날씨 및 명소 검색 앱

조민규*, 엄종욱*, 고희정*, 조희찬*, 김동년*, 이형봉*
*강릉원주대학교 컴퓨터공학과
{0215boc, whddnr1912.
gmlwjd1240. whgmlehs. chunchu0317. hblee}@gwnu.ac.kr

Weather and Famous Place Searching App

Min-Gyu Cho*, Jong-Wook Eom*, Hee-Jung Go*, Hee-Cha Cho*,
Dong-Nyeon Kim*, Hyung-Bong Lee*
*Dept. of Computer Science & Engineering, Gangneung-Wonju National University

요 약

이번 캡스톤 디자인에서는 식당, 카페 등 유명 장소 검색 시 날씨까지 함께 보여줄 수 있는 앱을 구현한다. 이 앱을 사용하는 사용자는 날씨 정보를 보면서 희망하는 장소를 검색하여 출장 업무나 여행 일정에 도움을 얻을 수 있다.

1. 서론

날씨를 확인하지 않고 외출하였을 때 급변하는 날씨로 인해 피해를 입었거나, 현재 위치에서 카페, 도서관, 식당 등 주위의 특정 장소에 대한 정보가 필요했던 경험을 다들 한 번쯤은 겪어봤을 것이다. 주변의 상인들도 날씨와 연관이 많다. 비가 오는 날에는 빗소리에 어울리는 파전과 막걸리 가게들이, 날씨가 맑으면 시원한 냉면이나 디저트 가게들의 검색이 잦을 것이다.

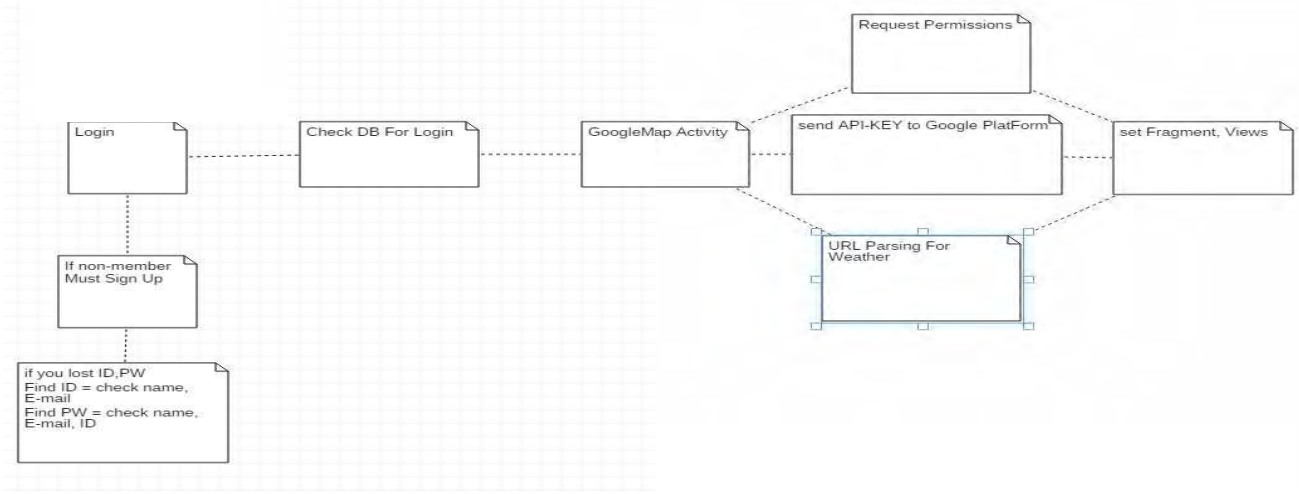
위와 같이 날씨와 장소 사이에는 긴밀한 관계가 있는데, 날씨를 보면서 장소를 검색할 수 있는 어플이 많지 않다. 웨어(where) 어플은 사용자 근처의 음식점을 찾아주고, 배달의 민족은 선택된 음식점으로 전화를 연결해주며, 하이 날씨는 실시간 기상 정보를 보여준다. 카카오 맵은 내비게이션 기능 중심으로 버튼을 누르면 간단한 날씨 정보를 보여준다.

따라서 이번 캡스톤 디자인에서는 날씨와 장소 정보를 동시에 제공하는 어플을 개발하도록 한다.

2. 어플 설계

어플의 전체적인 동작 구조는 아래와 같다(그림 1 참조).

- 로그인 단계
  - 회원이 아닐 경우 회원 가입 진행
  - ID 분실의 경우 이메일을 키로 검색



(그림 1) 어플 동작 구조 블록 다이어그램

- PW 분석의 경우 ID 및 이메일을 키로 검색
- DB에서 ID와 PW가 일치하는 경우 로그인 성공

• Activity 실행 단계
- 지도를 넣을 fragment, 초기 날씨, 버튼 출력
- 권한 요청(INTERNET, FINE_LOCATION, COARSE_LOCATION, NETWORK_STATE)
- 미리 준비한 키 값으로 구글 서버 지도 획득
- 날씨 URL 소스를 DOM 트리 형태로 구성해서 파싱한 뒤 날시 이미지와 기온 정보 획득
- 프로젝트 빌드 과정에서 noman GitHub 오픈 소스 연동

• Activity 설정 단계
- 획득한 날씨와 지도, 주변 지역 정보를 activity에 설정
- 버튼 listener 및 지도의 zoom in/out 활성화

• Camera 이동 단계
- 현재 위치로 카메라 이동
- 날씨 이미지를 세팅 및 기온 출력

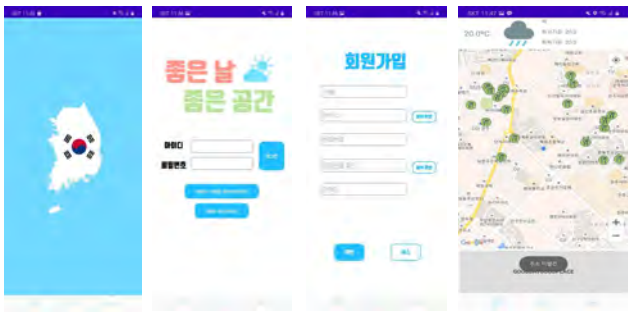
• Marking 단계
- 준비된 키 값으로 구글 서버에 Marker 요청

3. 어플 구현

2장에서 설계한 어플은 참고문헌 [1-4]를 참고하여 다음과 같이 구현하였다.

• 주요 UI

구현된 주요 UI는 그림 2와 같다.



(그림 2) 구현된 주요 UI

• 날씨정보 획득

소스를 DOM 트리 형태로 전환한 뒤 DOM tree를 순회하면서 파싱을 진행하는 Javax.xml.parsers.DocumentBuilderFactory 클라스를 활용하는 GetXMLTask 클라스를 그림 3과 같이 구현하였다.

```
@SuppressLint("NewApi")
private class GetXMLTask extends AsyncTask<String, Void, Document> {
    private Activity context;

    public GetXMLTask(Activity context) { this.context = context; }

    @Override
    protected Document doInBackground(String... urls) {

        URL url;
        try {
            url = new URL(urls[0]);
            DocumentBuilderFactory dbf = DocumentBuilderFactory
                    .newInstance();
            DocumentBuilder db;

            db = dbf.newDocumentBuilder();
            doc = db.parse(new InputSource(url.openStream()));
            doc.getDocumentElement().normalize();

        } catch (Exception e) {

            Toast.makeText(getBaseContext(), text: "Parsing Error",
                    Toast.LENGTH_SHORT).show();
        }
        return doc;
    }
```

(그림 3) GetXMLTask 클래스 활용 웹문서 파싱

• 지도 정보 획득

Google Maps API를 활용하기 위해 먼저 플랫폼에서 키를 얻어온 뒤에 해당하는 키를 참조하고, 이후에는 프로젝트 빌드에서 Google 컨텐츠를 활용하도록 적용시킨 뒤 위치 권한을 요청, 받아온 위치(location) 객체의 위도·경도를 가져오는 메서드를 그림 4와 같이 구현하였다.

```
@Override
public void onLocationResult(LocationResult locationResult) {
    super.onLocationResult(locationResult);

    List<Location> locationList = locationResult.getLocations();

    if (locationList.size() > 0) {
        location = locationList.get(locationList.size() - 1);
        //location = locationList.get(0);

        currentPosition
                = new LatLng(location.getLatitude(), location.getLongitude());

        String markerTitle = getCurrentAddress(currentPosition);
        String markerSnippet = "위도:" + String.valueOf(location.getLatitude())
                + " 경도:" + String.valueOf(location.getLongitude());

        Log.d(TAG, msg: "onLocationResult : " + markerSnippet);

        //현재 위치에 마커 생성하고 이동
        setCurrentLocation(location, markerTitle, markerSnippet);

        mCurrentLocatiion = location;
    }
}
```

(그림 4) 현재 위치를 가져오는 소스코드

지도 사용 권한을 받아오면 이후에는 Place API 도 플랫폼에서 키를 받아온 뒤 해당 키를 key 메서드에 할당해서 NRPlaces 객체에 접근한다. 이후에는 location의 위도·경도 정보를 latlng에 할당, 반경 500미터 의 RESTAURANT 타입의 한국 업체를 검색한 뒤 업체 정보 객체를 생성한다. 이에 대한 코드는 그림 5와 같은데, showPlaceInformation() 메소드에 onClickListener를 할당함으로 버튼 클릭 이벤트 발생 시 이 메소드가 동작하는 방식이다.

```
public void showPlaceInformation(LatLng location) {
    mMap.clear();

    if (previous_marker != null)
        previous_marker.clear();

    new NRPlaces.Builder()
            .listener(GoogleMap.this)
            .key("AIzaSyCfPk8_Zve8iaRv05D21bffqH_WeGShF7Y")
            .latlng(location.latitude, location.longitude)
            .radius(500)
            .type(PlaceType.RESTAURANT)
            .language( language: "ko",  countryCode: "KR")
            .build()
            .execute();
}
```

(그림 5) 현재 위치 정보에 따른 근방 업체 정보

버튼이 활성화되면 NRPlace 객체가 builder를 그림 5와 같은 조건에서 파싱하도록 설계하는데, 설계된 builder는 Place API로부터 받아오는 JSON 데이터를 분석함으로써 최종적인 인근지역 정보를 얻게 된다. 이 JSON 파일은 주변의 모든 지역정보를 가지고 있으며 그림 6의 밑줄 그어진 부분의 타입이 업종을 가리키며, 여기서는 대표적으로 RESTAURANT 타입으로 식당정보를 얻도록 했다.

(그림 6)지역 정보를 가지고 있는 데이터

• 마커 정보 획득

검색 버튼을 누르면 마커를 생성하는 코드를 그림 7과 같이 구현하였다. 여기서 마커는 음식점에 대한 정보를 제공하므로 그에 맞는 이미지로 변경했다. 안드로이드에서는 이미지 상으로 마커 사진의 크기를 변경할 수 없기 때문에 Bitmap Drawble()을 사용하게 되었다. 또한, Bitmap small marker()를 이용하여 마커 이미지를 사용자가 원하는 크기만큼 조절할 수 있으며, 코드를 추가하기 전 미리 준비해 놓은 foodsmarker 이미지(.png)를 drawble에 추가시키면 현재 위치에서 변경된 마커 이미지로 주변의 음식점 정보를 보이게 된다.

```
BitmapDrawable bitmapdraw = (BitmapDrawable) getResources().getDrawable(R.drawable.foodsmarker);
Bitmap b = bitmapdraw.getBitmap();
Bitmap smallMarker = Bitmap.createScaledBitmap(b,  dstWidth: 65,  dstHeight: 65,  filter: false);
markerOptions.icon(BitmapDescriptorFactory.fromBitmap(smallMarker));
```

(그림 7) 비트맵을 활용한 마커 이미지 변경

4. 결 론

이번 프로젝트로 기대되는 효과는 사용자의 위치 정보를 기반으로 주위의 날씨와 음식점을 동시에 제공함으로써 이용하려는 목적과 날씨 및 취향에 맞는 음식점을 빠른 시간 내에 찾을 수 있다는 편리함을 제공한다는 점이다. 검색 장소에 대한 카테고리 선택 기능을 구현하지 못했으나 앞으로 이 기능을 추가하여 날씨에 알맞은 장소 검색 기능을 제공할 예정이다.

참고문헌

[1] 천인국, 그림으로 쉽게 설명하는 안드로이드 프로그래밍, 생능출판사, 2018

[2] 정재곤, DO it 안드로이드 앱 프로그래밍(전면개정 7판), 이지스퍼블리싱, 2020

[3] 강성윤, 깡샘의 안드로이드 프로그래밍, 루비페이퍼, 2019

[4] 우재남, 박길식, Android Studio를 활용한 안드로이드 프로그래밍 5판, 한빛미디어, 2020