졸 업 작 품

최 종 보 고 서

지도 API 자체 제작을 통한 사용자 경험 개선 캠퍼스맵 어플리케이션

건국대학교 KU융합과학기술원 스마트ICT융합공학과

201812319

배성재

2023년 1학기

목차

제 1장 개요

1.1. 문제 인식

매년 또는 매학기 건국대학교 학생 커뮤니티 사이트에는 여러 질문들이 올라오곤 한다. 케이큐브 사용 방법과 위치 또는 건물에서 건물 간의 연강 가능성, '종강'이라는 건물에 대한 궁금증 등 학교 생활에 대한 질문들이 정기적으로 올라온다. 이런 지속적인 질문들에 대한 답변을 빠르게 학생들에게 주고자 건국대학교 캠퍼스 맵을 기획하였다.

현재 학생들은 크게 2가지의 문제점을 겪고 있다고 생각했다. 첫째로, 건국대학교 사이트에 있는 캠퍼스 안내도는 실질적 활용이 어렵다였다. 건국대학교 공식 캠퍼스 안내도에는 복사실, 편의점 등이 나타나있지 않았고, 실제 건물 배치와 다르기도 하였다. 과거의 이미지를 사용하여 지금과는 형태가 다른 건물 이미지도 존재하였다. 둘째로, 네이버 지도, 카카오 지도 등 상용화되어있는 어플리케이션 지도에는 건국대학교에 자세한 경로들이 나타나있지 않아 실제와는 달리 비효율적인 캠퍼스 경로를 계산하고 있다. 언덕길과 같은 작은 도보 등이 미반영되어있었고, 특히 교내 캠퍼스 내부 데이터는 반영되어있지 않고

있었다.

이런 문제점을 개선하기 위해 코로나로 인한 장기적인 거리두기(비대면)으로 캠퍼스에 익숙치 않은 학우들을 위한 '건국대 캠퍼스맵' 서비스 개발을 통해 교내외 구성원이 더 효율적으로 캠퍼스를 이용할 수 있도록 하는 것을 목표로 한다. 주요 기능으로 실제 보행시간 기준 장소 탐색, 장애학우를 위한 최단경로 등을 만들 계획이다.

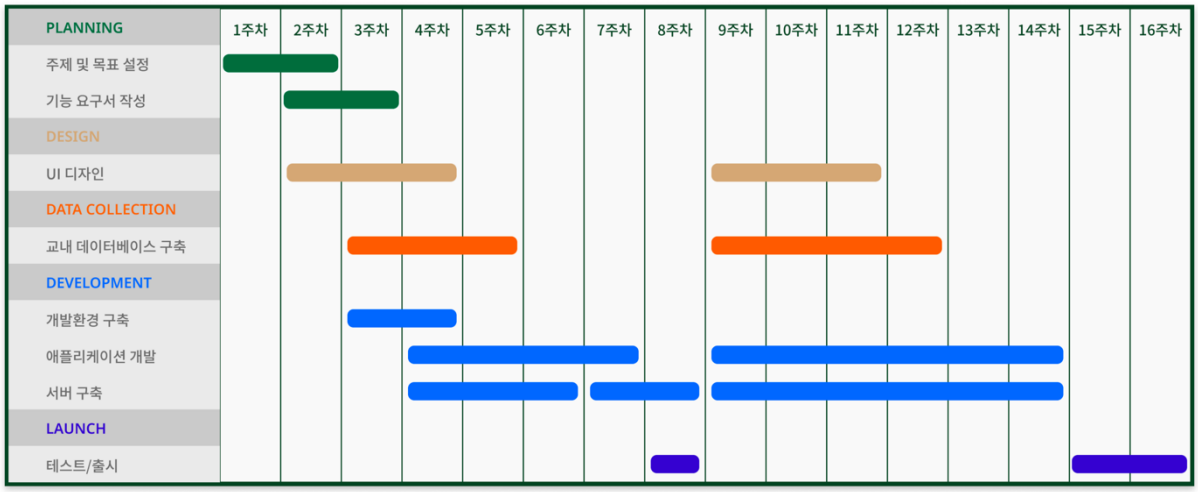
1.1.1 목적

이제 막 입학한 신입생, 인프라를 효율적으로 활용하고자 하는 재학생, 이동간 어려움이 있던 장애학우, 캠퍼스를 이용하는 교직원 등 교내외 구성원을 타겟으로 하며, 현재 상용 지도 서비스와 기존의 이미지 캠퍼스 맵에서 제공하지 못한 인프라 정보를 실시간으로 제공하는데 목표를 둔다.

1.1.2 기대효과

1) 애플리케이션 UI/UX 디자인 능력 함양
2) 지리 정보 시각화 관련 스터디 진행
3) 크로스플랫폼 개발 능력(Flutter) 증진
4) 소프트웨어 공학 능력 증진
5) 온/오프라인 지도 배부 및 학생이 학교에 기여하는 문화 확산
6) 지도 서비스 배포 및 100명의 초기 사용자 수 확보

1.2 팀 구조



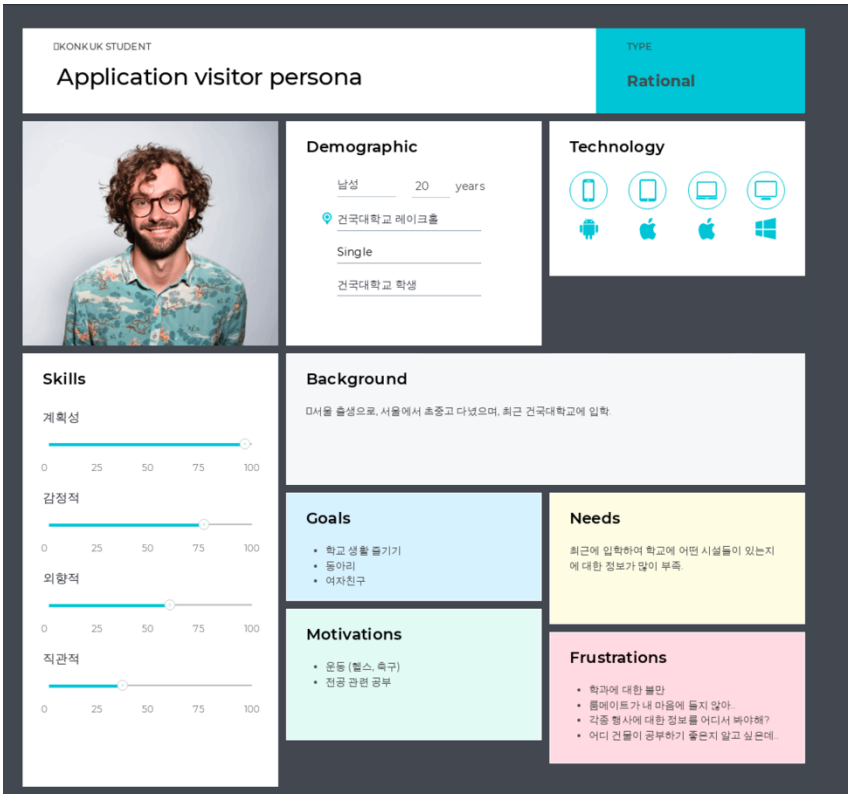
프로젝트 일정 관리를 위해 간트 차트를 작성하였다. 3주차 ~ 14주차 동안 계속해서 개발을 진행하고 마지막 15주차, 16주차에서는 단위 테스트, QA 등을 실시하여 안정성과 효율성을 갖춘 어플리케이션을 개발하도록 할 것이다. 나는 디자인과 프론트엔드 어플리케이션 개발을 맡았다.

제 2장 디자인

2.1 서비스 구조도

건국대학교 학생 커뮤니티와 주변 학우들을 통해 얻은 문제점은 크게 총 4가지였다. 1) 찾기 어려운 건물과 편의시설, 2) 상용서비스에 반영되지 않은 길, 3) 누군가에겐 소외된 길찾기, 4) 교내 행사에 대한 위치 및 정보 부족. 우리는 이런 문제점들을 해결하기 위해 UX 디자인 방법론 중 하나인 사용자 페르소나를 통해 진행하기로 하였다.

2.1.1 User Persona

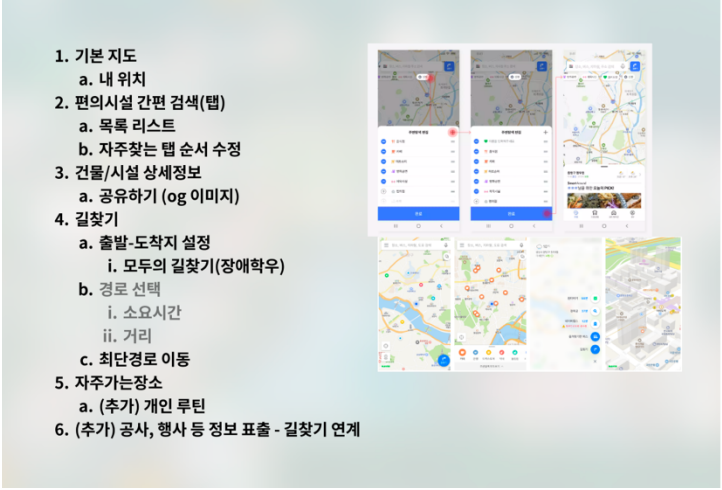


사용자 페르소나는 대상 고객을 나타내는 가상의 인물을 의미하며, 실제 사람이 아니지만 실제 인터뷰, 설문 조사 및 기타 형태의 사용자 연구를 통해 얻은 데이터와 사실을 기반으로 만들었다. 이를 통해 사용자의 한계, 힘든 점, 성공 및 목표를 연관시켜 고객의 요구 사항에 최적화된 맞춤형 사용자 환경을 구축할 수 있다.

나이, 성별, 사는 곳 등을 가상으로 생성하였으며, 어떤 니즈나 불만이 있는지 고객

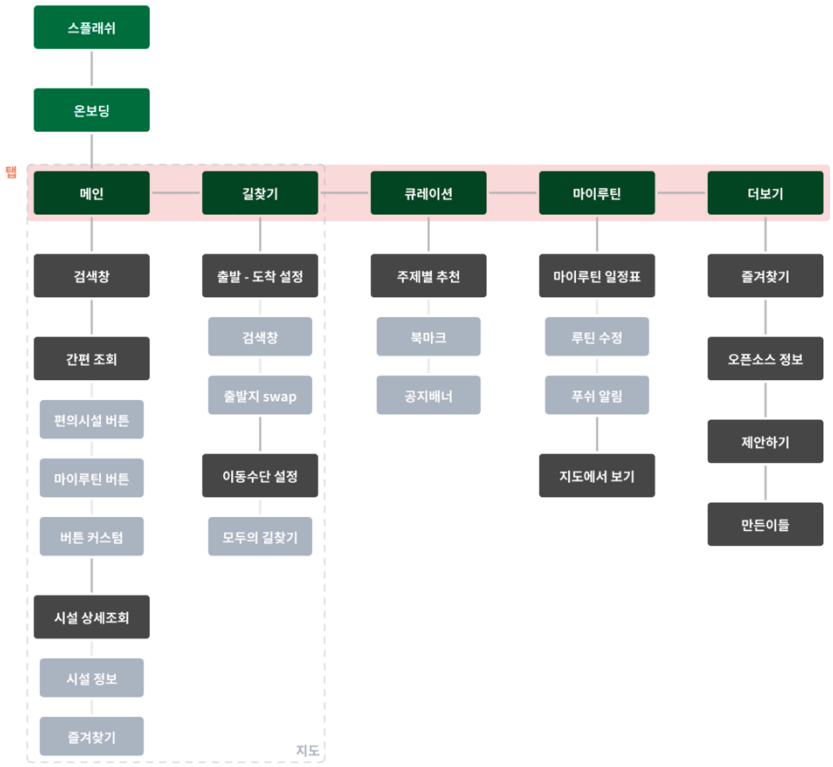
리서치에서의 답변들을 설정하였고, 위와 같은 페르소나를 만족하기 위한 기능을 정의하도록 하였다.

2.1.2 메뉴 구조도



큰 기능들로는 1) 기본 지도, 2) 편의시설 간편 검색, 3) 건물/시설 상세정보, 4) 길찾기, 5) 자주가는 장소로 5개의 기능이다. 2개의 추가 기능들도 작성하여 추후 개발하도록 진행하였다. 카카오 지도와 네이버 지도 어플리케이션(오른쪽에 있는 사진)을 참고하여 메뉴 구조도를 작성하였다.

2.1.3 화면 구조도

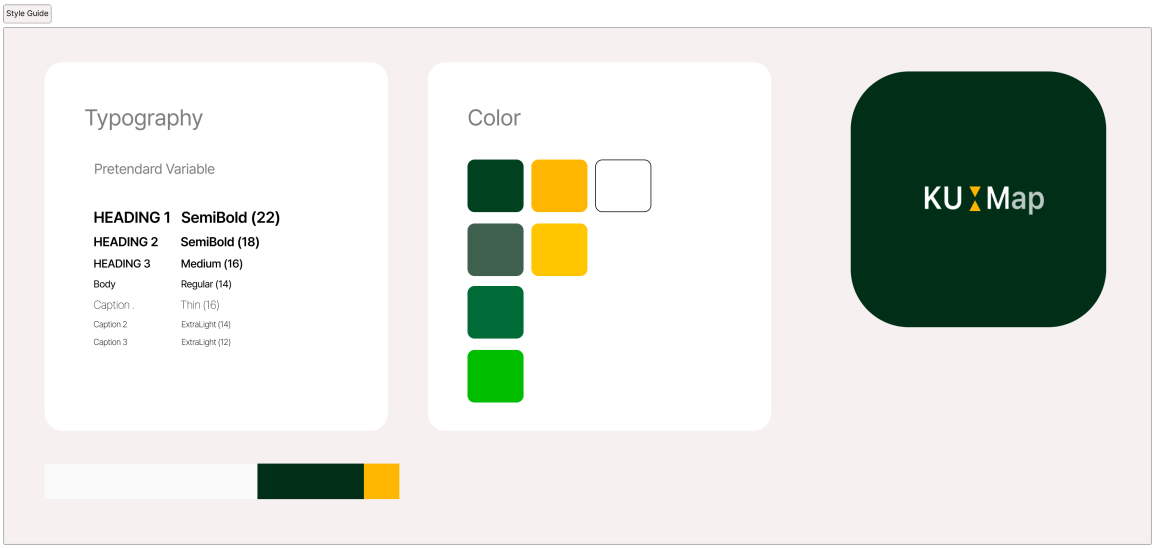


메뉴 구조도를 이용하여 화면 구조도를 완성하였다. 탭 화면, 일반 화면, 버튼, 들어갈 기능들에 대해서 정의하였다. 5개의 탭 화면으로 구성되어 메인 탭 화면에는 일반 지도가 보여지고 검색창과 간편 편의시설 조회, 시설 상세조회를 할 수 있다. 길찾기 탭 화면에는 출발지, 도착지 설정과 이동수단을 설정할 수 있다. 큐레이션 탭 화면에서는 주제별 추천하는 건물 정보와 북마트, 공지 배너를 볼 수 있다. 마이루틴 탭 화면에서는 마이루틴 일정표와 지도에서도 일정을 확인할 수 있다. 마지막으로 더보기 탭 화면에서는 오픈소스 정보와 만든이들 등 부가 정보들에 대해 확인할 수 있다.

같은 팀인 김현재 학우는 OSM(OpenStreetMap), Server 등 백엔드 파트를 담당하였고, 박정환 학우는 메인 탭 화면과 길찾기 탭 화면을 담당하였으며, 나는 UI 디자인과 큐레이션 탭 화면, 마이루틴 화면, 더보기 화면 개발을 담당하였다.

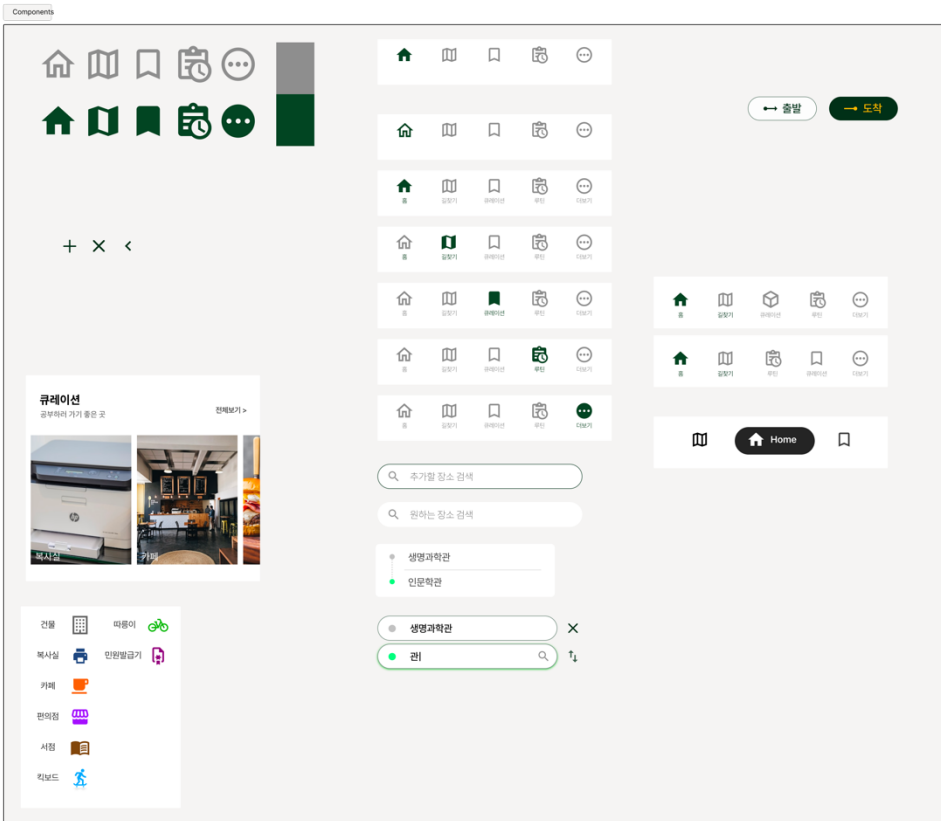
2.2 UI / UX

2.2.1 Design Guide



어플리케이션의 주요 색상은 3가지로, '#FEFEFE', '#0A2F19', '#FFBC00'이다. 이 3가지 색상은 건국대학교 로고에서 비롯되어으며, 6:3:1의 비율을 가지고 화면을 구성하였다. 이 3가지의 색상으로 화면을 구성하기에 부족할 경우에는 투명도를 75%로 정도 낮춰 추가 색상을 만들었다. UX 관점에서 완전 흰색('#FFFFFF')이 아닌 조금 덜 하얀 흰색('#FEFEFE')을 사용하여 사용자 눈의 피로감을 개선하였다.

앱 로고는 오른쪽 사진과 같이 제작하였다. 폰트는 Pretendard Variable을 사용하였고, 총 7가지의 폰트 디자인을 폰트 및 크기를 재조정하여 사용하였다. 특히 사용자들이 20대, 주로 대학생들이 타겟이기 때문에 작은 사이즈의 폰트를 사용하여도 무방하다고 생각하였다.



위와 같이 여러 컴포넌트를 구성하였다. 아이콘은 모두 Material design Icon을 사용하여 일관성있게 디자인하였다. 건물, 복사실 등의 편의시설 아이콘 같은 경우 눈에 띄는 색깔을 사용하여 사용자들에게 한 눈에 볼 수 있도록 집중도를 강화하였다.

탭 화면(Bottom navigation bar)은 선택되지 않았을 때는 '#8E8E8E' 의 회색 계열의 색깔을 사용하였고, 선택되었을 때는 '#0A2F19'로 포인트 색깔을 사용하여 현재 사용자가 어떤 탭 화면을 보고 있는지 잘 확인할 수 있도록 하였다.

검색 버튼은 총 4가지로 구성되어있다.

1) 맨 위의 검색 버튼는 사용자가 검색을 하기 위해 버튼을 클릭했을 때의 버튼이며, 사용자가 해당 부분을 잘 터치했다고 느낄 수 있게 경계선에 포인트 색상을 넣어주었다.

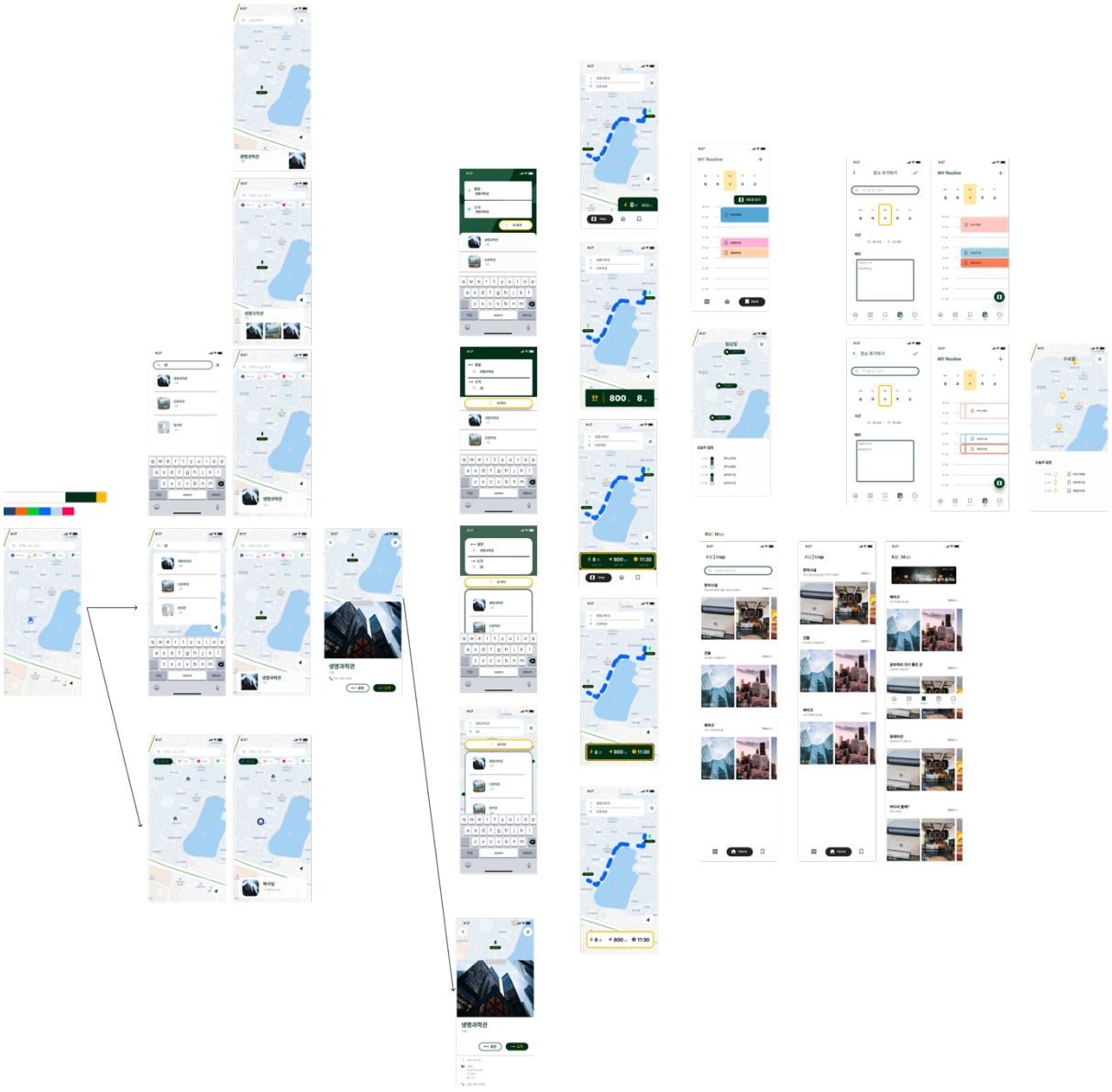
2) 두 번째 위의 검색 버튼은 일반 검색 버튼으로 사용자가 검색창을 활성화시키지 않았을 때의 상태이다.

3) 세 번째 위의 검색 버튼은 출발지와 도착지가 설정되어 있는 상태에서 사용자에게 직관적으로 어디에서 출발하고 어디에서 도착할 것인지를 보여주는 상태이다. 출발지는 회색,

도착지는 초록색을 사용하였다.

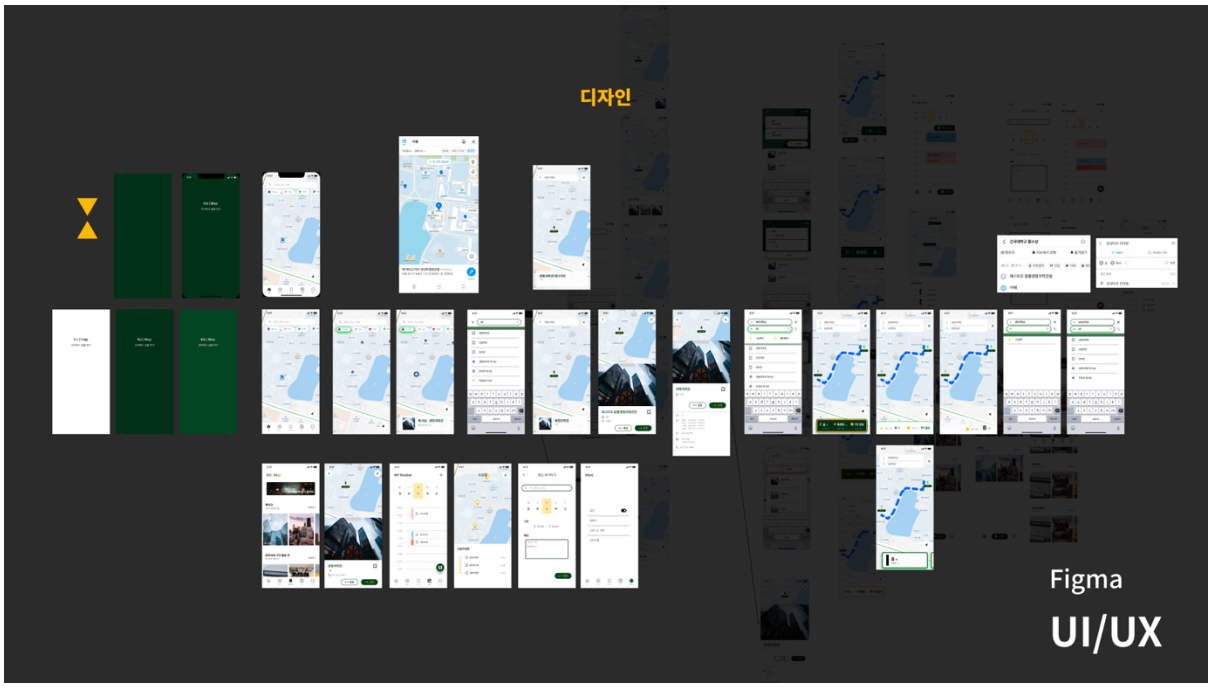
4) 맨 밑에 있는 검색 버튼은 출발지와 도착지를 설정할 수 있는 버튼으로 사용자가 출발지를 설정할 것인지, 목적지를 설정한 것인지에 따라 경계선의 색상이 회색이 될지, 초록색이 될지로 바뀐다.

2.2.2 전체 User Flow



우선 여러 화면을 디자인해보았다. 사용자가 어플리케이션을 시작하였을 때를 시작으로

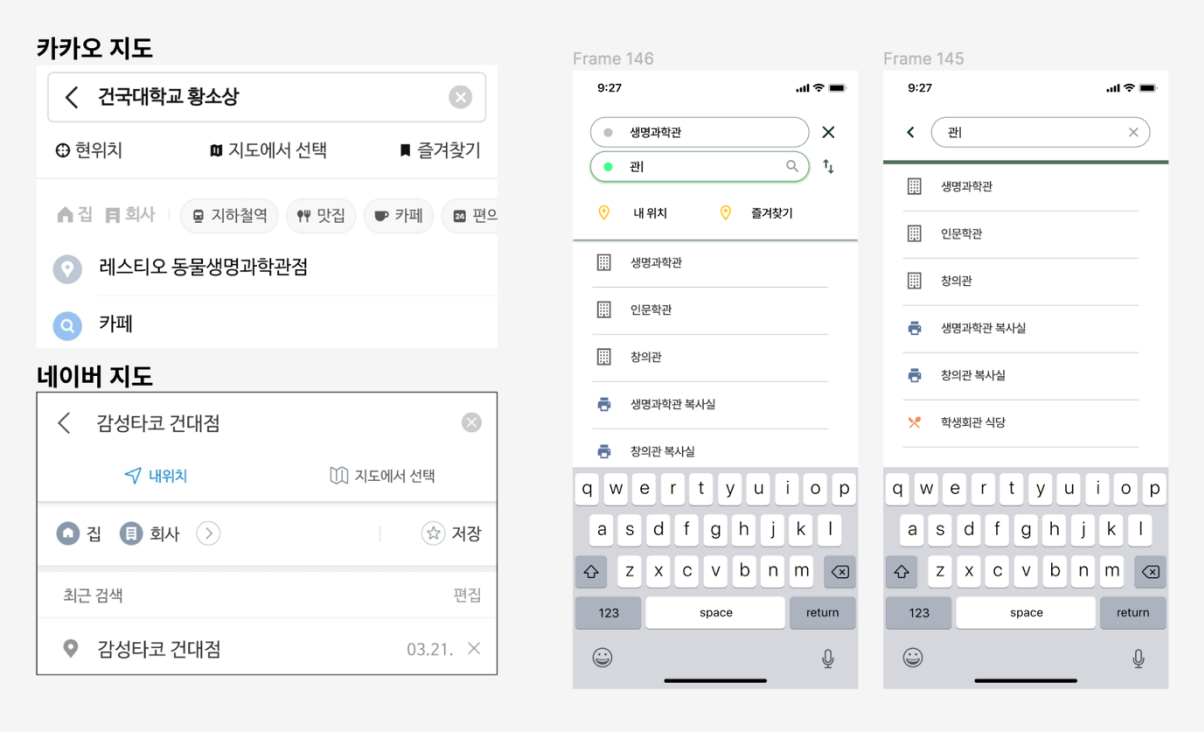
왼쪽에서 오른쪽 방향으로 유저 플로우를 생각하여 UI 디자인을 하였다. 각각의 디자인 형태마다 UX 관점이 달라 한 화면에 대해서 여러 디자인 형태를 구상해보았고, 팀원들과의 회의로 하나의 디자인을 골라 진행하였다.



위의 사진은 팀원들과 회의를 마친 후 최종 UI 디자인이다.

2.2.3 여러 화면에서 UX 디자인

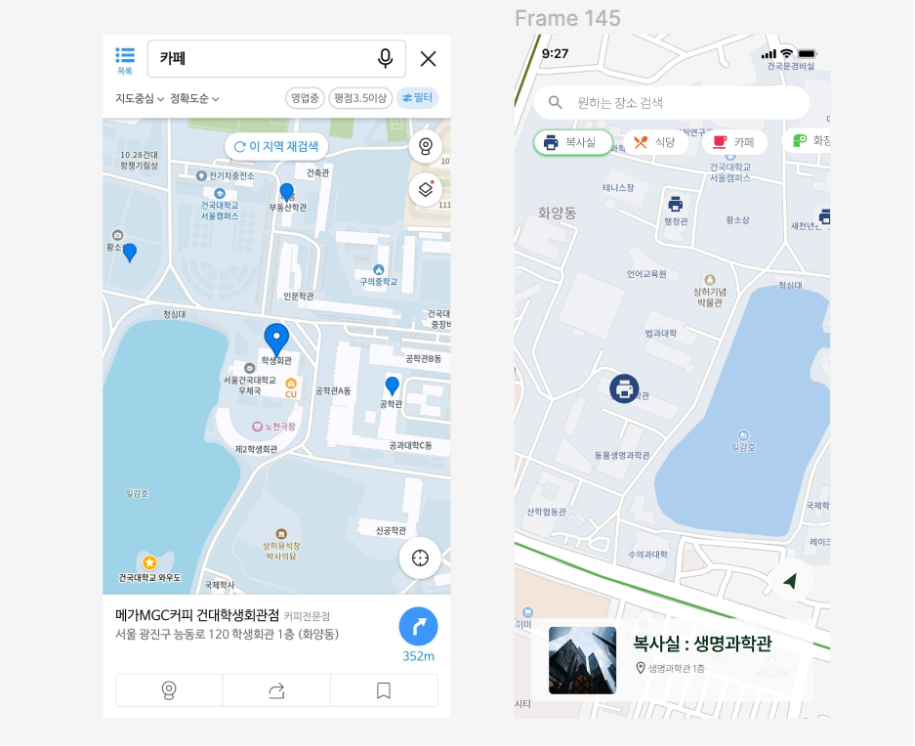
1) 검색창



Frame 145. 는 단일 검색 화면이다. 서버로부터의 데이터들을 불러와 건물 이름과 건물 아이콘을 사용자에게 보여준다. 네이버 지도와 카카오 지도 모두 뒤로 가는 버튼과 닫는 버튼(x 버튼) 2개를 동시에 사용하고 있어 이를 Frame 145. 화면에도 적용시켰다.

Frame 146. 는 길찾기 검색 화면이다. 회색점은 출발지, 초록색 점은 도착지이다. 출발지를 검색하거나 도착지를 검색할 경우 해당 검색창에 경계선의 색깔이 초록색으로 바뀌게 된다. 현재 Frame 146의 화면은 사용자가 도착지에 검색창을 눌렀을 때의 결과이다. 해당 화면에는 뒤로가기 버튼을 넣을 공간이 부족하여 삭제하였고, 닫는 버튼만 있는 상태이다. 닫는 버튼 밑에는 교차 버튼이 존재한다. 교차 버튼은 출발지와 도착지의 장소를 서로 바꿔주는 버튼이며, 이는 네이버 지도에 존재하고 많이 사용하기에 사용성을 높이고자 추가했다. 그 밑에 내 위치 버튼과 즐겨찾기 버튼이 있다. 내 위치 버튼은 나의 location(위도, 경도)값을 가져와 출발지 또는 도착지에 설정할 수 있게 해준다. 즐겨찾기 버튼은 내가 기존에 저장했던 나의 즐겨찾기 란에서 내가 선택하고자 하는 장소를 고를 수 있게 해주는 버튼이다. 이 또한 네이버 지도와 카카오 지도에서 실제로 쓰여지고 있는 버튼이다.

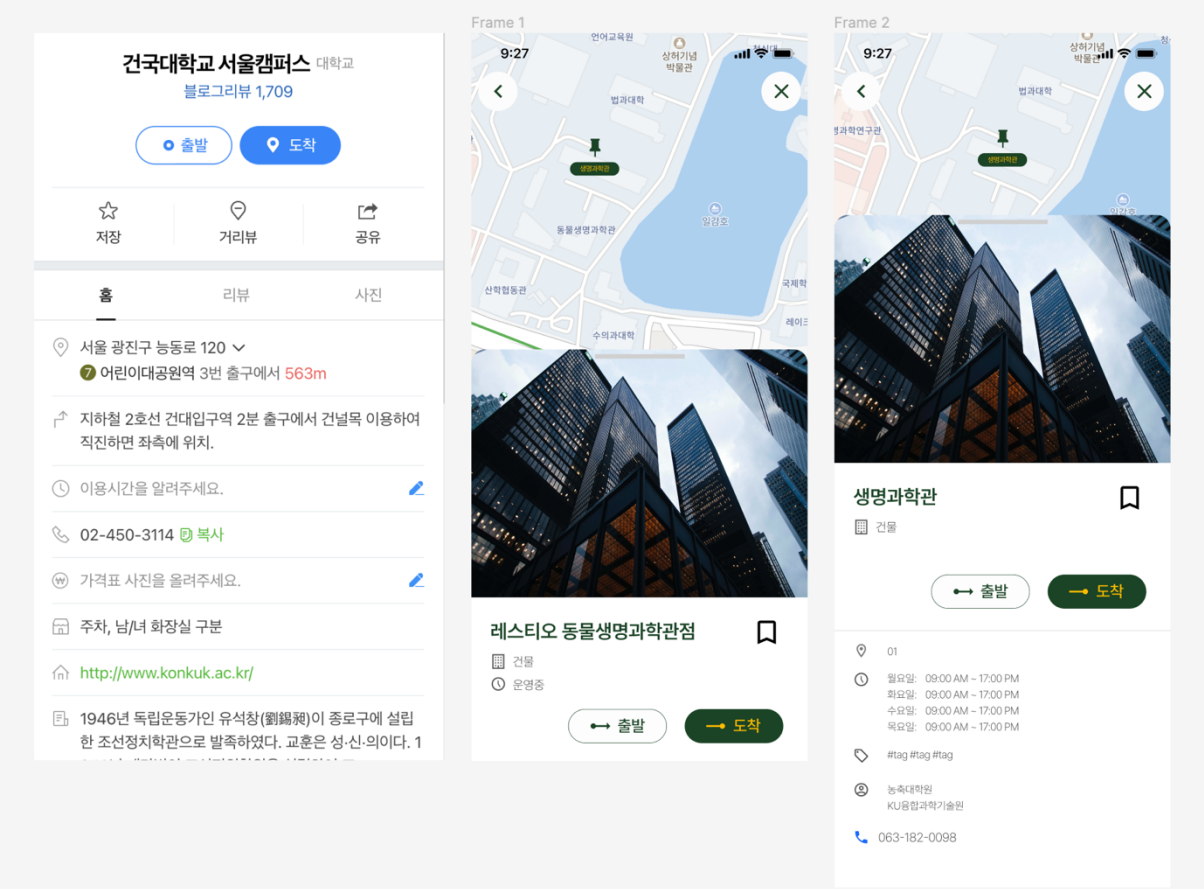
2) 간편 검색



간편 검색 같은 경우, 네이버 지도와 카카오 지도가 많은 차이를 가지고 있었고, 네이버 지도가 좀 더 20대 대학생들에게 보여지기 편한 UI를 가지고 있는 것 같아 이를 많이 참고하였다.

위의 사진 Frame 145. 복사실을 간편 검색하고 그 중 하나를 클릭하였을 때의 결과 화면이다. 복사실이 눌러졌다는 것을 사용자에게 알려주기 위해 경계선의 색깔이 초록색으로 바뀐 것을 확인할 수 있다. 해당 버튼을 누르면 복사실 마커가 여러 개 나타나게 되고, 그 중 하나의 마커를 눌러 해당 건물의 정보와 이름을 확인할 수 있다. 팀원들과 회의 끝에 해당 검색 결과 정보는 건물 사진, 건물 이름, 건물 정보를 확인할 수 있도록 하였다.

3) 상세 정보



다음은 검색 결과 상세 정보 화면이다. 네이버 지도를 많이 참고하여 상세 정보 UI를 디자인하였다.

Frame 1. 은 사용자가 검색 후 해당 건물 이름을 클릭하였을 때 나오는 화면이다. 건물 사진, 건물 이름, 건물 정보, 운영중 여부, 즐겨찾기 여부, 출발 버튼과 도착 버튼이 존재한다.

Frame 2. 는 Frame 1. 에서 사용자가 추가 정보를 얻고자하여 위로 슬라이드 했을 때 나오는 화면이다. Frame 1.보다 조금 더 다양하고 상세한 정보를 확인할 수 있다. 건물 번호, 운영시간, 태그 정보, 단과대 이름, 건물 전화번호 를 확인할 수 있다. 이는 모두 팀원들과 회의 끝에 어떤 정보를 삽입할지 결정하여 넣은 결과들이다. 출발, 도착 버튼을 눌렀을 때는 길찾기 검색 화면으로 이동하게 되며, 해당 건물 이름이 출발지 또는 도착지 검색창에 쓰이게 된다.

제 3장 개발

3.1 개발 환경 구축

IOS와 AOS에서 빌드를 하기 위해 크로스 플랫폼인 Flutter를 이용하여 개발을 하였다. Vscode 1.79.0에서 Flutter 3.7.7, Dart 2.19.4 환경에서 진행하였다. 컴퓨터 환경은 MacBook Pro 2019 – Ventura(Mac OS) 이다. 빌드 최소 타겟 환경은 IOS 13.0, Android 23 이다.

3.2 사용한 기술

1) Flutter

Flutter는 Google에서 개발한 오픈 소스 UI 소프트웨어 개발 킷이다. Flutter는 모바일 앱, 웹 앱 및 데스크톱 앱을 빌드하기 위한 크로스 플랫폼 프레임워크로, 하나의 코드 베이스에서 iOS, Android, 웹 및 기타 플랫폼용 애플리케이션을 개발할 수 있다.
Flutter는 Dart라는 프로그래밍 언어로 작성되며, 기본적으로 네이티브 코드가 아닌 UI를 렌더링하기 위해 Skia 그래픽 엔진을 사용한다. 이는 앱의 성능을 향상시켜 네이티브 앱과 유사한 사용자 인터페이스를 제공한다.

Flutter의 주요 특징은 다음과 같다:

1. 크로스 플랫폼 개발: 단일 코드 베이스로 iOS 및 Android 앱을 개발할 수 있다.
2. 빠른 개발: 핫 리로드 기능을 통해 앱의 변경 사항을 실시간으로 확인하고 반영할 수 있다.
3. UI: 네이티브 앱과 거의 동일한 룩 앤 필을 제공하는 아름다운 사용자 인터페이스를 구축할 수 있다.
4. 네이티브 성능: Flutter는 네이티브 코드와 통합하여 뛰어난 성능을 제공한다.
5. 다양한 위젯: 다양한 내장 위젯을 사용하여 앱을 구성하고, 필요한 경우 커스터마이징할 수 있다.
6. 확장 가능성: 플러그인 시스템을 통해 기능을 확장하고, 기존의 네이티브 코드와도 통합할

수 있다.

Flutter는 개발자들에게 빠르고 효율적인 앱 개발을 위한 강력한 도구를 제공하며, 많은 개발자들이 이를 사용하여 다양한 플랫폼용 앱을 개발하고 있다.

2) SQFlite

SQFlite는 Flutter 애플리케이션에 SQLite 데이터베이스를 간단하고 가벼운 라이브러리로 통합하는 Flutter 플러그인이다. SQLite는 인기 있는 관계형 데이터베이스 엔진으로, 구조화된 데이터를 저장, 검색 및 조작할 수 있게 해준다.

SQFlite 플러그인을 사용하면 Dart 프로그래밍 언어를 사용하여 Flutter 개발자가 SQLite 데이터베이스에서 생성, 읽기, 업데이트 및 삭제 (CRUD) 작업을 수행할 수 있다. Flutter 앱 내에서 데이터를 로컬로 영구적으로 저장하는 편리하고 효율적인 방법을 제공한다.

SQFlite의 주요 기능은 다음과 같다:

1. 가벼움: SQFlite는 가벼우며 효율적으로 설계되어 앱 성능과 리소스에 최소한의 영향을 미친다.
2. 비동기 작업: SQFlite는 비동기 데이터베이스 작업을 지원하여 개발자가 UI 스레드를 차단하지 않고 데이터베이스 트랜잭션을 수행할 수 있게 한다.
3. SQL 지원: SQL 쿼리 작성을 완전히 지원하여 복잡한 데이터베이스 작업을 수행하고 특정 조건에 따라 데이터를 검색할 수 있다.
4. 트랜잭션 및 일괄 작업: SQFlite는 단일 트랜잭션 또는 일괄 작업 내에서 여러 데이터베이스 작업을 실행할 수 있도록 해주어 Atomicity를 보장하고 성능을 향상시킨다.
5. 마이그레이션 지원: 플러그인은 데이터베이스 스키마 마이그레이션을 처리하는 메커니즘을 제공하여 앱이 시간에 따라 진화하여 데이터베이스 구조를 업그레이드하거나 수정할 수 있게 합니다.
6. 크로스 플랫폼 호환성: SQFlite는 iOS, Android, 웹을 포함한 Flutter에서 지원하는 다른

플랫폼에서 원활하게 작동하여 플랫폼간에 데이터베이스 코드를 공유할 수 있다.
전반적으로 SQFlite는 로컬 데이터베이스를 Flutter 앱에 통합하는 과정을 간소화하여 데이터의 영속성과 관리에 신뢰성 있고 효율적인 솔루션을 제공한다.

3) Provider

Flutter에서 Provider는 애플리케이션의 다른 부분 간에 상태(state)를 관리하고 공유할 수 있는 상태 관리 라이브러리이다. Provider는 Flutter에서 제공하는 InheritedWidget 개념을 따르며, 상태의 관리와 상태가 변경될 때 위젯을 업데이트하는 과정을 간소화한다.
Provider의 핵심 개념은 프로바이더(Provider)로, 프로바이더는 상태를 보유하고 의존하는 위젯이 해당 상태에 접근하고 사용할 수 있도록 한다. 프로바이더는 상태나 데이터를 저장하고 제공하는 컨테이너로 생각할 수 있다.

Provider는 사용 목적에 따라 다양한 유형의 프로바이더를 제공한다:
1. ChangeNotifierProvider : 이 프로바이더는 `ChangeNotifier` 클래스를 확장하는 클래스와 함께 사용된다. 변경 가능한 상태를 정의하고 관리하며, 다른 위젯이 해당 상태를 업데이트하고 수신하는 것을 가능하게 한다. 상태가 변경되면 프로바이더는 의존하는 위젯에게 다시 빌드하도록 알린다.
2. Provider : 이는 모든 유형의 데이터에 사용할 수 있는 일반적인 프로바이더이다. 데이터를 위젯에 노출하고, 데이터가 변경될 때 의존하는 위젯을 자동으로 업데이트한다. 상태 변경을 처리하기 위한 내장 메커니즘은 제공하지 않으므로 읽기 전용 또는 불변 데이터에 적합하다.
3. StreamProvider : 이 프로바이더는 위젯에 스트림 데이터를 노출하려는 경우 사용된다. 스트림을 수신하고 새 데이터가 사용 가능할 때마다 위젯을 업데이트한다.
4. FutureProvider : 이 프로바이더는 위젯에 Future 결과를 노출하려는 경우 사용된다. Future의 비동기적인 특성을 처리하고, Future가 완료될 때 위젯을 업데이트한다.

Provider를 사용하면 비즈니스 로직과 상태 관리를 UI 구성 요소와 분리할 수 있으므로 모듈화되고 유지 관리가 용이한 코드베이스를 구축할 수 있다. Provider는 단방향 데이터 플로우 개념을 장려하며, 확장 가능하고 성능이 우수한 Flutter 애플리케이션 개발에 도움을 준다.

4) Git

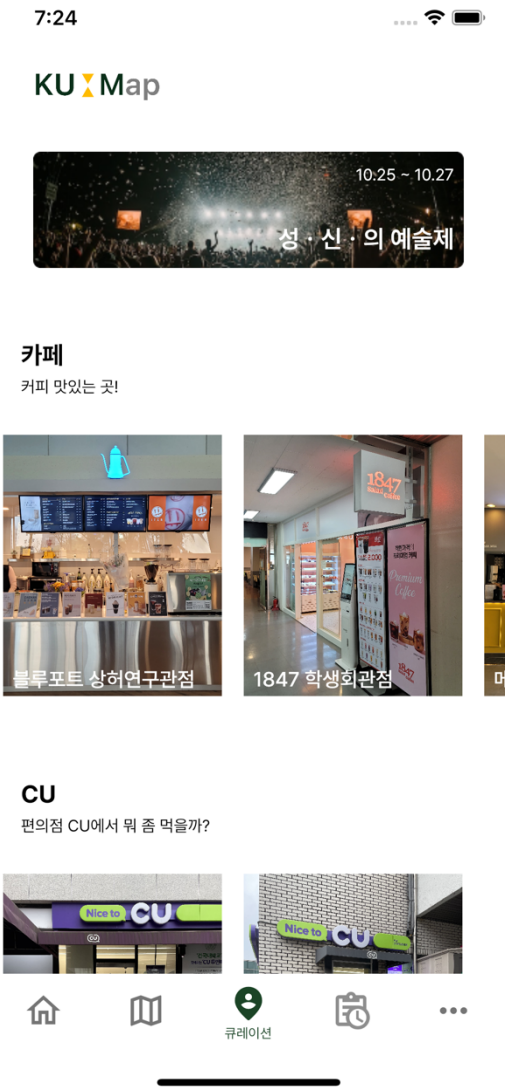
Git은 소프트웨어 개발 중 파일과 소스 코드의 변경 사항을 추적하는 분산 버전 관리 시스템이다. 여러 개발자가 협업하고 코드베이스의 다른 버전을 유지할 수 있게 해준다. Git을 사용하면 변경 사항의 이력을 추적할 수 있으며, 누가 언제 어떤 변경을 했는지 등을 확인할 수 있다. Git은 브랜치와 병합을 지원하여 개발자가 새로운 기능을 독립적으로 작업하고 주 코드베이스에 병합할 수 있게 한다. Git은 분산 시스템으로 각 사용자가 저장소의 완전한 복사본을 가지므로 오프라인 작업과 후에 동기화할 수 있다. 원격 저장소를 지원하여 코드 공유와 협업이 가능하다. Git은 변경 사항을 커밋하기 전에 준비하고 정리하는 스테이징 영역을 제공한다. 암호화 해시 함수를 통해 코드베이스의 무결성을 보장하며, Git은 속도, 확장성 및 유연성으로 널리 사용되고 알려져 있다.

5) screenutil 패키지

Flutter Screen Util 패키지는 개발자들이 Flutter 앱의 UI를 다양한 화면 크기와 밀도에 맞게 조정하는 데 도움을 주는 유틸리티 패키지이다. 이 패키지는 디바이스의 화면 속성을 기반으로 치수, 여백, 폰트 크기를 계산하고 설정하는 함수와 메서드 세트를 제공한다. 이 패키지는 반응형 레이아웃을 생성하는 과정을 간소화하고 다양한 디바이스에서 일관된 UI를 보장한다. 개발자는 고정된 픽셀이 아닌 논리적 단위를 사용하여 크기와 여백을 정의할 수 있으므로 앱이 다양한 화면 해상도에 잘 적응할 수 있다. Flutter Screen Util 패키지는 앱이 다양한 디바이스에서 잘 보이고 작동하여 사용자 경험을 향상시킨다.

3.3 각 화면 개발

3.3.1 큐레이션 탭 화면 개발



carousel slider 라는 flutter package를 사용하여 배너를 개발하였다.

현재 총 3개의 배너로 이루어져 있으며, 이는 3초에 한 번씩 자동으로 다른 배너로 바뀌며 재생된다.

현재 총 3개의 큐레이션으로 구성되어 있으며, 카페, CU, Kcube에 대해 소개해주고 있다. 한 큐레이션마다 제목, 부제, 장소 이름과 그에 해당하는 이미지를 보여주고 있다. 이미지는 팀원들과 같이 직접 찍은 사진을 이미지 서버에 올려 호스팅하고 있다.

서버에 있는 이미지를 링크를 가져와 extended_image라는 flutter package를 사용하여 앱 내에 불러온다(이 패키지는 개발자들이 쉽게 외부 이미지를 가져오도록 만들어진 패키지이다.). 또한 사용자들이 사용성을 높이기 위해 이미지를 캐시에 저장하여 다시 앱 내에서 다시 불러올 때 빠른 속도로 불러올 수 있도록 설정하였다.

3.3.2 마이루틴 탭 화면 개발

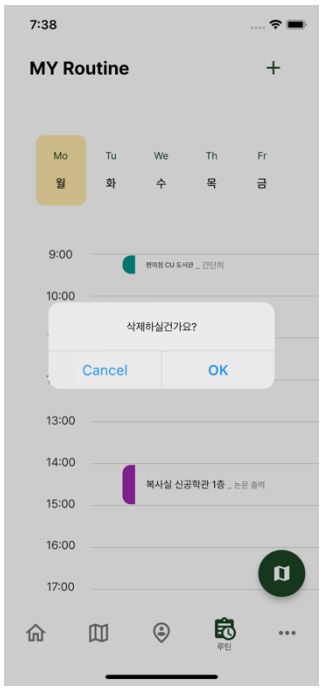
월, 화, 수, 목, 금에서 09 : 00 ~ 17 : 00 동안 사용자가 자신이 갈 장소를 미리 앱 내에 저장하여 시간표처럼 볼 수 있도록 하는 화면이다.

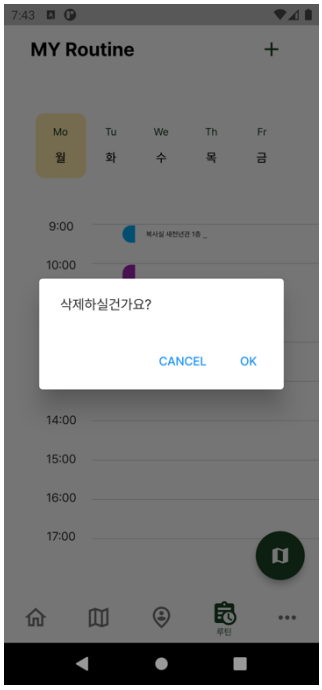
총 크게 4개의 버튼이 있다.

버튼 1. + 버튼은 사용자가 직접 자신의 일정과 장소를 추가하기 위한 화면으로 넘어가도록 하는 버튼이다.

버튼 2. 지도 모양의 초록색 floating 버튼은 하나의 요일에서 사용자가 설정한 장소들을 지도 상에서 보여주는 버튼이다.

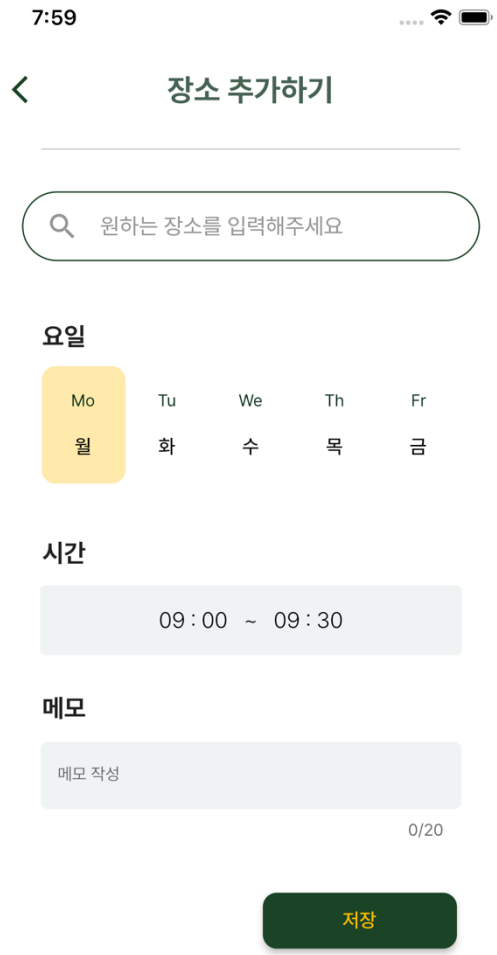
버튼 3. 월, 화, 수, 목, 금을 선택하는 버튼이다.





버튼 4. 사용자가 설정한 장소를 지울 수 있는 버튼이다. 해당 버튼은 사용자가 설정한 장소를 클릭한 경우 활성화된다. adaptive_dialog라는 flutter package를 이용하여 위의 사진들처럼 IOS와 Android에서 다이얼로그를 작동시켰다. 만약 삭제할 경우 Sqflite에 저장된 사용자들의 장소와 간단한 메모들을 지울 수 있게 된다.

이 화면에서는 총 3개의 Provider를 사용하였고, 모두 ChangeNotifier 형태를 사용하였다. 처음 마이루틴 탭 화면을 시작시켰을 때, Sqflite에 저장된 정보들을 조회하여 리스트에 저장하였고, 이를 rounded 사각형으로 표현해주었다. 특히 rounded 사각형 같은 경우 맨 앞쪽에 포인트 색깔을 위치시켜 시간의 범위나 시작 시간 등을 잘 표현되게 했다. 이 과정에서 30분 정도의 나의 루틴을 설정하여 추가 시켰을 경우, 글자가 깨지는 현상이 발생하여 텍스트의 사이즈와 padding을 변경시켜 예외 처리를 했다.



다음은 마이루틴 탭 화면에서 버튼 2를 눌러 장소를 추가할 때의 '장소 추가하기'화면이다. 총 6개의 버튼으로 구성되어있다.

버튼 1. 이전 화면으로 돌아가는 버튼이다.

버튼 2. 장소를 입력할 수 있는 입력창 버튼이다.

버튼 3. 요일을 설정할 수 있는 버튼이다.

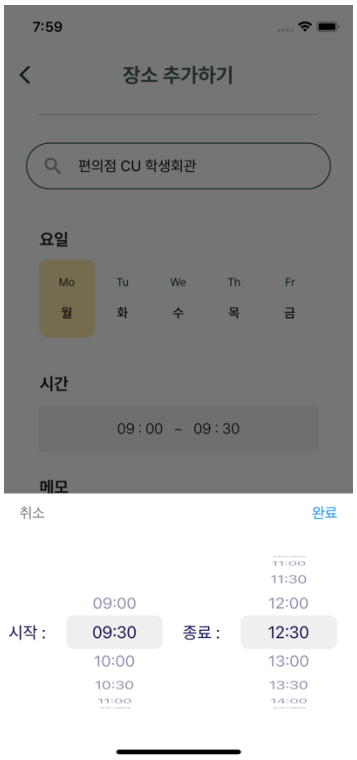
버튼 4. 시간을 설정할 수 있는 버튼이다.

버튼 5. 메모를 설정할 수 있는 버튼이다.

버튼 6. 화면에 있는 정보들을 sqflite에 저장할 수 있는 버튼이다.

버튼 2에서는 사용자가 원하는 장소를 입력할 수 있다. 그 과정에서 장소에 대한 위치(경도, 위도), 장소 이름을 불러온다.

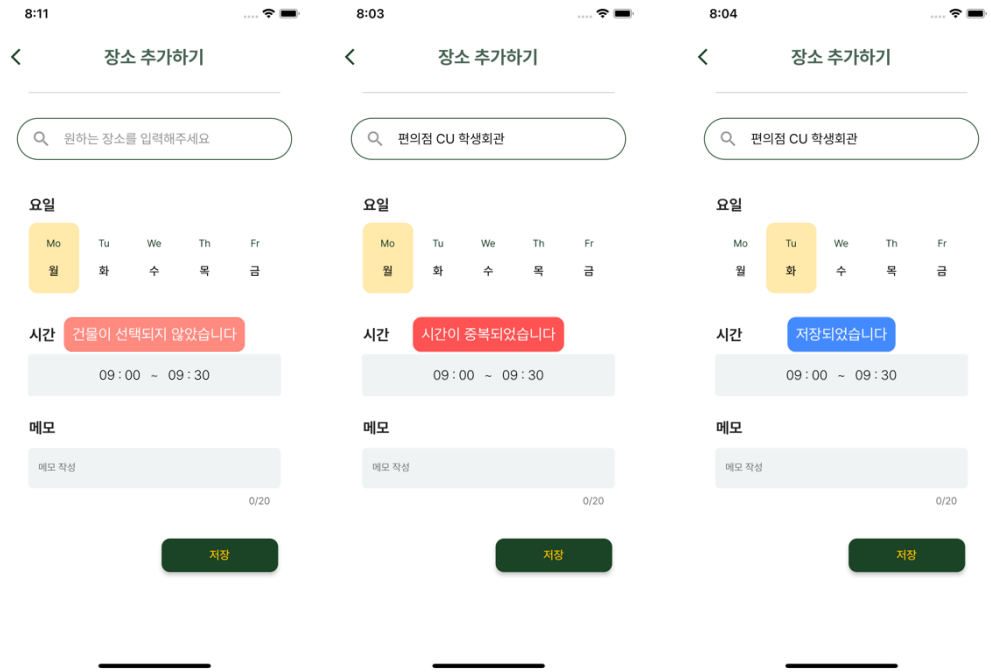
버튼 5에서는 TextFormField 위젯을 사용하여 사용자가 직접 키보드를 이용하여 자신의 간단한 메모를 작성하도록 설정하였고, 이는 20자로 제한을 두었다.




```dart
import 'package:flutter_datetime_picker/flutter_datetime_picker.dart';

class Routine_CustomPicker extends CommonPickerModel {
  String digits(int value, int length, int thirty) {
    if (thirty == 0) {
      return '$value'.padLeft(length, "0")+':00';
    } else {
      return '$value'.padLeft(length, "0")+':30';
    }
  }

  Routine_CustomPicker({DateTime? currentTime, LocaleType? locale}) : super(locale: locale) {
    this.currentTime = currentTime ?? DateTime.now();
    this.setLeftIndex(this.currentTime.hour);
    // this.setMiddleIndex(this.currentTime.hour*2);
    // this.setRightIndex(this.currentTime.hour*2);
    this.setMiddleIndex(0);
    this.setRightIndex(0);

  }

  // SJ : 시작 시간
  @override
  String? middleStringAtIndex(int index) {
    // print(index);
    if (index >= 0 && index < 16) {
      return this.digits((index~/2)+9, 2, (index%2));
    } else {
      return null;
    }
  }

  // SJ : 끝 시간
  @override
  String? rightStringAtIndex(int index) {
    if (index >= 1 && index < 17) {
      return this.digits((index~/2)+9, 2, (index%2));
    } else {
      return null;
    }
  }
```

버튼 4에서는 flutter_datetime_picker라는 패키지를 커스텀하여 사용하였다. 해당 패키지에는 총 3개의 picker가 존재하지만, 시작과 종료 시간만 필요하여 2개의 picker로 middle과 right를 사용하여 추가적으로 개발하였다. 시작 시간은 09:00~16:30까지 존재하고 종료 시간은 09:30~17:00까지 존재하고, 모두 30분 간격씩 설정되어있다.

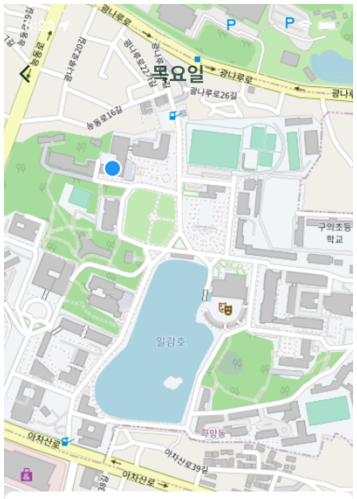
완료 버튼을 누를 경우 해당 시작 시간과 종료 시간은 '장소 추가하기' 화면에 나타나게 된다.

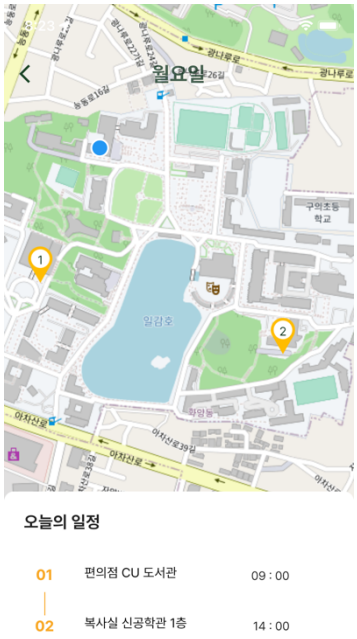


위의 사진 1. 은 '장소 추가하기' 화면에서 위치를 선택하지 않고 저장 버튼을 눌러 발생한 토스트 상태 메시지이다.

위의 사진 2. 는 기존에 사용자가 설정하였던 나의 루틴들과 새롭게 설정한 루틴의 시간이 겹쳐 발생한 상태 메시지이다. 이를 적용하기 위해 Provider를 이용하여 마이루틴 탭 화면에서 Sqflite에 저장된 정보들을 조회하고 저장한 리스트를 가져와 알고리즘을 적용하였다.

위의 사진 3. 은 장소가 잘 설정되어있고, 시간이 중복되지 않기 때문에 잘 저장되었다는 것을 사용자에게 알리는 토스트 상태 메시지이다. 사진 1, 사진 2와 달리 긍정적인 의미로 파란색을 사용하였다.





위의 사진들은 마이루틴 탭 화면에서 버튼 2. 지도 버튼을 클릭했을 때 나오는 화면이다.

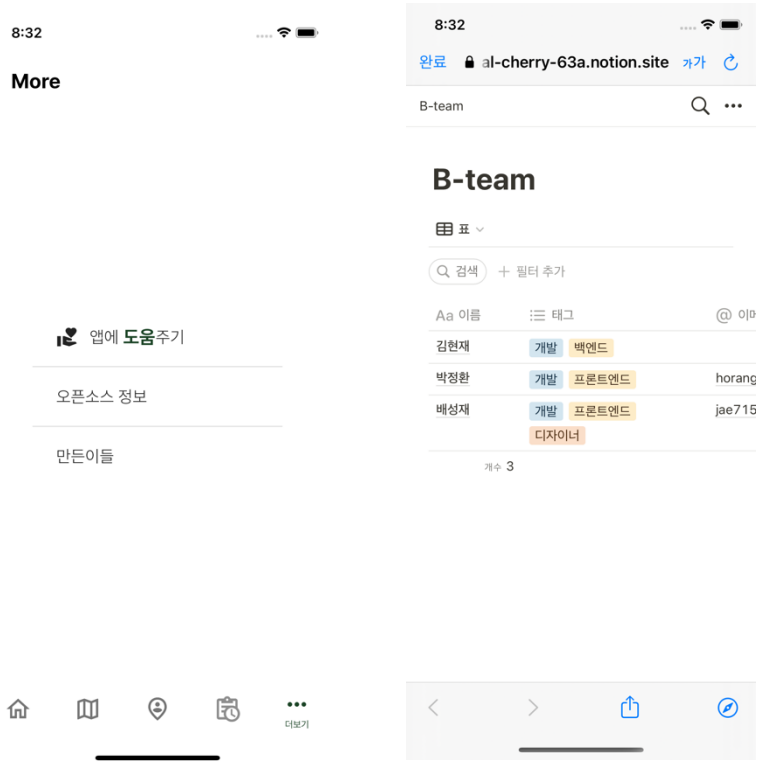
맨 위에 해당 요일을 보여주고, 뒤로 갈 수 있는 버튼을 넣어주었다. 마이루틴 탭 화면에서의 sqflite에 저장된 데이터 리스트를 가져와 사용하였다.

사진 1. 은 오늘의 일정이 없었을 때를 보여주는 화면이다. 리스트가 null일 때를 가져왔지만 예외처리를 하여 이상없이 나왔다.

사진 2. 는 일정이 2개일 때의 화면이다. 1번째로 갈 장소와 2번째로 갈 장소를 보여주고 있으며, 해당 위치와 건물 이름, 시작 시간을 보여주고 있다.

지도 같은 경우 김현재 학우가 만든 OSM을 이용하여 여러 타일들을 서버에서 가져오는 방식이고, 위에 일정 순서로 마커를 사용하여 사용자에게 나타냈다.

3.3.3 더보기 탭 화면 개발



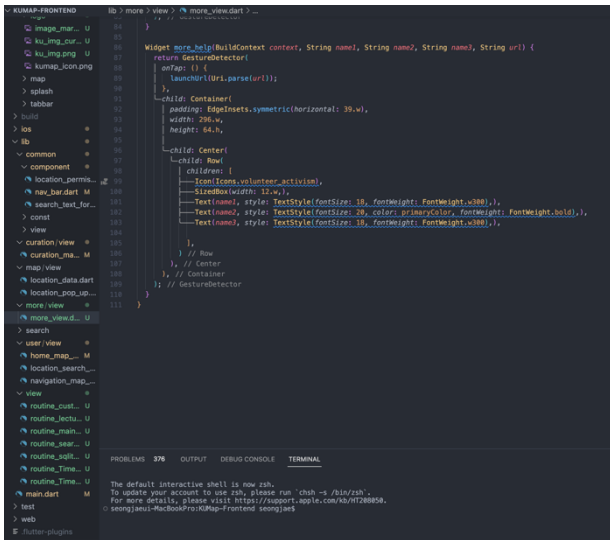
더보기 탭 화면은 오픈소스 정보나 만든이들 등을 소개해주려고 만든 간단한 탭 화면이다. 주로 노션의 외부 링크를 이용하여 보여주고 있다.

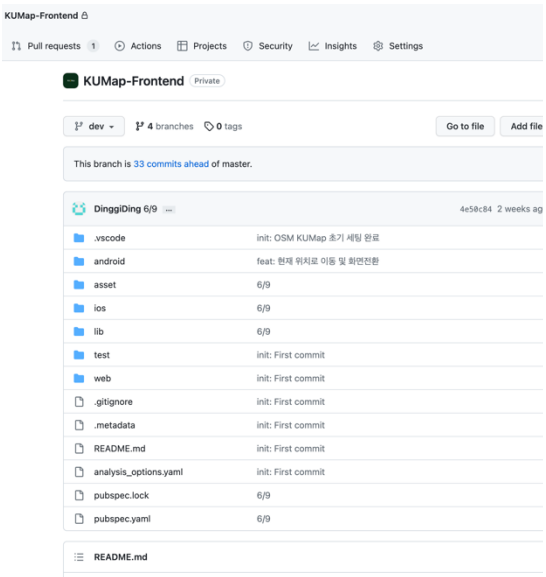
사진 2. 는 더보기 탭 화면에서 ‘만든이들’ 버튼을 클릭하였을 경우에 노션 링크로 이동하여 보여지게 된다.

3.3.4 Sqflite DB 구조

```
CREATE TABLE Specs(
  id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
  build TEXT,
  lat REAL,
  lng REAL,
  icon TEXT,
  build_detail TEXT,
  day INTERGER NOT NULL,
  startTime INTERGER NOT NULL,
  endTime INTERGER NOT NULL
)
```

3.4 코드 통합

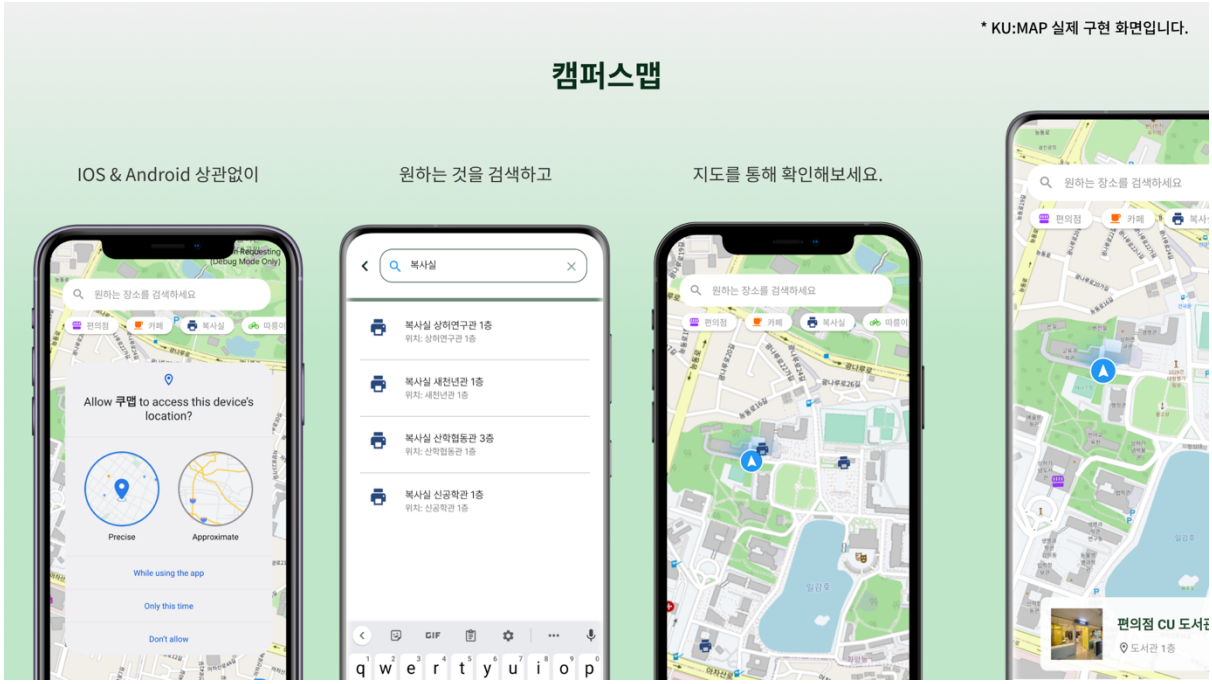


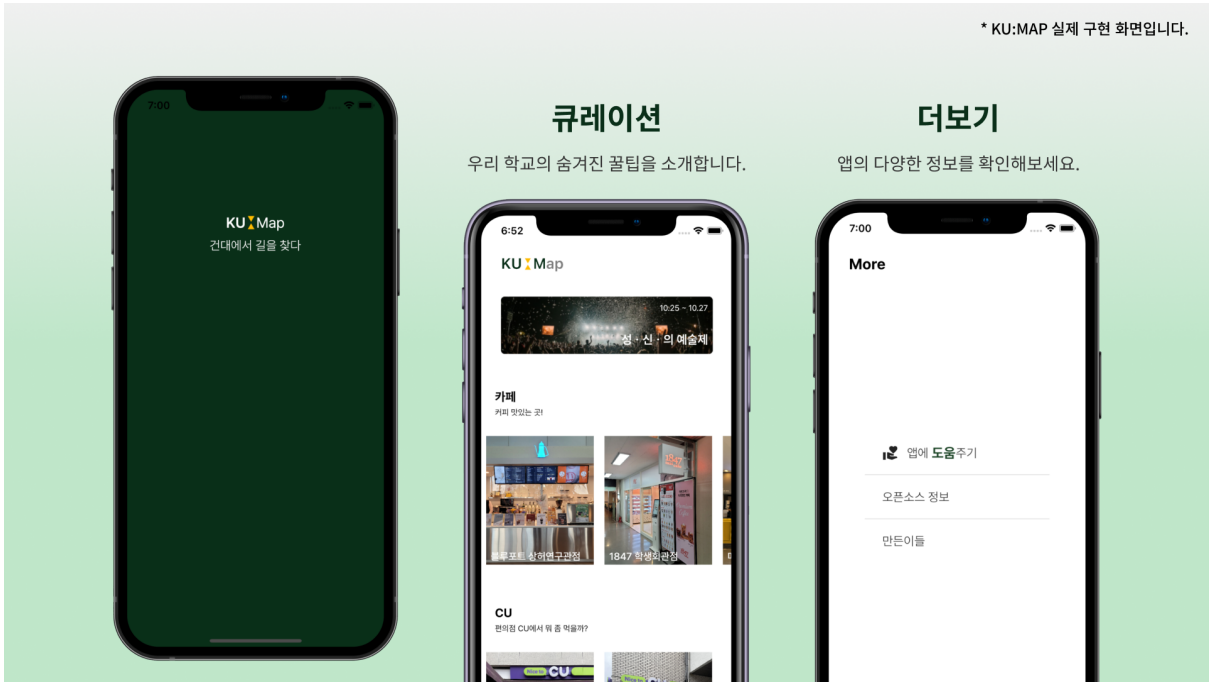


박정환 학우가 만든 메인 탭 화면, 길찾기 탭 화면과 내가 만든 3개의 탭 화면을 통합하기 위해 Git을 이용하여 서로가 만든 코드 등을 확인하고 이를 통합하였다. 이미지 소스와 사용했던 flutter package, 내부 설정 코드, 변수 이름, IOS 권한 설정 등 팀원과 조율하여 코드 통합을 마쳤다.

제 4장 결과

4.1 최종 결과

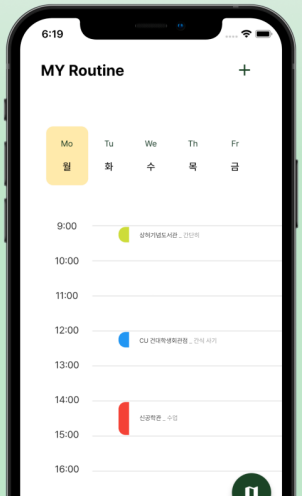




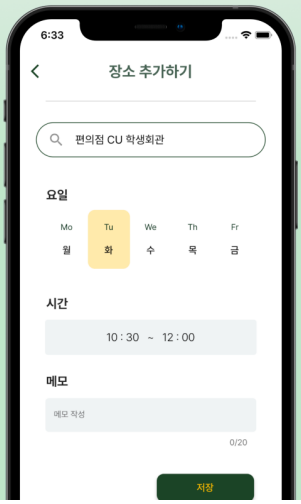
* KU:MAP 실제 구현 화면입니다.

마이 루틴

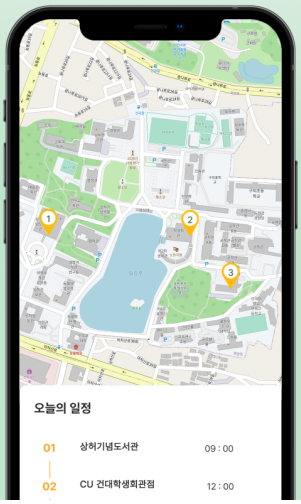
주기적인 학교 일정 까먹지 않도록



자주 가는 장소까지 연결해서



지도에서 확인하는 나의 루틴



* KU:MAP 실제 구현 화면입니다.

KU:Map

우리가 만들어가는 건국의 지도