

REPORT

최종 보고서

- Flutter OSS 활용 반려 식물 관리 앱 '식물집사' 개발 -



주 제 : 프로젝트 최종 보고서
교 수 : 송인식 교수님
학 과 : 소프트웨어학과
학 번 : 32193469
이 름 : 이정현
제 출 : 2023. 06. 05.(월)

목 차

1장 서론

01 프로젝트 개요	-----	3p
02 프로젝트 기술 스택	-----	3p
03 프로젝트 목표 및 기대 효과	-----	4p

2장 요구사항 분석

01 앱의 주요 기능 및 요구사항 목록	-----	5p
02 사용자 시나리오	-----	6p

3장 설계

01 앱의 전체 구조 및 아키텍처	-----	7p
02 화면 설계 및 UI/UX 구성 요소	-----	11p
03 데이터베이스 설계	-----	12p

4장 구현

01 사용된 변수 및 자료형	-----	13p
02 주요 코드 구현 사항 및 기능	-----	13p
03 문제 해결 과정 및 알고리즘	-----	14p

5장 테스트

01 사용된 테스트 전략 및 방법	-----	16p
02 단위 테스트 및 통합 테스트 결과 요약	-----	16p
03 버그 및 이슈 관리	-----	16p

6장 결과

01 프로젝트 후기 및 결과 평가	-----	17p
02 프로젝트 요약	-----	19p

1. 서론

1.1 프로젝트 개요

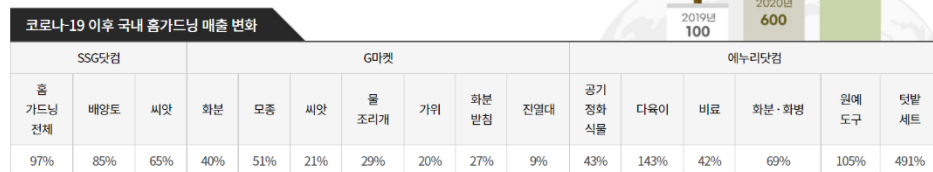
① 반려식물 관리 서비스 '식물집사'

최근, 코로나 19로 인한 실내생활이 증가함에 따라 무료함을 달래기 위한 취미 생활로 "반려식물 키우기(홈 가드닝)"에 대한 관심이 급부상하고 있다. 필자 또한 실제로 초보 식물 집사로서 방울토마토를 집에서 재배해보며 많은 불편함을 느꼈다. 식물의 관리법에 대한 지식이 부족할 뿐 더러, 물은 몇 번 줘야 하는지 쉽게 알 수 없기 때문에 일부 식물을 죽이기도 했다. 하지만, 쉽게 쓸 수 있는 식물 관리 플랫폼이 있다면 보다 쉽게 식물을 관리할 수 있고, 이는 추가적인 식물 재배에 대한 "지속적인 관심"으로 이끌 것이다.

반려식물 관리 서비스 '식물집사'의 기획 단계에서, 관리 서비스로서의 적절한 기능을 사용자에게 제공하기 위해서는 사용자가 상시에 확인할 수 있는 '알림' 기능이 최적화된 모바일 앱 환경에서 구현하는 것이 효과적일 것이라고 판단했다. 그 중에서도 하나의 프로그래밍 언어와 하나의 코드베이스를 사용하여 Android, iOS 등 여러 플랫폼용 어플리케이션을 구축할 수 있는 오픈소스 프레임워크인 'Flutter'를 사용해 프로젝트를 진행했다.

코로나-19로 집에서 시간을 보내는 사람들이 늘어나면서
식물 키우기가 인기 취미로 떠오르고 있고, 관련 산업 매출은
급격하게 증가중입니다.

코로나-19 이후 국내 홈가드닝 매출은 급격하게 증가 중이며,
향후 2023년은 지금의 매출 규모의 8배를
전망하고 있습니다.

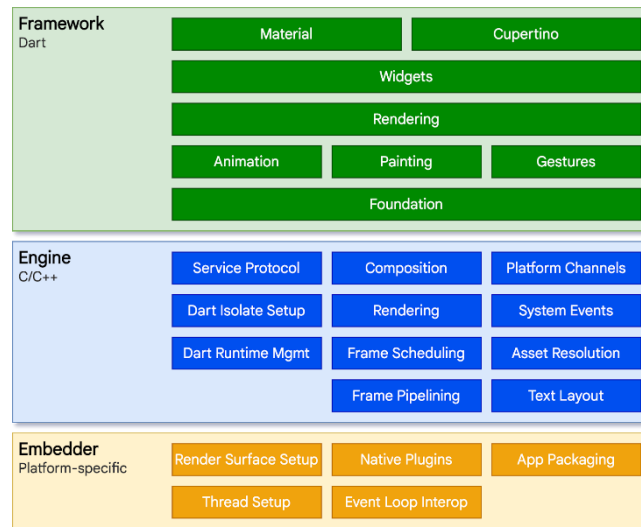


[그림 1] 국내 홈 가드닝 시장 규모

1.2 프로젝트 기술 스택

① Flutter

Google에서 개발 및 지원하는 오픈소스 프레임워크로서 다양한 플랫폼에 대한 애플리케이션의 UI를 단일 코드 베이스로 구축 가능하다. 네이티브 앱 개발과 유사한 성능, 빠르고 일관적인 렌더링, 위젯 등 개발자에게 편리한 도구를 제공해주는 특징이 있다.



[그림 2] Flutter 아키텍처: Layer Model

② Android Studio

안드로이드 및 안드로이드 전용 앱 제작을 위한 공식 통합 개발 환경이다. 다양한 기능 중 안드로이드 에뮬레이터는 앱을 테스트해볼 수 있는 도구로 배터리, 네트워크, 전원 상태를 조작할 수 있어 각 상황에 앱이 어떻게 대응하는지를 확인할 수 있는 특징이 있다.

③ GitHub

Git 기반의 버전 관리와 협업 기능을 제공하는 소스 코드 호스팅 플랫폼으로, Git 리모트 저장소, 코드 리뷰, 프로젝트 관리, 문서화 위키, 오픈 소스 프로젝트 등 다양한 기능을 지원 본 프로젝트도 GitHub에 PR 생성 후 주요 기능마다 브랜치를 생성해, 수정 사항이 있을 때마다 commit 하며 지속적인 코드 관리를 수행했다.

④ Firebase

모바일 및 웹 앱 구축을 위한 실시간 데이터베이스이다. 쿠키-세션을 주로 사용하는 웹과 토큰을 주로 사용하는 모바일 플랫폼에서 통일된 OAuth API를 사용할 수 있기 때문에 모바일과 웹을 동시에 서비스할 경우 유용하다는 특징이 있다.

1.3 프로젝트 목표 및 기대 효과

① 프로젝트 목표

본 프로젝트는 반려식물을 재배하기 시작한 2-30대의 “초보 식물집사”에게 가장 큰 걸림돌이 되는 “식물 관리법 미숙”의 문제를 아래와 같은 3가지 기능을 통해 해결하고자 한다.

1) 식물 프로필 및 매뉴얼: 등록된 식물의 관리 매뉴얼 자동 제공 기능

→ 사용자가 프로필에서 식물 종을 선택하면 해당 식물의 관리 매뉴얼이 자동으로 등록된다. 이를 통해 사용자는 해당 식물의 관리법을 직접 찾을 필요 없이 쉽게 얻을 수 있다.

2) 식물 성장 다이어리: 반려식물 성장 기록 및 알림 기능

→ 사용자에게 캘린더 및 타임라인 형식의 기록공간을 제공하여 사진과 글을 통해 식물의 성장을 꾸준히 기록하고 관찰할 수 있게 한다.

→ 물주기 알림 및 물주기 후 체크 기능을 제공하여 바쁜 일상 속에서도 잊지 않고 급수 관리를 하도록 돕는다.

3) 식물 집사 커뮤니티: 전문가 상담을 통한 반려식물 솔루션 제공 기능

→ 식물재배 중 이상징후 발생 시, 사용자가 커뮤니티에 글을 등록하면 전문가가 댓글을 통해 식물 상태를 진단하고 알맞은 솔루션을 제공하여 위급상황에 대처하도록 돕는다.

② 프로젝트 기대 효과

- 사용자는 손쉽게 식물을 관리하고, 전문적인 솔루션을 통해 식물 관리법 관련 지식 습득
- 지속적인 홈 가드닝 활동 장려로 반려식물 시장의 확대
- 본 프로젝트에서 GNU 기반 GPL 3.0 라이선스 사용으로 누구나 개발자(이하 이정현)의 코드를 공개 · 수정 · 배포하여 지속적 수정을 통한 기술 혁신 독려

2. 요구사항 분석

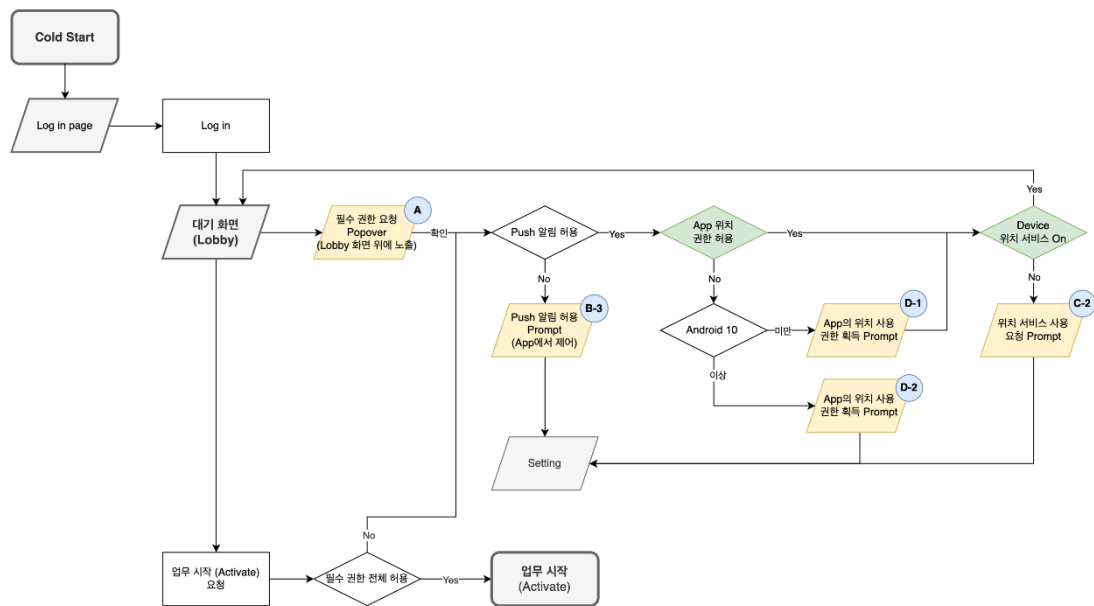
2.1 앱의 주요 기능 및 요구사항 목록

본 프로젝트의 핵심 기능인 식물 관리 매뉴얼 제공, 식물 다이어리 기록, 식물집사 커뮤니티 형성 기능을 구현하기 위해 필요한 세부사항은 다음과 같다.

주요 요구사항	요구사항 상세설명	참고자료
1) 식물 다이어리 기능	사진, 글 등을 통해 식물의 성장을 캘린더 형태 속에서 지속적으로 기록할 수 있는 공간 구현	[MZ세대의 기록에 대한 수요 및 트렌드] www.2e.co.kr/news/
2) 전문가 상담	식물 문제 발생 시, 게시글에 식물 사진을 올려 전문가가 솔루션을 답글 형태로 제공 가능한 공간 구현	[식물 재배 어려움 근거자료] https://live.lge.co.kr/tiun-1/
3) 커뮤니티	게시판, 식물 사진을 통해 사용자들 간의 홈 가드닝 정보 공유 및 소통을 위한 공간 구현	[관심사 기반 커뮤니티의 중요성 통계자료] https://www.sisajournal.com/news/
4) 식물 관리 매뉴얼	식물 관리 정보 수집 (명칭, 꽃말, 온도, 습도, 일조량, 급수량, 분갈이 주기 등)	[식물 데이터 수집 출처_국가생물종 지식정보시스템] https://www.nature.go.kr/kbi/plant/ [식물 RDBMS 이미지]
5) 식물 정보 불러오기	사용자가 등록한 식물에 따른 관리법을 포함한 정보를 자동으로	[식물 정보 빅데이터 검색 결과자료] https://www.nongsaro.go.kr/portal/ps

생성		
6) 사용자 구분	일반 사용자와 전문가를 구분할 수 있는 유저 등급 필요	 Expert /  전문가
7) 식물 재배 수요 확인	홈 가드닝 시장의 확대에 대한 근거 자료 필요	[홈가드닝 수요증가 통계자료] https://www.nongsaro.go.kr/portal/
8) 전문가 섭외	사용자의 질문 글에 솔루션 제공이 가능한 전문가 섭외	[양재 꽃시장 내 전문가 분포 자료] https://flower.at.or.kr/yfmc/front/market/
9) 앱 디자인 컨셉	사용자의 글에 답글을 달 수 있는 전문가 집단 섭외 '힐링'의 컨셉을 극대화하는 초록 색상과 차분한 분위기의 UI/UX 개발	[Z세대에게 각광을 받는 반려식물 트렌드] https://www.20slab.org/Archives/38217

2.2.1 사용자 시나리오



[그림 2] 사용자 시나리오 흐름도

2.2.2 상세 사용자 시나리오

① 사용자 접속:

사용자는 어플리케이션에 접속하여 구글 로그인 및 회원가입을 진행

② 회원 정보 저장:

회원가입 데이터는 Firebase의 데이터베이스(Fire Store)에 저장되고, 해당 정보가 인증이 되면 메인 화면으로 이동

③ 식물 추가:

메인 화면에서 사용자는 본인의 식물을 추가 가능. 이때 선택한 식물의 정보가 자동 생성되며 식물의 이름, 종류, 사진, 물주기 등을 직접 설정 가능

④ 식물 정보 저장:

입력된 식물 데이터는 Firebase의 데이터베이스에 저장되며 메인 뷰(view)에 새로운 식물을 추가

⑤ 상세 페이지 이동:

사용자는 메뉴 버튼을 통해 캘린더 혹은 커뮤니티 페이지에 접근

⑥ 성장 다이어리 작성:

캘린더 페이지에 접속할 경우 앞서 추가한 식물에 대한 물주기 날짜가 달력에 표시되며, 사용자는 이를 확인. 또한 캘린더 하단에 타임라인 형태로 해당 식물에 대한 성장 다이어리를 작성하여 추가 가능 (다이어리 내용으로는 제목 내용 사진 등을 입력. 이 데이터는 모두 Firebase에 저장)

⑦ 게시판 작성 및 수정:

커뮤니티 페이지에 접속할 경우, Firebase로부터 각종 글들을 불러옴. 각 글은 제목, 내용, 댓글, 좋아요를 포함하고 있으며, 사용자는 식물에 대한 궁금증이 생기거나 본인 식물에 이상 징후가 발생할 경우 커뮤니티에 글을 작성하여 사용자 혹은 전문가로부터 솔루션을 제공받을 수 있음 (작성한 글은 모두 Firebase에 저장되어 다른 유저들도 확인 가능)

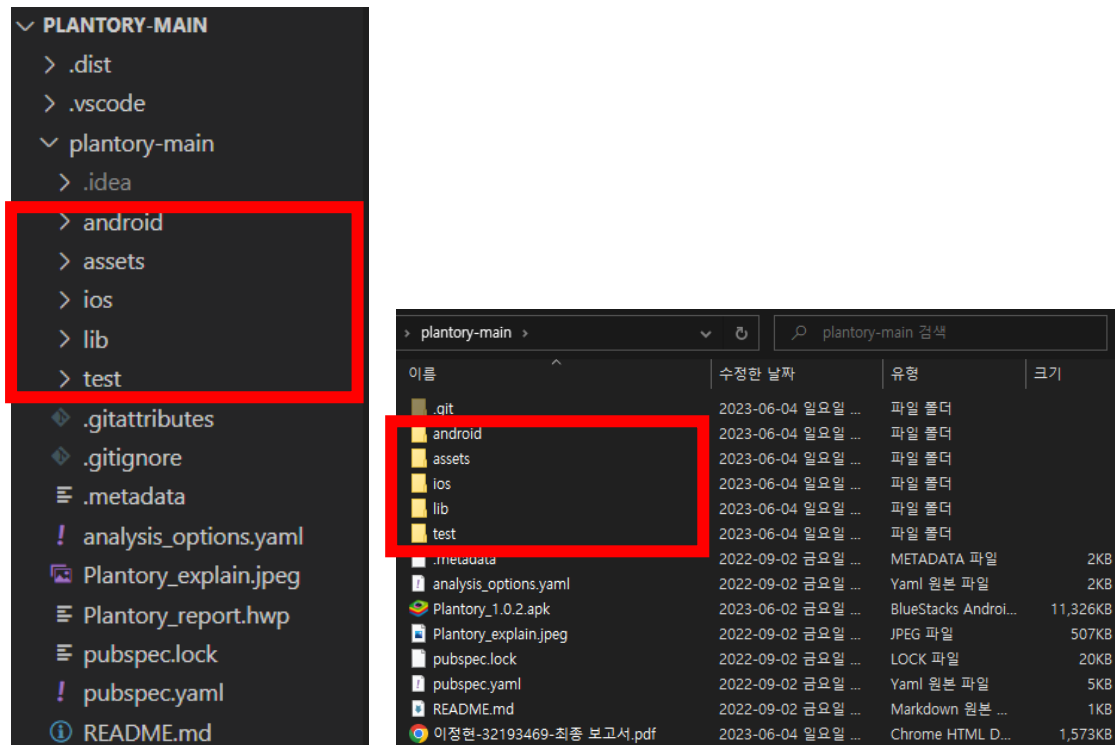
3. 설계

3.1 앱의 전체 구조 및 아키텍처

① 프로젝트 구조

1. 개발이 이루어지는 lib 폴더
2. 테스트를 진행하는 test 폴더
3. 플랫폼별 빌드와 배포가 이루어지는 build , andriod, ios 폴더가 존재한다.

해당 프로젝트에서는 lib 폴더와 최상위에 존재하는 pubspec.yaml 을 가장 많이 사용하고 수정했다. 또한 실질적인 라이브러리들은 Extenal Libraries 에 존재한다.



[그림 3] 식물집사 Flutter 디렉토리 구조

② 식물집사 앱의 구조: pubspec.yaml

- pubspec.yaml 은 gradle 이나 maven 처럼 패키지 모듈을 설치하고, 버전에 의존성 등을 관리해주는 설정 파일이다.
- 배포하게 되는 앱의 패키지 일부인 name, version, desc , author (이름, 버전정보, 설명, 만든이)에 대한 정보가 포함된다.
- 플러터 안에서 사용되는 assets (이미지, 폰트, 미디어 파일)을 사용하고 관리하는 파일

```

name: <APP_NAME>
description: A new Flutter project.

# 다음은 앱 버전 및 빌드 번호를 정의합니다.
# 버전은 점으로 구분된 세개의 숫자입니다. (예: 1.2.43)
# 그 다음 빌드 번호가 '+' 뒤에 표시됩니다. (옵션입니다.)
# 버전과 빌드 번호는 각각 --build-name과 --build-number를 지정하여 오버라이딩할 수 있습니다.
# semver.org에서 버전관리에 대해 자세히 알아보세요.
version: 1.0.0+1

environment:
  sdk: '>=2.0.0-dev.68.0 <3.0.0'

dependencies:
  flutter:
    sdk: flutter

# 다음은 앱에 Cupertino 아이콘 폰트를 추가합니다.
# iOS 스타일 아이콘으로 CupertinoIcons 클래스를 이용해서 사용하세요.
cupertino_icons: ^0.1.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  
```

[그림 4] pubspec.yaml 선언부

③ 식물집사 Flutter 아키텍처

1) main 메소드

```
void main() {
  runApp(const MyApp());
}
```

- 플러터 앱이 실행되면, 가장 먼저 실행되는 main 메소드
- 메인 메소드는 runApp 함수를 통해서 아래에 작성돼있는 MyApp() 클래스를 호출한다.
- main 에 의해서 호출되는 Myapp 에서 사용할 테마와 위젯을 지정한다.

2) MyApp(메인 앱)

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

본 프로젝트에서는 구글에서 제공하는 MaterialApp 디자인을 사용, 다음과 같이 앱의 제목, 테마, 위젯을 지정

- MyApp 클래스: 특정 화면에 대한 키가 명시됨 (Navigator 화면의 전환을 위해 약속된 Key)
- build 메소드: 앱의 실행에서 생명주기상에 가장 중요한 메소드로 UI 코드가 작성
- MaterialApp 하위: 앱 테마와 home 화면에 연결
- MaterialApp Widget: 초기 프로젝트의 UI 에 최상단에는 일반적으로 MaterialApp Widget 이 사용, 여기서 앱의 전체적인 테마도 지정하며, 화면의 구성과, 라우팅, 초기화면을 연결. 여기서는 home 옵션에 MyHomePage 가 연결되어 있다.

3) MyHomePage

```
class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key, required this.title}) : super(key: key);
  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }
}
```

setState() 메소드가 호출되면, 변화를 감지하여 앱의 화면이 순식간에 reBuild되므로 앱 화면 사용자에게 클릭에 바로 변경됨을 확인할 수 있다.

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Text(
            'You have pushed the button this many times:',
          ),
          Text(
            '$_counter',
            style: Theme.of(context).textTheme...,
          ),
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: const Icon(Icons.add),
    ), // This trailing comma makes auto-formatting nicer for build methods.
  );
}
```

여기서는 FloatingActionButton 버튼을 누르면 연결된 `_incrementCounter` 메소드가 호출되는데 작성된 메소드 안을 보면, `setState` 가 있음을 알 수 있다.

- 플러터 앱의 상태관리(Application State Management)¹를 담당
- 앱 내에서 바뀐 상태를 즉각적으로 반영해주는 방법: `StatefulWidget + setState()` 메소드
- 상태변화 반영 방법: `StatefulWidget` 하위에서 `setState` 메소드에 의해 변화된 부분은 즉각적인 `reBuild` 되어 앱의 상태변화를 반영

3.2 화면 설계 및 UI/UX 구성 요소

④ 화면 설계: Google Material 디자인

본 프로젝트에서는 안드로이드 앱 UI/UX 에 익숙한 사용자를 위해 구글 안드로이드 앱에서 자주 사용하는 텍스트, 박스, 버튼 위젯이 포함된 Material App 디자인을 사용해 화면을 설계했다.

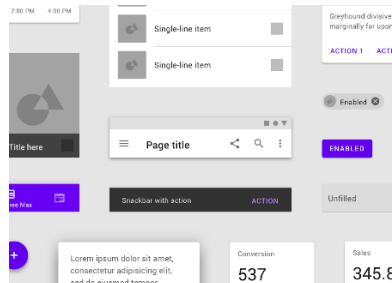
`MaterialApp` 은 구글이 지향하는 Material Design 을 지향 할 수 있게 해주는 클래스로, `main.dart` 에 위치하며 Navigator 라우팅, 공통 Theme 와 같은 코드를 분리 할 수 있도록 해준다. 이런 `MaterialApp` 에 짝꿍처럼 따라다니는 클래스가 있는데 바로 `Scaffold` 이다. `Scaffold` 는 사전에 찾아보면 처형대, 조립식 발판의 의미를 가지고 있다. html 으로 따지면 `<body></body>` 태그이다.

¹ 상태관리: 인터랙티브하게 사용자의 행동에 따라 값, UI, 화면의 변화를 실시간으로 관리

② UI/UX 구성요소: Scaffold Components

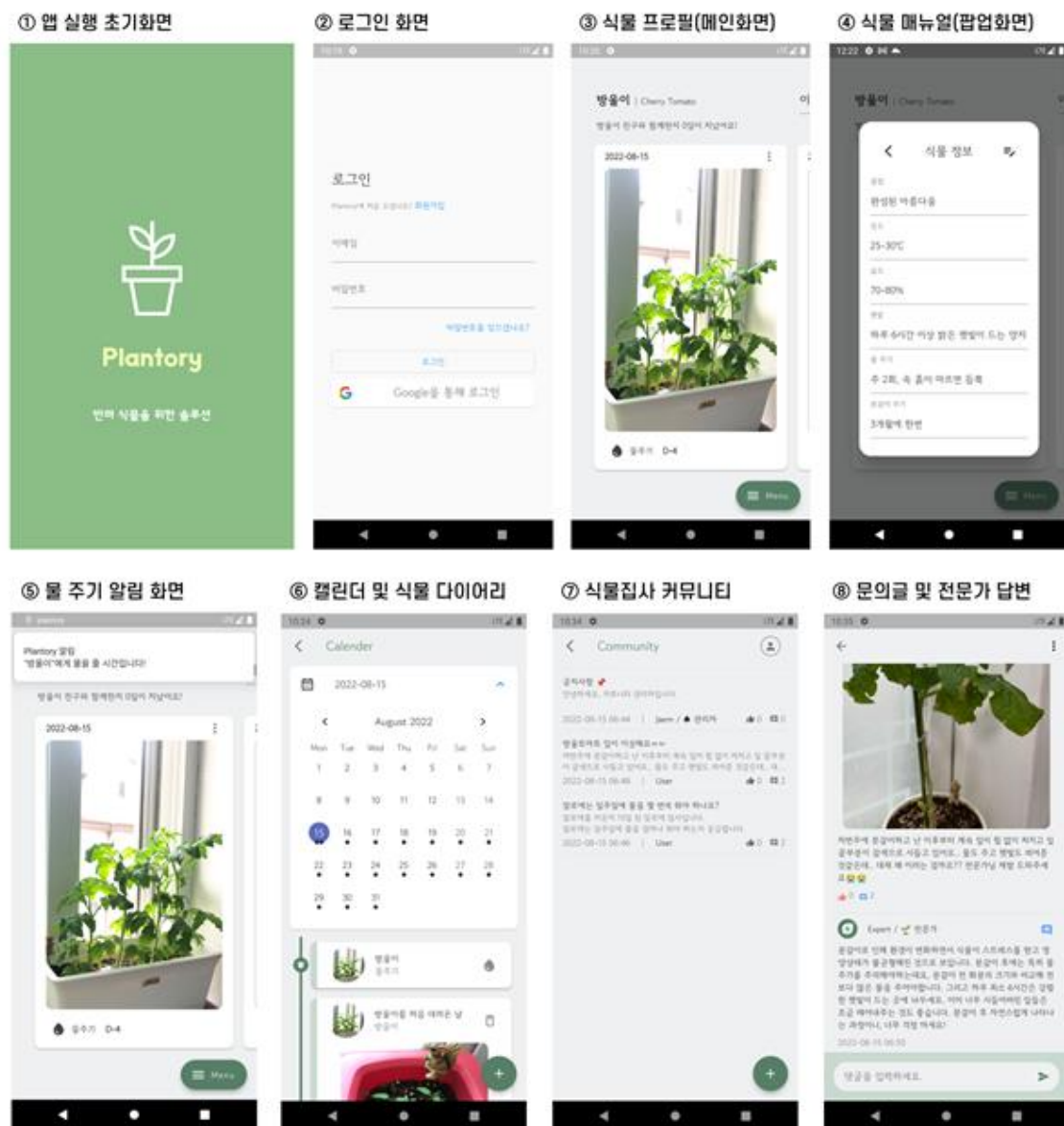
MaterialApp() 아래에 존재해, 텍스트/박스/버튼 위젯 형태로 많은 기본 기능들을 제공한다.

ex) Material Design 의 AppBar, BottomNavigationBar, FloatingActionButton

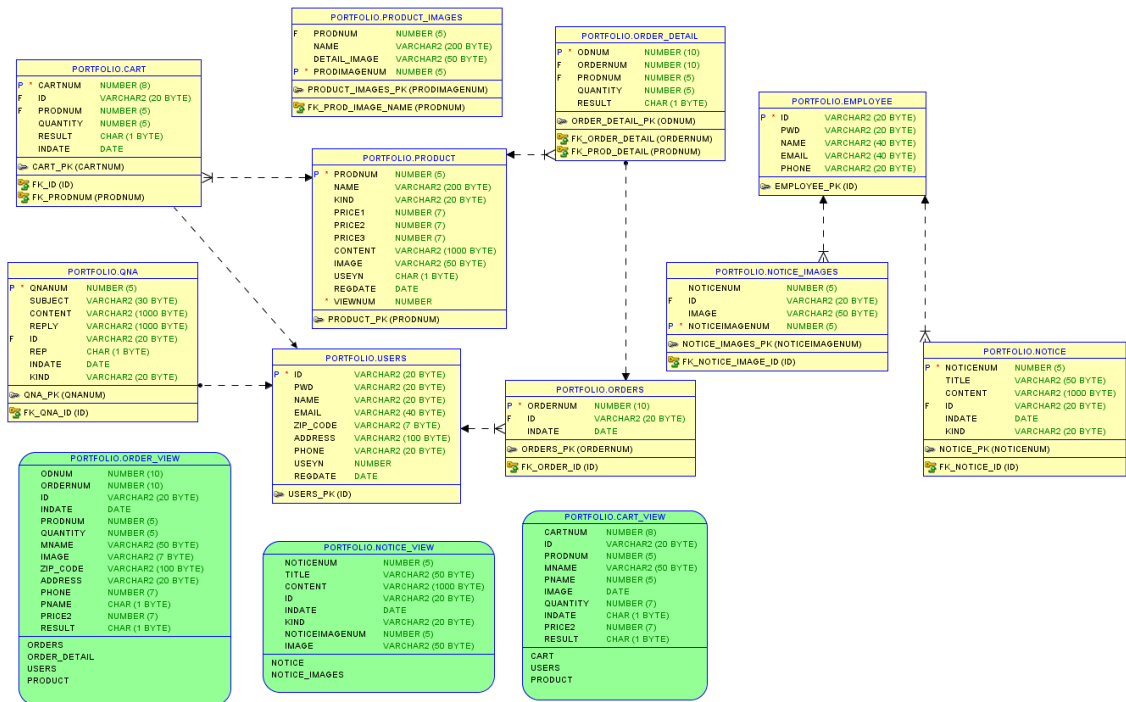


[그림 5] Google에서 제공하는 Material 위젯 예시

③ 화면 설계 결과



3.3 데이터베이스 설계



[그림 6] Oracle Database를 이용한 물리적 ERD 작성 결과

① 회원 정보 테이블

컬럼명	타입	설명	비고
ID	varchar2(20)	회원 아이디	primary key
PWD	varchar2(20)	회원 비밀번호	
NAME	varchar2(20)	회원 이름	
EMAIL	varchar2(40)	회원 이메일	
ADDRESS	varchar2(100)	회원 주소	
PHONE	varchar2(20)	회원 전화번호	
USEYN	number	회원 탈퇴여부	default 1 (가입: 1, 탈퇴: 2)
REGDATE	date	회원 가입일	default sysdate

② 식물 정보 테이블

컬럼명	타입	설명	비고
PID	varchar2(20)	식물 이름	primary key
PTYPE	varchar2(20)	식물 종류	
PLANG	varchar2(40)	꽃말	
PTEMP	varchar2(20)	온도	

PHUMI	varchar2(20)	습도
PSUN	varchar2(20)	햇빛양
PWATER	varchar2(20)	물 주기
PCYCLE	varchar2(20)	분갈이 주기
PCHAR	varchar2(50)	특징
PBUG	varchar2(20)	병해충

4. 구현

4.1 사용된 변수 및 자료형

Oracle Database를 이용한 ERD 작성 시 사용했던 회원 정보 테이블, 식물 정보 테이블, 계 시판 정보 테이블의 컬럼명을 변수의 이름으로, 타입을 자료형으로 사용해 데이터베이스 설계와 실제 Flutter 구현에 동일한 이름을 사용해 유지보수에 용이하도록 통일성을 높였습니다.

4.2 주요 코드 구현 사항 및 기능

① Stateful 위젯의 상세 생명주기

구분	주요 기능
createState()	상태 생성 · StatefulWidget이 생성될 때 반드시 호출되는 함수
mounted == true	setState() 호출 가능 체크 · State가 생성되면 mounted property가 true로 설정 · 위젯을 제어할 수 있는 buildContext 클래스에 접근 가능해지며 setState()를 호출할 수 있게 된다. · setState() 호출전에 mounted property를 체크하면 더 안전한 코드를 작성할 수 있다.
initState()	위젯 초기화 · 위젯 초기화시 1번 호출한다. · 위젯에 필요한 데이터를 준비하고 작성하기 좋은 구간
didChangeDependencies()	의존성 변경 시 호출 · initState() 이후에 호출된다. · 데이터에 의존하는 위젯이라면 화면에 표시하기 전에 반드시 호출 · 상속받은 위젯을 사용할때 Super(Parent)가 변경되면 호출
build()	위젯 빌드 반환 · 위젯을 만들어주는 메소드. 위젯이 렌더링 되어 반환

① Stateful 위젯의 상세 생명주기(Cont.)

구분	주요 기능
didUpdateWidget()	<p>위젯 갱신 시 호출되는 메소드</p> <ul style="list-style-type: none"> Parent 위젯이나 데이터가 변경되어 위젯을 갱신해야 할 때 호출 위젯이 생성되고 나서 갱신을 해야 할 때는 initState()가 아니라, didUpdateWidget()을 호출해야 함
setState()	<p>실행 중에 변경사항을 알리는 메소드</p> <ul style="list-style-type: none"> 데이터가 변경되었다는 것을 알리고 다시 Build() 메소드를 실행시켜, UI를 갱신함 상태 갱신이 필요한 부분에 사용
deactivate()	<p>상태 관리를 일시적으로 멈추는 메소드</p> <ul style="list-style-type: none"> State가 소멸전에 플러터의 구성 트리로부터 제거되는 메소드 따라서, 관리되지 않으나 메모리 해제까지 한 것은 아니라 사용 가능
dispose()	<p>상태 관리 종료하는 메소드</p> <ul style="list-style-type: none"> State 위젯을 완전히 소멸 위젯을 없앨 때 해줘야 하는 작업이 있다면 여기에 작성 deactivate() 된 위젯을 다른 트리에서 재사용하는 경우 dispose() 호출되지 않을 수 있음
mounted == false	<p>setState() 호출불가</p> <ul style="list-style-type: none"> 더이상 stateful 위젯이 존재하지 않으므로, setState() 메소드 사용이 불가함을 설정함.

4.3 문제 해결 과정 및 알고리즘

① 이슈 1. 텍스트 크기를 최소 혹은 최대로 설정하면 앱 화면이 제대로 표시되지 않음

iOS와 Android 앱은 기기 설정에서 디스플레이 및 텍스트 크기를 변경할 수 있다. 기기 설정에서 이 옵션을 최소 및 최대로 설정할 경우 화면이 제대로 표시되지 않는 이슈가 발생했다. 이를 수정하기 위해 MediaQuery 위젯을 이용해서 textScaleFactor를 0.7~1.4로 제한하고, 필요에 따라 고정된 텍스트 크기로 표시되는 위젯을 만들어 해당 이슈를 해결했다.

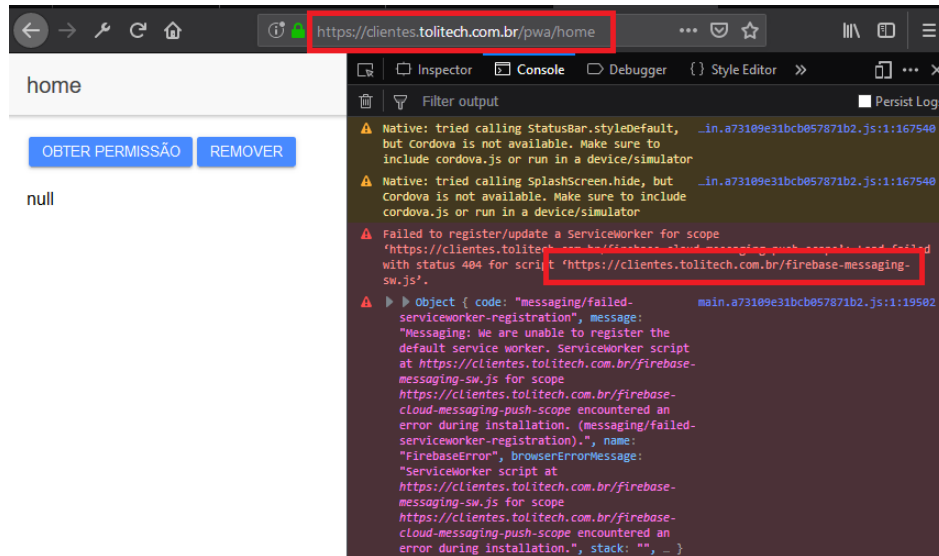


[그림 7] 이슈 1. MediaQuery 위젯의 textScaleFactor 범위 제한

② 이슈 2. 백그라운드 상태에서 FCM(Firebase Cloud Messaging) 수신 시 isolate

firebase_messaging 패키지를 이용하면 앱이 백그라운드에 있는 상태에서 FCM 푸시 수신 처리가 가능하다. 그런데 이때 SharedPreferences 에 접근할 경우, 아래 메시지와 함께 오류가 발생한다.

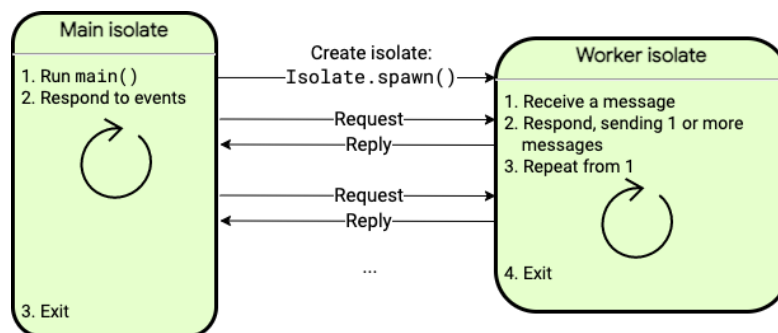
'When received, an isolation is spawned. Since the handler runs in its own isolate outside your applications context, it is not possible to update application state or execute any UI impacting logic.'



[그림 8] 이슈 2. firebase_messaging 패키지의 SharedPreferences 접근

Flutter 구현에 필요한 Dart 앱은 단일 스레드 프레임워크이며 스레드 대신 isolate 에서 작동한다. Isolate 를 새로 만들고 서로 통신할 수는 있지만 각 isolate 는 고유한 메모리 힙을 할당받기 때문에 공유하는 메모리는 없다. 백그라운드에서의 푸시 콜백은 새로운 isolate 에서 전달하는 것이기 때문에 메인 isolate 의 메모리 데이터를 사용할 수 없어서 위와 같은 오류가 발생한다.

이 문제는 앱 디렉터리 하위에 JSON 형식의 파일로 데이터를 저장하고 관리할 수 있는 유틸리티를 만들어서 FCM 수신 시 로컬에 데이터 저장이 필요한 경우에는 이 유틸리티를 사용하는 것으로 해결했다.



[그림 9] Dart 의 Isolate 작동 방식

5. 테스트

5.1 사용된 테스트 전략 및 방법

① 통합 테스트(Integration Test)

단위 테스트보다 더 큰 동작을 달성하기 위해 여러 모듈들을 모아 이들이 의도대로 협력하는지 확인하는 테스트를 '식물집사' 로그인 접근 위한 회원 생성 요청 과정에서 진행했다.

```

public static ExtractableResponse<Response> 회원_생성_요청(MemberRequest memberRequest) {
    return RestAssured
        .given()
        .contentType(MediaType.APPLICATION_JSON_VALUE)
        .body(memberRequest)
        .when().post("/api/members")
        .then()
        .extract();
}
  
```

[그림 10] RestAssured로 만든 인수 테스트 결과

5.2 단위 테스트 및 통합 테스트 결과 요약

Table 13 — Methods for software integration testing

Methods		ASIL			
		A	B	C	D
1a	Requirements-based test ^a	++	++	++	++
1b	Interface test	++	++	++	++
1c	Fault injection test ^b	+	+	++	++
1d	Resource usage test ^{cd}	+	+	+	++
1e	Back-to-back comparison test between model and code, if applicable ^a	+	+	++	++

Table 15 — Structural coverage metrics at the software architectural level

Methods		ASIL			
		A	B	C	D
1a	Function coverage ^a	+	+	++	++
1b	Call coverage ^b	+	+	++	++

[그림 11] 통합 테스트 결과 요약

5.3 버그 및 이슈 관리

테스트 단계	자동화 도구	도구설명
테스트 계획	요구사항관리	· 고객 요구사항 정의 및 요구사항관리를 지원
테스트 분석/설계	테스트케이스 생성	· 테스트 기법에 따른 테스트 케이스 작성과 테스트 데이터 생성을 지원
테스트 수행	테스트 자동화	· 기능 테스트와 UI 테스트 등 단위 테스트 및 통합테스트를 지원
	정적분석	· 코딩표준, 런타임 오류 등을 검증
	동적분석	· 대상시스템 시뮬레이션을 통한 오류 검증
	성능 테스트	· 부하생성기 등을 이용하여 가상 사용자 생성하고, 시스템의 처리능력을 측정하는 도구
테스트 관리	모니터링	· 시스템 자원(CPU, Memory 등)의 상태 확인 및 분석 지원
	커버리지 측정	· 테스트 완료 후 테스트 중 분성 여부 검증 지원
	형상관리	· 테스트 수행에 필요한 도구, 데이터 및 문서 관리
	결함 추적/관리	· 테스트에서 발생한 결함 추적 및 관리 활동 지원

6. 프로젝트 결과

6.1 프로젝트 후기 및 결과 평가

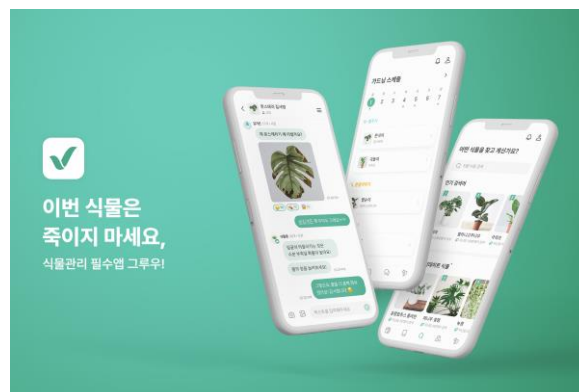
① 프로젝트 후기

본 오픈소스SW 프로젝트에서 지속적인 코드 수정과 앱 스토어 배포가 모두 용이한 '앱'을 개발해보고 싶다는 생각에, 프로젝트 설계 단계에서 Flutter 오픈소스 프레임워크를 선택했습니다. Flutter는 iOS, Android, 웹, Windows, MacOS, Linux까지 총 여섯 가지 플랫폼에 대한 다양한 어플리케이션 개발 환경을 지원합니다.

저도 이러한 Flutter의 가장 큰 장점인 크로스 플랫폼 앱 개발과, 6개의 플랫폼에 걸쳐 일관된 애플리케이션 UI를 만들 수 있다는 점 때문에 React-native, Swift가 아닌 Flutter로 프로젝트를 진행했습니다. 본 프로젝트에서는 한정된 기간에 빠른 결과물을 보여줘야 하기 때문에 Flutter의 위젯을 사용하여 UI 레이아웃을 설계하고 구현했습니다. Flutter 위젯은 개발자가 손쉽게 사용자 지정할 수 있도록 설계되었습니다. 즉, 대부분의 위젯은 작은 위젯으로 구성되며, 개발자가 위젯을 결합하거나 편집하여 새 위젯을 만들 수 있습니다. 그 중에서도 저는 구글 안드로이드의 UI에 익숙하기 때문에 iOS 스타일의 위젯인 Cupertino보다는 Material Components 위젯을 사용해 해당 Flutter 프로젝트의 UI/UX를 구현했습니다.

본 프로젝트를 진행하면서 개발이 가장 편리했던 점은 위젯을 구성할 때 사용되는 AppBar 레이아웃이나 요소를 배열 형태로 정렬하는 Child, Children 문법 등 Flutter에서 사용되는 Dart의 문법이 Google과 Reddit, Discord, Slack, Stack Overflow, Gitter의 오픈소스 커뮤니티에서 쉽게 확인하고 질문하는 과정을 통해 오류가 나더라도 빠르게 해결할 수 있다는 점이었습니다. 이런 Flutter 커뮤니티에서는 개발자가 빠른 개발을 할 수 있도록 수 천 가지의 서드 파티 패키지를 지원하며, 이러한 라이브러리는 pub.dev에서 직접 사용해볼 수 있습니다. 예시로 Dart의 Sound Null Safety를 통해 null error를 쉽게 감지할 수 있게 하여, 개발자가 디버깅하는 시간을 줄이고 앱 개발 구축에 집중할 수 있게 도와줄 수 있었습니다.

끝으로, 본 프로젝트 종료 이후에도 구글 플레이 스토어, 애플 앱 스토어 등에 '식물집사' 앱을 배포하고 사용자의 피드백을 받는 과정을 통해 사용자가 사용하기 최적화된 앱으로 수정하며 지속적으로 유지보수 하며 스케일업 해보고 싶습니다.



② 결과 평가

번호	프로젝트 목표 달성 여부 판단 기준	점수
설계		
1	OSS 표준단체 OSI(Open Source Initiative)에 등록된 오픈소스 프레임워크를 사용해서 프로젝트를 진행했는가?	5 점
2	구현한 앱을 GitHub 에 올리는 과정에서 오픈소스 라이선스를 사용해 업로드했는가?	5 점
3	프로젝트에 사용된 변수와 자료형을 보여주는 물리적 ERD 는 데이터베이스 테이블 간의 데이터 무결성, 기밀성, 가용성을 모두 만족하며, 기본키와 외래키 관계가 명확하게 정의되었는가?	4 점
구현		
4	프로젝트 제안서에서 기획했던 UI/UX 메인 화면과 프로젝트의 주요 기능을 모두 구현했는가?	5 점
5	앱 구현 시 단순 화면 구현만 구현한 네이티브 앱이 아닌 로그인/회원가입/게시판 등 DB 연동 및 서버와의 통신하는 앱을 구현했는가? => Firebase 로 로그인/회원가입 시 고객 정보를 고객 DB 에 저장	5 점
6	사용자가 편리하게 사용할 수 있는 UI/UX 디자인을 반영해 앱을 구현했는가? => 안드로이드 유저에 최적화된 구글 제공 material App Design 을 반영해 친숙한 구글 위젯을 사용해 앱을 개발	4 점
테스트		
7	앱을 apk 파일로 빌드하는 과정에서 에러 없이 정상적인 빌드가 이루어졌는가?	5 점
8	구현한 앱은 안드로이드 환경에서 정상적으로 작동하는가?	5 점
보고서		
9	프로젝트 결과 보고서에는 사용자와 개발자 가이드를 포함했는가?	5 점
10	OSS 프로젝트 보고서의 표준인 설계, 구현, 테스트, 결과 양식에 맞춘 보고서를 작성했는가?	5 점

plantory

반려식물 맞춤 솔루션 서비스

1. 개발배경

1) 시장 측면

"증가하는 홀가드닝 수요 대비 관리체계 부족"

"기존 어플의 전문솔루션 기능 부재"

"초보 식물집사에게 낮은 전문지식 접근성"

"식물 집사의 약 60%가 식물 관리의 어려움 경험"

2) 소비자 측면

Q: 식물 재배는 관리가 어려운가?

응답	비율
그렇다	59.3%
아니다	35.5%
잘 모름	5.2%

2. 핵심기능

식물 전문가와 사용자를 연결하여 전문적인 솔루션을 제공
&
홀가드닝에 대한 일시적 수요를 지속적인 수요로 이끄는 서비스가 필요

Q1

식물 관리 매뉴얼

Q2

성장 기록 다이어리

물 주기 알림

사용자 커뮤니티

Q3

전문가 상담 솔루션

3. 기대효과 & 확장가능성

식물 관리의 어려움 해결

꽃집 홍보

전문 솔루션 제공

잠재적 소비자

코로나19로 인해서 원예시장이 많이 침체되었는데, 플랜토리를 통해 초보 식물집사들과 소통하고, 꽃집 홍보도 하고 1석 2조 효과를 볼 수 있을 것 같아서 기대가 됩니다!



김성빈
전문가

☎ 010-5689-4985
🏠 사문화원(경기 고양시 일산동구 백마로 547)

전문분야: 생화·관엽, 등·서양란, 조경
경력사항: 식물보호기사, 원예기능사, 조경기능사, 화훼장식 기능사 취득
네이버 엑스퍼트 전문가 등록: 온라인 공개강좌, 상시상담 진행중

7. 참고 문헌

<단행본>

- [1] 플러터 인 액션(안드로이드와 iOS 애플리케이션을 한 번에 개발하는 완벽 가이드)
- [2] Do it! 플러터 앱 프로그래밍(개정판) : 오픈 API 활용 + 파이어베이스 + 구글 맵 + 광고 수익까지

<웹사이트>


- [1] LINE Engineering, "Flutter 패키지로 공통 모듈 리팩토링하기", 2023.01.06.
- [2] Martin Heller, "안드로이드부터 윈도우, iOS까지" 구글이 만든 멀티플랫폼 개발 툴 '플러터', 2022.07.06.
- [3] FlutterFire Requesting permission (Apple & Web)
<https://firebase.flutter.dev/docs/messaging/usage/#background-messages>
- [4] Google Material Design
<https://m2.material.io/design/introduction>

<기사>

- [1] 안재형, "홈가드닝·플랜테리어... 대세가 된 반려식물", 매일경제, 2022.04.26.
- [2] Everything new for developers at Google I/O: Android Studio, Flutter, AI, and more
<https://9to5google.com/2023/05/10/google-io-2023-developer-keynote-recap/>

8. 소스 코드 GitHub 링크

- [1] <https://github.com/sump99/PlantButler3.0>

	sump99 Update README.md	4a4012c 2 days ago	🔄 2 commits
📁	android	Flutter OSS project final upload	2 days ago
📁	assets	Flutter OSS project final upload	2 days ago
📁	ios	Flutter OSS project final upload	2 days ago
📁	lib	Flutter OSS project final upload	2 days ago
📁	test	Flutter OSS project final upload	2 days ago
📄	.metadata	Flutter OSS project final upload	2 days ago
📄	Plantory_1.0.2.apk	Flutter OSS project final upload	2 days ago
📄	Plantory_explain.jpeg	Flutter OSS project final upload	2 days ago
📄	README.md	Update README.md	2 days ago
📄	analysis_options.yaml	Flutter OSS project final upload	2 days ago
📄	pubspec.lock	Flutter OSS project final upload	2 days ago
📄	pubspec.yaml	Flutter OSS project final upload	2 days ago
📄	이정현-32193469-최종 보고서.pdf	Flutter OSS project final upload	2 days ago