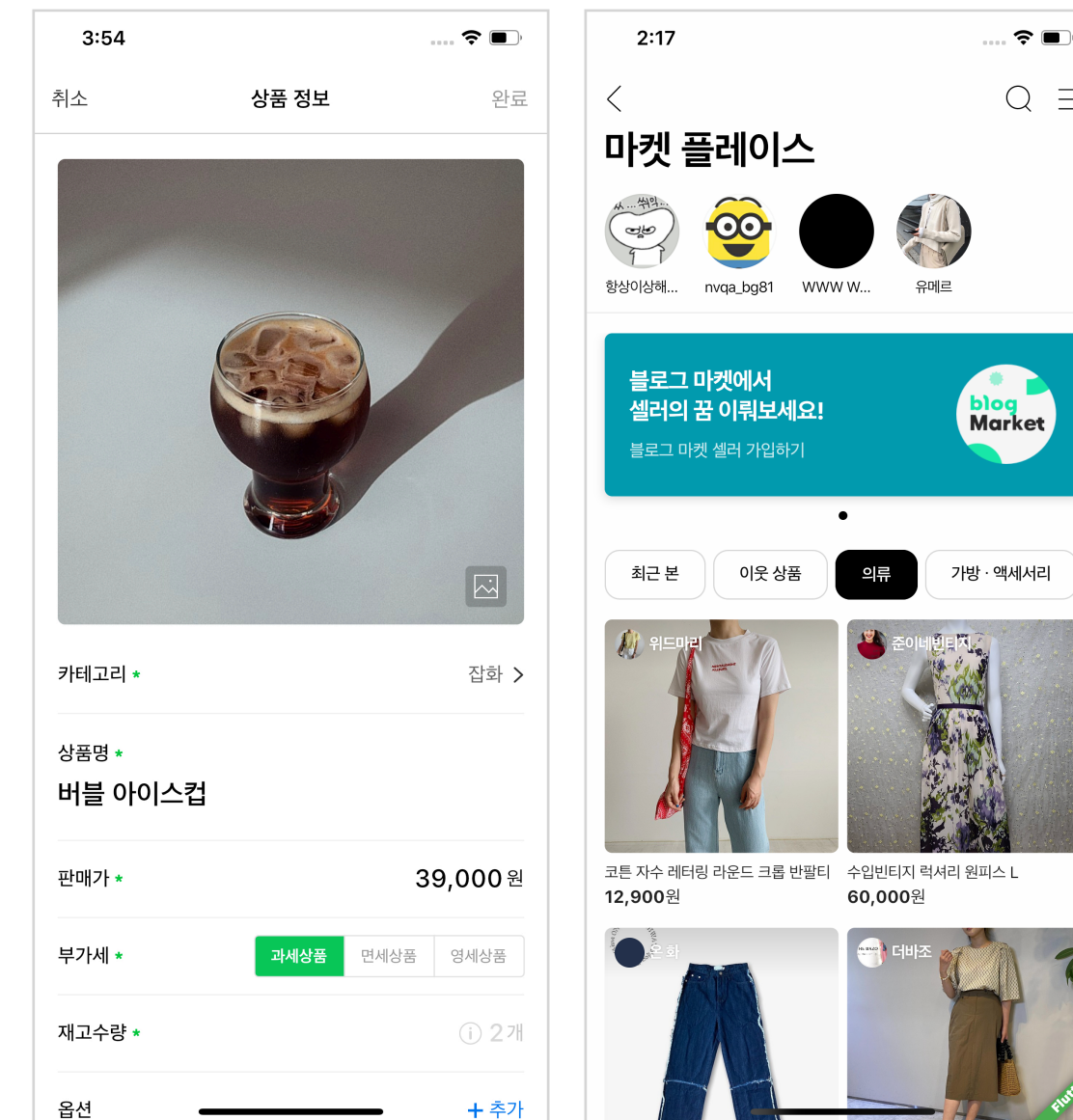
1.1 블로그앱과 서비스 현황

지금도 꾸준히 새로운 시도를 하는 중

- PC, 모바일웹의 비중도 높긴하지만 모바일 양 플랫폼 합산 DAU 100만 이상인 서비스.



모먼트



블로그마켓

모먼트 에디터와 같이 앱을 통해서만 이용 가능한 기능도 존재.

1.1 블로그앱과 서비스 현황

블로그앱에 플러터를 도입한 배경: 서비스 측면

- 오래된 서비스의 유지보수가 필요한 상황이 증가.
- 새로운 기능이나 시도, 사용성 개선에 대한 빠른 서포트가 요구 됨.

유지보수 상황
증가

빠른 서포트의
필요성

1.2 조직과 업무 현황

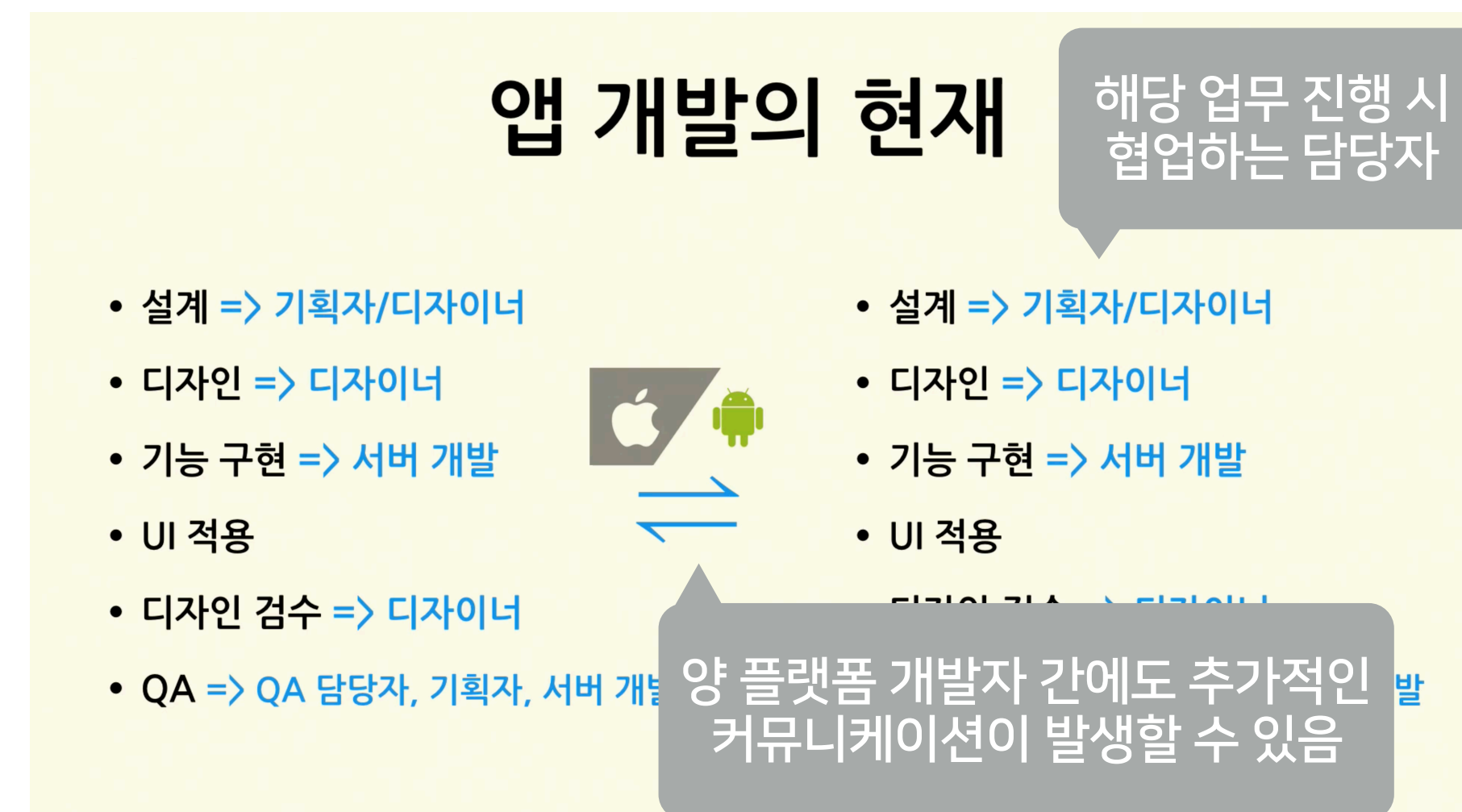
제한 된 개발 인원, 효율적인 작업 필요.

- 편중 된 플랫폼 개발 인력.
- 플랫폼 간 중복된 업무 프로세스.



부족하고 균형이 맞지 않는 개발 인력 현황

NAVER Engineering Day 발표 내용 중 일부 (2019.4 by 추홍엽님)



서비스 개발 업무 시에 많은 유관부서와 함께 일하며
단지 기능 구현 외에도 해야할 업무가 많다.

1.2 블로그앱에 플러터를 도입한 배경

조직과 업무 측면

- 한정된 자원 속 효율적인 업무의 필요성.

한정된
개발 자원

업무 효율성
증대 필요

즉시 적용이
가능한 플러터

1.3 지식인 플러터 앱 런칭

지식인 플러터 앱 런칭

- 그 동안 안드로이드 버전으로 단순 기능만 제공 중이던 지식인 앱을 플러터로 새로 개발하여 양 플랫폼에 출시.
- 런칭 경험을 토대로 블로그앱 플러터 적용 때도 개발 환경 등 서포트.



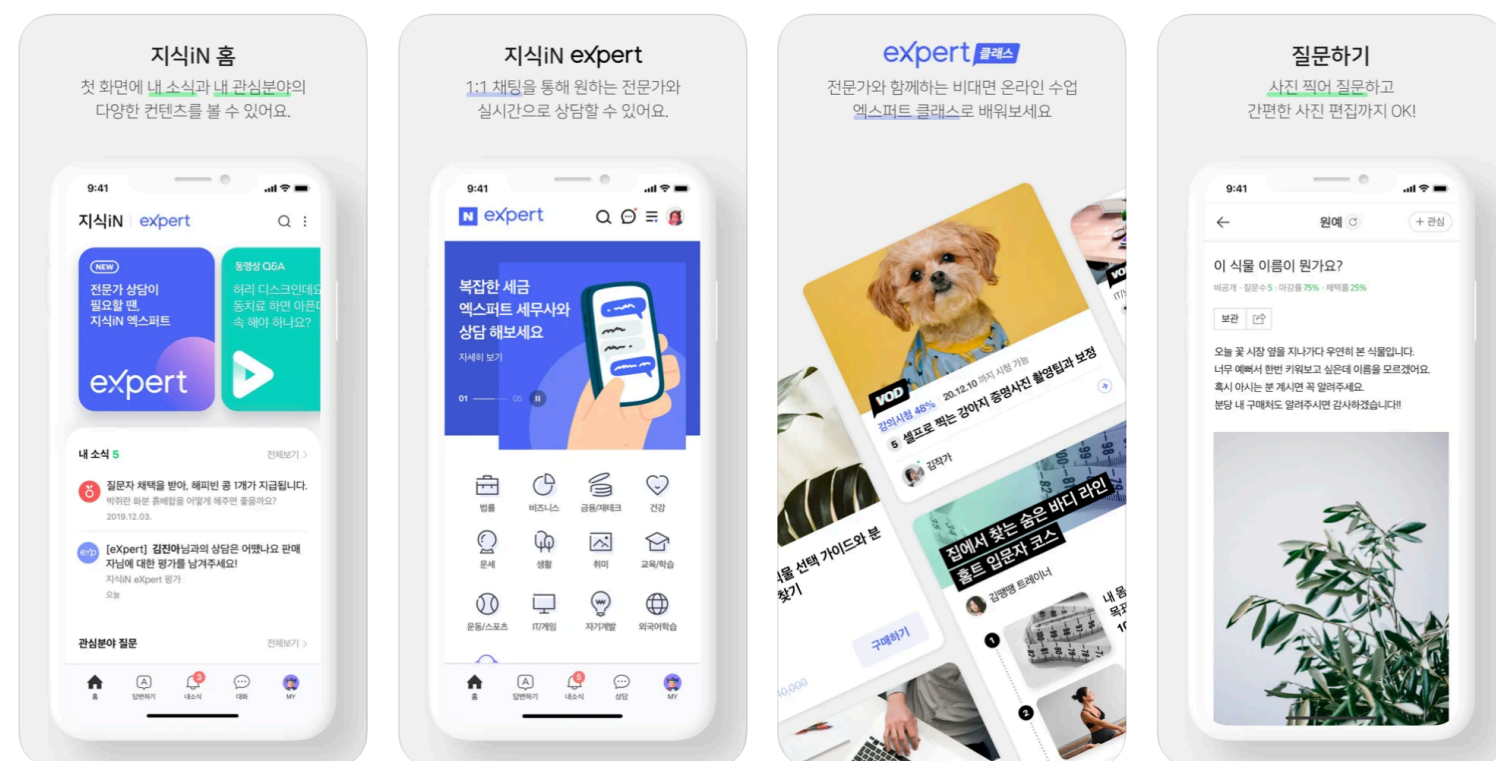
네이버 지식iN, 엑스퍼트 12+

NAVER Corp.

소셜 네트워킹 앱 94위
★★★★★ 4.7 • 1.7천개의 평가

무료

스크린샷 iPhone iPad



네이버 지식iN, 엑스퍼트

NAVER Corp. 소셜

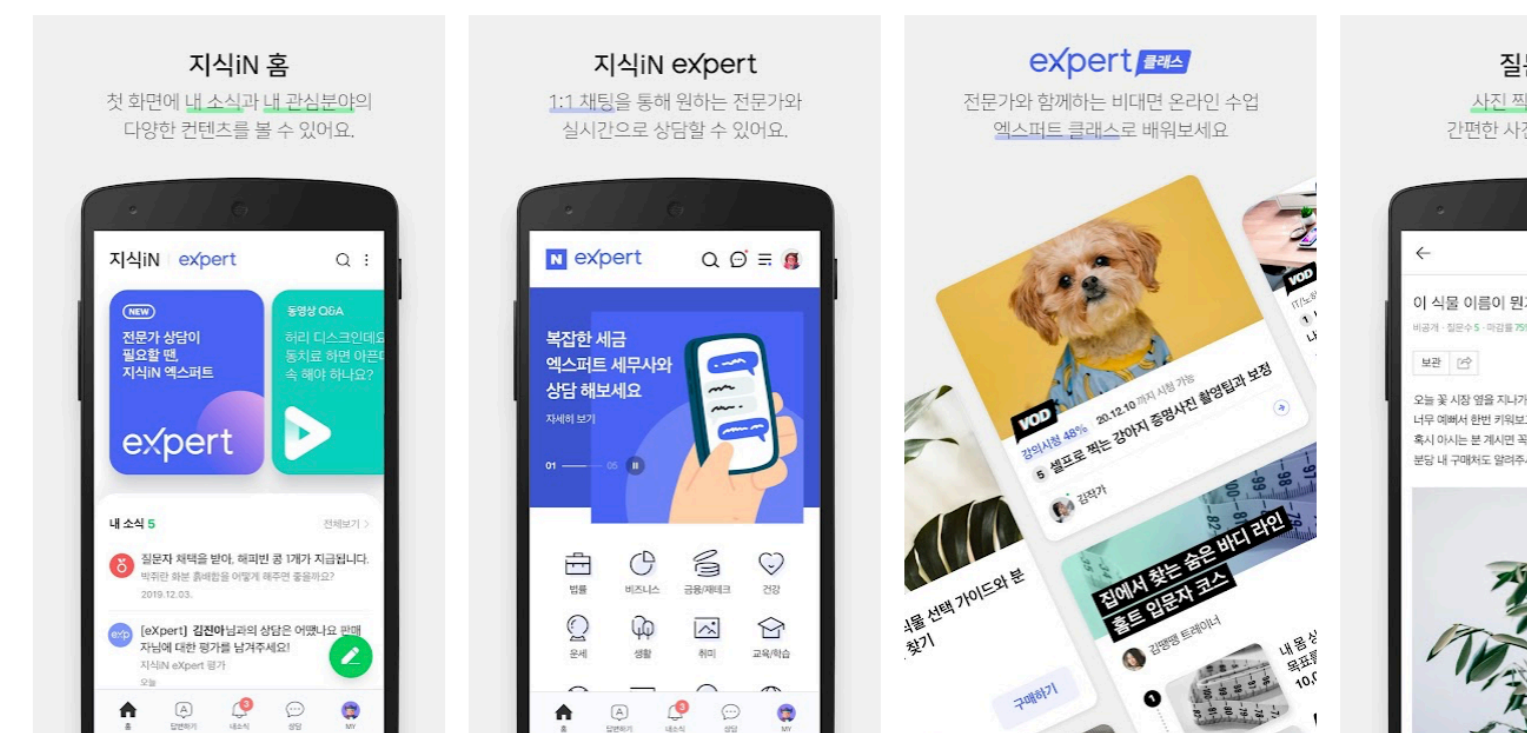
★★★★★ 15,621

전체이용가

내 모든 기기에서 앱을 사용할 수 있습니다

위시리스트에 추가

설치

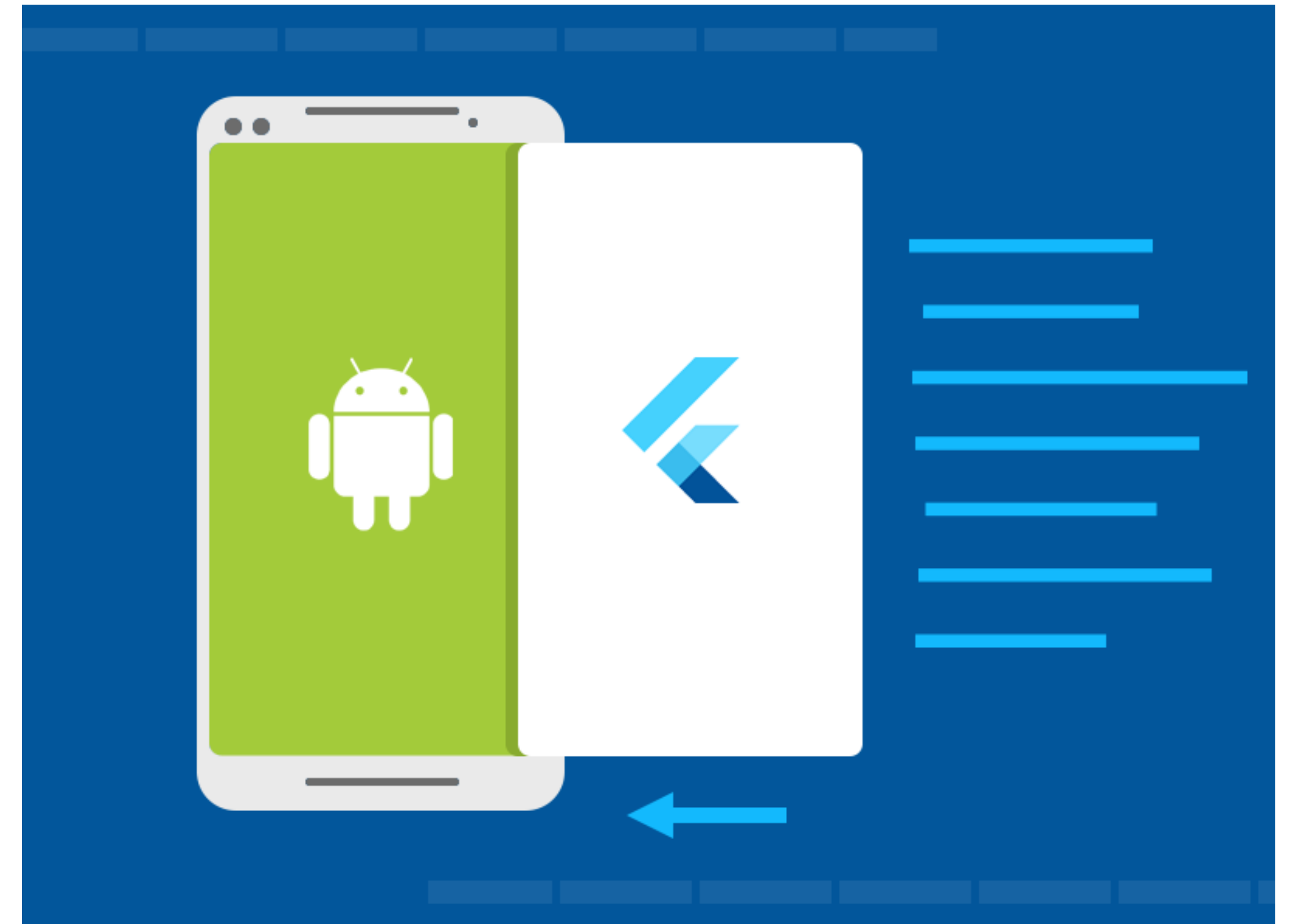


2. 블로그앱에 플러터를 적용한 방법

2.1 즉시 적용 가능한 플러터 Add-to-app

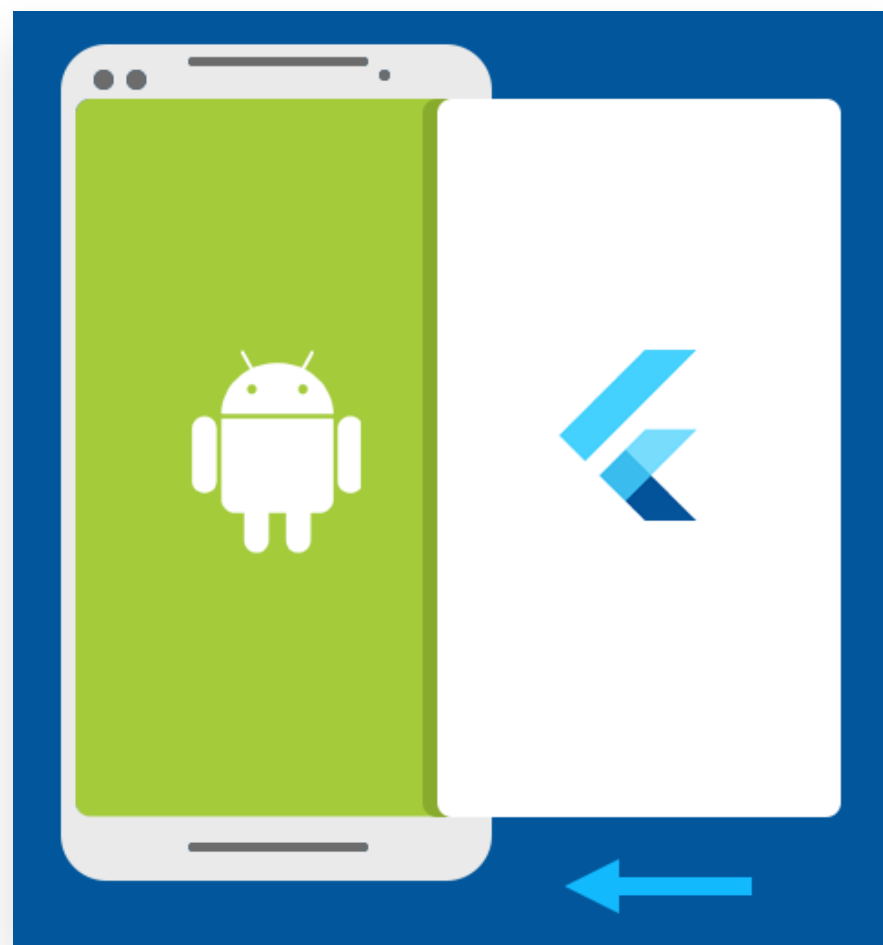
플러터로 앱 개발 시 적용 가능한 방법

- 네이티브 앱 프로젝트에 플러터가 모듈이나 라이브러리 형태로 포함 되는 방식.
- 앱 UI의 일부나 공유 Dart 로직을 사용할 수 있다.
- 네이버 블로그 모바일앱은 Add-to-app 방식으로 점진적으로 플러터로 전환하는 중.

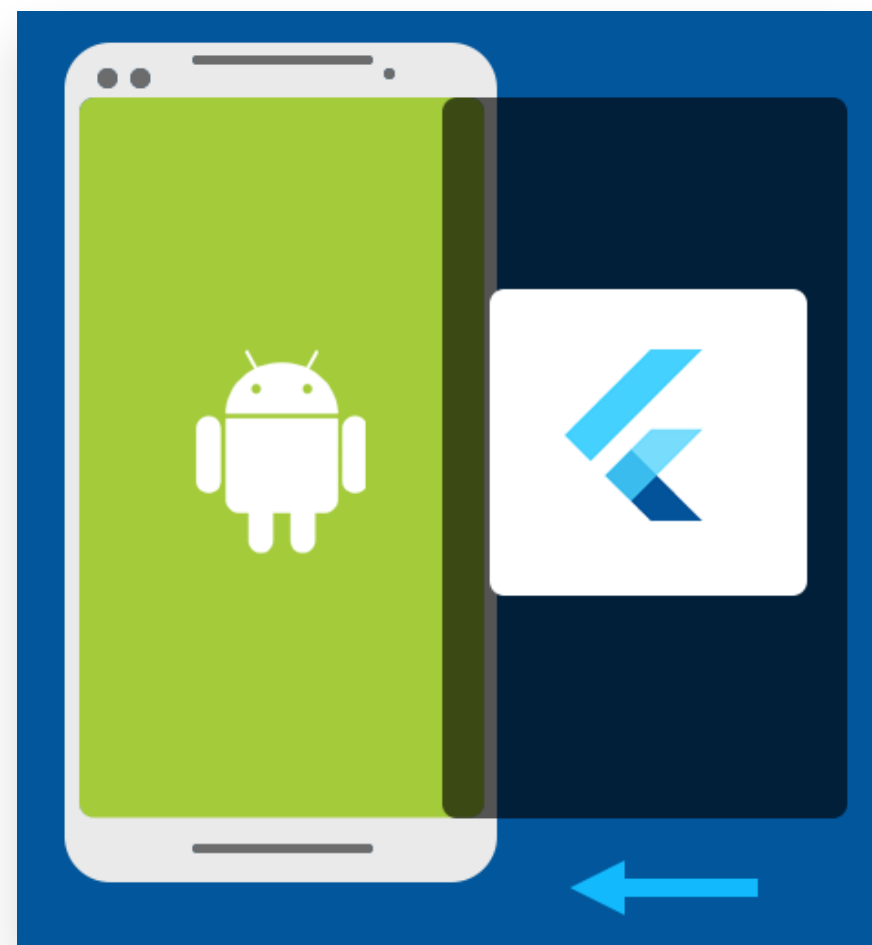


2.2 Add-to-app에서 지원하는 기능

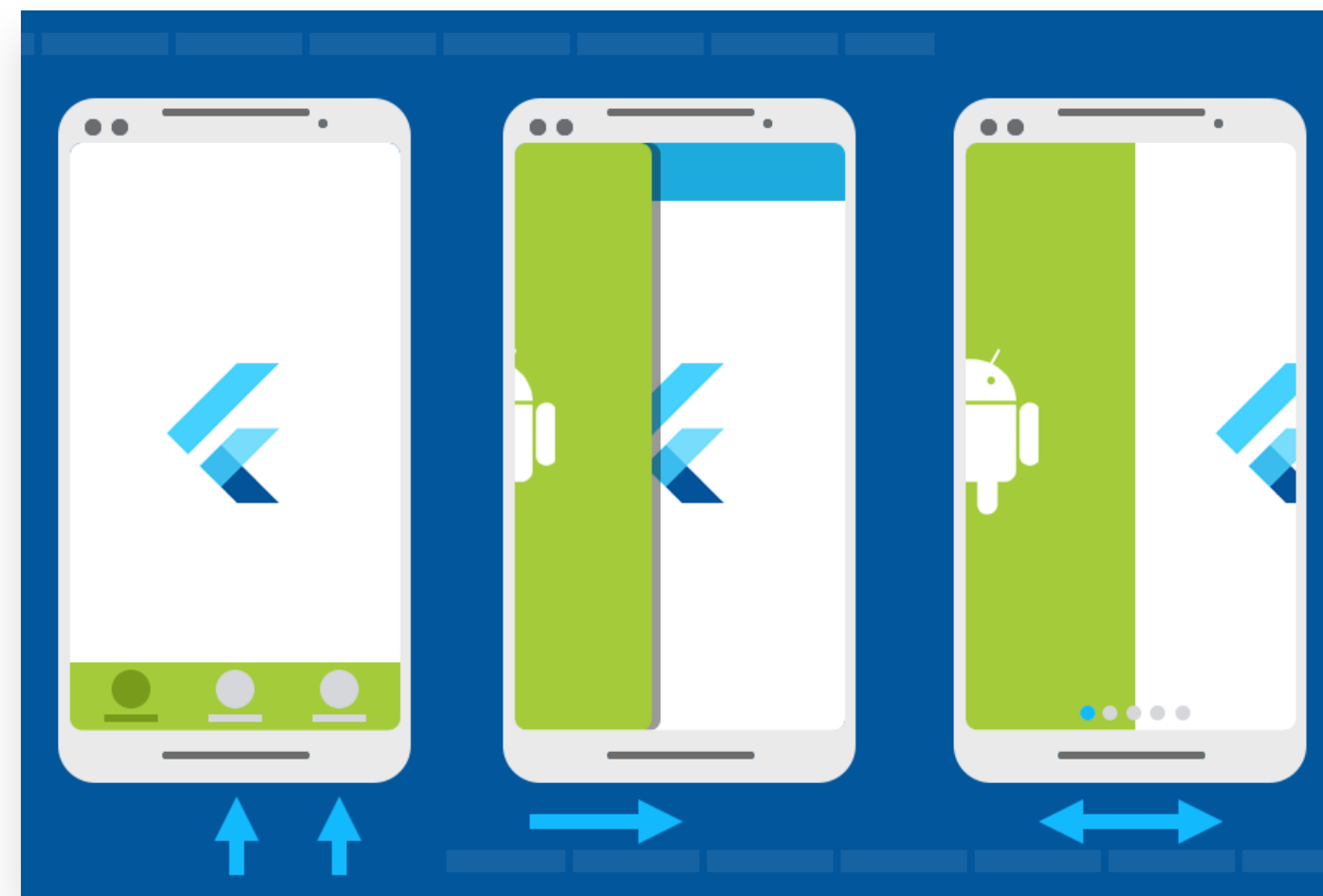
네이티브 앱과의 연동 방식



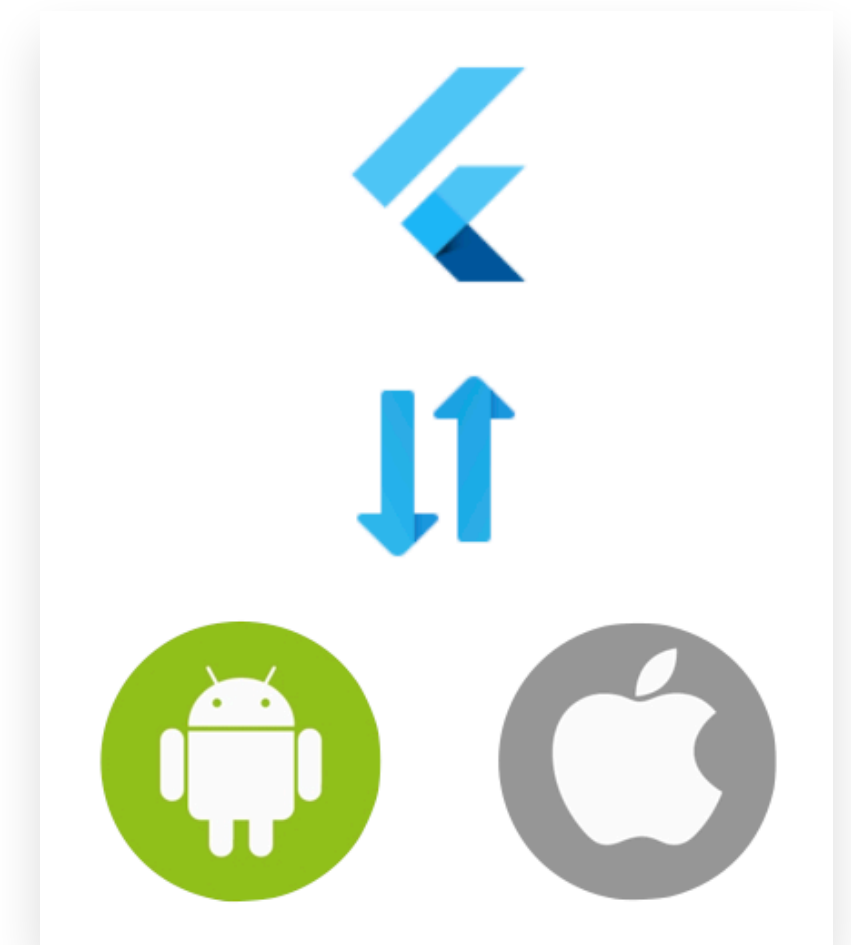
일부 화면의 전체를 플러터로 구현



투명 배경으로 표시 가능
(다이얼로그나 바텀시트 용도)



안드로이드 일부 Fragment를 플러터로 구현



플러터와 네이티브 간 데이터 전달
(공통의 비즈니스 로직 수행)

3. 블로그앱에 플러터를 적용한 진행 과정

3.0 진행 과정

프로젝트 세팅
& 학습

작은 UI 위젯 구현

화면 단위 개발

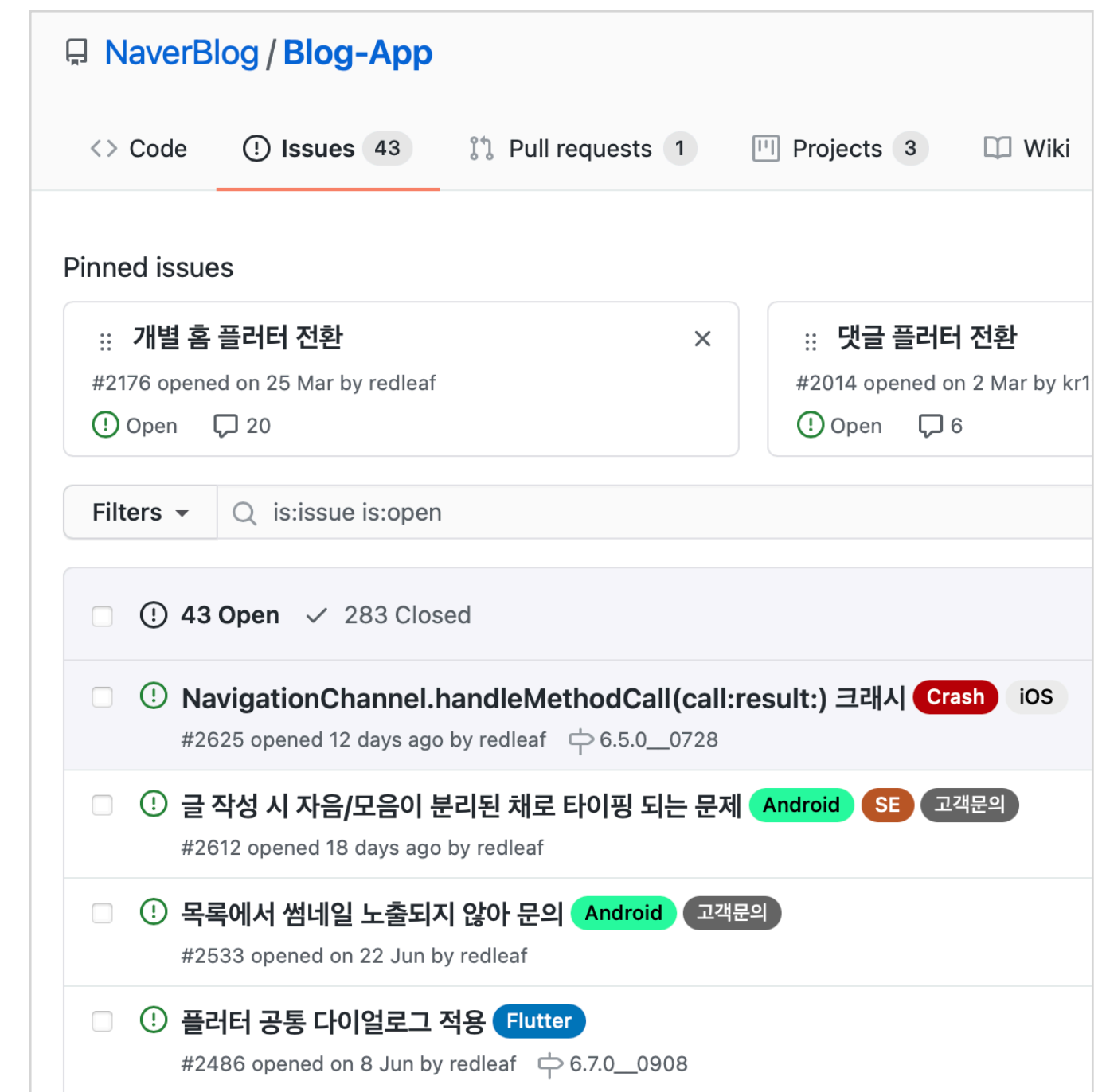
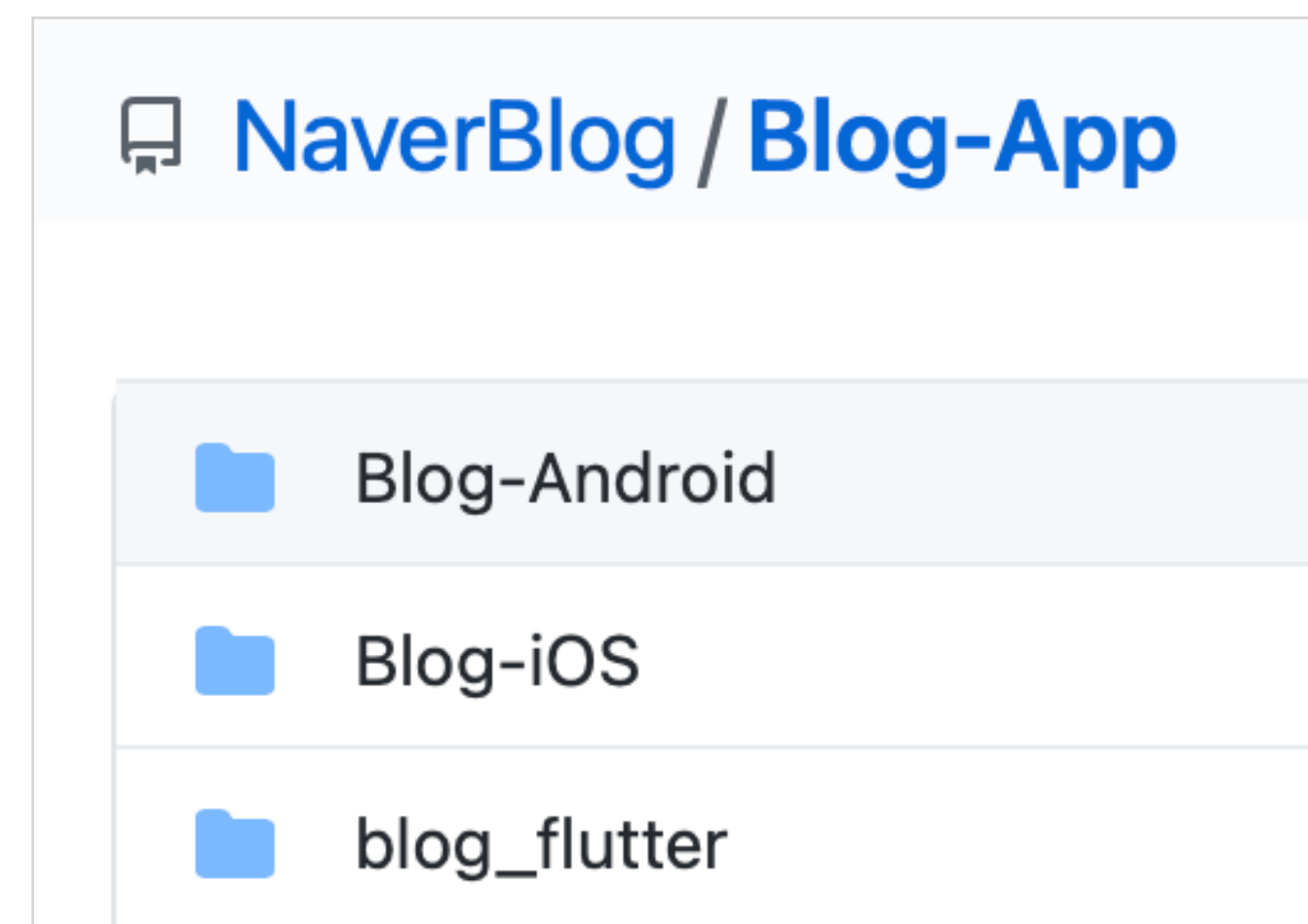
네이티브앱 연동

QA & 배포

3.1 프로젝트 세팅 & 학습

프로젝트 세팅 & 저장소 단일화

- 지식인 플러터 개발 인력들이 초기 세팅에 도움을 줌.
- 저장소 통합, 버저닝 단일화도 함께 진행.
 - 이슈 관리, PR 리뷰도 한 곳에서 모든 팀원들이 함께 참여가 가능하니 이슈 파악 등 커뮤니케이션 관점에서도 이점.



저장소 통합으로 커뮤니케이션 비용 감소 효과도 획득

3.1 프로젝트 세팅 & 학습

Lint 적용

- 똑같은 UI를 구현하더라도 여러 방식이 존재하기에 린트를 통해 약속된 코드를 작성하도록 유도.
- lints, flutter_lints 라이브러리를 통해 약속된 부분만 지정 가능.
- 초반의 낮은 학습도 상태에서의 작업에 대해서도 어느정도 커버 가능.

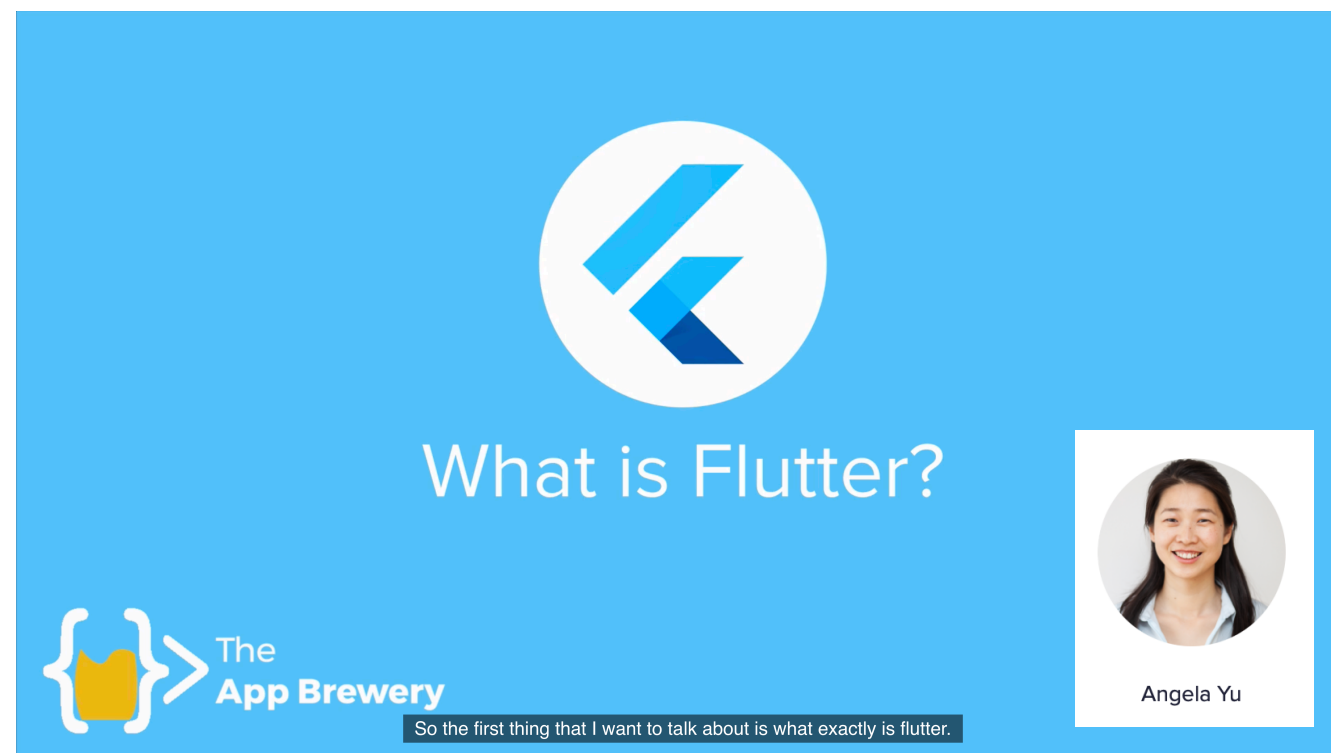
avoid_unnecessary_containers Avoid unnecessary containers. Details Avoid wrapping widgets in unnecessary containers. Wrapping a widget in <code>Container</code> with no other parameters set has no effect and makes code needlessly more complex.	BAD: <pre>Widget buildRow() { return Container(child: Row(children: <Widget>[const MyLogo(), const Expanded(child: Text('...'),),],),); }</pre>	GOOD: <pre>Widget buildRow() { return Row(children: <Widget>[const MyLogo(), const Expanded(child: Text('...'),),],); }</pre>
---	---	---

Ex) 불필요한 Container 사용을 금지하는 린트

3.1 프로젝트 세팅 & 학습

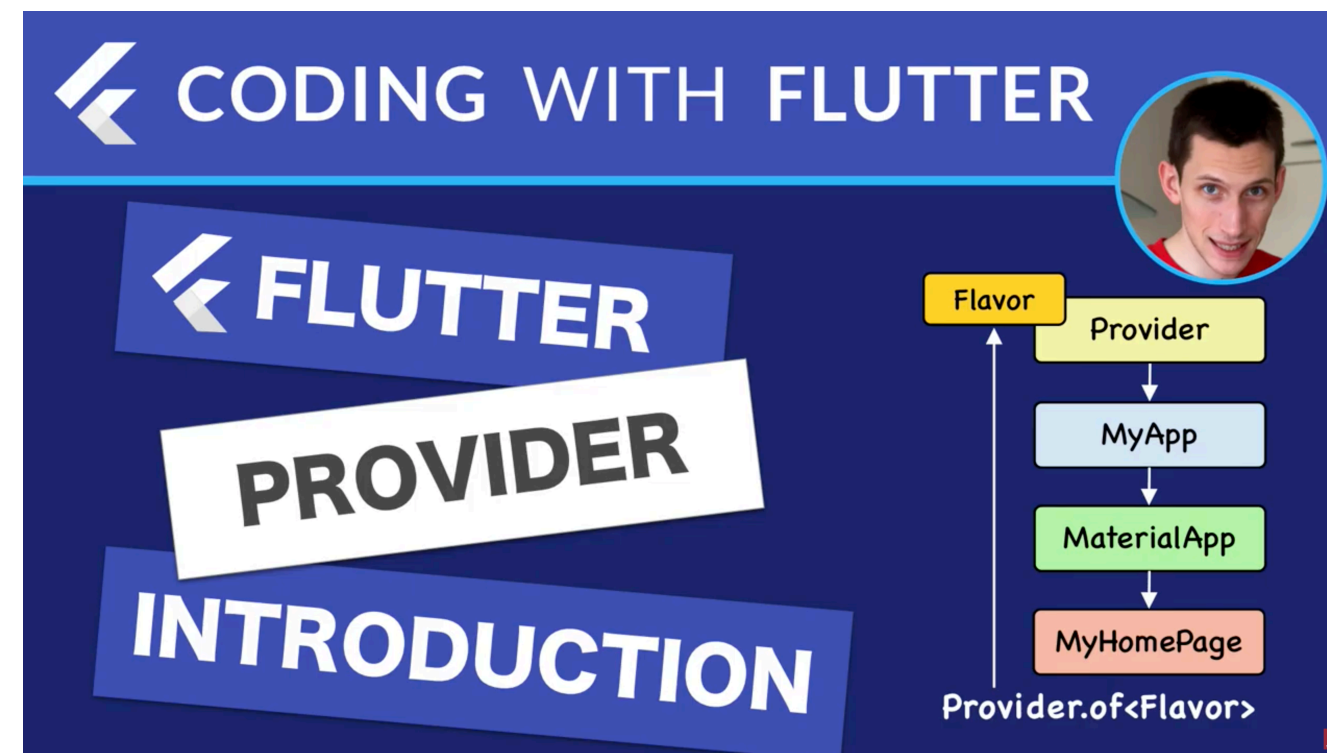
Dart & Flutter 학습

- 공식 문서, 유튜브 영상 등을 통해 학습.



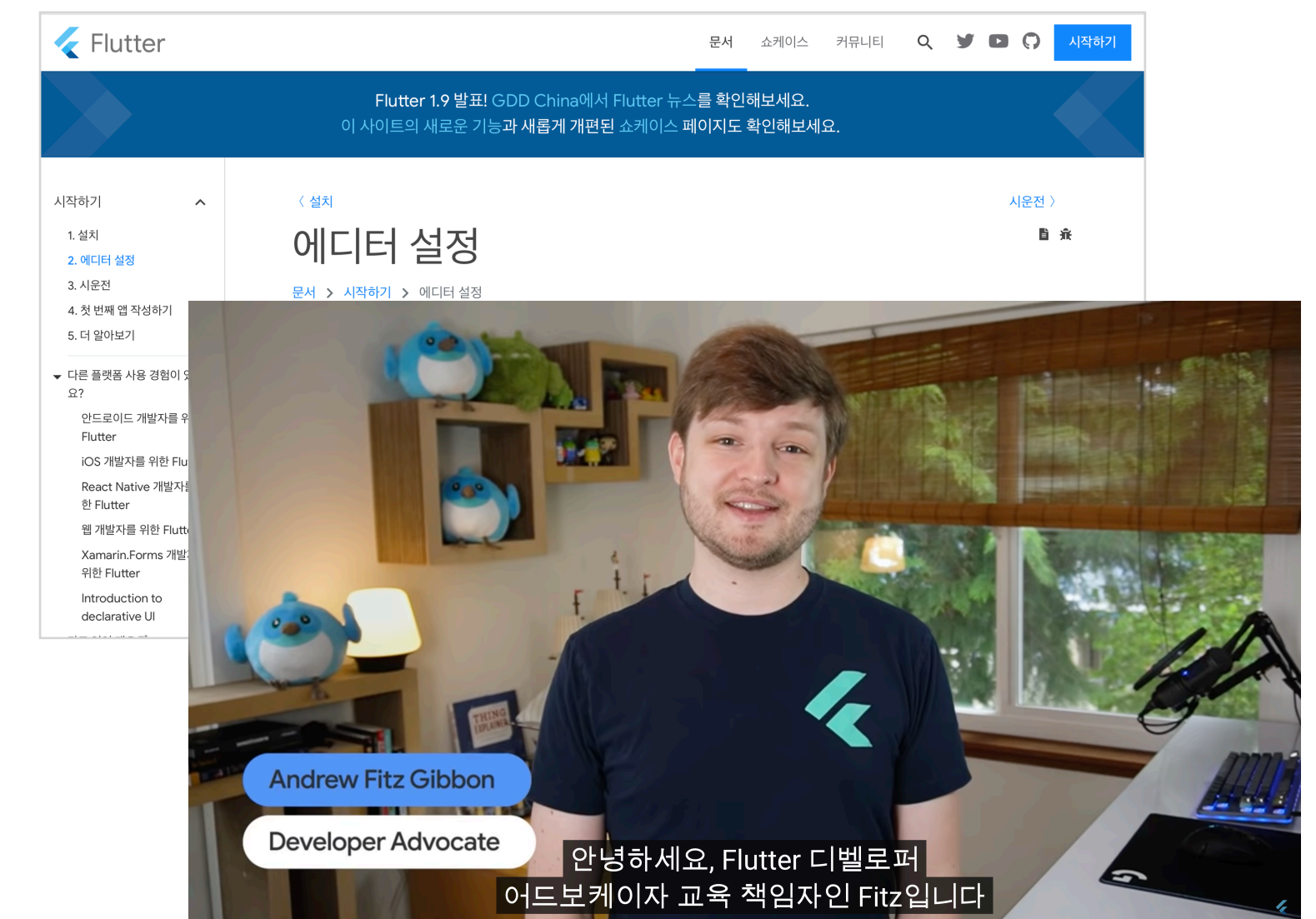
AppBrewery - Introduction to Flutter Development Using Dart

안젤라 선생님의 발음은 어찌나 정확한지
내가 영어를 잘하는 것으로 착각하게 만들 정도.



Youtube - Flutter Provider: Introduction

Provider 개념에 관해 도움을 주시는 또 다른 안 선생님, 안드레아



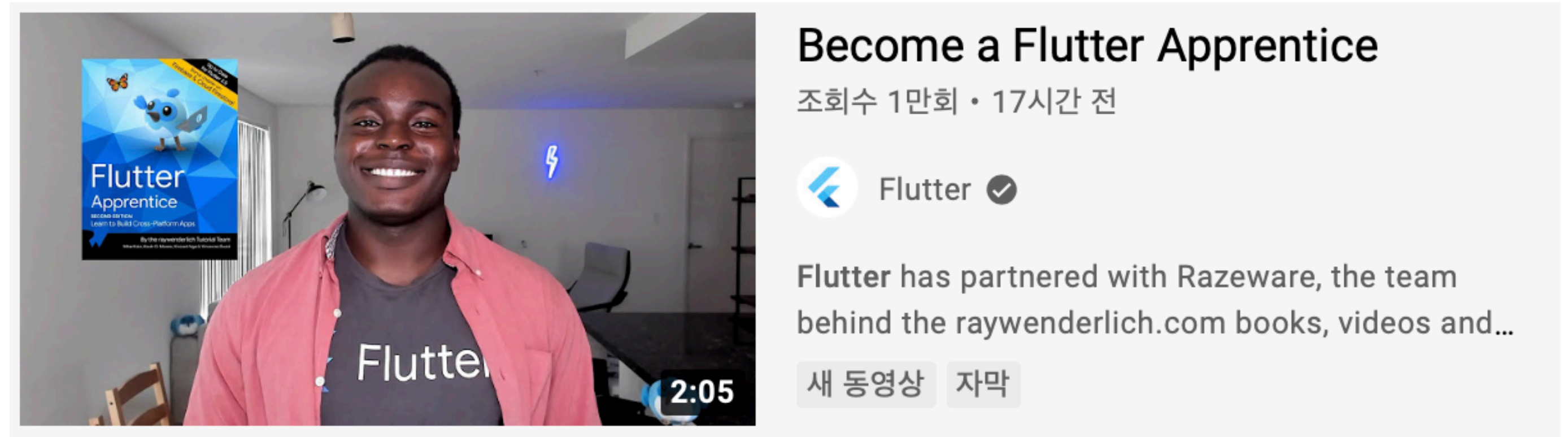
Flutter.dev / Flutter Official YouTube

공식 사이트, 공식 유튜브에서도
한글 문서를 어렵지 않게 접할 수 있다.

3.1 프로젝트 세팅 & 학습

Dart & Flutter 학습

- 2022년 1월 6일까지 Ray Wenderlich 책 무료 열람 가능.
- <https://flutter.dev/apprentice-giveaway/>



Youtube Flutter 채널을 구독하면 이런 유익한 꿀소식도 접할 수 있다.

3.2 작은 위젯부터 구현

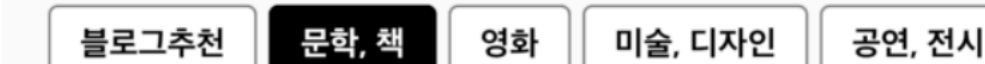
블록 쌓듯 차근차근 물밑 작업과 함께

공통 위젯 카탈로그 #65



redleaf commented on 18 May 2020 • edited ▾

BGKeywordList

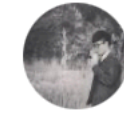


- 추천 탭 관심주제 목록, 태그 상세 등 사용 예정
- wrap 이 false (default)면 횡스크롤 생기고, true 로 지정하면 줄바꿈
- selectedIndex 를 지정하면(null 이 아니면) 해당 아이템 하이라이팅

```
BGKeywordList(
  keywords: ['블로그추천', '문학, 책', '영화', '미술, 디자인', '공연, 전
  wrap: true,
  selectedIndex: 3, // nullable (Default)
  onTap: (index) { ... },
  ....
)
```



2



gasigogi commented on 26 May 2020 • edited ▾

BGTextField

주소를 입력하세요

- 플레이스 홀더나 글자 수 제한 같은 옵션들을 좀 더 사용하기 쉽게 포팅해놓은 위?

```
BGTextField(
  hintText: 'hintText',
  hintColor: Colors.black,
  maxLines: 2,
  maxLength: 20,
  autoFocus: false,
  readOnly: false,
  prefix: Container(),
  textAlign: TextAlign.right,
  controller: TextEditingController(),
  focusNode: FocusNode(),
  keyboardType: TextInputType.text,
  onTap: () {},
  onChanged: (text) {},
  onEditingComplete: () {},
  onSubmitted: (text) {},
)
```



3

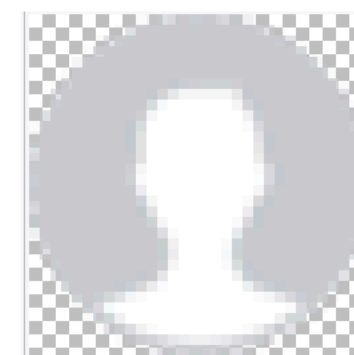


emit-steam commented on 26 May 2020 • edited ▾

ProfileImageView



- 공통 프로필 뷰
- size와 profileUrl을 넣어주면, Circle 모양(회색 테두리를 가진) 으로 로딩하여 보여줍니다.
- placeholder, errorView 경우 default profile이미지 나오게 처리되어있습니다.



```
ProfileImageView(
  size: 32,
  profileUrl: profileUrl,
),
```



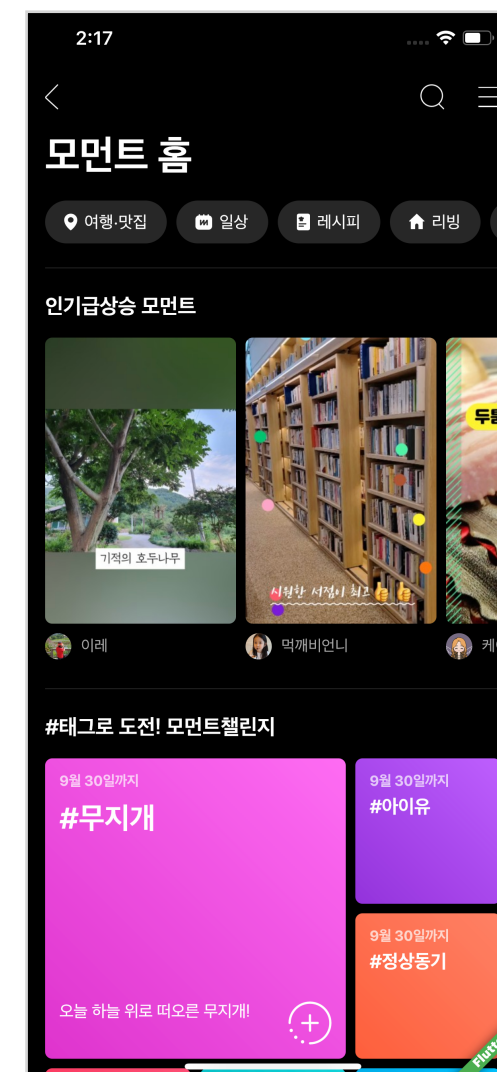
3

3.3 화면 단위 개발 돌입

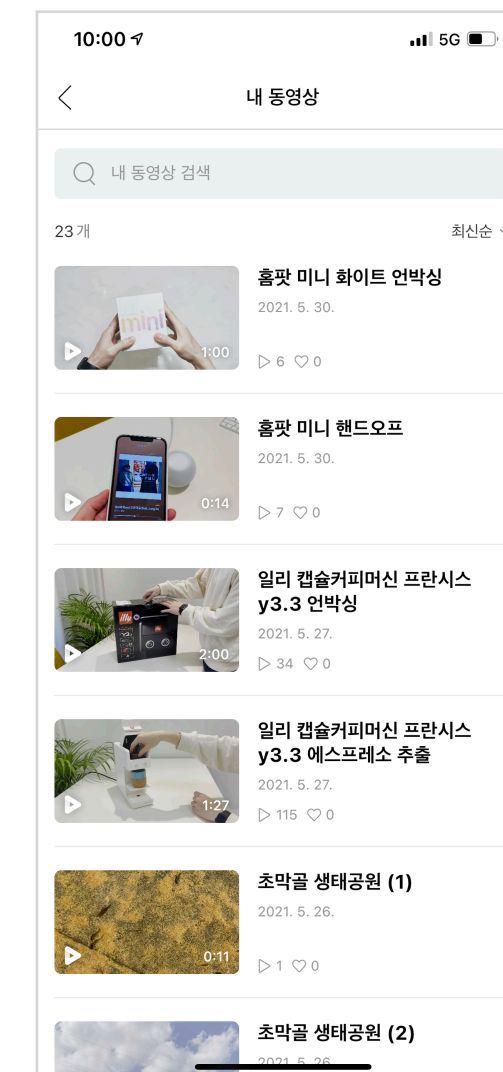
플러터 개발 우선순위와 기준을 정하기

1. 새로 개발하는 화면은 플러터로 개발하는 것을 기본 전제.
 - 부득이하게 네이티브로 개발해야 한다면 비즈니스 로직과 같이 공통화 가능한 부분은 다트로 작업하여 연동 -> 중복 작업 최소화.
2. 유지 보수가 필요한 화면을 우선으로 전환.
3. 중요도나 비중이 낮은 화면을 차선으로 전환.
4. 기존의 주요 화면을 플러터로 전환.

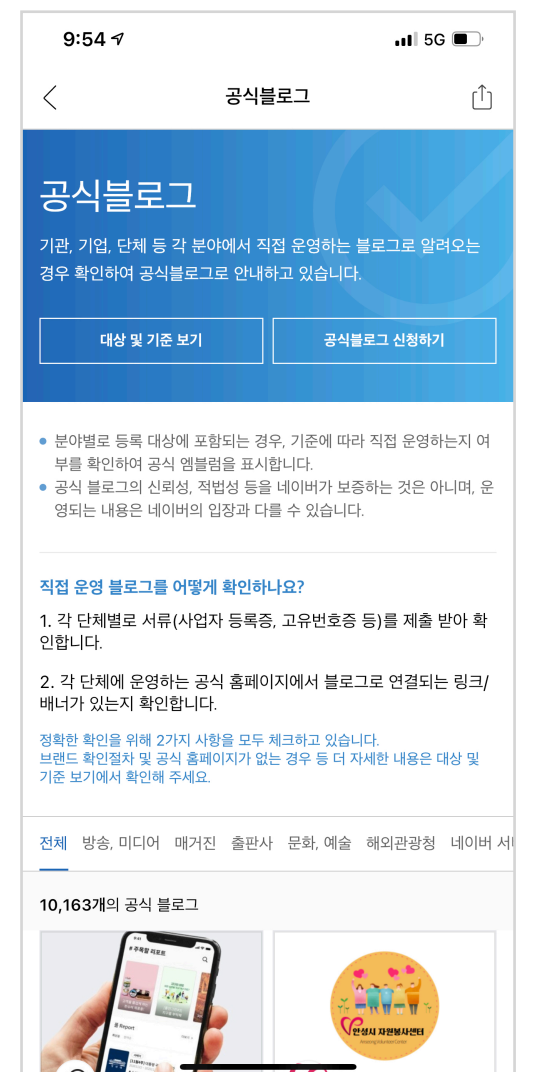
중복 업무 감소, 리스크 최소화



신규 화면이었던 '모먼트홈'



네이티브였지만 스펙 변경으로 플러터로 우선전환 된 '내 동영상'



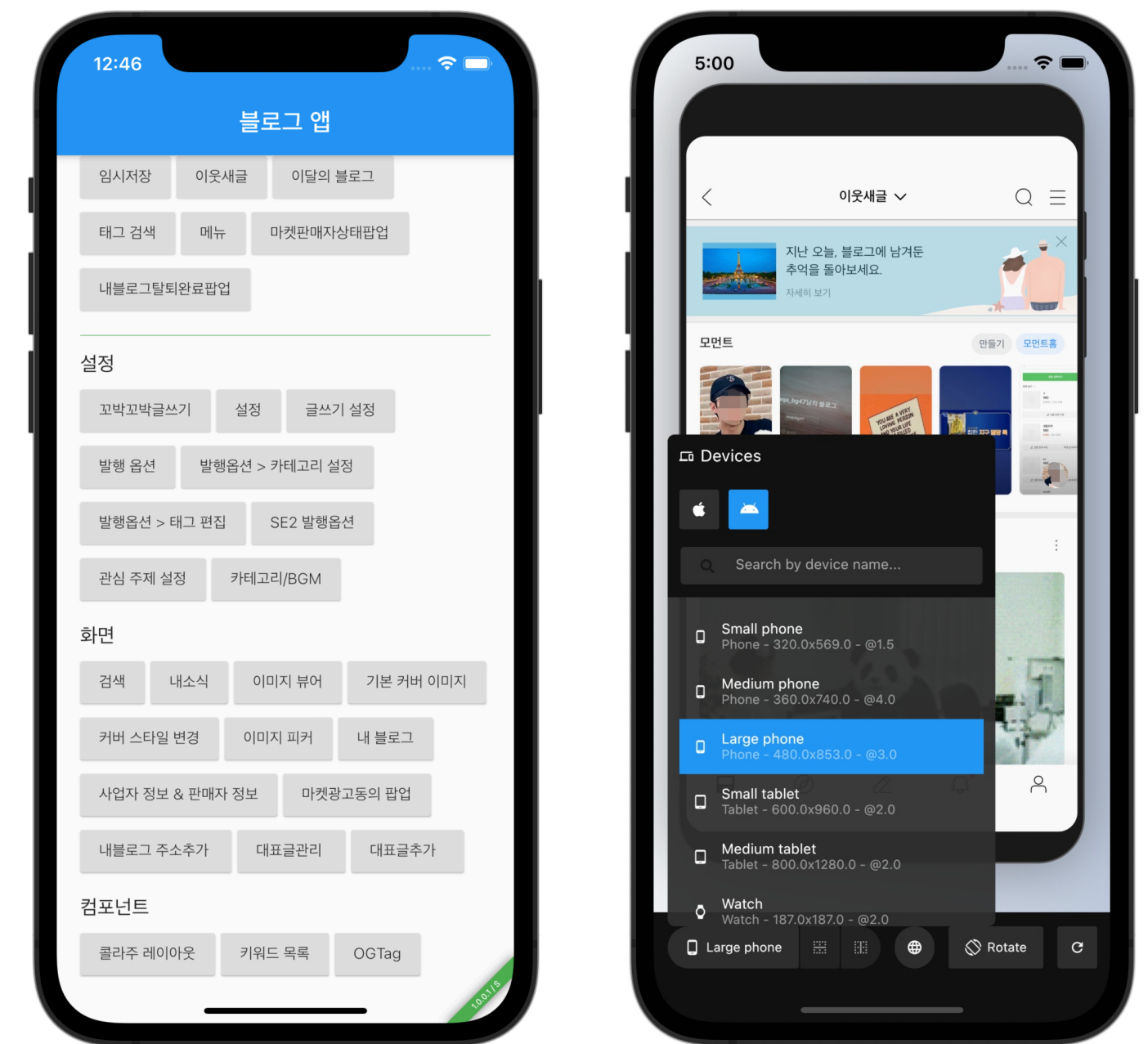
중요도가 낮은 '공식블로그' 메뉴

3.3 화면 개발 진행

UI 개발은 플러터 프로젝트 내에서 빠르게

- 더 빠른 개발 속도를 위해 UI 작업 과정에는 플러터 단독 프로젝트로 빌드: 빠른 빌드의 장점
- device_preview 라이브러리를 통해 어떤 디바이스에서 도 양 플랫폼, 타블렛 UI 실시간 확인 가능.
- 이미지는 최대한 svg를 통합하여 적용. 그 외에는 devicePixelRatio 값을 이용하여 1x, 2x, 3x로 구분.

플러터 단독앱으로
가볍게 UI 작업



device_preview를 통해 하나의 환경에서
다른 플랫폼, 다른 디바이스 확인 가능

3.3 화면 개발 진행

상태 관리

- Provider 라이브러리를 통한 상태 관리. (사실상 국룰)
 - Widget 클래스에서는 UI와 사용자 인터랙션만 구현.
 - Provider를 통해 데이터로직과 UI로직을 나누고 위젯이 직접 상태관리를 하지 않도록 개발.



플러터 공식 인증 맛집(?)

Simple app state management



[Docs](#) > [Development](#) > [Data & backend](#) > [State management](#) > Simple app state management

Now that you know about [declarative UI programming](#) and the difference between [ephemeral](#) and [app state](#), you are ready to learn about simple app state management.

On this page, we are going to be using the [provider](#) package. If you are new to Flutter and you don't have a strong reason to choose another approach (Redux, Rx, hooks, etc.), this is probably the approach you should start with. The [provider](#) package is easy to understand and it doesn't use much code. It also uses concepts that are applicable in every other approach.

Flutter 공식 사이트에서도 Provider를 이용하여 손쉽게 상태관리 방법을 가이드 중

3.4 네이티브앱에 연동

플러터 빌드 및 네이티브앱에 모듈 연동

- 플러터 변경 내역을 빌드하고 네이티브앱 프로젝트에 적용.
 - 안드로이드: 원스텝 빌드를 통해 안드로이드 프로젝트 빌드 시, 플러터 프로젝트도 함께 빌드.
 - iOS: 플러터 프로젝트를 프레임워크로 미리 빌드하고 네이티브 앱에 링크.

※ 원스텝 빌드는 로컬에 플러터 개발 환경이 필요하기 때문에 타 개발 조직과 함께 작업을 하는 경우 적합하지 않을 수 있음.

3.4 네이티브앱에 연동

네이티브와 플러터와의 데이터 연동

- 플러터 모듈로부터 MethodChannel을 통해서 양쪽 간 통신 가능.
- : 고유한 채널을 통해 비동기로 값을 주고 받음.
- 데이터는 Map과 같은 타입으로부터 직렬화 처리 후 전달, 각 플랫폼에서 HashMap이나 Dictionary로 재처리
- : 미리 양쪽 플랫폼에 정해진 채널명, 메서드명이 필요하고 그에 맞는 처리 로직을 정의해야 함.
- : 블로그앱에서는 목적에 따라 여러 MethodChannel을 구현하여 사용 중.

```
.whenComplete(
    () {
        // 모먼트 삭제 이후, 모먼트 서버에 삭제 요청을 위해 각 네이티브 삭제로직
        호출 필요.
        FunctionChannel.instance.call(
            'didDeleteMoment',
            model: FunctionChannelModel(
                parameters: {
                    'momentNo': momentNo.toString(),
                },
            ),
        );
    }
);
```

플러터에서 메서드채널을 통해 네이티브로 요청하면

```
private func handleMethodCall(call: FlutterMethodCall, result: @escaping
FlutterResult) {
    switch FunctionChannelMethod(rawValue: call.method) {
    case .didDeleteMoment:
        guard let parameters = (call.arguments as? String)?.parseJSON(),
        let momentNoString = parameters["momentNo"] as? String,
        let momentNo = Int64(momentNoString) else { return }
        MomentViewerViewController.deleteMoment(with: momentNo)
    }
}
```

네이티브에선 채널과 메서드명에 맞는 로직을 실행



블로그앱에서 현재 운용 중인 MethodChannel 종류

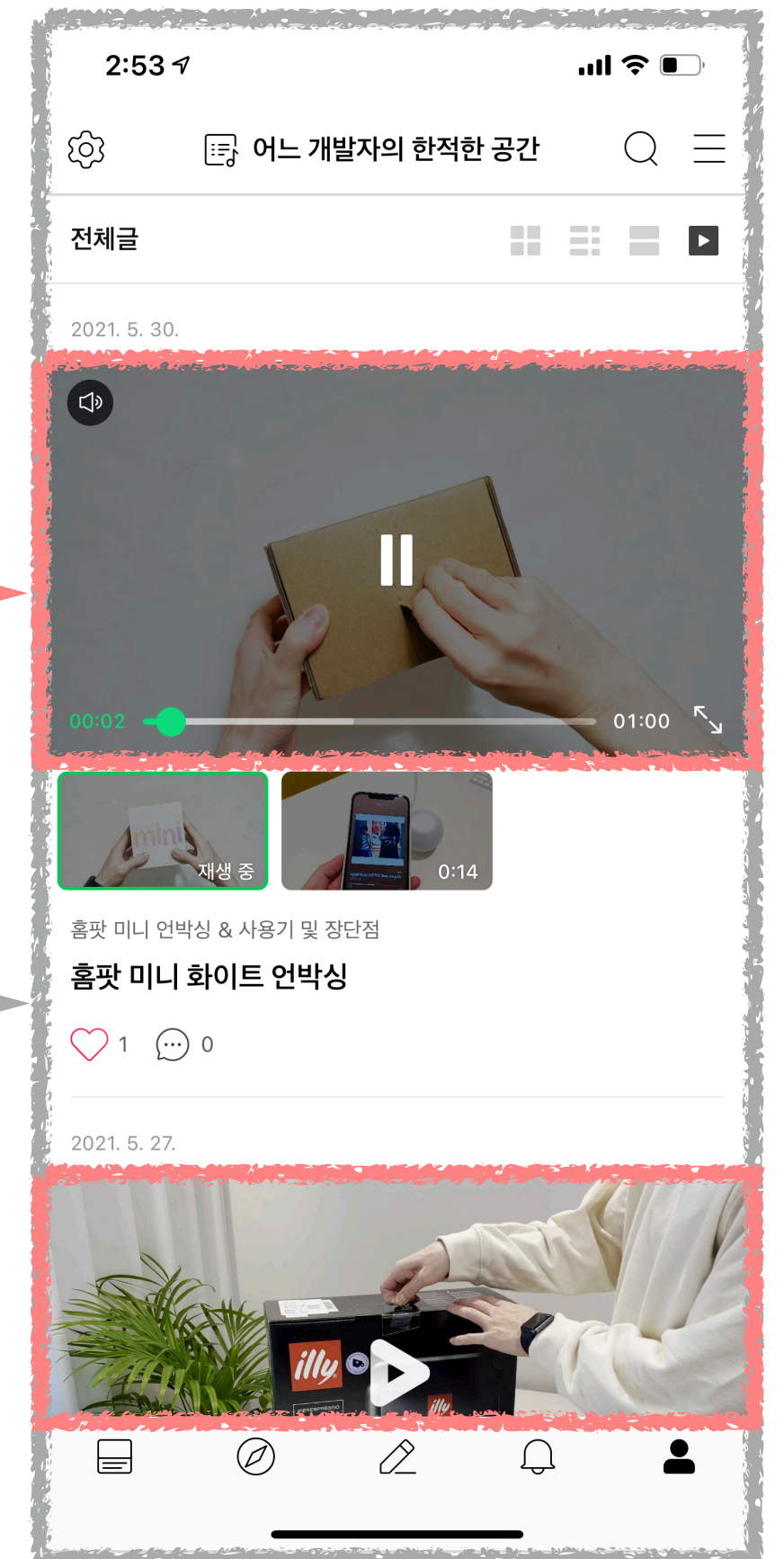
3.5 네이티브에서 사용 중인 라이브러리나 프레임워크는 어떻게? **N DEVVIEW 2021**

platform-view

- 기존의 블로그앱에는 프리즘플레이어를 통해 인라인에서 동영상을 재생시키는 기능이 존재.
- 플러터로 포팅할 수 없는 수준의 프레임워크.
- 플러터에서 제공하는 platform-view를 통해 플러터 위젯 상에서 네이티브앱의 UI를 직접 표시할 수 있음.

프리즘플레이어 영역
: 네이티브

나머지 영역
: 플러터

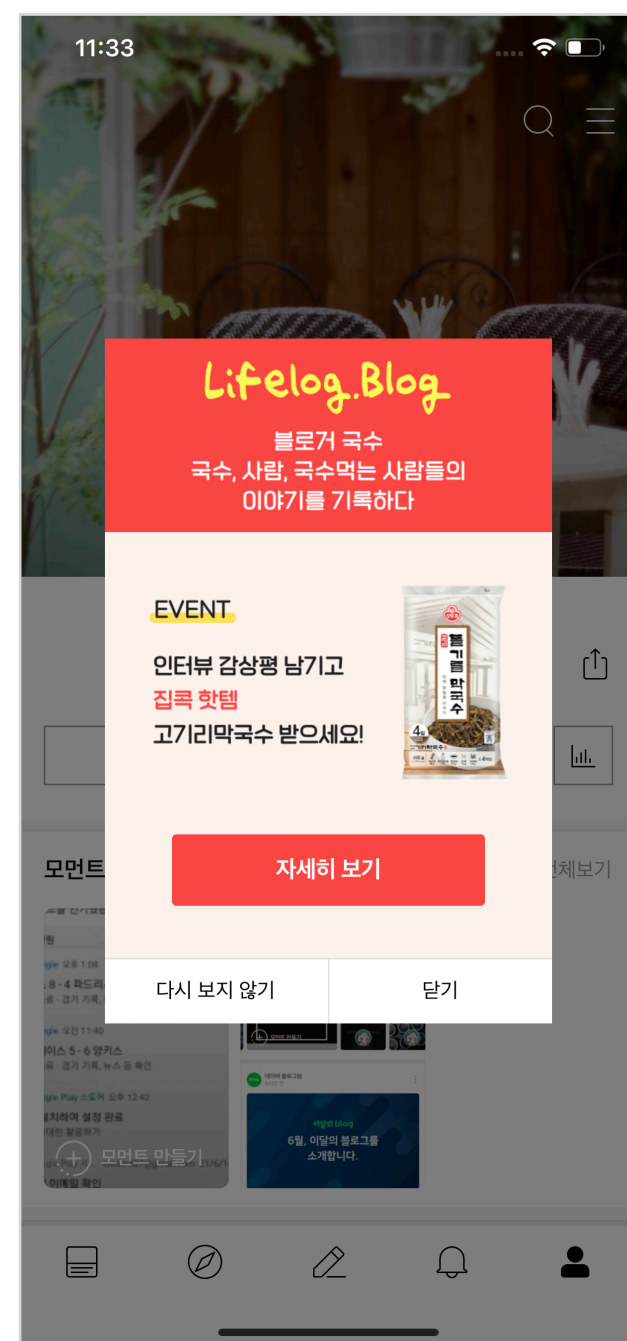


'내 블로그' 메뉴 내 글 목록(동영상 타입)

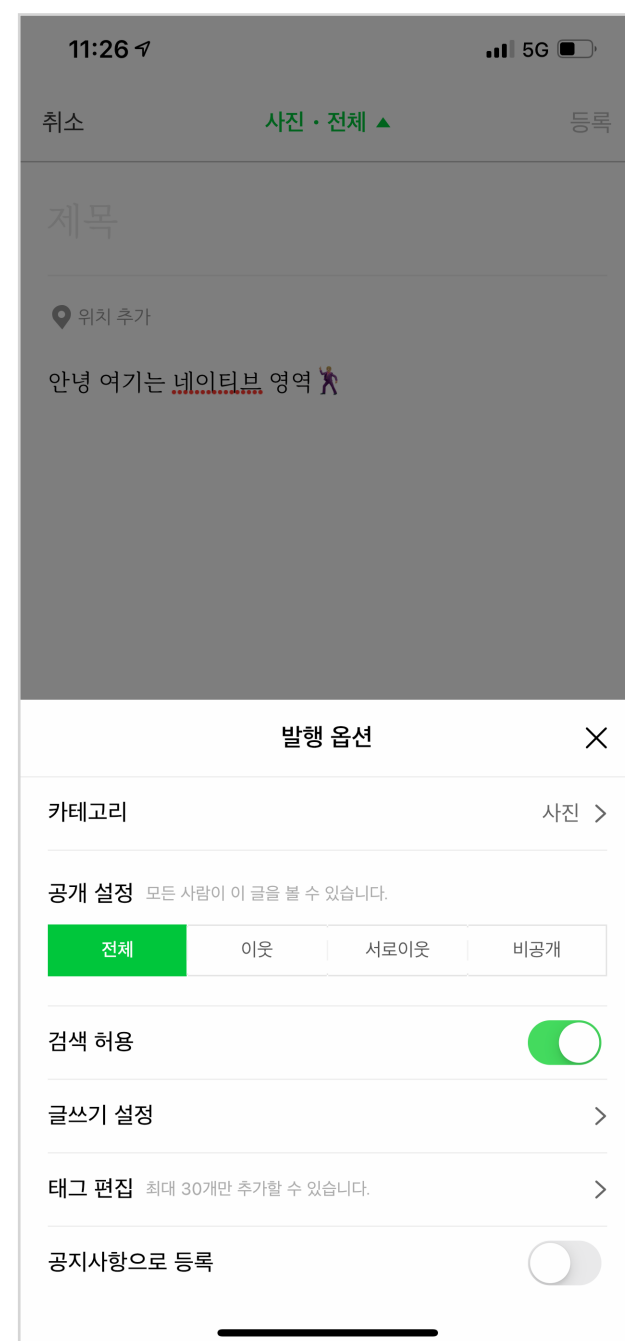
4.플러터 적용 결과

4.1 '플러터로 개발한 화면은 티가 나지 않을까요?' **N DEVIEW 2021**

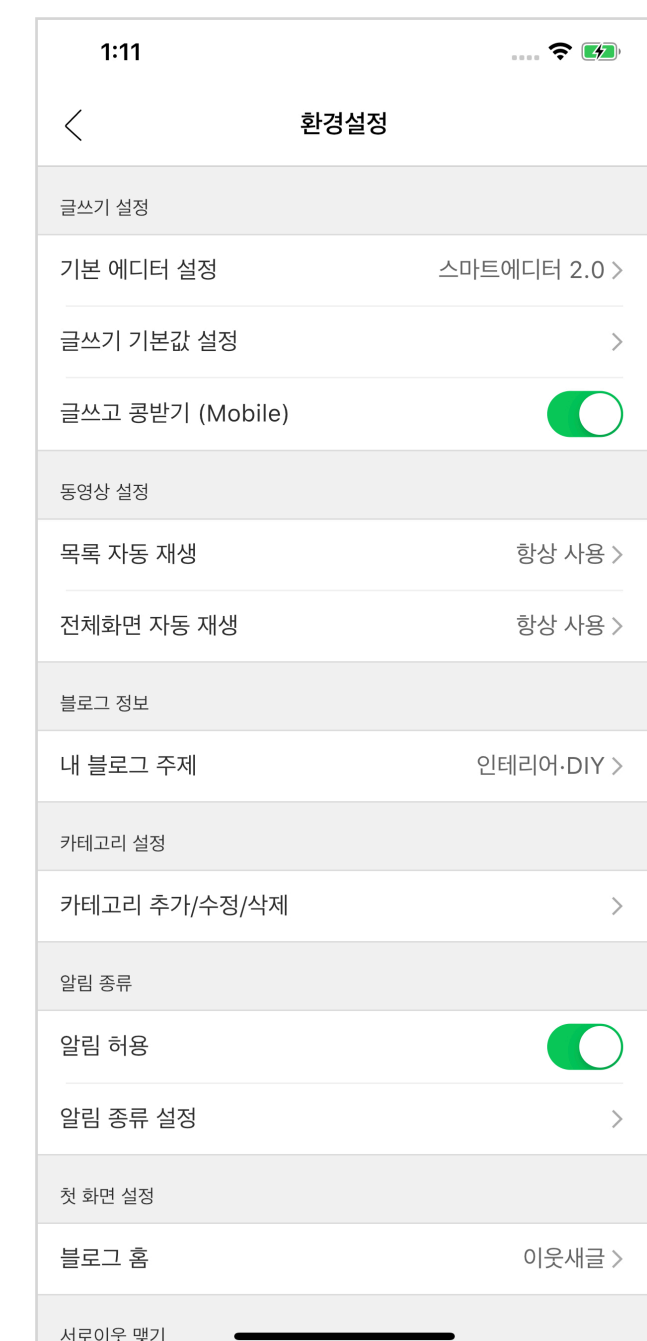
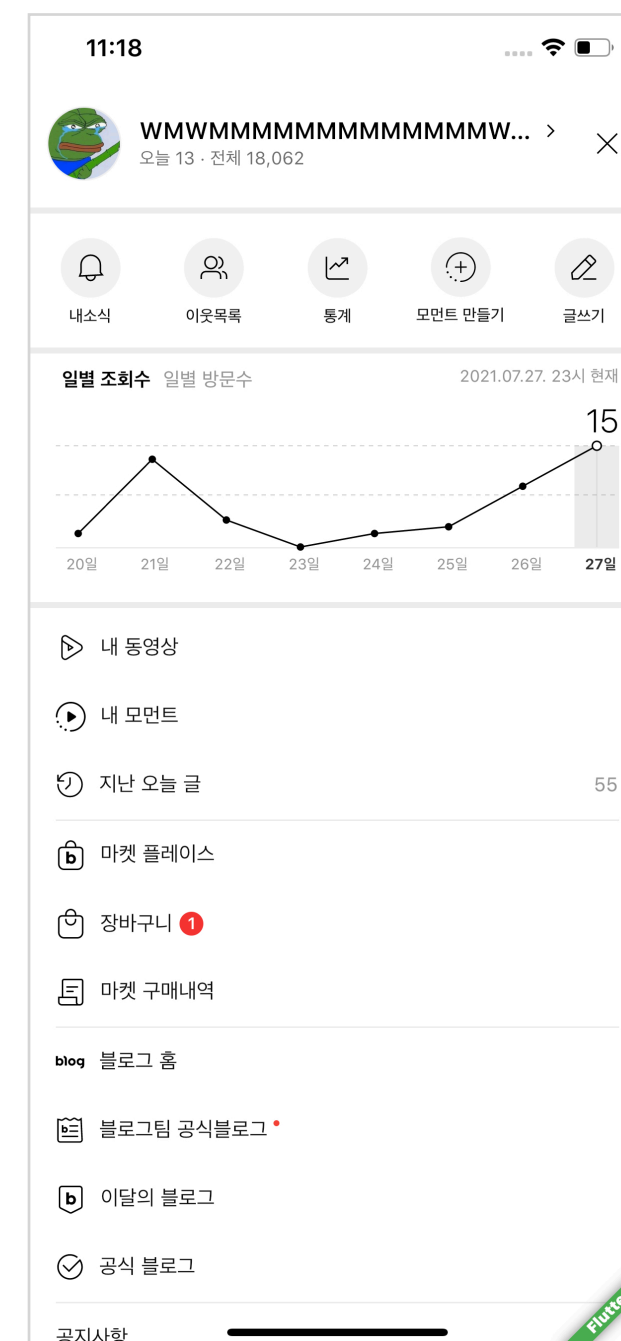
- 투명 배경 위에 다이얼로그 형태로 플러터 위젯을 구현하여 네이티브에 연동한 화면.
- 스펙이 바뀌면서, 기존 네이티브로 작업된 화면을 플러터로 전환하여 적용한 화면.



네이티브앱 화면 상에 투명 배경 위에 위젯으로 표시된 예시



네이티브에서 플러터로 전환 된 사이드메뉴



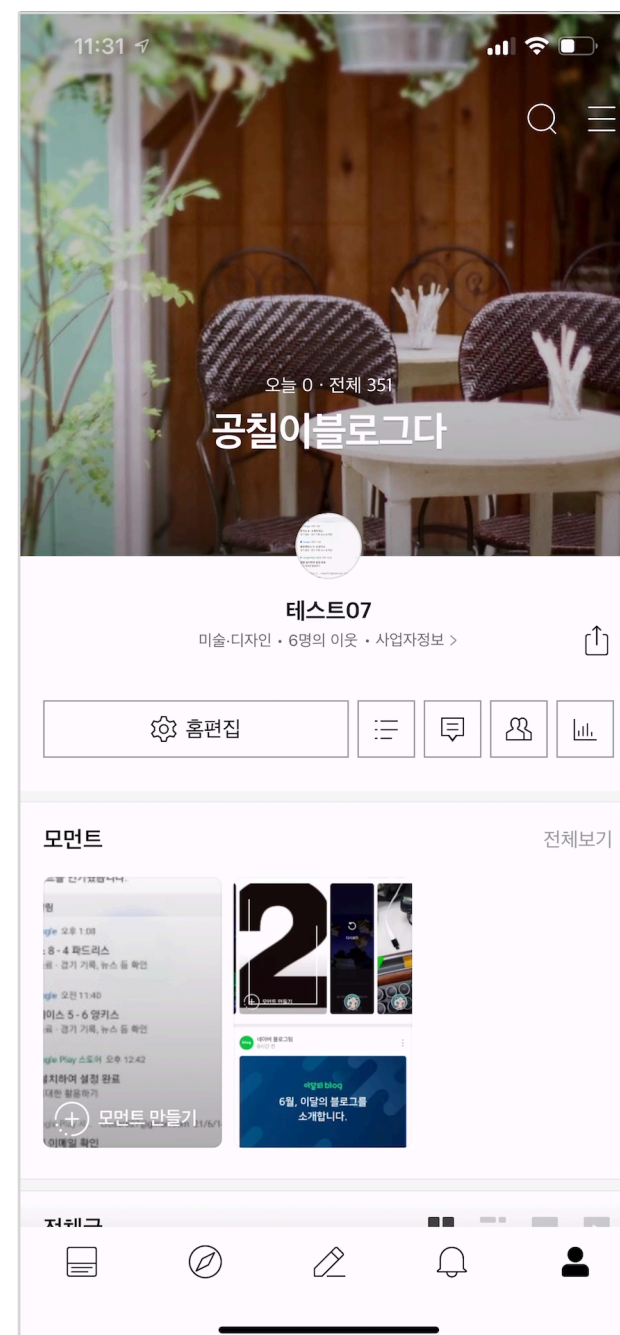
네이티브에서 플러터로 전환된 앱 설정 화면

4.2 네이티브와 플러터 화면 간 전환 사례

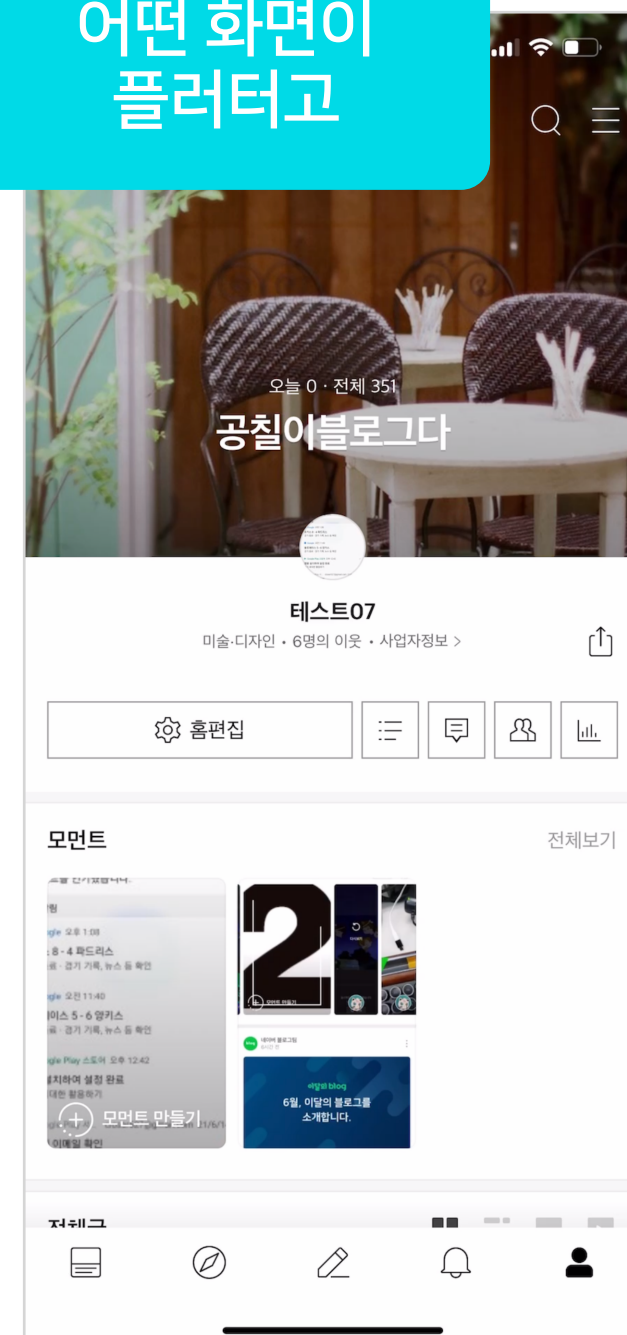
- 네이티브 화면과 플러터 화면 간 전환,
 - 플러터로 작업한 화면의 실제 인터랙션,
- : 현재 블로그앱은 네이티브와 플러터 화면이 혼용되어 적용 중.

어떤 화면이
네이티브일까요?

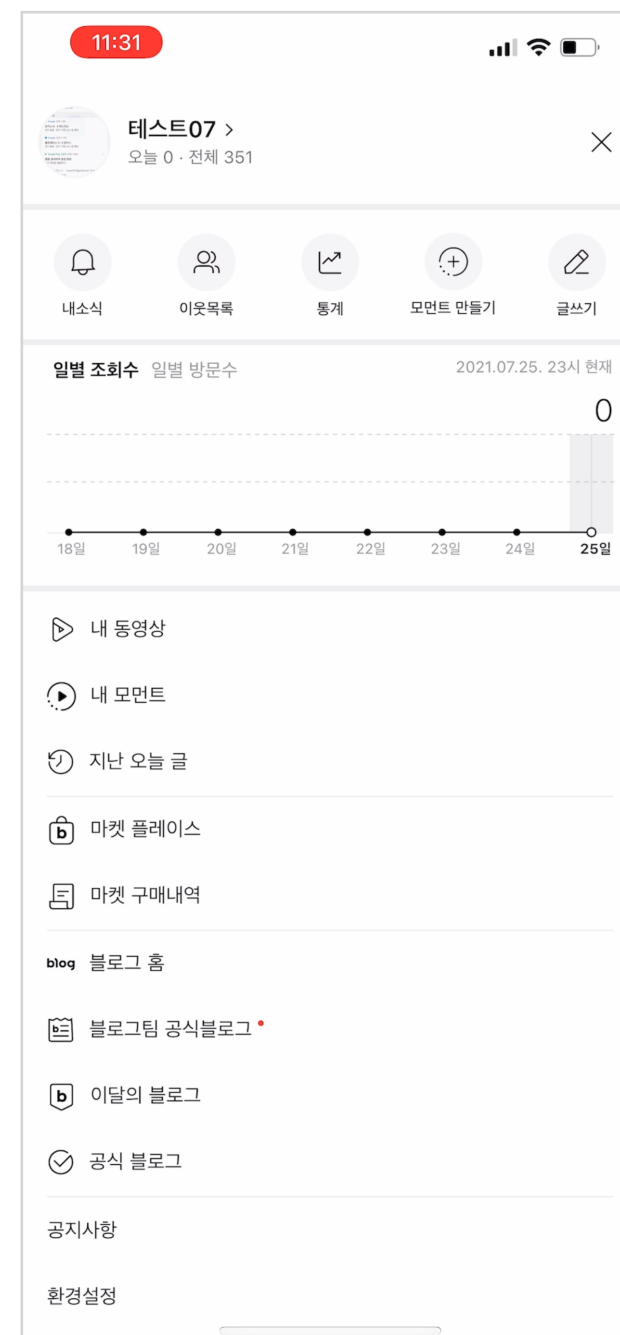
어떤 화면이
플러터고



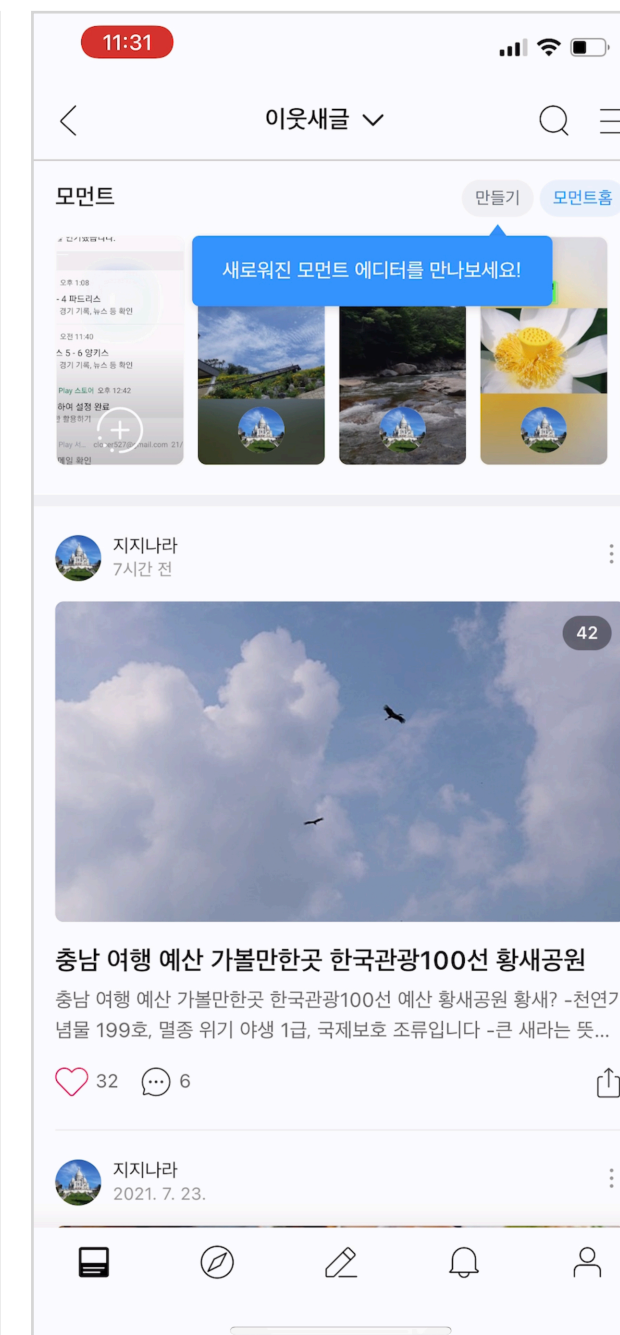
네이티브와 플러터 화면 간 전환 예시



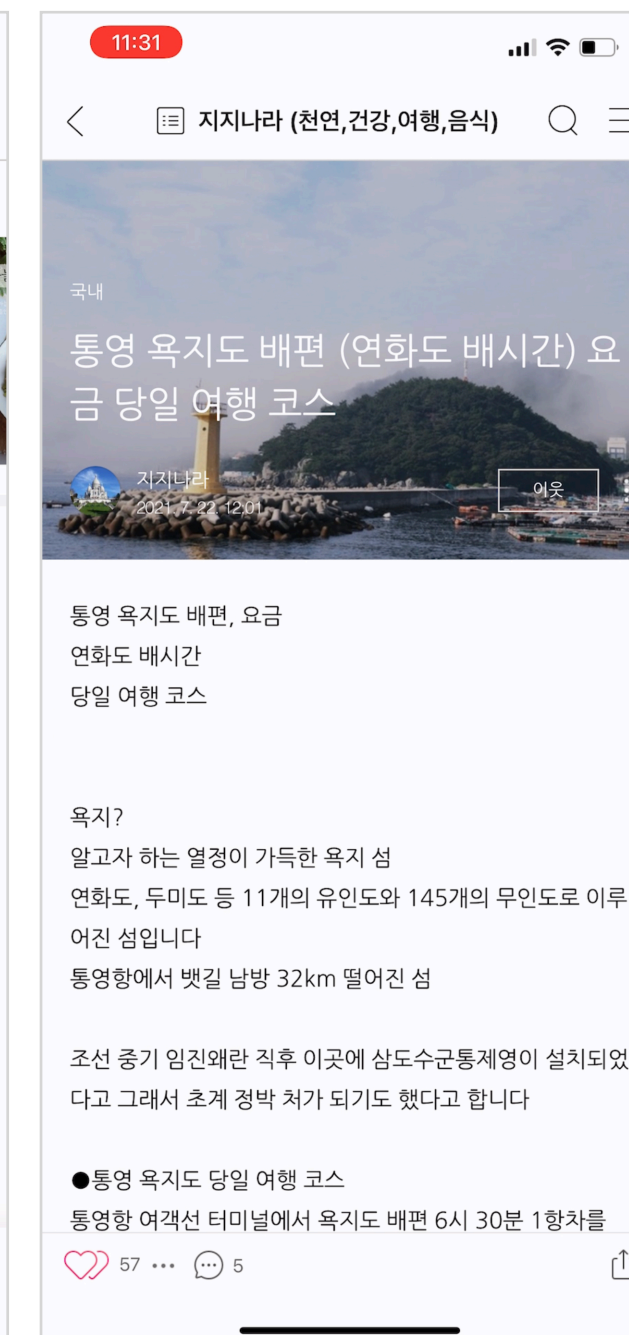
네이티브



플러터



플러터



네이티브

5. 플러터 도입에 대한 회고

5.1 UI 개발 측면에서 최고

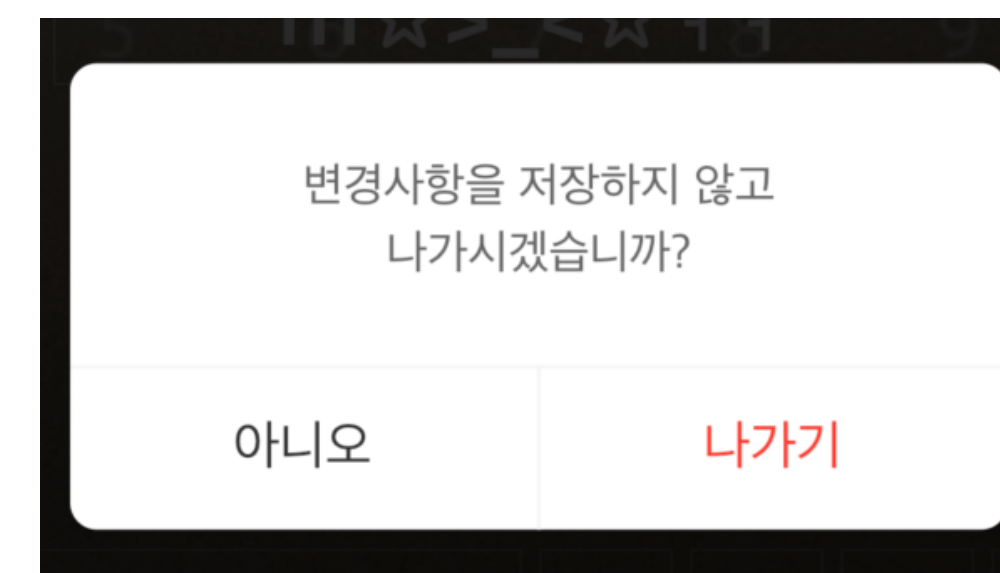
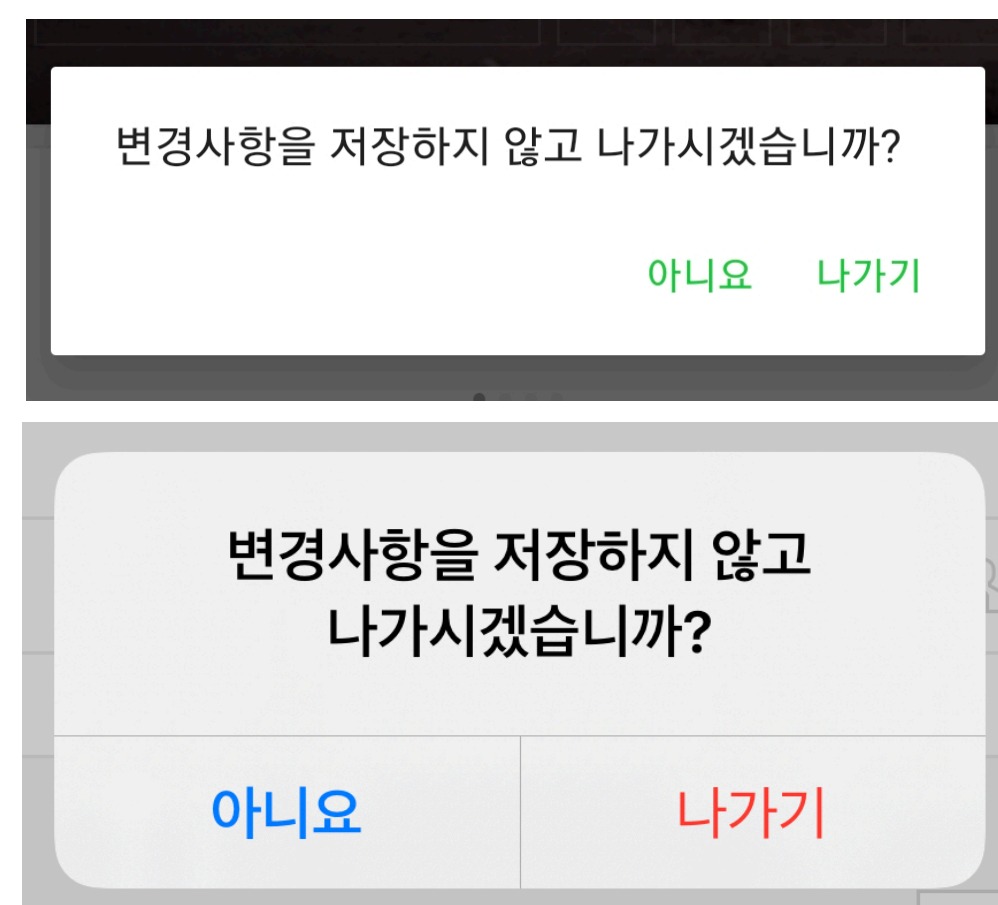
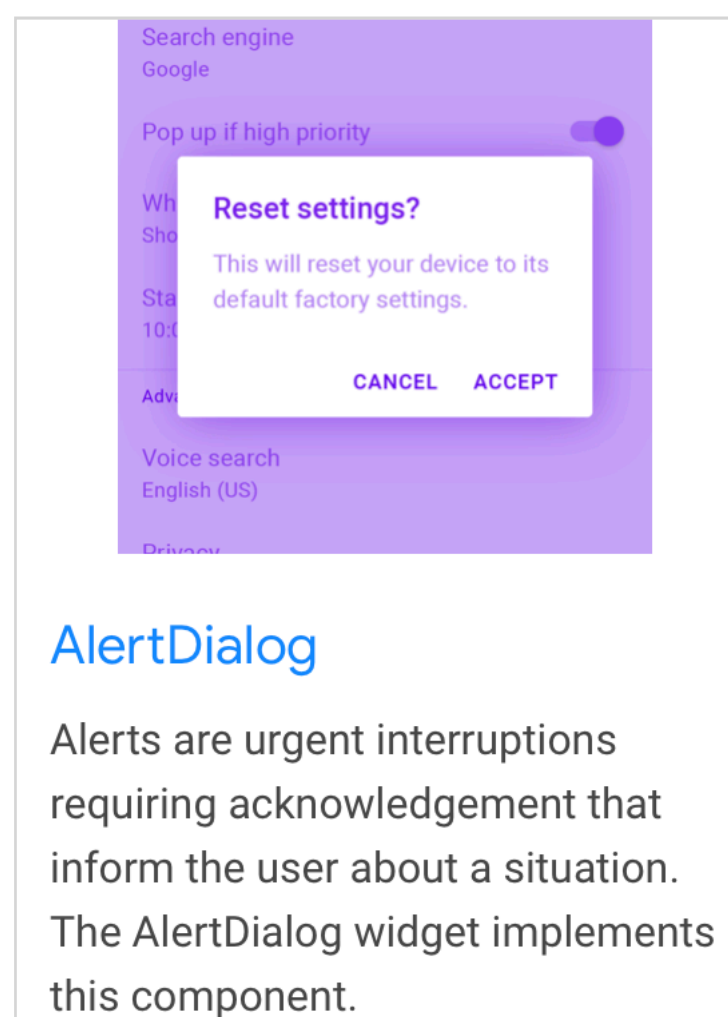
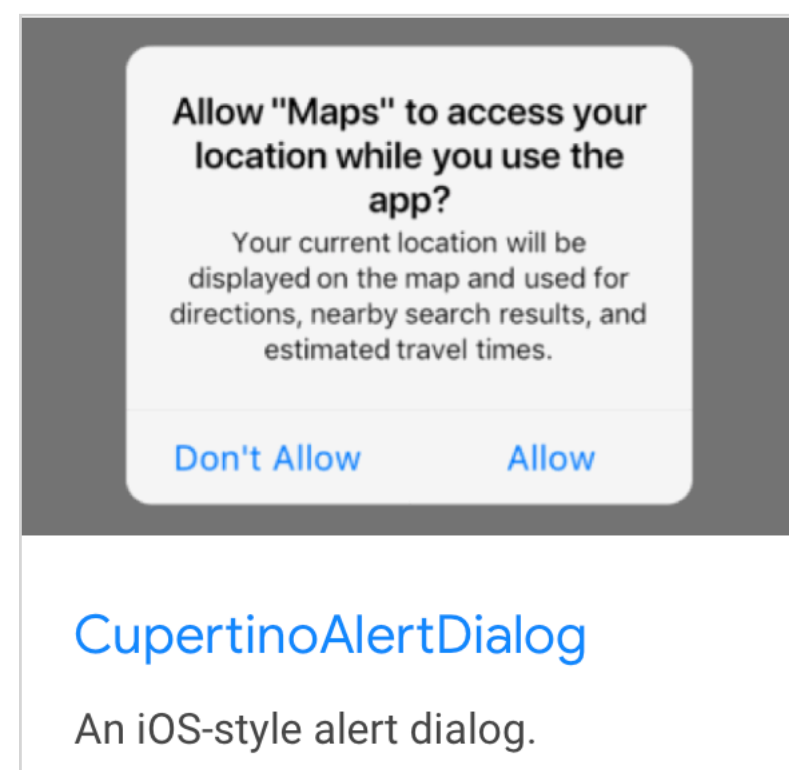
- UI 개발 생산성이 매우 좋아진다.
- Declarative UI 개발은 코드 작성도 직관적이고 가독성도 좋다.
- UI는 편하게 조립하고 상태 관리에만 신경 쓰면 OK.
- 재사용성 높은 위젯 방식.
- 무거운 프로젝트일수록 더 소중할 핫리로드.
: 단순한 시간절약을 넘어 흐름을 끊지 않고 업무 집중에도 긍정적 영향.
- 플러터에서 제공하는 다양한 위젯을 적극 활용하는 것을 추천.



특) 진짜 별거 다있음

5.2 기본 UI 컴포넌트의 모순?

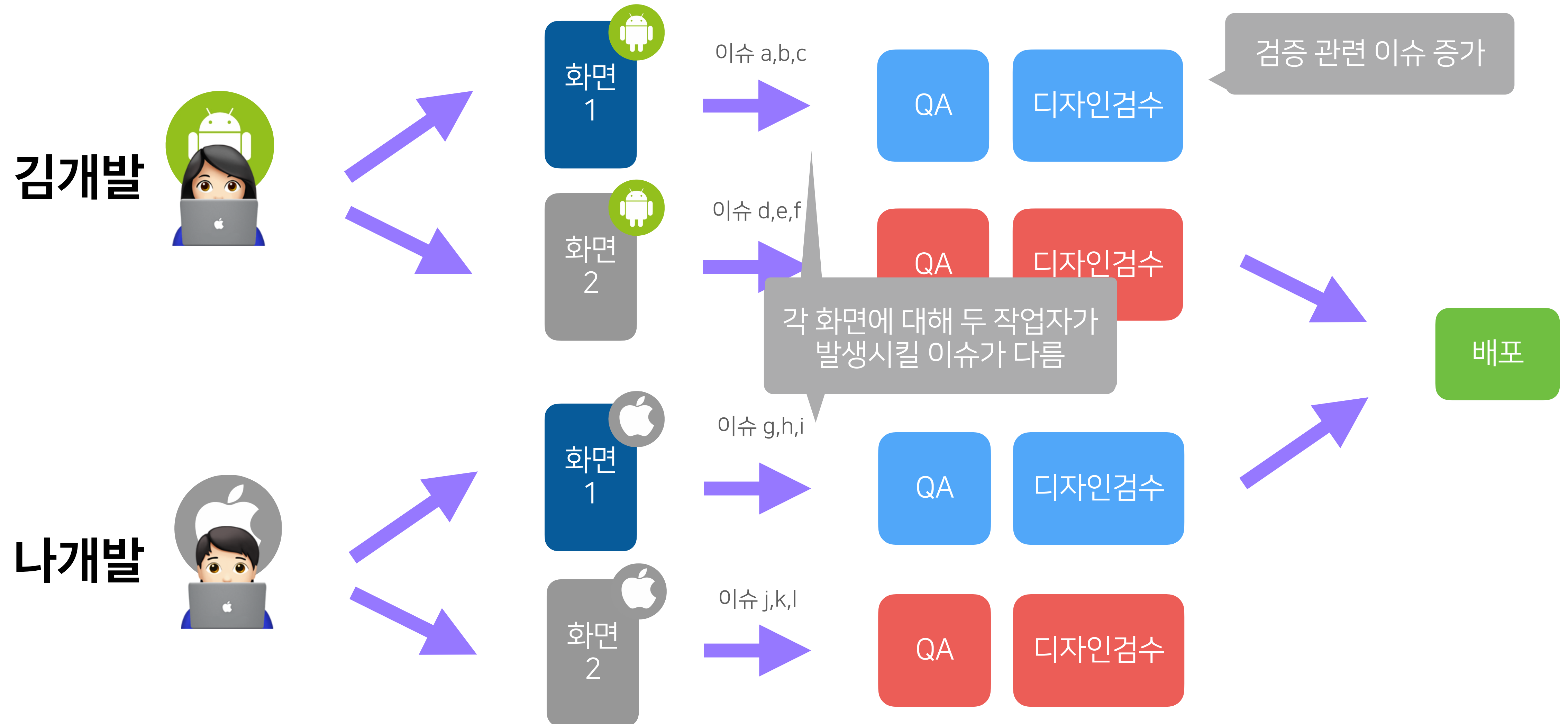
- iOS나 안드로이드 플랫폼의 기본 UI 컴포넌트를 많이 쓸수록 작업에 불리.
- 시스템 얼럿, 버튼 등; 각 플랫폼 별로 분기 처리해야 함.
- 오히려 커스텀 UI로 양 플랫폼 통합하는 편이 좋을수도?
- 그러나 각 플랫폼 별 기본적인 사용 행태나 관습을 무시해선 안될 것.
- 블로그에서는 기획/디자인과 논의하여 커스텀 UI으로 통합하는 방향으로 논의 중.



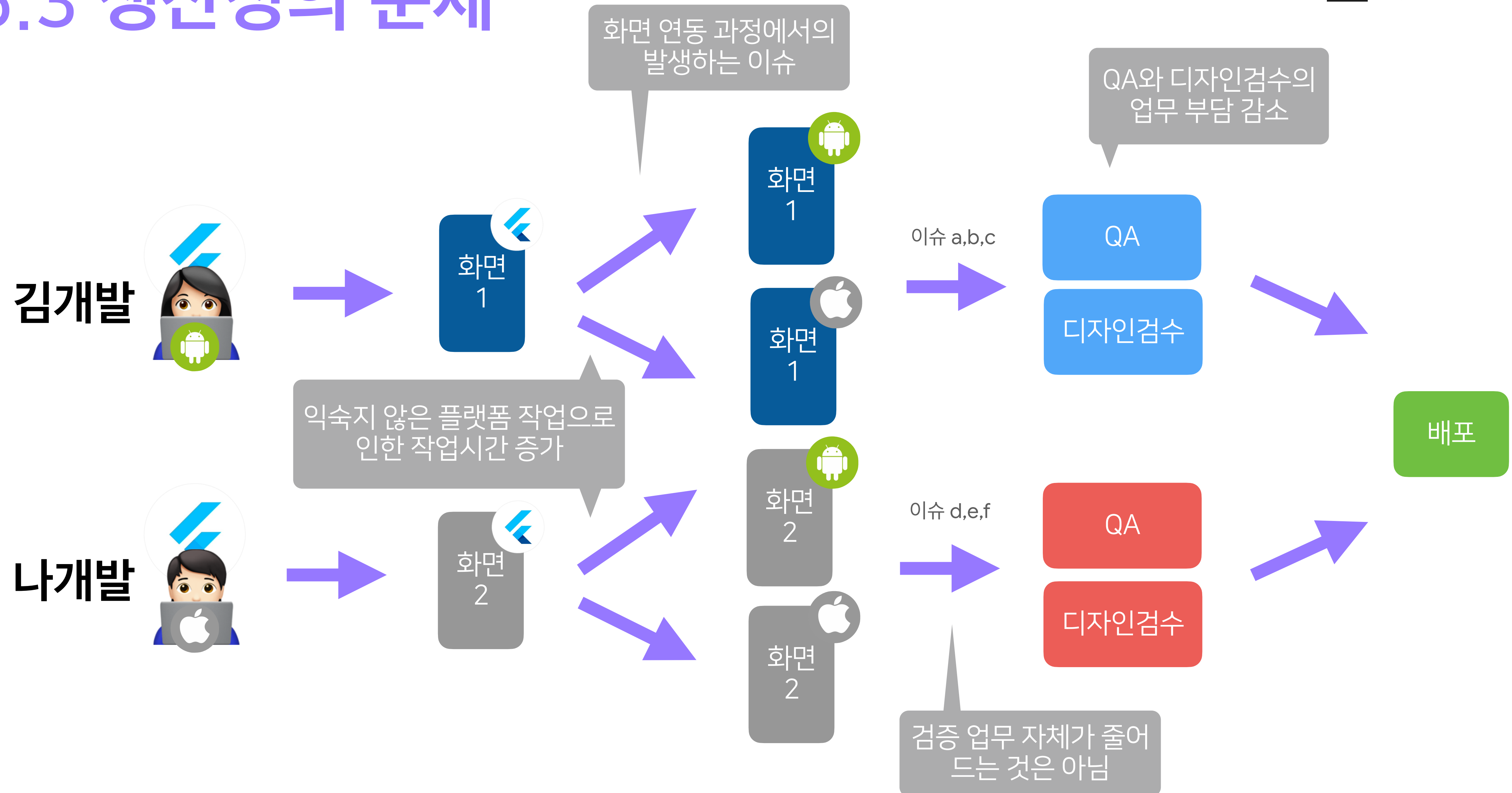
통합 UI로 전환한 예시

플러터에서 제공하는
각 플랫폼 스타일의 AlertDialog 위젯

5.3 생산성의 문제



5.3 생산성의 문제



5.4 개인적인 소회

아쉬운 부분

- 현재는 1인 1화면 개발로 속도전 중.
- 조직 전체의 효율은 증가했지만 작업자 개인의 효율로 이어지려면...
 - 네이티브 연동이 쉽지 않았다. Add-to-app의 사례가 많지 않아서 우여곡절.
 - 숙련도가 떨어지는 플랫폼 작업은 확실히 더뎴. (학습만이 살 길...)
 - 가능하다면 각 플랫폼 별로 한 명씩 정/부를 나누어 페어로 작업하는 것이 좋을 듯.
 - 공유의 중요성.

커리어 측면

- iOS도 아직 미숙한데 다른 플랫폼을 한다면 커리어에 불리하지 않을까?
- 결국은 기존 플랫폼의 앱으로 배포하는만큼 네이티브앱 개발 스킬도 배제할 수 없음.
- 기술적 시야 넓히는데 도움.

5.5 우리팀에도 플러터를 도입해도 될까?

나름 적합했다고 볼 수 있는 블로그앱

- 블로그앱은 글과 이미지와 같은 미디어를 주로 소비하고 탐색하는 서비스.
- 복잡한 비즈니스 로직, 디바이스 성능 < 사용자가 소비하는 UI와 사용자경험의 비중

어떤 서비스가 적합할까?

- 커스텀 UI가 많은 서비스.
- 오래되어 레거시 개선이 필요한 서비스.
- 개발 인력이 부족하거나 불균형이 있는 조직.
- 조직 구성원들의 의사도 중요. (양쪽 플랫폼을 다루는 능력이나 의향이 있는지, 새로운 커리어 영역의 도전)
- 스타트업과 같은 조직, 빠른 증명이 필요한 서비스나 기능을 개발이 필요한 부서.



Thank You

네이티브앱 내 플러터 화면 전환 연동

- Flutter 공식 홈페이지: <https://flutter.dev/>
 - Add To Existing App: <https://flutter.dev/docs/development/add-to-app>
 - Flutter architectural overview: <https://flutter.dev/docs/resources/architectural-overview>
- Flutter Official Youtube: <https://www.youtube.com/channel/UCwXdFgeE9KYzIDdR7TG9cMw>
 - FlutterLogo (100번째 이 주의 위젯 영상!): <https://youtu.be/aAmP-Wcl6dg>
 - Building platform adaptive apps | Session: <https://youtu.be/RCdeSKVt7LI>
- Flutter Appbrewery 기본코스: <https://www.appbrewery.co/p/intro-to-flutter>
- Provider 개념 이해: <https://codewithandrea.com/videos/flutter-provider-essential-guide>