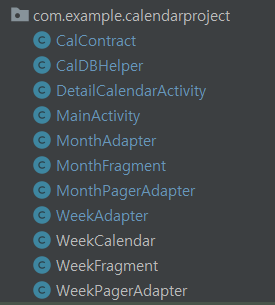
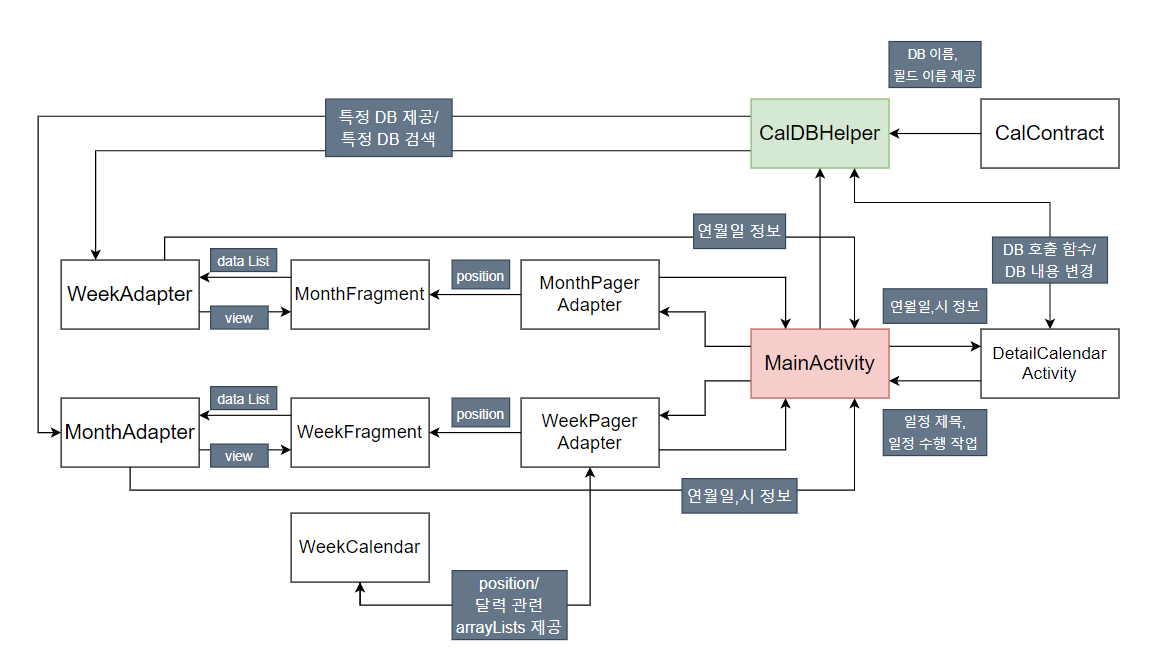
**안드로이드 최종 프로젝트**

**2071359 이서연**

**\*개요**

안드로이드에서 실행되는 일정 관리 앱을 만든다.

**\*프로그램의 구조(클래스 및 클래스 간의 관계)**

**+)클래스 설명**

**CalContract**: DB 테이블과 필드 이름, 속성을 정의하는 클래스.

**CalDBHelper**: sql문법으로 DB테이블 레코드를 읽고 수정하는 클래스.

**DetailCalendarActivity**: 일정을 추가, 수정하는 서브 액티비티.

**MainActivity**: 메인 액티비티.

**MonthAdapter**: 월간 달력에 대한 사용자 정의 어댑터.

**MonthFragment**: 한 달에 대한 월간 달력을 표시하는 프레그먼트.

**MonthPagerAdapter**: 스와이핑 가능한 월간 달력의 프레그먼트들을 생성하는 클래스.

**WeekAdapter**: 주간 달력에 대한 사용자 정의 어댑터.

**WeekCalendar**: 받은 포지션에 대해 날짜 정보 리스트를 생성하는 클래스.

**WeekFragment**: 한 주에 대한 월간 달력을 표시하는 프레그먼트.

**WeekPagerAdapter**: 스와이핑 가능한 주간 달력의 프레그먼트들을 생성하는 클래스.

**DB를 사용하는(CalDBHelper를 통해) 주요 클래스**

-MainActivity: 월간 달력의 세부 일정 표시할 때 DB를 확인해서 일정 리스트 생성.

-DetailCalendarActivity: DB 일정 확인, 추가, 업데이트, 삭제 수행

-WeekAdapter/MonthAdapter: getView함수 내부에서 해당 연월일, 시에 대한 일정 확인

**\*각 기능별로 코드의 구조 설명**

**1. 일정추가 메뉴 표시 및 일정추가 액티비티 실행**

일정추가 메뉴를 표시하기 위해 플로팅 버튼을 추가하였고, 기존의 Linear레이아웃을 사용하면

플로팅 버튼이 프레그먼트에 의해 가려져서 Coordinator를 메인 레이아웃으로 사용함.

텍스트이(가) 표시된 사진

자동 생성된 설명

일정추가 액티비티를 실행할 때, 클릭한 날짜의 연월일시 정보를 인텐트로 넘겨야 DB를 읽어올 수 있기 때문에 주간-월간 프레그먼트에 함수형 인터페이스를 정의하고 메인에서 이를 상속받아

달력 아이템이 클릭되면 메인의 selectYMD, selectH의 값을 클릭된 연월일시로 갱신되게 하였음. (아무런 선택이 없는 초기상태엔 디폴트를 1일과 0시로 하여 오류가 나지 않도록 함)

텍스트이(가) 표시된 사진

자동 생성된 설명

플로팅 버튼을 클릭하면 일정추가 액티비티를 실행해야 되는데 몇 가지 경우를 나누었음.

1)**월간 달력**이고 선택된 날짜에 다른 일정이 있을 때(기존 일정 표시-다이얼로그)

-DB에서 해당 날짜에 대한 모든 레코드를 가져와서 다이얼로그에 표시할 리스트를 생성하고

다이얼을 표시한다. 다이얼에서 리스트 아이템을 클릭하면 일정 액티비티 실행

2)**월간 달력**이고 선택된 날짜에 다른 일정이 없을 때(새로운 일정)

3)**주간 달력**일 때(새로운 일정인지 아닌지는 일정 액티비티에서 한 번 더 검사함)

-선택 정보가 저장된 selectYMD와 selectH로 인텐트를 만들고 액티비티 실행

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

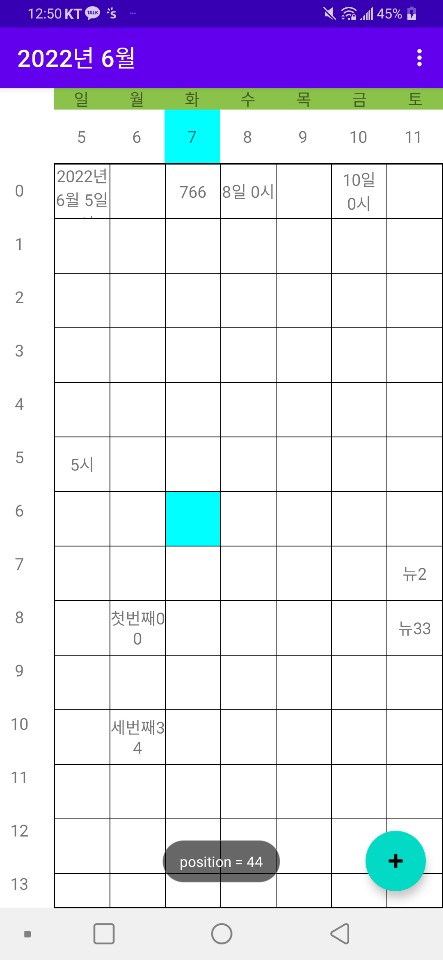
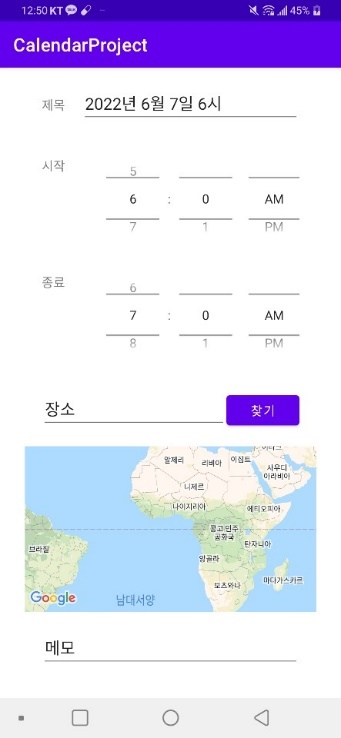
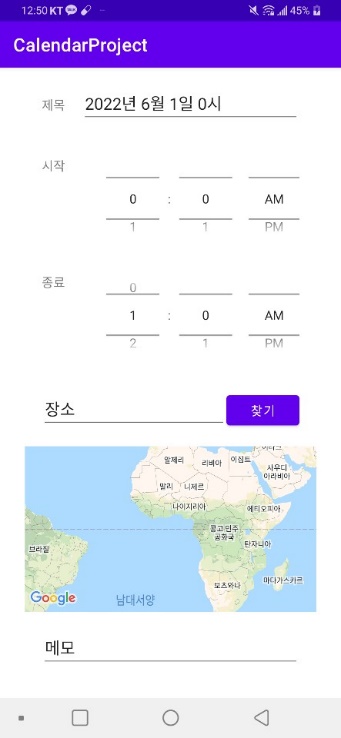
일정추가 액티비티의 경우 EditText에 무언갈 입력하거나 뷰에 의해서 아래가 잘리는 문제가 있었는데 ScrollView를 추가해 스크롤이 가능하도록 하였음.

일정추가 액티비티가 실행되면 인텐트 정보를 받아서 저장하고, 해당 연월일시가 DB에 존재하는 레코드인지 확인하고 boolean변수를 준다.

존재한다면 그 레코드 내용으로 제목, 시간, 지도등을 변경하고, 아니라면 디폴트 값을 넣어준다.

시작시간 종료시간을 표시하는 것은 NumberPicker을 각각 3개씩 사용하였는데, 시작시간의 디폴트는 받아온 시간 값으로, 종료시간은 +1인 시간 값으로 하였음

**실행결과**

****

**2. 상세일정 작성 및 저장**

**일정 액티비티 간단 설명:** 제목과 장소, 메모는 EditText로 해서 글자를 쓰고 지울 수 있음

시간-종료는 NumberPicker 사용, 스크롤도 시간 지정 가능

지도 표시는 fragment, 구글 맵 api 사용함.

텍스트이(가) 표시된 사진

자동 생성된 설명

**지도 표시:** 찾기 버튼을 눌렀을 때나 액티비티를 실행하고 맵 동기화가 완료되었을 때 getAddress함수 실행.

이 함수는 장소 EditText의 문자열을 받고, 그 문자열로 위치 검색을 하여 표시해주는 함수임.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명여러 위치 중에서 bestResult 하나만 사용하고, 해당 Lat과 Long값으로 구글 맵 마커를 설정해서 구현함.

**시작-종료시간:** 위 이미지는 NumberPicker을 세팅하는 부분. 시간은 0~11/분은 0~59/오전-오후는 0~1로 최소-최대값을 지정하였음. wrapSelector는 시작과 끝을 연결할 지에 대한 설정으로 false로 해 둠.

시작에 대한 NumberPicker 3개를 DB에 따로따로 저장하기는 복잡해서 하나의 문자열로 합친

결과를 DB에 저장하고 받아올 땐 문자열.split()을 이용해 나누는 방식으로 Picker를 사용하였음.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**텍스트이(가) 표시된 사진

자동 생성된 설명**

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

**저장과 업데이트**: 저장하기와 업데이트는 하나의 버튼을 공유하기 때문에 구분해주어야 했는데

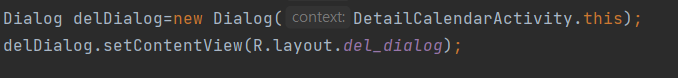
액티비티가 실행될 때 DB검사로 OldCal에 불리언 값을 주었던 것을 이용함.

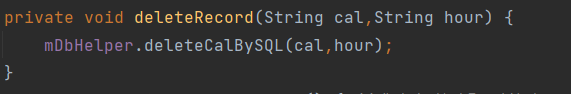
**해당 프로젝트는 여러 클래스에서 DB를 호출하는데 id값을 사용하면 사용이 복잡할 것 같아서 전체 DB에서 하나씩만 존재할 수 있는 연월일+시 필드를 id 대신 사용하였음.**

저장과 업데이트 둘다 일정추가 액티비티에 입력한 내용들을 읽어서 CalDBHelper의 함수에 전달하는 방식으로 구현함. (업데이트는 시간도 갱신해야되니 예전과 현재시간 함께 넘김)

Picker12to24 함수는 액티비티에서의 시간부분이 0~11과 AMPM으로 표현되는데 오후11시를 23시로 변환하는 코드임.

텍스트이(가) 표시된 사진

자동 생성된 설명 

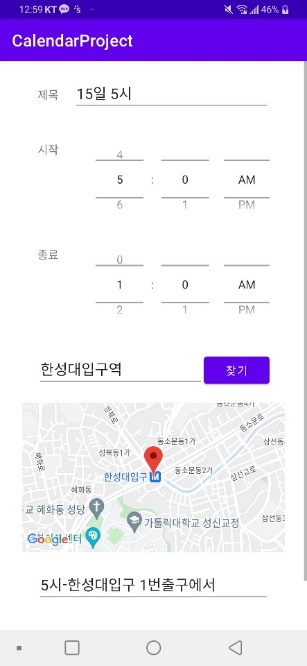
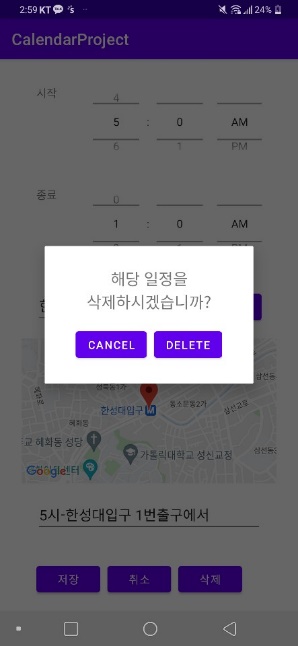


**삭제:** 삭제버튼을 눌렀을 때, 한 번 더 확인해주는 다이얼로그를 띄워 줌.

삭제 다이얼로그에 대한 xml 파일을 정의하고 다이얼로그 객체의 view로 설정해준 후, 취소와 삭제에 대한 클릭이벤트를 처리하여 구현하였음.

취소하면 다이얼로그만 종료되고, 삭제하면 레코드를 지우고 반환 인텐트를 설정한 후 종료한다.

**실행결과**



**3. 달력에 일정제목 표시 및 저장된 상세일정 표시 기능** **텍스트이(가) 표시된 사진

자동 생성된 설명**

**달력에 일정 제목을 표시하는 부분은** 각각의 달력 어댑터에서 수행하였음.

달력 아이템들이 getView를 통해 그려질 때 CheckDB함수로 해당 position 날짜에 대한 DB를

검사하고 있다면 표시, 없다면 공백으로 표시함.

**\***이 방법은 viewPager2 특성 상, 미리 프레그먼트를 생성하는 것 때문에 현재 필요하지 않은 달력 부분까지 DB를 검사하는 문제가 있었는데 프레그먼트의 생성당시 position과

현재 페이지 positon을 비교하는 방법으로 해결하였음(현재와 앞뒤만 DB검사) **텍스트이(가) 표시된 사진

자동 생성된 설명**

텍스트이(가) 표시된 사진

자동 생성된 설명

**주간 달력은** 한 아이템당 일정이 하나뿐이라 이전 xml을 그대로 사용했지만

**월간 달력**은 일정을 최대 2개까지 표시해야 되어 TextView 2개를 추가하였고,

paadding으로 내부 공간을 만들고 흰색-파란색으로 변경하여 테두리를 표현하였음.

DB를 읽어 cursor객체를 받아오고 while문을 돌면서 값을 변경하는데 count변수로 0번이면 날짜 아래 첫번째 텍스트 뷰의 배경색과 값을 변경하고 1번이면 두번째 것을 변경하여 표시함.

세부 기능으로 5글자가 넘으면 일정제목이 표시될 때 그 뒤를 자르고 ‘…’을 넣어 주었음.

텍스트이(가) 표시된 사진

자동 생성된 설명

**상세일정 표시**는 메인 액티비티에서 해당 일에 대한 DB를 검사하여 리스트를 만들고 다이얼로그의 리스트에 넘기는 방법으로 구현하였음. **기능1**과 이어지는 부분인데 플로팅 버튼이 클릭 될 때 일정이 1개 이상이면 이 함수가 실행된다.

calItem, hourItem 2개를 생성한 이유는 리스트 뷰가 클릭되면 해당일정으로 넘어갈 때 시간정보가 필요하기 때문이다. 첫번째 리스트가 클릭되면 해당 연월일과 시간이 넘겨지고 그 내용으로

**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**일정 액티비티에서 DB를 확인한다. (인덱스 정보를 넘겨준 것은 저장/삭제될 때 해당 부분을 수정하기 위해서)

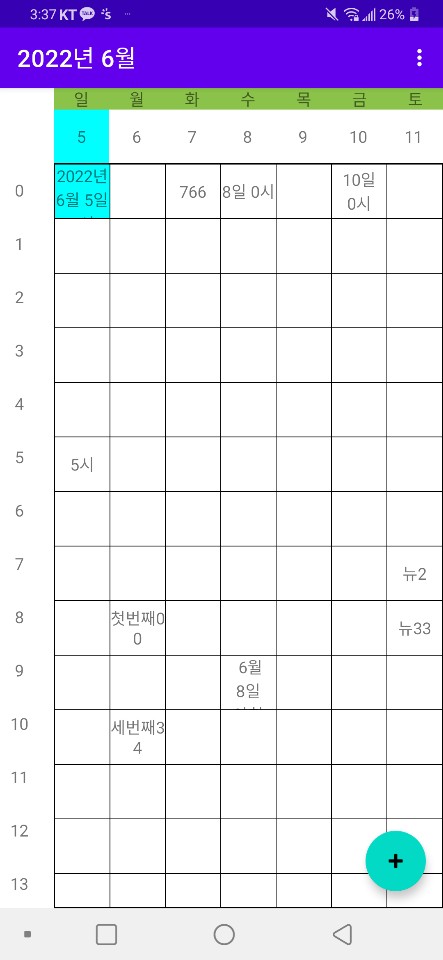
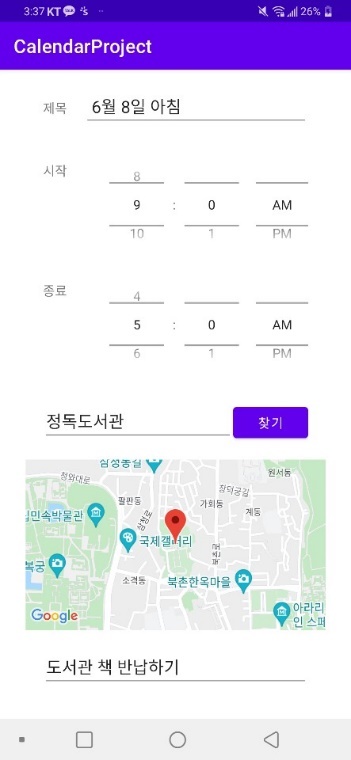
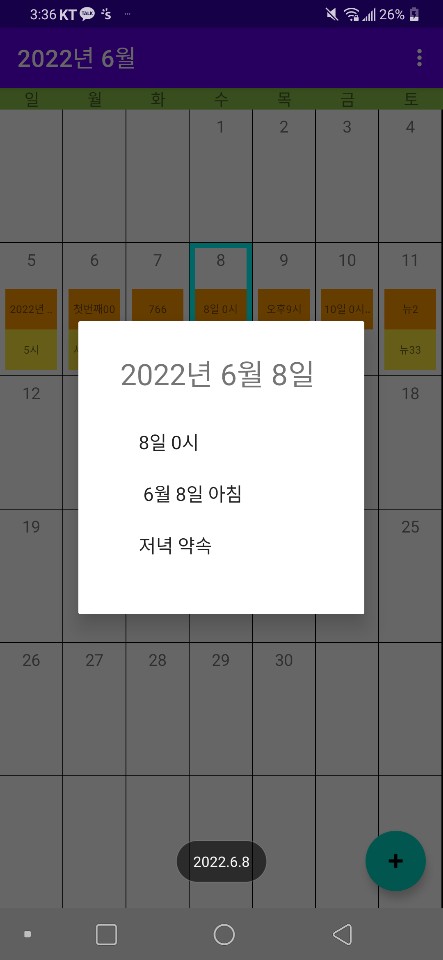
**저장/삭제/업데이트 후:** 일정 수정 후 메인 액티비티로 돌아오는데 다시 로드되는 것이 아니기 때문에 수정 전 내용이 남아있다. DB내용은 이미 변경되었지만 이 부분도 변경시켜 주기 위해

onActivityResult 함수에서 받은 intent 내용으로 작업을 수행하였다.

삭제코드 “CAL\_DELETE”가 있으면 해당 내용 (월간이면 맞는 텍스트뷰)를 삭제하고

없으면 생성 및 업데이트로 처리하여 전달된 제목을 표시한다.

**실행결과**

****

**\*자기 평가 및 느낀점**

-git으로 코드를 공유하며 팀 작업을 하는 것은 처음이었는데 팀원의 오류를 해결해주거나 기본 구조를 짜는 등 제 몫을 충분히 했다고 생각합니다. 팀 프로젝트라서 좀 더 적극적으로 서로의 코드와 의견을 공유하고 싶었는데 그 부분이 잘 이루어지지 못한 것은 아쉬움이 남습니다.

이후에도 팀 프로젝트를 진행한다면 잘 분담해서 해보고 싶다는 생각이 들었습니다.

-기말 프로젝트까지 진행하며 매 프로젝트의 구조를 짜고 자잘한 오류를 수정하며 이런 부분이 문제가 될 수 있다는 것도 알 수 있었고, 문제가 생긴 부분에 대해 타인의 코드와 여러 아이디어를 찾아보고 이해하여 자신의 프로젝트에 적용해보는 과정도 도움이 많이 되었습니다.

**\*Github Repository 주소**

<https://github.com/seoyeon1359/Calendar>

->최종 프로젝트에 대한 깃헙 주소

<https://github.com/Junghayeon/AndroidTeamProject>

->팀원과의 깃헙 주소(분담 작업을 하다가 변경이 있어 기말프로젝트는 대부분 개인작업이 됨)