



## 졸업작품 최종보고서

### 반려견 동반 가능장소 검색 앱

지도 교수 : 서 장 원 교수님

제 출 일 : 2021.12.13

제 출 자 : 1906097 최예진

# 목차

요 약

I. 개요 .....	1
I-1 참여 인원	
I-2 프로젝트 내용(개발 필요성)	
I-3 기대 효과	
II. 요구 분석 .....	2
II-1 기능적 요구	
II-1-1 인터페이스 요구	
II-1-2 기능 요구	
II-2 비기능적 요구	
III. 설계 제작 .....	3
III-1 Loading Activity (로딩 화면)	
III-2 Login, Register Activity (로그인, 회원가입)	
2-1 LoginRequest, Login.php	
2-2 RegisterRequest, Register.php	
III-3 Main_Activity (메인 액티비티)	
III-4 Home_fragments (홈 화면 프라그먼트)	
4-1 SliderAdapter (자동 슬라이더 뷰)	
4-2 jsonArrayRequest, item, adapterphone2 (리사이클러뷰)	
III-5 Cafe_Activity, Park_Activity, Restirant_Activity (장소 갈래)	
5-1 item, adapterphone1	
5-2 shopinfo_Activity	
III-6 Search_Activity (검색)	
6-1 Search_Filter	
6-2 SeacrhRequest_filter	
III-7 Fragment_Map (지도)	
III-8 Fragment_Like (찜)	
III-9 Fragment_Mypage (마이페이지)	
9-1 DogInsert	
9-2 DogRequest	
III-10 CustmizeActivity (맞춤)	
10-1 dogProfile_item, CustomizedAdapter	
10-2 CustomizedRequest	
10-3 CustomResultRequest, Custom.php	
10-4 CustomResultRequest	

III-11 DB 설계 (테이블 구조 및 제약 조건)	
11-1 Member 테이블	
11-2 Place 테이블	
11-3 Dog 테이블	
11-4 Liked 테이블	
IV. 테스트 [시험, 성능 평가]	4
IV-1 테스트 방법	
IV-2 테스트 과정	
IV-3 테스트 결과	

## 요 약

### 제 목 : 워드펫 (반려견 동반 장소 검색 앱)

농림축산식품부의 '2019년 동물보호 국민의식조사' 에 따르면 반려동물의 수는 약 1,000만 마리이며, 양육 인구 수는 1,500만 명을 돌파 한 것으로 밝혀졌다.

양육 인구수가 증가하면서 관련 산업의 수요도 증가하고 있다. 해당 기관의 발표에 따르면 2021년 반려동물 산업의 규모는 3조 7000억원 이다. 개중에서는 반려동물 관련 IT 서비스 산업 또한 포함되어 있다.

이 반려동물 중 가장 많은 수를 차지하는 것은 반려견이다(589만 마리). 반려견의 경우 양육시 산책이 필수적이기 때문에 산책에 도움을 주는 어플리케이션이 시장에 다양하게 나와 있다. 반려견과 동반 가능한 장소를 검색하는 기능이 일반적인데, 일각에서는 이 기능이 부족하다는 의견이 나오고 있다.

일반적으로 동반 장소는 입장이 가능한 견종, 크기, 몸무게, 입마개 여부 등의 조건을 제시한다. 제시한 조건에 부합하지 않으면 입장이 거부될 수 있다. 산책 관련 어플리케이션에선 이러한 입장 조건을 가게 정보에 포함하여 보여준다. 앱 사용자는 입장 조건과 장소의 위치를 확인하고, 반려견과 산책을 나가게 된다. 반려견과의 산책시 장소의 위치 와 입장 조건이 가장 중요하다는 것이다.

현재 시장에 나온 어플리케이션의 경우(\*피,하\*독,반\*\*활) 검색시 적용 가능한 요소는 현위치, 장소 래래, 장소 이름, 지역 등 이다. 입장 조건은 가게 정보에서 확인할 수 있다. 중요한 **입장 조건으로 검색**은 불가능했다. 때문에 우리는 입장 조건 및 지역으로 검색이 가능한 앱을 제작하기로 하였다.

앱의 기능은 크게 검색, 강아지 조건 입력, 강아지 조건 맞춤 검색, 동반 장소 정보제공, 지도, 회원가입/로그인, 회원페이지, 찜한 장소모아보기 로 구성되어 있다.

각 기능들은 기본적으로 서버와의 통신을 통해 구현 된다. MySQL로 DB를 관리하고, 닷홈 서버와 PHP 를 사용하였다. Andoridstudio JAVA 로 코드를 작성하였으며 다양한 라이브러리를 사용하였는데, Volley와 Glade 라이브러리를 가장 많이 사용하였다.

앱의 정체성이자 차별점인 입장 조건 검색은 구현을 완료하였으나 앱의 UI 디자인과 사용되는 데이터, 보안, 일부 기능 등에서 부족함이 있었다. 아쉬움이 남지만 앱을 제작하는 과정에서 java와 Androidstudio, PHP, MySQL을 비교적 심도있게 공부할 수 있었다. 1인 개발이 아닌 팀 프로젝트로 개발을 진행하였기 때문에 협업의 중요성 또한 깨달을 수 있는 소중한 경험이었다.

## 개요

### I-1 참여 인원

- 팀장

소속	분반	학번	성명
컴퓨터소프트웨어학과	C2	1906097	최예진

- 팀원

소속	분반	학번	성명
컴퓨터소프트웨어학과	C2	1906097	구가은
컴퓨터소프트웨어학과	D2	1906099	김민주
컴퓨터소프트웨어학과	C2	1906101	이보미
컴퓨터소프트웨어학과	D2	1906106	권지현
컴퓨터소프트웨어학과	C2	1906121	전희선

### I-2 프로젝트 내용(개발 필요성)

농촌 경제 연구원에 따르면 국내 반려동물 수는 1,000만 마리를 넘어섰으며, 양육 인구수는 1,500만 명을 돌파한 것으로 알려졌다. 반면, 반려동물 관련 서비스 산업은 이러한 증가 추세를 따라오지 못하고 있다. 따라서 불편함을 느끼는 반려동물 양육 인구 또한 증가하게 되었다.

이러한 불편함을 스마트폰 애플리케이션이라는 매체가 가진 대중성을 통해 해소 할 수 있을 것으로 생각한다.

### I-3 기대 효과

- 반려동물과 사용자가 함께 할 수 있는 장소에 대하여 나의 강아지 프로필을 등록하거나 검색 필터를 설정함으로써 내가 원하는 장소를 빠르고 정확하게 검색할 수 있다.

- 해당 가게에 대한 정보를 제공받을 수 있으며, 지도를 통하여 가게의 위치 또한 편리하게 찾아볼 수 있다.

- 따라서 반려동물 양육 인구들이 반려동물과 함께 할 수 있는 여러 장소들을 찾는 데에 있어 어려움을 줄여주고 반려견과의 추억을 쌓는데 도움을 줄 수 있다.

## 요 구 분 석

### II-1 기능적 요구

위드펫 앱의 목적에 따른 기능적/비기능적 요구 사항을 분석한다.

#### 1-1 인터페이스 요구

- |          |          |                    |
|----------|----------|--------------------|
| 1. 시작화면  | 4. 장소 검색 | 7. 탈퇴 안내 및 컨텍스트 예제 |
| 2. 회원 가입 | 5. 홈 화면  | 8. 장소 상세 페이지       |
| 3. 지도    | 6. 마이페이지 |                    |

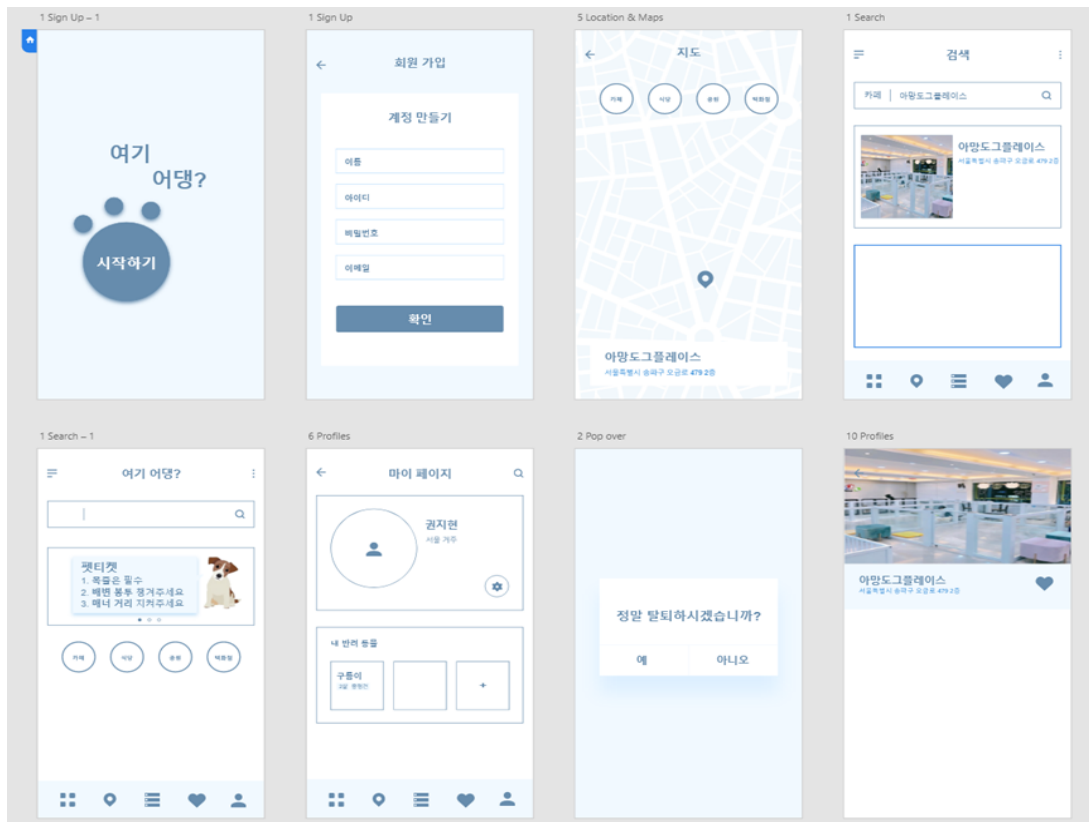


그림 인터페이스 요구

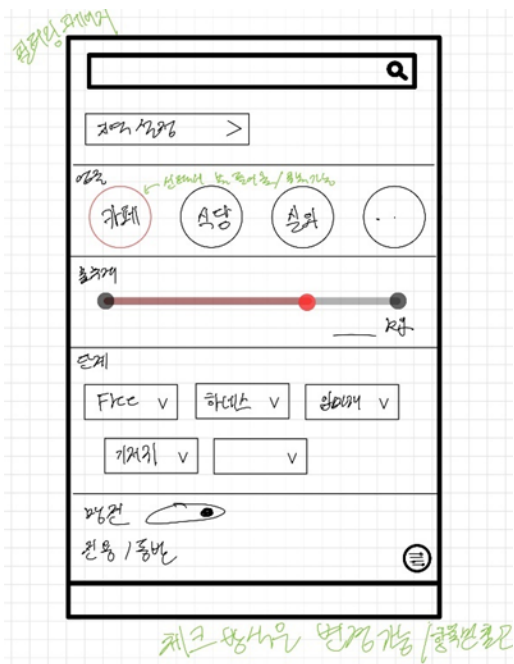


그림 9. 검색 조건 필터



그림 10. 맞춤 검색

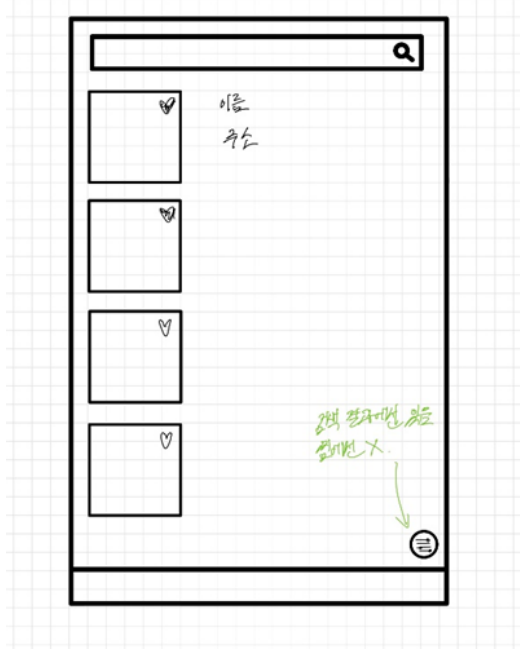


그림 11. 찜 리스트

## II-1-2 기능 요구

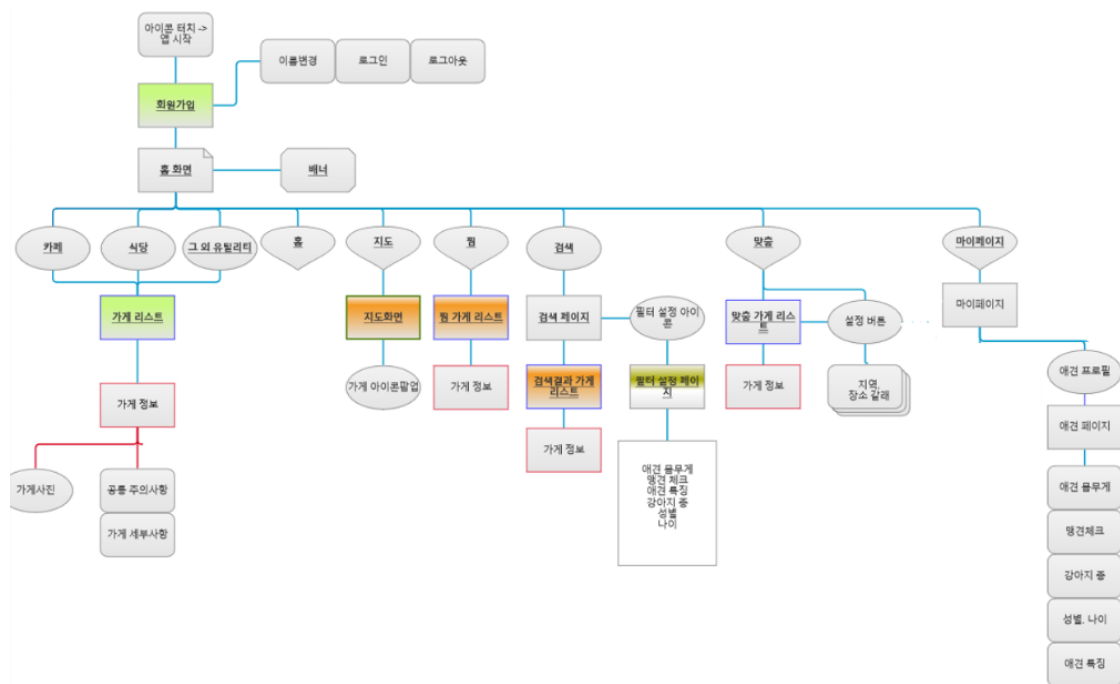


표 1 - 페이지 흐름도

1. 회원가입: 사용자는 이름, 아이디, 비밀번호 입력을 통해 회원가입을 할 수 있다. 이 정보는 서버에 저장되어 관리자가 관리할 수 있다.
2. 로그인: 사용자는 회원가입을 할 때 입력한 정보로 로그인을 할 수 있다. 서버에 저장된 정보와 대조하여 로그인한다.
3. 반려동물 관리: 사용자는 자신의 반려동물 이름, 몸무게, 망건 여부, 성별, 나이, 특징을 저장할 수 있다. (여러 마리 등록 가능)
4. 검색: 사용자는 카테고리, 입장 제한 몸무게, 지역, 전용/동반 여부, 망건 여부 등에 따라 장소를 검색할 수 있다.
5. 맞춤 검색: 기능 요구 '3. 반려동물 관리'에서 등록한 반려동물의 조건을 가져와 검색마다 조건을 입력하지 않아도 필터 설정이 되어 검색할 수 있다. 단, 위치, 카테고리 등은 반려동물 조건이 아니므로 자동 맞춤 되지 않는다.
6. 지도 : 지도 화면은 사용자의 현재 위치 중심으로 나타나며 지도 위의 마커를 통해 장소들을 파악 할 수 있다.
7. 찜: 검색 결과에서 하트를 누르면 찜 리스트에 등록되며 찜 탭화면에서 모아져 볼 수 있다.
8. 장소 상세 페이지: 검색 결과에서 장소를 누르면 해당 장소의 상세 페이지로 이동한다. 해당 장소의 정보를 확인할 수 있다.



## II-2 비기능적 요구 -----

- 반응 시간, 처리 소요 시간은 3초 이내로
- 처리율은 1건/ 3초
- 3.2 H/W 요구(기억 장치 규모, 통신수용도)
- HDD 사용량 최대 200M
- 트래픽 사용량 최대 300M
- 예외 조건 및 이의 처리
- 시스템이 기능을 수행하지 못하면, 일정한 횟수로 반복 시도하고 그래도 불가능하면 오류 메시지를 출력한다. 사용자가 오류 메시지를 인식하고 시스템을 마친다.
- 자원, 인력에 대한 제약 조건
- 프로젝트인원이 각자 월20시간 이상씩 투자
- COVID19로 인해 대면 회의가 불가하여 온라인 통화를 이용한 비대면 회의 혹은 소규모 회의로 실시

### III-1 Loading Activity (로딩 화면)-----



그림 1-1.로딩 페이지

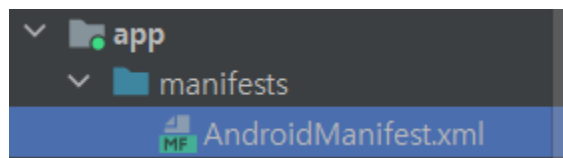


그림 1-2.안드로이드 매니페스트 위치

Loading Activity.java

```
public class Activity_Loading extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_loading);

        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(intent);
                finish();
            }
        }, 3000);

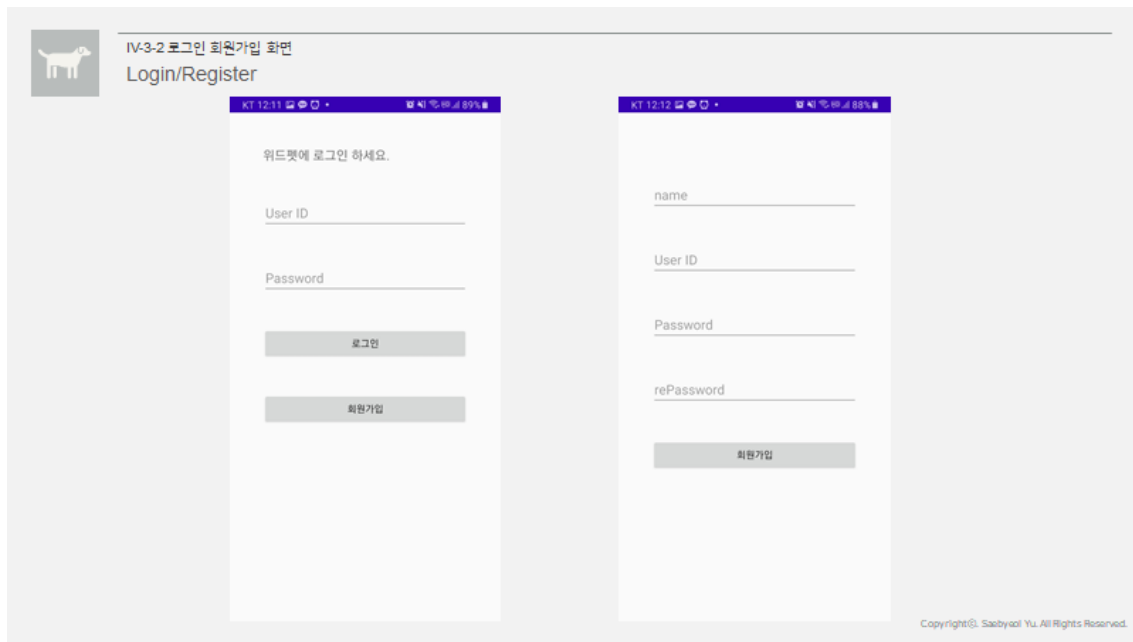
        @Override
        protected void onPause() {
            super.onPause();
            finish();
        }
    }
}
```

위는 앱의 첫 번째 페이지인 로딩 화면의 자바 클래스이다.

다음 사진 1-2의 위치에 있는 AndroidManifest.xml 에서 Intent Filter(1) 를 설정하여, 앱 시작 시 첫 화면을 사진 1-1의 로딩화면이 뜰 수있도록 한다.

위의 자바 클래스 코드를 통해, 스레드 및 인텐트 처리를 하여 3000밀리초(3초) 후 LoginActivity가 실행될 수 있도록 한다.

### III-2 Login , Register Activity (로그인, 회원가입)-----



#### Login Activity.java

```
public class LoginActivity extends AppCompatActivity {
    private EditText et_id, et_pass;
    private Button btn_login, btn_register;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        et_id = findViewById(R.id.userid1);
        et_pass = findViewById(R.id.password1);
        btn_login = findViewById(R.id.btnlogin);
        btn_register = findViewById(R.id.btnregs);

        // 회원가입 버튼을 클릭 시 수행
        btn_register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
                startActivity(intent);
            }
        });

        btn_login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                // EditText에 현재 입력되어있는 값을get(가져온다)해온다.
                String ID = et_id.getText().toString();
                String PW = et_pass.getText().toString();

                Response.Listener<String> responseListener = new Response.Listener<String>() {
```

```

@Override
public void onResponse(String response) {
    try {
        // TODO : 인코딩 문제때문에 한글DB인 경우 로그인 불가
        System.out.println("hongchul" + response);
        JSONObject jsonObject = new JSONObject(response);
        boolean success = jsonObject.getBoolean("success");
        if (success) { // 로그인에 성공한 경우
            String ID = jsonObject.getString("ID");
            String PW = jsonObject.getString("PW");

            Toast.makeText(getApplicationContext(),"로그인에
            성공하였습니다.",Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(LoginActivity.this, MainActivity.class);
            intent.putExtra("ID", ID);
            intent.putExtra("PW", PW);
            startActivity(intent);
        } else { // 로그인에 실패한 경우
            Toast.makeText(getApplicationContext(),"로그인에
            실패하였습니다.",Toast.LENGTH_SHORT).show();
            return;
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
};

LoginRequest loginRequest = new LoginRequest(ID, PW, responseListener);
RequestQueue queue = Volley.newRequestQueue(LoginActivity.this);
queue.add(loginRequest);

});

}

```

위 코드는 로그인 화면의 자바 클래스 코드이다. 사용자가 EditText 에 입력한 값을 저장한 뒤, 해당 값을 2-1 LoginRequest 클래스를 이용해 서버와 통신하여 로그인 가능 여부를 확인한다. 각각 로그인 성공과 실패에 따라 Toast 메시지로 사용자에게 성공 여부를 알려 준다.

로딩화면 다음으로 사용자에게 보여지는 화면이며, 하단의 ‘회원 가입하기’ 버튼을 통해 회원가입 화면(Register\_Activity)으로 이동한다.

Register Activity java

```

public class RegisterActivity extends AppCompatActivity {

    private EditText et_id, et_pass, et_name;
    private Button btn_register;

    @Override
    protected void onCreate(Bundle savedInstanceState) { // 액티비티 시작시 처음으로
        실행되는 생명주기!
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        // 아이디 값 찾아주기
        et_id = findViewById(R.id.userid);
        et_pass = findViewById(R.id.password);
        et_name = findViewById(R.id.username);

        // 회원가입 버튼 클릭 시 수행
    }
}

```

```

btn_register = findViewById(R.id.btnsignup);
btn_register.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
// EditText에 현재 입력되어있는 값을get(가져온다)해온다.
String ID = et_id.getText().toString();
String PW = et_pass.getText().toString();
String Name = et_name.getText().toString();

Response.Listener<String> responseListener = new Response.Listener<String>() {
@Override
public void onResponse(String response) {
try {
JSONObject jsonObject = new JSONObject(response);
boolean success = jsonObject.getBoolean("success");
if (success) { // 회원등록에 성공한 경우
Toast.makeText(getApplicationContext(),"회원 등록에
성공하였습니다.",Toast.LENGTH_SHORT).show();
Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
startActivity(intent);
} else { // 회원등록에 실패한 경우
Toast.makeText(getApplicationContext(),"회원 등록에
실패하였습니다.",Toast.LENGTH_SHORT).show();
return;
}
} catch (JSONException e) {
e.printStackTrace();
}
}
};
// 서버로Volley를 이용해서 요청을 함.
RegisterRequest registerRequest = new RegisterRequest( ID, PW, Name,
responseListener);
RequestQueue queue = Volley.newRequestQueue(RegisterActivity.this);
queue.add(registerRequest);
});
}
}

```

위 코드는 회원가입 화면의 자바 클래스 코드이다. 로그인 기능과 유사하게, 사용자가 EditText에 입력한 값을 저장한 뒤, 해당 값을 2-2 RegisterRequest 클래스를 이용해 서버와 통신하나, 사용자의 정보가 서버에 저장되어 다음 로그인 시 해당 컬럼과 비교하여 로그인된다는 점이 다르다. 로그인 가능 여부를 확인한다. 각각 로그인 성공과 실패에 따라 Toast 메시지로 사용자에게 성공 여부를 알려 준다.

로딩화면 다음으로 사용자에게 보여지는 화면이며, 하단의 ‘회원 가입하기’ 버튼을 통해 회원가입 화면(Register\_Activity)으로 이동한다.

## 2-1 LoginRequest, Login.php-----

LoginRequest.java

```
//로그인 요청을 함
public class LoginRequest extends StringRequest {

    // 서버URL 설정( PHP 파일 연동)
    final static private String URL = "http://heremong.dothome.co.kr/Login.php";
    private Map<String, String> map;

    public LoginRequest(String ID, String PW, Response.Listener<String> listener) {
        super(Method.POST, URL, listener, null);

        map = new HashMap<>();
        map.put("ID", ID);
        map.put("PW", PW);
    }

    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        return map;
    }
}
```

III-2 Login Activity에서 서버와의 통신을 위해 사용하는 LoginRequest 클래스이다. 앱의 기본적인 통신은 Volley 라이브러리를 이용한다. 해당 라이브러리의 StringRequest 클래스를 상속받은 클래스로, String 데이터 타입의 데이터를 서버에서 응답받을 수 있다. map 메서드를 이용하여 III-2 에서 저장한 값을 서버로 보낸다.

이후 URL = "http://heremong.dothome.co.kr/Login.php";

php에서 쿼리 질의를 한 후, String 값을 반환한다.

III-2 의 boolean success = jsonObject.getBoolean("success"); 함수로 로그인 가능 여부를 판단한다.

## Login.php

```
<?php
$con = mysqli_connect("localhost", "heremong", "////////", "heremong");
mysqli_query($con,'SET NAMES utf8');

$ID = $_POST["ID"];
$PW = $_POST["PW"];

$stmt = mysqli_prepare($con, "SELECT * FROM Member WHERE ID = ? AND PW = ?");
mysqli_stmt_bind_param($stmt, "ss", $ID, $PW);
mysqli_stmt_execute($stmt);

mysqli_stmt_store_result($stmt);
mysqli_stmt_bind_result($stmt, $ID, $PW, $Name);

$response = array();
$response["success"] = false;

while(mysqli_stmt_fetch($stmt)) {
    $response["success"] = true;
    $response["ID"] = $ID;
    $response["PW"] = $PW;
    $response["Name"] = $Name;
}

echo json_encode($response);

?>
```

상기 **LoginRequest** 에서 사용된 전송 방식은 Post 방식이다. 서버 URL에 데이터가 노출되지 않아 Post 방식을 사용하였다.

SELECT 문을 이용하여 받은 데이터(ID, PW)를 질의하고, 존재하는 경우 success를 반환하고, 해당 ID와 PW가 일치하지 않거나 존재하지 않는 경우

```
    $response["success"] = false;
```

코드에 따라 false 값을 반환한다.

## 2-2 RegisterRequest, register.php-----

RegisterRequest.java

```
public class RegisterRequest extends StringRequest {  
    // 서버URL 설정( PHP 파일 연동)  
    final static private String URL = "http://heremong.dothome.co.kr/Register.php";  
    private Map<String, String> map;  
  
    public RegisterRequest(String ID, String PW, String Name, Response.Listener<String>  
listener) {  
        super(Method.POST, URL, listener, null);  
  
        map = new HashMap<>();  
        map.put("ID", ID);  
        map.put("PW", PW);  
        map.put("Name", Name);  
    }  
  
    @Override  
    protected Map<String, String> getParams() throws AuthFailureError {  
        return map;  
    }  
}
```

III-2 Register Activity에서 서버와의 통신을 위해 사용하는 RegisterRequest 클래스이다. 2-1 LoginRequest와 기본 구조가 같으나, 통신하는 URL의 php 주소가 달라 질의에 차이가 있다.

register.php

```
<?php  
$con = mysqli_connect("localhost", "heremong", "////////", "heremong");  
mysqli_query($con, 'SET NAMES utf8');  
  
$ID = $_POST["ID"];  
$PW = $_POST["PW"];  
$Name = $_POST["Name"];  
  
$statement = mysqli_prepare($con, "INSERT INTO Member VALUES (?,?,?)");  
mysqli_stmt_bind_param($statement, "sss", $ID, $PW, $Name);  
mysqli_stmt_execute($statement);  
  
$response = array();  
$response["success"] = true;  
  
echo json_encode($response);  
?>
```

사용자가 회원가입 시 입력하는 정보인 ID, PW, Name 을 앱으로 부터 Post 타입으로 받게 되면, Member 테이블 에 Inset 문으로 데이터를 넣는다.

기본키/고유키/외래키 등이 Member 테이블에 설정되어 있어, 이미 존재하는 ID의 경우 회원가입을 할 수 없다.



### III-3 Main\_Activity-----

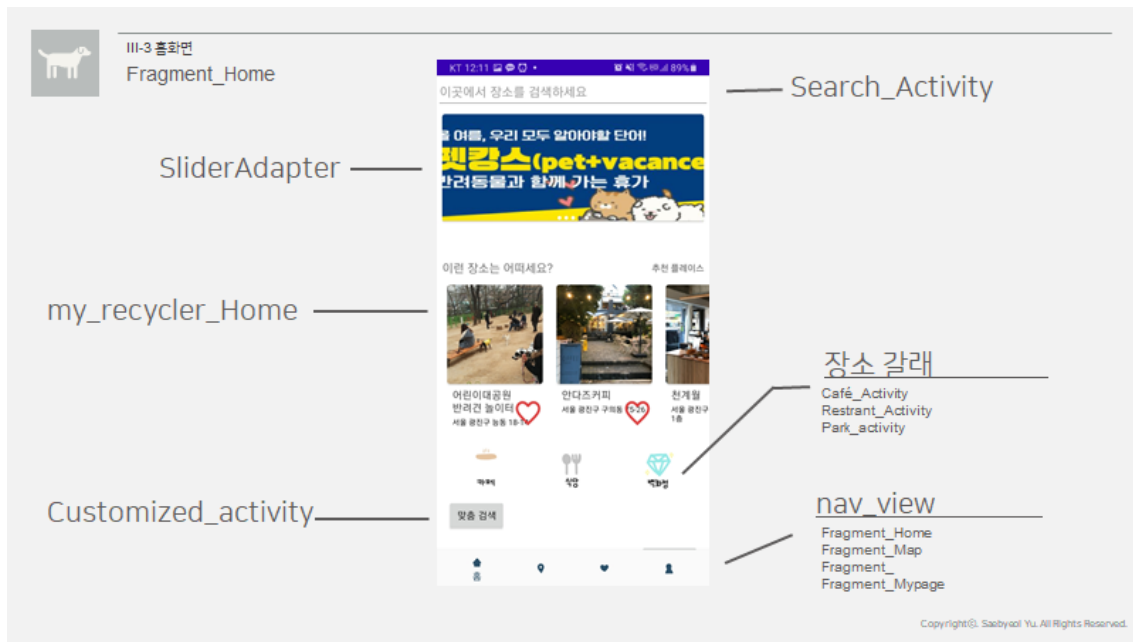


그림 3-1

사용자가 로그인을 성공한 후 보여지게 되는 화면이다. 각 요소는 그림 3-1의 설명대로 해당 목차로 설명한다.

Main\_Activity.java

```
public class MainActivity extends AppCompatActivity {
    private BottomNavigationView mBottomNV;
    private Toolbar toolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mBottomNV = findViewById(R.id.nav_view); //하단 네비바id
        mBottomNV.setOnNavigationItemSelectedListener(new
        BottomNavigationView.OnNavigationItemSelectedListener() { //NavigationItemSelected
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                BottomNavigate(menuItem.getItemId());
            }
        });

        mBottomNV.setSelectedItemId(R.id.navigation_Home);
    }

    //BottomNavigation 프라그먼트 변경
    private void BottomNavigate(int id) {
        String tag = String.valueOf(id); //tag는 고유명사로id의string 값

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

```

//이곳에서 기본 프래그먼트 매니저와 트랜잭션을 설정함.
Fragment currentFragment = fragmentManager.getPrimaryNavigationFragment();
if (currentFragment != null) {
    fragmentTransaction.hide(currentFragment);
}

Fragment fragment = fragmentManager.findFragmentByTag(tag);

if (fragment == null) {
    if (id == R.id.navigation_Home) { //홈화면 아이콘 터치
        fragment = new Fragment_Home();
    } else if (id == R.id.navigation_Map){ //지도화면
        fragment = new Fragment_Map();
    }

    else if (id == R.id.navigation_Like){ //찜 화면

        fragment = new Fragment_Like();
    }else if (id == R.id.navigation_MyPage){ //마이페이지 화면

        fragment = new Fragment_Mypage();
    }

    fragmentTransaction.add(R.id.content_layout, fragment, tag);
} else {
    fragmentTransaction.show(fragment);
}

fragmentTransaction.setPrimaryNavigationFragment(fragment);
fragmentTransaction.setReorderingAllowed(true);
fragmentTransaction.commitNow();
}

@Override
public void onBackPressed() {
}

@Override
protected void onDestroy() {
    super.onDestroy();
}
}

```

우선 **Main\_Activity**에서 사용되는 기능 중 설명할 것은 크게 2가지이다.

- 1.Fragment
- 2.BottomNavigate

### 1. Fragment

프래그먼트는 앱의 화면인 Activity 위에 부분적으로 그려지는 화면이다. 별도의 xml과 자바 클래스를 가져 코드의 모듈화 및 재사용성이 좋다. 다른 특성이 많지만 중요한 것은 액티비티 변경 없이 화면을 전환 할 수 있다는 점과 별도의 자바 클래스를 가질 수 있다는 점이 유용하여 위드젯에 사용하게 되었다.

Main\_Activity에의 xml에 프래그먼트가 들어갈 칸을 마련한 뒤,

```
fragmentTransaction.add(R.id.content_layout, fragment, tag);
```

위의 코드로 필요한 프래그먼트가 들어가도록 한다.

특히, 위드젯은 다양한 페이지를 구현해야 하므로, **Main\_Activity** 위에서 기능 대부분을 구동할 수 있도록 제작하였다.

Main\_Activity 위에서 프라그먼트를 변경하는 것은 BottomNavigate를 이용하였다.

## 2.BottomNavigate

안드로이드 스튜디오는 빌드 시 Gradle 을 이용한다. 이곳에 필요한 라이브러리를 추가하여 관리 할 수 있다. 위의 Volley 또한 이곳에 코드를 추가하여 사용하였다.

BottomNavigate 의 gradle코드는 다음과 같다.

```
implementation 'com.google.android.material:material:1.0.0'
```

이 라이브러리는 대략적으로 하단네비게이션 바의 외적인 이미지 및 효과,

onNavigationItemSelectedListener 메서드 등을 제공하여서 한 액티비티 위에서 프라그먼트 이동이 용이하도록 한다.

BottomNavigate함수를 간단히 설명하자면, 우선 각 하단 바의 아이콘(홈, 지도, 찜, 마이페이지)이 선택되면 각 아이콘의 ID를 반환한다. 반환된 ID를 매개변수로 위 BottomNavigate 함수가 작동되는데, if 문으로 ID를 대조하며 해당 Id에 맞는 프라그먼트의 태그 값을 반환하고, 반환한 태그 값을 위의 fragmenttransaction.add로 액티비티 위에 적절한 프라그먼트가 띄워지도록 한다.

또한 하단 네비게이션 바는 별도의 xml 파일이 있는데. 위치는 사진 3-2 와 같다.

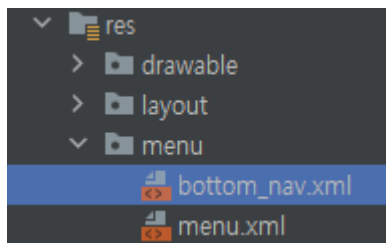


그림 3-2

### III-4 Home\_fragemnt (홈 화면 프라그먼트) -----

해당 코드는 양이 많아 세부 코드는 하위 목차에서 설명하도록 한다.

사용자가 로딩 화면을 지나, 로그인을 성공한 뒤 보이는 **Main\_Activity** 위에 가장 처음 그려지는 프라그먼트이다.

Home\_fragemnt-SliderAdapter.java

```
public class SliderAdapter extends SliderViewAdapter<SliderAdapter.Holder>{
    int[] images;

    public SliderAdapter(int[] images){
        this.images = images;
    }

    @Override
    public Holder onCreateViewHolder(ViewGroup parent) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.slider_item,parent,false);
        return new Holder(view);
    }

    @Override
    public void onBindViewHolder(Holder viewHolder, int position) {
        viewHolder.imageView.setImageResource(images[position]);
    }

    @Override
    public int getCount() {
        return images.length;
    }

    public class Holder extends SliderViewAdapter.ViewHolder{
        ImageView imageView;

        public Holder(View itemView){
            super(itemView);
            imageView = itemView.findViewById(R.id.image_view);
        }
    }
}
```

Gardle

```
implementation 'com.github.smarteist:autoimageslider:1.4.0' //이미지 슬라이더 배너
```

상단 슬라이더뷰에 필요한 코드는 다음과 같다. **autoimageslider**라는 오픈 소스를 사용하여 구현하였다. 슬라이더뷰의 이미지는 서버와의 통신 없이, drawable 폴더의 이미지를 사용하여, 앱 자체에 저장되어 있다.

#### 4-2 jsonArrayRequest, item, adapterphone2-----

위드젯의 리사이클러뷰는 서버로부터 받은 데이터를 내용으로 그려진다. 때문에 통신리퀘스트 부분, 데이터 저장 및 개별 아이템뷰 부분, 어댑터 부분으로 나뉜다.

Home\_fragment-jsonArrayRequest.java

```
new Thread(new Runnable() {
boolean isRun = true;

@Override
public void run() {
while ((isRun)){

handler1.post(new Runnable() {
@Override
public void run() {

//서버주소
String serverUrl="http://heremong.dothome.co.kr/PlaceList2.php";

JSONArrayRequest jsonArrayRequest= new JSONArrayRequest(Request.Method.POST,
serverUrl, null, new Response.Listener<JSONArray>()

@Override
public void onResponse(JSONArray response) {

items2.clear();
adapter2.notifyDataSetChanged();

try {

for(int i=0;i<response.length();i++){
JSONObject jsonObject= response.getJSONObject(i);

int Place_id= Integer.parseInt(jsonObject.getString("Place_id"));
String P_name=jsonObject.getString("P_name");
String P_address=jsonObject.getString("P_address");
String P_image=jsonObject.getString("P_image");

items2.add(0,new Item(Place_id, P_name, P_address, P_image));
}
} catch (JSONException e) {e.printStackTrace();}

}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error) {
}
});

RequestQueue requestQueue=
Volley.newRequestQueue(getActivity().getApplicationContext());
requestQueue.add(jsonArrayRequest);

}
});
try {
Thread.sleep(1000);
}catch (Exception e){ }
```

```
}  
}).start();
```

**Home\_fragment.java** 코드 내부에 있는, 앞서 설명한 통신 리퀘스트 부분에 해당하는 코드이다.

Jsonarrayrequest를 상속받아 jsonarray 타입의 데이터를 응답받는 코드이다. array인 만큼 해당 테이블(가게리스트)의 모든 데이터를 받아오는데, 이를 가게 id 기준으로 잘라 각각의 가게가 리사이클러뷰 아이템 하나를 차지하도록 한다. 가게의 사진, 주소 등등을 받고, 하단의 item 클래스를 호출하여 데이터를 잘라 넣어둔다.

또한 **Main\_Activity** 위의 **Home\_fragment**에서 구동되기 때문에, 메인 액티비티의 스크린에서 질의를 하게 되는데, 답변이 지연되어 구동이 되지 않는 경우가 있으므로 별도의 스크린으로 구현하였다.

item.java

```
public class Item {  
  
    int Place_id;  
    String P_name;  
    String P_address;  
    String P_image;  
  
    public Item() {  
    }  
  
    public Item(int Place_id, String P_name, String P_address, String P_image) {  
        this.Place_id = Place_id;  
        this.P_name = P_name;  
        this.P_address = P_address;  
        this.P_image = P_image;  
    }  
  
    public int getPlace_id() {  
        return Place_id;  
    }  
  
    public String getP_name() {  
        return P_name;  
    }  
  
    public void setP_name(String P_name) {  
        this.P_name = P_name;  
    }  
  
    public String getP_address() {  
        return P_address;  
    }  
  
    public void setP_address(String P_address) {  
        this.P_address = P_address;  
    }  
  
    public String getP_image() {  
        return P_image;  
    }  
  
    public void setP_image(String P_image) {  
        this.P_image = P_image;}}  
}
```

상기 jsonarrayrequest 코드에서 설명한 대로, 위에서 잘라준 가게 데이터의 총 개수만큼 위의 **item** 클래스를 호출해 리사이클러뷰 아이템에 들어가기 알맞게 해준다. 이후 해당 데이터는 하단의 **adapterphone2** 코드에서 배열로 다시 묶여 리사이클러뷰로 그려지게 된다.

adapterphone2.java

```
public class adapterphone2 extends RecyclerView.Adapter<RecyclerView.ViewHolder> {
    Context context;
    ArrayList<Item> phoneLaocations; //아이템을 넣은 배열
    LikeButton likeButton;
    ToggleButton like_toggle;

    public adapterphone2(Context context, ArrayList<Item> phoneLaocations) {
        this.phoneLaocations = phoneLaocations;
        this.context = context;
    }

    //수평 리사이클러뷰 어댑터
    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        LayoutInflater inflater= LayoutInflater.from(context);

        View view =
        LayoutInflater.from(parent.getContext()).inflate(R.layout.phonerecyclercard_2, parent,
        false);

        VH holder = new VH(view);
        return holder;
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
        VH vh= (VH) holder;

        Item phonehelper = phoneLaocations.get(position);
        vh.tvName.setText(phonehelper.getP_name());
        vh.tvMsg.setText(phonehelper.getP_address());

        Glide.with(context).load(phonehelper.getP_image()).into(vh.iv);

        //세부페이지를 위해 추가한부분
        vh.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String tvName = vh.tvName.getText().toString();
                String tvMsg = vh.tvMsg.getText().toString();
                //받은 이미지뷰.
                ImageView imageView = vh.iv;

                Intent intent;
                intent = new Intent(context, ShopInfo_Activity.class);
                intent.putExtra("tvName", tvName);
                intent.putExtra("tvMsg", tvMsg);

                BitmapDrawable drawable = (BitmapDrawable) imageView.getDrawable();
                Bitmap bitmap = drawable.getBitmap();
```

```

ByteArrayOutputStream stream = new ByteArrayOutputStream();
bitmap.compress(Bitmap.CompressFormat.JPEG, 100, stream);
byte[] byteArray = stream.toByteArray();
intent.putExtra("tvImage",byteArray);

context.startActivity(intent);
});
}

@Override
public int getItemCount(){
return (phoneLaocations != null ? phoneLaocations.size() : 0);
}

@Override
public long getItemId(int position) {
return position;
}

class VH extends RecyclerView.ViewHolder implements View.OnClickListener{

TextView tvName;
TextView tvMsg;
ImageView iv;
ToggleButton like_toggle;

public VH(@NonNull View itemView) {
super(itemView);

tvName=itemView.findViewById(R.id.tv_name2);
tvMsg=itemView.findViewById(R.id.tv_msg2);
iv=itemView.findViewById(R.id.iv2);
}

@Override
public void onClick(View v) {

int clickedPosition = getAdapterPosition();
}
}

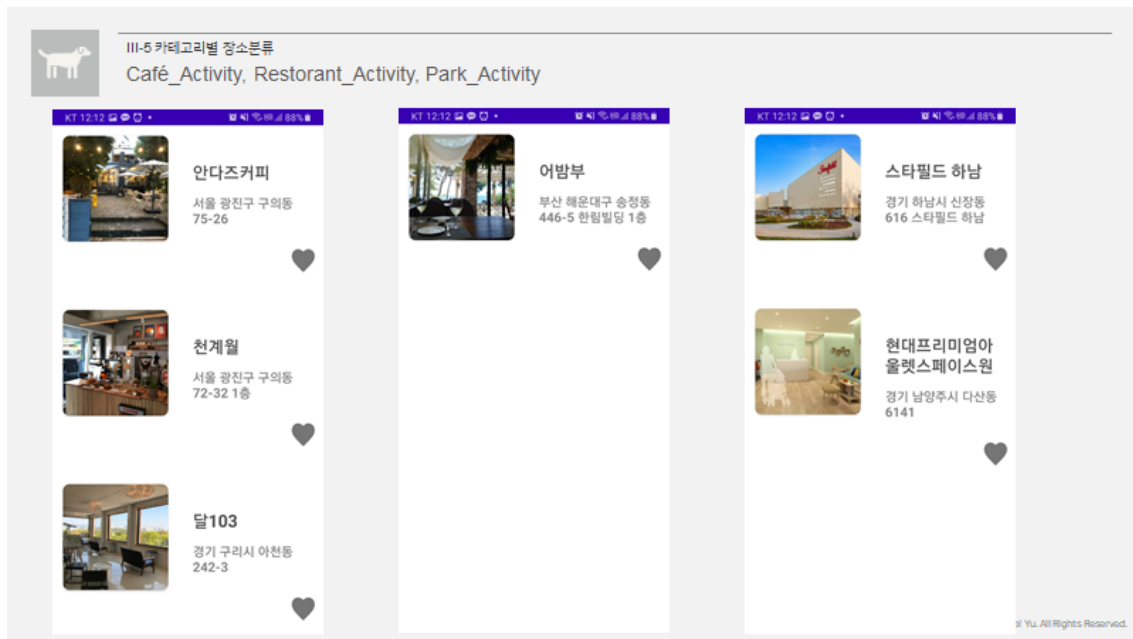
```

리사이클러뷰의 Adapter에 해당하는 코드이다.  
리사이클러뷰의 방향(수평,수직)에 따라 Adapter를 두었으며, 위 **adapterphone2**는 수평 리사이클러뷰에 사용된다. 방향에 따라 사용하는 아이템 xml이 다르기 때문이다. 위 itme 클래스를 배열로 하는 **ArrayList<Item> phoneLaocations**를 데이터로 넣어 뷰를 그린다. 이때 뷰홀더를 사용한다.

또한 **vh.itemView.setOnClickListener(new View.OnClickListener()** 부분을 추가하여, 각 가게 아이템 클릭시 하단에 설명될 ShopInfo 액티비티로 이동하도록 설계하였다.



### III-5 Cafe\_Activity, Park\_Activity, Restirant\_Activity-----



Cafe\_Activity, Park\_Activity, Restirant\_Activity는 같은 구조를 갖되, 호출하는 php와 DB 테이블이 달라 내용과 용도가 다르다. 여기서는 Cafe\_Activity java 코드를 기준으로 설명한다.

장소 갈래의 기능은 해당 갈래의 가게리스트를 보여주고, 아이템 클릭 시 해당 가게의 상세 페이지로 이동한다.

가게리스트를 보여주는 기능 자체는, 수평 리사이클러 어댑터인 adapterphone1을 사용하는 점 외에는 같기에 간략히 다루도록 한다.

Cafe\_Activity.java

```
public class Cafe_Activity extends AppCompatActivity {
    RecyclerView recyclerView1;
    ArrayList<Item> items = new ArrayList<>();

    adapterphone1 adapter;
    LinearLayoutManager layoutManager2=new
    LinearLayoutManager(this,LinearLayoutManager.VERTICAL,false);

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cafe);

        recyclerView1 = findViewById(R.id.recyclerview_cafe);
        adapter = new adapterphone1(this, items);
        recyclerView1.setAdapter(adapter);
        recyclerView1.setLayoutManager(layoutManager2);

        //서버주소
        String serverUrl = "http://heremong.dothome.co.kr/SearchCafe.php";

        JSONArrayRequest jsonArrayRequest = new JSONArrayRequest(Request.Method.POST,
```

```

serverUrl, null, new Response.Listener<JSONArray>() {

@Override
public void onResponse(JSONArray response) {

items.clear();
adapter.notifyDataSetChanged();

try {

for (int i = 0; i < response.length(); i++) {
JSONObject jsonObject = response.getJSONObject(i);

int Place_id = Integer.parseInt(jsonObject.getString("Place_id"));
String P_name = jsonObject.getString("P_name");
String P_address = jsonObject.getString("P_address");
String P_image = jsonObject.getString("P_image");

items.add(0, new Item(Place_id, P_name, P_address, P_image));
adapter.notifyItemInserted(0);

}} catch (JSONException e) {
e.printStackTrace();
}

}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error) {
}
});

//실제 요청 작업을 수행해주는 요청큐 객체 생성
RequestQueue requestQueue = Volley.newRequestQueue(this);

//요청큐에 요청 객체 생성
requestQueue.add(jsonArrayRequest);

}
}

```

상기 리사이클러뷰 설명에서 사용된 구조와 같다. 사용되는 serverUrl이 다르므로 데이터가 다르다.

## 5-1 item, adapterphone1-----

item 의 경우 같은 클래스를 사용하여 코드를 생략한다.

adapterphone1 의 경우 adapterphone2 와 똑같으나, 설정된 xml만이 달라 코드를 생략한다.

## 5-2 shopinfo\_Activity-----

리사이클러뷰의 아이템을 클릭하면 호출되는 가게 상세 페이지 액티비티 이다.

shopinfo\_Activity.java

```
public class ShopInfo_Activity extends AppCompatActivity {
    private Intent intent;
    ImageButton c_back;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_shop_info);

        c_back = (ImageButton) findViewById(R.id.Btn_Back_press1);

        TextView tvName = (TextView)findViewById(R.id.tv_name2);
        TextView tvMsg = (TextView)findViewById(R.id.tv_msg2);
        ImageView iv2 = (ImageView) findViewById(R.id.iv2);

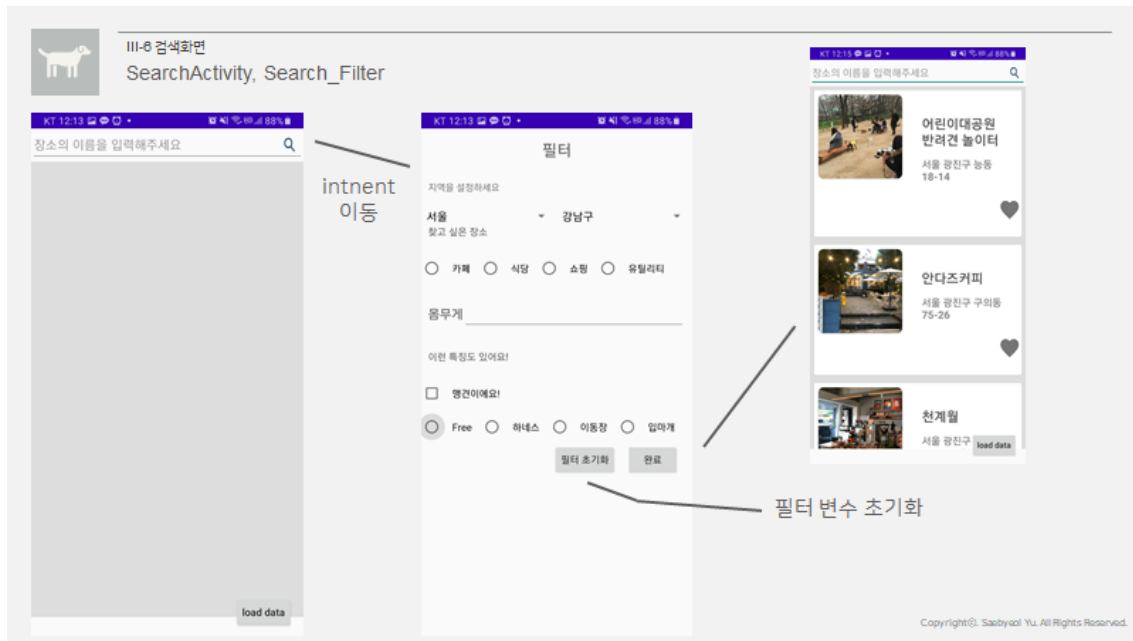
        Bitmap image;

        Intent intent = getIntent();
        tvName.setText(intent.getStringExtra("tvName"));
        tvMsg.setText(intent.getStringExtra("tvMsg"));
        byte[] arr = getIntent().getByteArrayExtra("tvImage");
        image = BitmapFactory.decodeByteArray(arr, 0, arr.length);
        iv2.setImageBitmap(image);

        c_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onBackPressed();
            }
        });
    }
}
```

가게리스트 데이터를 요청한 페이지(Home\_fragment, Cafe\_fragment)들은 질의 결과에 맞는 장소 데이터들을 4-2의 설명에 따라 리사이클러뷰로 출력시킨다. 해당 데이터들은 intent를 이용하여 shopinfo\_Activity에 전달한 뒤, 전달받은 데이터를 토대로 가게별 상세 페이지를 생성한다. 이때 ImageView 의 경우, bitmap 형식으로 변환하여 intent 한다.

### III-6 Search\_Activity (검색)-----



검색 기능은 사용자가 설정한 조건과 검색어에 따라 해당하는 검색 결과를 노출 시킨다. 돋보기를 누르면 필터페이지로 이동하며 완료 시 'load data' 버튼을 누르면 검색이 실행된다.

SearchActivity.java

```
public class SearchActivity extends AppCompatActivity {  
    RecyclerView recyclerView;  
    ArrayList<Item> items = new ArrayList<>();  
    adapterphone1 adapter;  
    EditText search_bar;  
    ImageButton filterbutton;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_search);  
  
        recyclerView = findViewById(R.id.recycler2);  
        adapter = new adapterphone1(this, items);  
        recyclerView.setAdapter(adapter);  
        search_bar = findViewById(R.id.search_bar1);  
        filterbutton = findViewById(R.id.filterbutton);  
  
        //리사이클러뷰의 레이아웃 매니저 설정  
        LinearLayoutManager layoutManager = new LinearLayoutManager(this,  
        LinearLayoutManager.VERTICAL, false);  
        recyclerView.setLayoutManager(layoutManager);  
    }  
}
```

위의 코드는 해당 Activity에서 쓰이는 객체와 리사이클러뷰의 레이아웃 매니저를 설정한다.

```
//필터 설정으로 넘어가기  
filterbutton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent(getApplicationContext(), Search_Filter.class);  
        startActivity(intent);  
    }  
});  
}
```

위의 코드는 ‘돋보기’ 버튼을 눌렀을 때 실행되며 Search\_Filter로 인텐트 이동을 하게 하는 부분이다.

```

public void clickLoad(View view) {
    String SB = search_bar.getText().toString();
    Intent GetIntent = getIntent();

    String Dkg = GetIntent.getStringExtra("Dkg");
    String Dsav = GetIntent.getStringExtra("Dsav");
    String Grade = GetIntent.getStringExtra("Grade");
    String Cate = GetIntent.getStringExtra("Cate");
    String Paddress = GetIntent.getStringExtra("Paddress");

    Response.Listener<String> responseListener = new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Toast.makeText(SearchActivity.this, response.toString(), Toast.LENGTH_SHORT).show();
            //불러들이는지 확인하기 위해 해당내용을 토스트메세지로 출력하도록 함(개발편의)
            System.out.println("답변옴");
            items.clear();
            adapter.notifyDataSetChanged();
            try {
                JSONArray searcharray = new JSONArray(response);
                for(int i=0;i<response.length();i++){
                    JSONObject jsonObject = searcharray.getJSONObject(i);

                    int Place_id= Integer.parseInt(jsonObject.getString("Place_id"));
                    String P_name=jsonObject.getString("P_name");
                    String P_address=jsonObject.getString("P_address");
                    String P_image=jsonObject.getString("P_image");

                    items.add(0,new Item(Place_id, P_name, P_address, P_image));
                    adapter.notifyItemInserted(0);
                }
            } catch (JSONException e) {e.printStackTrace();}
        }
    };
    // 서버로Volley를 이용해서 요청을 함.
    SearchRequest_filter searchRequest_filter = new SearchRequest_filter(ID, SB, Dkg,
    Grade, Dsav, Cate, Paddress, responseListener);
    RequestQueue queue = Volley.newRequestQueue(SearchActivity.this);
    queue.add(searchRequest_filter);
}
}

```

위의 코드는 'load data' 버튼을 누르면 실행된다.

검색어와 해당 조건들을 인텐트 통신으로 받아 변수에 저장한 뒤 해당하는 장소들을 **responseListener**를 통해 받아 와 Item 클래스에 저장하여 리사이클러뷰로 출력한다.

이때 요청과 처리는 volley 라이브러리를 사용했으며 **SearchRequest\_filter**에서 Map 방식으로 DB에 정보를 요청한다.

## 6-1 Search\_Filter-----

Search\_Filter.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_search_filter);

    Spinner aspinner = findViewById(R.id.Aspinner);
    Spinner aspinner2 = findViewById(R.id.Aspinner2);

    spinner = ArrayAdapter.createFromResource(this, R.array.spinner_do,
        android.R.layout.simple_spinner_dropdown_item);

    spinner.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    aspinner.setAdapter(spinner);

    aspinner.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int i, long l) {
            if(spinner.getItem(i).equals("서울")) {
                choice_do = "서울";
                spinner2 = ArrayAdapter.createFromResource(Search_Filter.this,
                    R.array.spinner_do_seoul, android.R.layout.simple_spinner_dropdown_item);
                spinner2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                aspinner2.setAdapter(spinner2);

                //변수에"도" 저장
                adress_do = aspinner.getSelectedItem().toString();

                aspinner2.setOnItemClickListener(new AdapterView.OnItemClickListener() {
                    @Override
                    public void onItemClick(AdapterView<?> parent, View view, int i, long l) {
                        choice_se = spinner2.getItem(i).toString();

                        //변수에"시/군/구" 저장
                        adress_se = aspinner.getSelectedItem().toString();
                    }
                });
            } else if (spinner.getItem(i).equals("경기")) {
                choice_do = "경기";
                spinner2 = ArrayAdapter.createFromResource(Search_Filter.this,
                    R.array.spinner_do_gyeonggi, android.R.layout.simple_spinner_dropdown_item);
                spinner2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
                aspinner2.setAdapter(spinner2);

                //변수에"도" 저장
                adress_do = aspinner.getSelectedItem().toString();

                aspinner2.setOnItemClickListener(new AdapterView.OnItemClickListener() {
                    @Override
                    public void onItemClick(AdapterView<?> parent, View view, int i, long l) {
                        choice_se = spinner2.getItem(i).toString();

                        //변수에"시/군/구" 저장
                        adress_se = aspinner.getSelectedItem().toString();
                    }
                });
            }
        }
    });
}
```

```

public void onNothingSelected(AdapterView<?> parent) {
}
});

@Override
public void onNothingSelected(AdapterView<?> parent) {
}
});

//카테고리 선택
place = findViewById(R.id.place);
place1 = findViewById(R.id.place1);
place2 = findViewById(R.id.place2);
place3 = findViewById(R.id.place3);
place4 = findViewById(R.id.place4);

place.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
@Override
public void onCheckedChanged(RadioGroup group, int checkedId) {

if (checkedId == R.id.place1){
//임시로 카테고리 변수 설정
Cate = "카페";

}else if(checkedId == R.id.place2){
Cate = "식당";

}else if(checkedId == R.id.place3){
Cate = "쇼핑";

}else if(checkedId == R.id.place4){
Cate = "유틸리티";

}
}
});

//맹견 여부 체크
ferodog = findViewById(R.id.ferodog);
ferodog.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
@Override
public void onCheckedChanged(CompoundButton buttonView, boolean b) {
Dsav = "0";
if (ferodog.isChecked()){
Dsav = "1";
}else if(ferodog.isChecked()==false) {
Dsav = "0";
}
}
});

//등급설정
grade_group = findViewById(R.id.grade);
grade0 = findViewById(R.id.grade0);
grade1 = findViewById(R.id.grade1);
grade2 = findViewById(R.id.grade2);

```



```

grade3 = findViewById(R.id.grade3);

grade_group.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()
{
    @Override
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        if (grade0.isChecked()){
            Grade = "0";
            Toast.makeText(Search_Filter.this, Grade, Toast.LENGTH_SHORT).show();
        }else if(grade1.isChecked()){
            Grade = "1";
        }else if(grade2.isChecked()){
            Grade="2";
        }else if(grade3.isChecked()){
            Grade="3";
        }
    }
});

```

위의 코드는 필터페이지에서 여러 조건을 설정할 때 사용자의 입력을 받아 변수에 입력하는 코드이다.

순서대로 지역(Spinner), 카테고리(RadioButton), 몸무게(EditText), 맹견 여부(CheckBox), 장소 등급(RadioButton)의 조건 설정 객체들이다.

```

//필터 설정 완료 버튼
filtercomplete = findViewById(R.id.filtercomplete);

filtercomplete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        et_weight = findViewById(R.id.et_weight);
        Dkg = et_weight.getText().toString();

        Intent myIntent = new Intent(Search_Filter.this, SearchActivity.class);

        myIntent.putExtra("Dkg", Dkg);
        myIntent.putExtra("Dsav", Dsav);
        myIntent.putExtra("Grade", Grade);
        myIntent.putExtra("Cate", Cate);
        myIntent.putExtra("SB", SB);
        myIntent.putExtra("ID", ID);
        myIntent.putExtra("Paddress", adress_se);
        startActivity(myIntent);

        onBackPressed();
    }
});

```

위의 코드는 필터페이지의 '완료' 버튼을 누르면 실행되며 설정 조건들을 변수에 저장하며 인텐트 통신으로 값들을 **SearchActivity**에 전송한다. 설정이 완료되면 이전의 페이지인 **SearchActivity**로 이동한다.

```
//필터 초기화 버튼
delete = findViewById(R.id.delete);

delete.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
address_do = "서울";
address_se = "서울";
Cate = "카페";
Dsav = "0";
Dkg = "1";
Grade = "3";
Toast.makeText(Search_Filter.this, "필터가 초기화 되었습니다.",
Toast.LENGTH_SHORT).show();
}
});
```

위의 코드는 필터페이지의 ‘필터 초기화’ 버튼을 누르면 실행되며 변수에 저장된 설정 조건들을 기본값 혹은 null 값으로 초기화시킨다.

## 6-2 SearchRequest\_filter, SearchF.php-----

SearchRequest\_filter.java

```
public class SearchRequest_filter extends StringRequest {
// 서버URL 설정( PHP 파일 연동)
final static private String URL = "http://heremong.dothome.co.kr/SearchF.php";
private Map<String, String> map;

public SearchRequest_filter(String ID, String SB, String Dkg, String Grade, String Dsav,
String Cate, String Paddress, Response.Listener<String> listener){
super(Method.POST, URL, listener, null);

map = new HashMap<>();

map.put("ID",ID);
map.put("SB",SB);
map.put("Dkg",Dkg);
map.put("Grade",Grade);
map.put("Dsav",Dsav);
map.put("Cate",Cate);
map.put("Paddress",Paddress);
}

@Override
protected Map<String, String> getParams() throws AuthFailureError {
return map;
}
}
```

SearchRequest\_filter는 서버의 데이터베이스에 필터에서 설정한 조건들과 함께 받아올 결과에 대한 질의를 보내는 클래스이다.

Map을 이용하여 변수에 저장된 값을 보내며 해당 값을 받아 처리하는 것은 URL의 php이다. 설정 조건을 보내 해당하는 장소 값을 받아온다.

Search.php

```
<?php
/*
강아지의 정보와 조건을 읽어와서DB에서 조회 후 출력(검색, 필터)
*/

header("Content-Type:text/html; charset=UTF-8");

$conn= mysqli_connect("localhost","heremong","////////","heremong");

mysqli_query($conn, "set names utf8");

$ID = $_POST["ID"];
$SB = $_POST["SB"];
$Dkg = intval($_POST["Dkg"]);
$Dsav = intval($_POST["Dsav"]);
$Grade = intval($_POST["Grade"]);
$Cate = $_POST["Cate"];
$Paddress = $_POST["Paddress"];

//$sql= "select Place_id, P_name, P_address, P_image from Place where P_name
LIKE '%{$SB}%'";

$sql= " SELECT DISTINCT Place_id, P_name, P_address, P_image
FROM Place, Dog
WHERE( Place.P_kg NOT BETWEEN 0 AND '{$Dkg}'
OR Place.p_kg IS NULL
AND Place.P_sav <= '{$Dsav}'
AND Place.p_cate= '{$Cate}'
AND Place.p_grade <= '{$Grade}'
AND Place.P_name LIKE '%{$SB}%'
AND Place.P_address LIKE '%{$Paddress}%' )";

$result=mysqli_query($conn, $sql);

$rowCnt= mysqli_num_rows($result);

$arr= array(); //빈 배열 생성

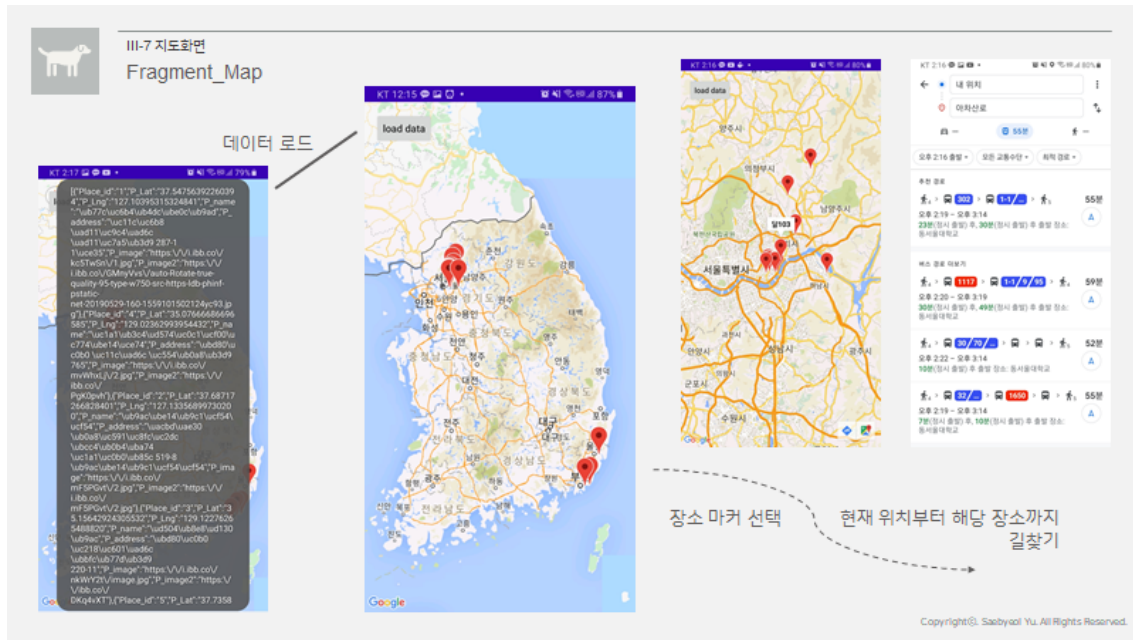
for($i=0;$i<$rowCnt;$i++){
$row= mysqli_fetch_array($result, MYSQLI_ASSOC);
//각 각의row를$arr에 추가
$arr[$i]= $row;
}

//배열을json으로 변환하는 함수가 있음.
$jsonData=json_encode($arr); //json배열로 만들어짐.
echo "$jsonData";

mysqli_close($conn);
?>
```

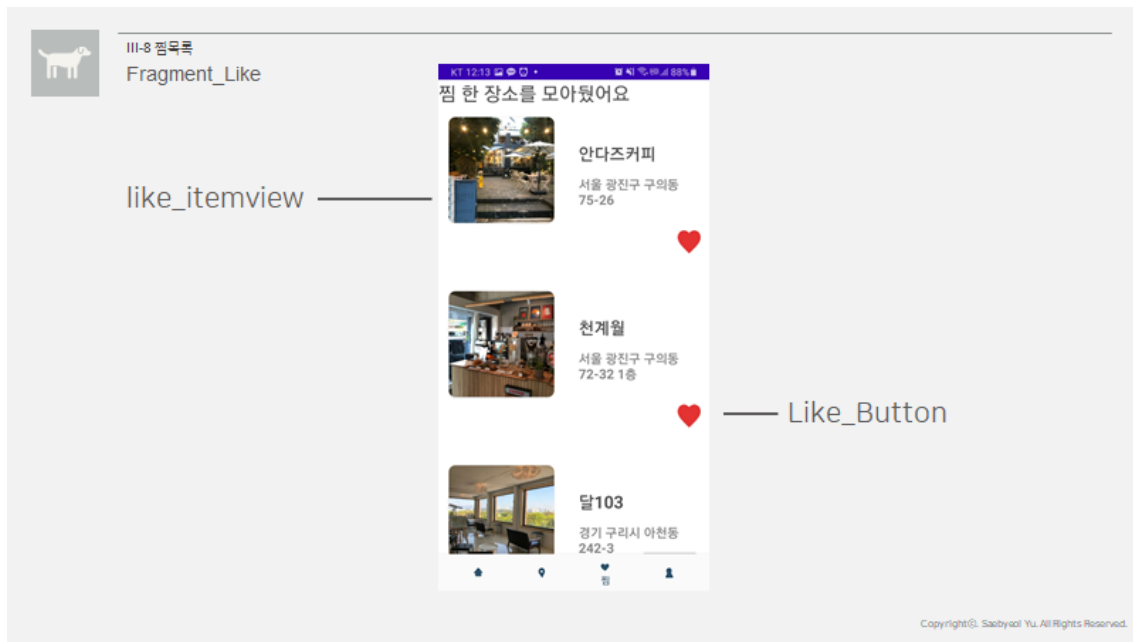
**Seacrhf.php**는 설정한 각 조건의 변수값을 받아오며 해당 조건에 맞는 조건들을 질의한다.  
질의 결과는 json 형태로 장소들을 배열로 만들어 답변한다.

### III-7 Fragment\_Map



지도 기능은 데이터 베이스에 저장된 가게의 위치를 확인하기 위한 기능이다. AndoridStudio 기본 제공 템플릿중 Google Maps Activity 를 이용하여 구현하였다. 구글 Map API 를 사용하였는데, 이를 사용하기 위해서는 API Key 를 발급받아야 한다. 이후 각 장소들의 위도와 경도를 설정하여 마커를 찍는다. 구글 Map 자체의 기능으로, 출발장소와 목적지를 설정하여 길찾기 기능을 사용 할 수 있다.

### III-8 Fragment\_Like -----



가게리스트를 보여주는 구조 역시 위에서 설명한 리사이클러뷰의 설명과 기본 코드 구조가 같다.

Fragment\_Like.java

```
public class Fragment_Like extends Fragment {

    private View view2;
    RecyclerView recyclerView1;
    ArrayList<Item> items = new ArrayList<>();

    adapterphone1 adapter;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        @Nullable Bundle savedInstanceState) {
        view2 = inflater.inflate(R.layout.fragment_like, container, false);
        LinearLayoutManager layoutManager2 = new
            LinearLayoutManager(getActivity().getApplicationContext(), LinearLayoutManager.VERTICAL, false);

        recyclerView1 = view2.findViewById(R.id.recyclerview_like);
        adapter = new adapterphone1(getActivity().getApplicationContext(), items);
        recyclerView1.setAdapter(adapter);
        recyclerView1.setLayoutManager(layoutManager2);
        String serverUrl = "http://heremong.dothome.co.kr/Like.php";

        JsonRequest jsonArrayRequest = new JsonRequest(Request.Method.POST,
            serverUrl, null, new Response.Listener<JSONArray>() {

            @Override
            public void onResponse(JSONArray response) {

                items.clear();
                adapter.notifyDataSetChanged();

                try {

                    for (int i = 0; i < response.length(); i++) {
                        JSONObject jsonObject = response.getJSONObject(i);

                        int Place_id = Integer.parseInt(jsonObject.getString("Place_id"));
                        String P_name = jsonObject.getString("P_name");
                        String P_address = jsonObject.getString("P_address");
                        String P_image = jsonObject.getString("P_image");

                        items.add(0, new Item(Place_id, P_name, P_address, P_image)); // 첫 번째 매개변수는
                        // 몇번째에 추가 될지, 제일 위에 오도록
                        adapter.notifyItemInserted(0);
                    }
                } catch (JSONException e) {e.printStackTrace();}
            }

        }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError error) {

            }

        }));

        //실제 요청 작업을 수행해주는 요청큐 객체 생성
        RequestQueue requestQueue =
            Volley.newRequestQueue(getActivity().getApplicationContext());
```

```
//요청큐에 요청 객체 생성
requestQueue.add(jsonArrayRequest);

return view2;
}
```

앞서 설명한 내용과 같이, 사용하는 `serverUrl`이 달라 띄워지는 내용과 용도가 다르다.

**Like.php**를 사용하여 데이터베이스의 **Liked** 테이블의 데이터를 응답받는다.  
그러나 안드로이드 스튜디오 코드의 문제로 무결성 검사를 자체적으로 하지 못하여 데이터 신뢰성이 떨어진다.

Like.php

```
<?php
/*
아이디를 읽어와서 해당 사용자의 찜 목록을DB에서 조회 후 출력
*/

header("Content-Type:text/html; charset=UTF-8");

$conn= mysqli_connect("localhost","heremong","////////","heremong");

mysqli_query($conn, "set names utf8");

$ID = $_POST["ID"];

$sql= " SELECT DISTINCT Place.Place_id, Place.P_name, Place.P_address,
Place.P_image
FROM Place, Liked
WHERE Place.Place_id IN (Select Place_id From Liked Where ID = '{ $ID}' )";

$result=mysqli_query($conn, $sql);

$rowCnt= mysqli_num_rows($result);

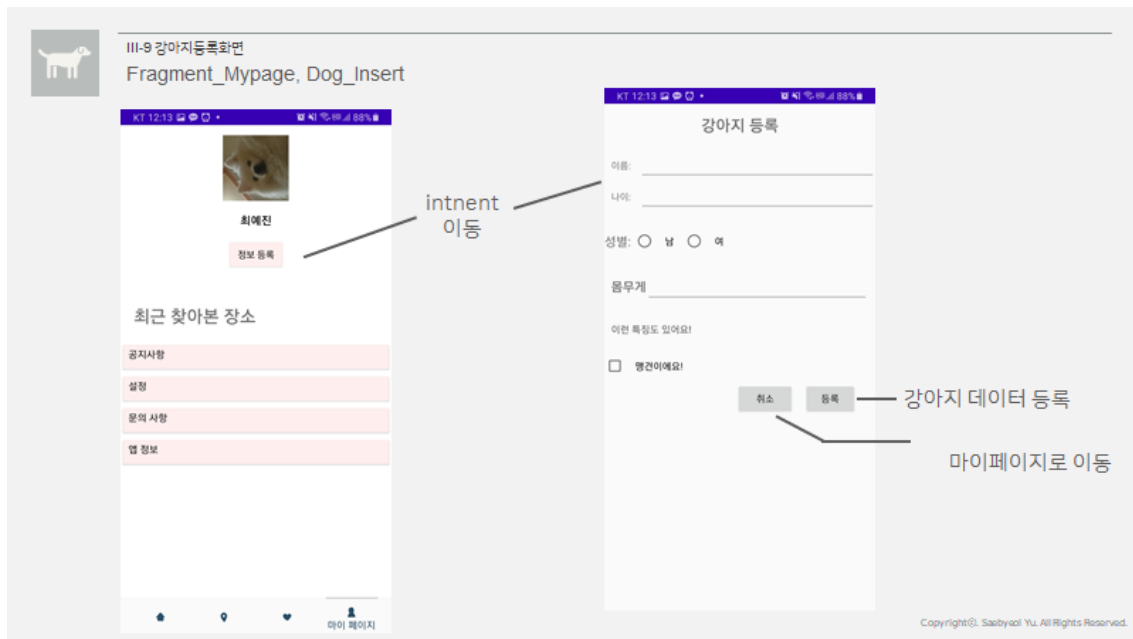
$arr= array(); //빈 배열 생성

for($i=0;$i<$rowCnt;$i++){
$row= mysqli_fetch_array($result, MYSQLI_ASSOC);
//각 각의row를$arr에 추가
$arr[$i]= $row;
}

//배열을json으로 변환하는 함수가 있음.
$jsonData=json_encode($arr); //json배열로 만들어짐.
echo "$jsonData";

mysqli_close($conn);
?>
```

### III-9 Fragment\_Mypage (마이페이지)-----



마이페이지에서는 정보등록 버튼을 통해 사용자가 자신의 반려견을 등록할 수 있으며 해당 반려견의 프로필은 맞춤 검색에서 사용된다. 취소 시 입력한 내용은 사라지며 마이페이지로 돌아가며 등록 버튼을 누를 시 강아지 데이터가 데이터베이스에 등록된다.

<input type="checkbox"/>	수정	복사	삭제	19	id	하늘	12.0	0	1	2.0	NULL
--------------------------	----	----	----	----	----	----	------	---	---	-----	------



Fragment\_Mypage.java

```
public class Fragment_Mypage extends Fragment {

    private View view2;
    RecyclerView phoneRecycler;
    RecyclerView.Adapter adapter;

    Button btn;
    TextView textView1;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        @Nullable Bundle savedInstanceState) {
        view2 = inflater.inflate(R.layout.fragment_mypage, container, false);

        btn = view2.findViewById(R.id.btn_dogprofile);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getActivity().getApplicationContext(), DogInsert.class);
                startActivity(intent); //액티비티 이동
            }
        });
        return view2;
    }
}
```

위의 코드 부분은 해당 Fragment에서 쓰이는 객체 등을 설정한다. ‘정보등록’ 버튼을 누르면 강아지 등록 페이지(DogInsertActivity)로 인텐트 이동한다.

## 9-1 DogInsert-----

DogInsert.java

```
public class DogInsert extends AppCompatActivity {

    ArrayAdapter<CharSequence> spinner, spinner2;

    String Sex; //장소 카테고리
    String Kg; //몸무게
    String Grade; // grade 하네스 이동장 여부
    TextView sizeresult;
    Button btn_cancel;
    Button btn_regi;
    RadioGroup sex;
    RadioButton sex1, sex2;
    CheckBox ferodogi;
    String Sav = "0";
    EditText et_name, et_age, et_weight;
    String ID, namei, agei;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.doginsert);

        //성별 선택
        sex = findViewById(R.id.sex);
        sex1 = findViewById(R.id.sex1);
        sex2 = findViewById(R.id.sex2);

        sex.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {

                if (checkedId == R.id.sex1){
                    //임시로 카테고리 변수 설정
                    Sex = "0";
                }else if(checkedId == R.id.sex2){
                    Sex = "1";
                }
            }
        });

        //맹견 여부 체크
        ferodogi = findViewById(R.id.ferodogi);
        ferodogi.setOnCheckedChangeListener(new
        CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean b) {
                Sav = "0";
                if (ferodogi.isChecked()){
                    Sav = "1";
                }else if(ferodogi.isChecked()==false) {
                    Sav = "0";
                }
            }
        });

        et_name = findViewById(R.id.et_name);
        et_age = findViewById(R.id.et_age);
        et_weight = findViewById(R.id.et_weight);
    }
}
```

위의 코드는 등록 페이지에서 사용자의 입력을 받아 변수에 입력하는 코드이다.

순서대로 이름(EditText), 나이(EditText), 성별(RadioButton), 몸무게(EditText), 맹견 여부(CheckBox)의 강아지 정보 등록 객체들이다.

```
//등록 완료 버튼
btn_regi = findViewById(R.id.btn_regi);
btn_regi.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {

    namei = et_name.getText().toString();
    agei = et_age.getText().toString();
    Kg = et_weighti.getText().toString();

    Response.Listener<String> responseListener = new Response.Listener<String>() {
@Override
public void onResponse(String response) {

        try {
            JSONObject jsonObject = new JSONObject(response);
            boolean success = jsonObject.getBoolean("success");
            if (success) { // 등록에 성공한 경우
                Toast.makeText(DogInsert.this,"반려견 등록에
                성공하였습니다.",Toast.LENGTH_SHORT).show();
                onBackPressed();

            } else { // 등록에 실패한 경우
                Toast.makeText(DogInsert.this,"반려견 등록에
                실패하였습니다.",Toast.LENGTH_SHORT).show();
                return;
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
});

    DogRequest dogRequest = new DogRequest( ID, namei, Kg, Sav, Sex, agei,
    responseListener);
    RequestQueue dogqueue = Volley.newRequestQueue(DogInsert.this);
    dogqueue.add(dogRequest);
}
```

위의 코드는 강아지 정보등록 페이지의 ‘등록’ 버튼을 누르면 실행되며 설정 조건들을 변수에 저장하며 데이터베이스에 등록한다. 앞서 말한 III-2 Login ,Register Activity(로그인, 회원가입)의 매커니즘과 같다. ResponseListener와 DogRequest를 활용하여 요청 및 응답한다.(상기 내용은 III-8-2 DogRequest, DogInsert.php 항목에서 세부적으로 다루도록 한다) 등록에 성공 시 토스트 메시지로 짧게 “반려견 등록에 성공하였습니다.”라고 출력하며 실패 시 “반려견 등록에 실패하였습니다.”라고 출력된다.

```
//취소 버튼
btn_canceli = findViewById(R.id.btn_canceli);

btn_canceli.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
    onBackPressed();
}
});
```

위의 코드는 강아지 정보등록 페이지의 ‘취소’ 버튼을 누르면 실행되며 입력 값은 사라지며 이전의 페이지인 마이페이지로 돌아간다.

## 9-2 DogRequest, DogInsert.php-----

DogRequest.java

```
public class DogRequest extends StringRequest {  
    // 서버URL 설정( PHP 파일 연동)  
    final static private String URL = "http://heremong.dothome.co.kr/DogInsert.php";  
    //강아지 등록php  
    private Map<String, String> map;  
  
    public DogRequest(String ID, String namei, String Kg, String Sav, String Sex, String  
    agei, Response.Listener<String> listener) {  
        super(Method.POST, URL, listener, null);  
  
        map = new HashMap<>();  
        map.put("ID",ID);  
        map.put("namei",namei);  
        map.put("Kg",Kg);  
        map.put("Sav",Sav);  
        map.put("Sex",Sex);  
        map.put("agei",agei);  
    }  
  
    @Override  
    protected Map<String, String> getParams() throws AuthFailureError {  
        return map;  
    }  
}
```

DogRequest는 서버의 데이터베이스에 강아지 등록 페이지에서 사용자가 입력한 정보들과 함께 받아올 결과에 대한 질의를 보내는 클래스이다.

Map을 이용하여 변수에 저장된 값을 보내며 해당 값을 받아 처리하는 것은 URL의 php이다. 해당 클래스는 사용자의 ID 값과 강아지 정보를 보내 해당 사용자에게 반려견의 정보를 등록한다.

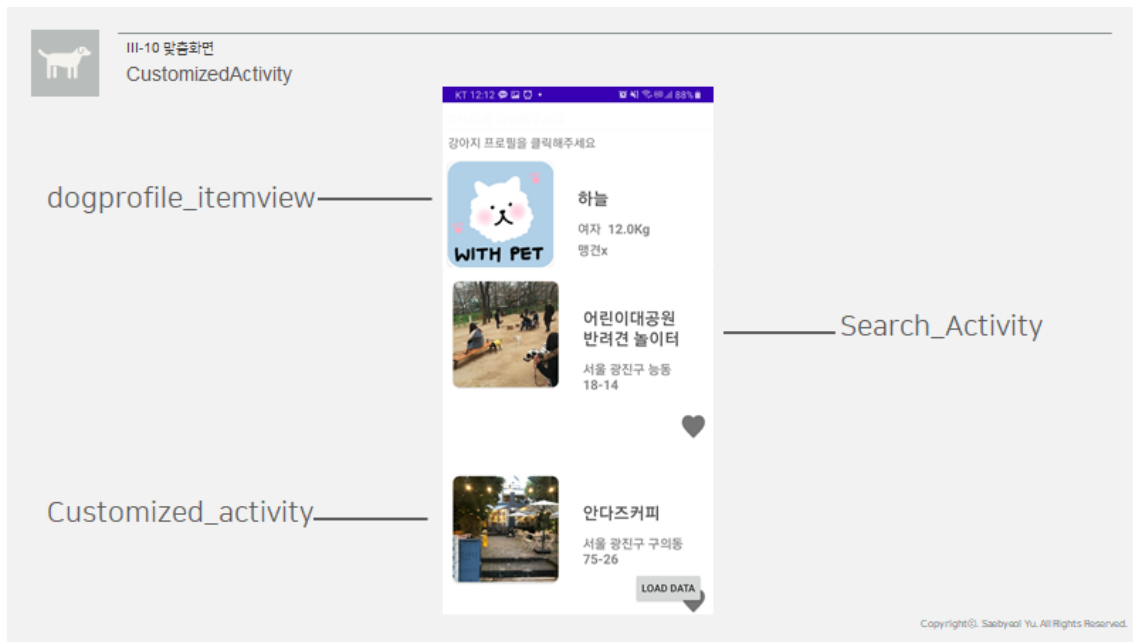
DogInsert.php

```
<?php  
/*  
해당 사용자의ID를 읽어와서 강아지의 프로필 출력(ex. 아이디가'a'인 사람의 강아지 정보  
출력)  
맞춤 상단의 수평 리사이클러뷰에 사용  
*/  
  
header("Content-Type:text/html; charset=UTF-8");  
$conn= mysqli_connect("localhost","heremong","\\\\\\\\\\\\\\\\","heremong");  
mysqli_query($conn, "set names utf8");  
  
$ID = $_POST["ID"];  
$Kg = intval($_POST["Kg"]);  
$namei = $_POST["namei"];  
$Sav = intval($_POST["Sav"]);  
$Sex = intval($_POST["Sex"]);  
$agei = intval($_POST["agei"]);  
  
$sql= "INSERT INTO Dog(ID, D_name, D_kg, D_sav, D_sex, D_age) VALUES ('{$ID}',  
'{$namei}', '{$Kg}',  
'{$Sav}', '{$Sex}', '{$agei}')";  
  
$result=mysqli_query($conn, $sql);
```

```
$response = array();  
$response["success"] = true;  
  
echo json_encode($response);  
  
mysqli_close($conn);  
?>
```

**DogInsert.php**는 사용자의 ID 값, 입력한 강아지 정보 값을 받아 와 질의한다. 해당 질의가 성공적으로 끝나면 success 값의 불린 값을 true로 반환한다.

### III-10 CustmizeActivity -----



맞춤 페이지는 검색에서 여러 조건을 일일이 입력하는 번거로움을 덜기 위한 기능이다. 마이페이지에서 등록한 자신의 강아지의 프로필이 상단에 뜨며 프로필을 클릭 시 해당 강아지의 조건들을 자동으로 입력시켜 준다. 별도의 입력 없이 강아지 프로필과 검색 버튼을 누르면 '맞춤' 결과를 노출 시킨다.

CustomizedActivity.java

```
public class CustomizedActivity extends AppCompatActivity {
    RecyclerView recyclerView;
    ArrayList<dogProfile_Item> items = new ArrayList<>();
    CustomizeAdapter adapter;
    EditText search_bar;
    Button btn_customintent;

    private FragmentManager fragmentManager;
    private FragmentTransaction transaction;
    Customized_Fragment customized_fragment;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customized);

        String ID ="id";

        recyclerView = findViewById(R.id.recycler_dogprofile);
        adapter = new CustomizeAdapter(this, items);
        recyclerView.setAdapter(adapter);
        search_bar = findViewById(R.id.search_bar2);
    }
}
```

```

//리사이클러뷰의 레이아웃 매니저 설정
LinearLayoutManager layoutManager = new LinearLayoutManager(this,
LinearLayoutManager.HORIZONTAL, false);
recyclerView.setLayoutManager(layoutManager);

fragmentManager = getSupportFragmentManager();
customized_fragment = new Customized_Fragment();

transaction = fragmentManager.beginTransaction();
transaction.replace(R.id.custom_result,
customized_fragment).commitAllowingStateLoss();

Response.Listener<String> responseListener = new Response.Listener<String>() {
@Override
public void onResponse(String response) {
Toast.makeText(CustomizedActivity.this, response.toString(),
Toast.LENGTH_SHORT).show();
//볼러들이는지 확인하기 위해 해당내용을 토스트메세지로 출력하도록 함(개발편의를 위한 것)
System.out.println("답변옴");
items.clear(); //일단 깨끗하게 처리함
adapter.notifyDataSetChanged();
try {
JSONArray dogarray = new JSONArray(response);
for(int i=0;i<response.length();i++){
JSONObject jsonObject = dogarray.getJSONObject(i);

String Dog_id = jsonObject.getString("Dog_id");
String D_name = jsonObject.getString("D_name");
String D_sex =jsonObject.getString("D_sex");
String D_sav = jsonObject.getString("D_sav");
String D_kg = jsonObject.getString("D_kg"); //no가 문자열이라서 바꿔야함.

items.add(0, new dogProfile_Item(Dog_id,D_name, D_sex, D_sav, D_kg)); //
adapter.notifyItemInserted(0); //   이는 삽입된item(여기선 고유명사item 임)이
있다는걸 알려주는것

//
}
} catch (JSONException e) {
e.printStackTrace();
}

};
// 서버로Volley를 이용해서 요청을 함.
CustomizedRequest customizedRequest = new CustomizedRequest(ID,responseListener
);
RequestQueue queue = Volley.newRequestQueue(CustomizedActivity.this);
queue.add(customizedRequest);
}
}

```

위의 코드 부분은 해당 Activity에서 쓰이는 객체와 리사이클러뷰의 레이아웃 매니저, 프라그먼트를 설정한다.

리사이클러뷰로 강아지 프로필을 띄우는 부분은 Activity에서 **responseListener**를 이용하는 검색(III-6)와 같다.

## 10-1 dogProfile\_item, CustomizedAdapter-----

dogProfile\_item.java

```
public class dogProfile_Item {

String D_name,ID,D_sav,D_sex,D_kg,Dog_id;

//강아지 무게, 이름, 유저아이디, 맹견여부, 성별

int Place_id;

public dogProfile_Item() {

}

public dogProfile_Item(String Dog_id,String D_name, String D_sex,String D_sav,String
D_kg) {
this.D_name = D_name;
this.D_sex = D_sex;
this.D_sav = D_sav;
this.D_kg = D_kg;
this.Dog_id = Dog_id;
}

public String getDog_id() {
return Dog_id;
}

public void setDog_id(String D_name) { this.Dog_id = Dog_id; }

public String getD_name() {
return D_name;
}

public void setD_name(String D_name) { this.D_name = D_name; }

public String getD_sav() {
return D_sav;
}

public void setD_sav(String P_image) {
this.D_sav = D_sav;
}

public String getD_sex() {
return D_sex;
}

public void setD_sex(int Dog_id) {
this.D_sex = D_sex;
}

public String getD_kg() {
return D_kg;
}
}
```



CustomizedAdapter.java

```
public class CustomizeAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    Context context;
    ArrayList<dogProfile_Item> dogProfile_items; //아이템을 넣은 배열. 강아지 프로필용ITMe
    Customized_Fragment fragment_customized;

    public CustomizeAdapter(Context context, ArrayList<dogProfile_Item> dogProfile_items)
    {
        this.dogProfile_items = dogProfile_items;
        this.context = context;
    }

    //수평 리사이클러뷰 어댑터
    @NonNull
    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
    viewType) {
        LayoutInflater inflater= LayoutInflater.from(context);

        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.dogprofile_itemview,
        parent, false);

        VH holder = new VH(view);
        return holder;
    }

    @Override
    public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
        VH vh= (VH) holder;

        dogProfile_Item dogProfile_item = dogProfile_items.get(position);
        vh.D_name.setText(dogProfile_item.getD_name());
        vh.D_kg.setText(dogProfile_item.getD_kg()+"Kg");
        vh.Dog_id.setText(dogProfile_item.getDog_id());

        String i =dogProfile_item.getD_sav();

        if (i.equals("0")){
            vh.D_sav.setText("맹견x");
        } else{
            vh.D_sav.setText("맹견");
        }
        String e = dogProfile_item.getD_sex();

        if (e.equals("0")){
            vh.D_sex.setText("남자");
        } else {
            vh.D_sex.setText("여자");
        }

        //세부페이지를 위해 추가한부분
        vh.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String Dog_id = dogProfile_item.getDog_id();
            }
        });
    }

    @Override
    public int getItemCount(){
```

```

return (dogProfile_items != null ? dogProfile_items.size() : 0);
}

class VH extends RecyclerView.ViewHolder implements View.OnClickListener{

    TextView D_name;
    TextView D_sex;
    TextView D_kg;
    TextView D_sav;
    TextView Dog_id;

    ImageView iv;

    public VH(@NonNull View itemView) {
        super(itemView);

        D_name=itemView.findViewById(R.id.dv_name);
        D_sex=itemView.findViewById(R.id.dv_sex);
        D_kg=itemView.findViewById(R.id.dv_kg);
        D_sav=itemView.findViewById(R.id.dv_sav);
        Dog_id=itemView.findViewById(R.id.dv_dogid);
    }

    @Override
    public void onClick(View v) {

        int clickedPosition = getAdapterPosition();
    }
}

```

위의 dogProfile\_item 코드와 CustomizedAdapter 코드는 CustomizedActivity의 리사이클러뷰 responseListener에서 뷰에 값을 넣기 위해 존재하는 클래스이다. 앞서 III-4-2 item, adapterphone2(리사이클러뷰) 항목에서 언급한 바와 내용이 동일하며 각 item의 변수 데이터 타입과 변수값만이 다르다.

## 10-2 CustomizedRequest, DogProfile.php-----

CustomizedRequest.java

```
public class CustomizedRequest extends StringRequest {

    // 서버URL 설정( PHP 파일 연동)
    final static private String URL = "http://heremong.dothome.co.kr/DogProfile.php";
    private Map<String, String> map;

    public CustomizedRequest( String ID, Response.Listener<String> listener) {
        super(Method.POST, URL, listener, null);
    }

    map = new HashMap<>();

    map.put("ID",ID);

}

@Override
protected Map<String, String> getParams() throws AuthFailureError {
    return map;
}
}
```

CustomizedRequest는 서버의 데이터베이스에 필터에서 설정한 조건들과 함께 받아올 결과에 대한 질의를 보내는 클래스이다.

Map을 이용하여 변수에 저장된 값을 보내며 해당 값을 받아 처리하는 것은 URL의 php이다. 해당 클래스는 사용자의 ID값을 보내 해당 사용자에게 등록된 반려견의 프로필 정보를 받아와 출력한다.

DogProfile.php

```
<?php
/*
해당 사용자의ID를 읽어와서 강아지의 프로필 출력(ex. 아이디가'a'인 사람의 강아지 정보
출력)
맞춤 상단의 수평 리사이클러뷰에 사용
*/

header("Content-Type:text/html; charset=UTF-8");

$conn= mysqli_connect("localhost","heremong","\\\\\\\\\\\\\\\\","heremong");

mysqli_query($conn, "set names utf8");

$ID = $_POST["ID"];

$sql= "SELECT Dog_id, D_name, D_kg, D_sav, D_sex, D_age
FROM Dog
WHERE ID = '{$ID}'";

$result=mysqli_query($conn, $sql);

$rowCnt= mysqli_num_rows($result);

$arr= array(); //빈 배열 생성

for($i=0;$i<$rowCnt;$i++){
$row= mysqli_fetch_array($result, MYSQLI_ASSOC);
//각 각의row를$arr에 추가
$arr[$i]= $row;
}

//배열을json으로 변환하는 함수가 있음.
$jsonData=json_encode($arr); //json배열로 만들어짐.
echo "$jsonData";

mysqli_close($conn);
?>
```

**DogProfile.php**는 사용자의 ID값을 받아 와 해당 ID에 등록된 반려견 정보를 질의한다. 질의 결과는 json 형태로 프로필들을 배열로 만들어 답변한다.

### 10-3 Customized\_Fragment-----

Customized\_Fragment.java

```
public class Customized_Fragment extends Fragment {

    private View view1;

    RecyclerView recyclerView1;
    ArrayList<Item> items = new ArrayList<>();

    adapterphone1 adapter;

    String SB = " ";
    String Dog_id;
    Button btn_custom;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        view1 = inflater.inflate(R.layout.fragment_customized_, container, false);
        recyclerView1 = view1.findViewById(R.id.recyclerview_CustomResult);
        adapter = new adapterphone1(getActivity().getApplicationContext(), items);
        recyclerView1.setAdapter(adapter);
        recyclerView1.setHasFixedSize(true);
        LinearLayoutManager layoutManager2 = new
        LinearLayoutManager(getActivity().getApplicationContext(),
        LinearLayoutManager.VERTICAL, false);
        recyclerView1.setLayoutManager(layoutManager2);

        btn_custom = view1.findViewById(R.id.btn_custom);
        btn_custom.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String SB = "";
                String Dog_id = "1";

                Bundle extra = getArguments();
                if (extra != null){
                    Dog_id = extra.getString("Dog_id");

                    Toast.makeText(getActivity().getApplicationContext(), Dog_id,
                    Toast.LENGTH_SHORT).show();
                }

                Response.Listener<String> responseListener = new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        Toast.makeText(getActivity().getApplicationContext(), response.toString(),
                        Toast.LENGTH_SHORT).show();

                        System.out.println("답변옴");
                        items.clear();
                        adapter.notifyDataSetChanged();
                        try {
                            JSONArray customarray = new JSONArray(response);
                            for (int i = 0; i < response.length(); i++) {
                                JSONObject jsonObject = customarray.getJSONObject(i);

                                int Place_id = Integer.parseInt(jsonObject.getString("Place_id"));
                                String P_name = jsonObject.getString("P_name");
                                String P_address = jsonObject.getString("P_address");
                                String P_image = jsonObject.getString("P_image");
```

```

items.add(0, new Item(Place_id, P_name, P_address, P_image)); // 첫 번째 매개변수는
// 몇번째에 추가 될지, 제일 위에 오도록
adapter.notifyItemInserted(0);
}
} catch (JSONException e) {
e.printStackTrace();
}
}
};
// 서버로Volley를 이용해서 요청을 함.
CustomResultRequest customResultRequest = new CustomResultRequest(SB, Dog_id,
responseListener);
RequestQueue queue = Volley.newRequestQueue(getActivity().getApplicationContext());
queue.add(customResultRequest);

}
});

return view1;
}
}

```

위의 Customized\_Fragment 코드는 CustomizedActivity와 Fragment라는 점을 제외하곤 매우 유사하다. 해당 Fragment에서 쓰이는 객체와 리사이클러뷰의 레이아웃 매니저 등을 설정하였다.

리사이클러뷰로 강아지 조건에 맞는 장소를 띄우는 부분은 Activity에서 responseListener를 이용하는 검색(III-6)과 같으며 앞서 III-4-2 item, adapterphone2(리사이클러뷰) 항목에서 언급한 바와 내용이 동일하다.

#### 10-4 CustomResultRequest, Custom.php-----

CustomResultRequest.java

```
public class CustomResultRequest extends StringRequest {

    // 서버URL 설정( PHP 파일 연동)
    final static private String URL = "http://heremong.dothome.co.kr/Custom.php";
    private Map<String, String> map;

    public CustomResultRequest( String SB, String Dog_id, Response.Listener<String>
    listener) {
        super(Method.POST, URL, listener, null);

        map = new HashMap<>();

        map.put("SB",SB);
        map.put("Dog_id",Dog_id);
    }

    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        return map;
    }
}
```

CustomResultRequest는 서버의 데이터베이스에 검색어와 선택된 강아지의 아이디 값과 함께 받아올 결과에 대한 질의를 보내는 클래스이다.

Map을 이용하여 변수에 저장된 값을 보내며 해당 값을 받아 처리하는 것은 URL의 php이다. 해당 클래스는 선택된 강아지의 아이디 값을 보내 해당 반려견의 조건과 일치하는 장소 정보를 받아 와 출력한다.

Custom.php

```
<?php
/*
강아지의 정보를 읽어와서DB에서 조회 후 출력(맞춤)
*/

header("Content-Type:text/html; charset=UTF-8");

$conn= mysqli_connect("localhost","heremong","\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\","heremong");

mysqli_query($conn, "set names utf8");

$SB = $_POST["SB"];
$Dog_id = intval($_POST["Dog_id"]);

//$sql= "select Place_id, P_name, P_address, P_image from Place where P_name
LIKE '%{$SB}%'";

$sql= " SELECT Place_id, P_name, P_address, P_image
FROM Place, Dog
WHERE (Place.P_kg NOT BETWEEN 0 AND Dog.D_kg
OR Place.p_kg IS NULL
AND Place.P_sav <= Dog.D_sav)
AND Dog.Dog_id = '{$Dog_id}' ";

$result=mysqli_query($conn, $sql);

$rowCnt= mysqli_num_rows($result);

$arr= array(); //빈 배열 생성

for($i=0;$i<$rowCnt;$i++){
$row= mysqli_fetch_array($result, MYSQLI_ASSOC);
//각 각의row를$arr에 추가
$arr[$i]= $row;
}

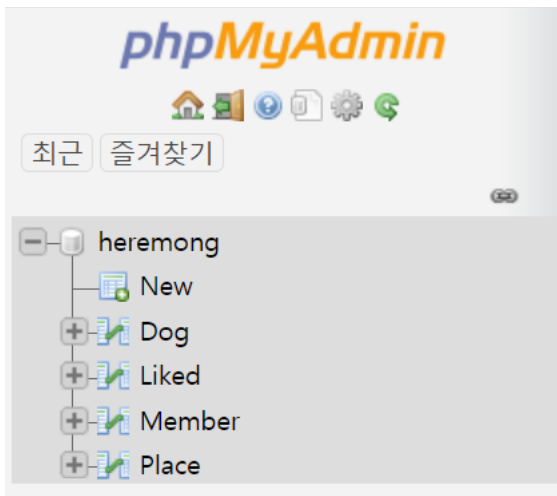
//배열을json으로 변환하는 함수가 있음.
$jsonData=json_encode($arr); //json배열로 만들어짐.
echo "$jsonData";

mysqli_close($conn);
?>
```

**Custom.php**는 선택된 강아지의 아이디 값을 받아 와 해당 반려견 조건과 같은 장소를 질의 한다. 질의 결과는 json 형태로 프로필들을 배열로 만들어 답변한다.



### III-11 DB 설계 -----



워드펫 프로젝트에서 사용된 데이터베이스의 구조는 위의 사진과 같다.

#### 11-1 Member테이블 -----

#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
1	<b>ID</b>	varchar(15)	utf8mb4_general_ci		아니오	없음		
2	<b>PW</b>	varchar(15)	utf8mb4_general_ci		아니오	없음		
3	<b>Name</b>	varchar(10)	utf8mb4_general_ci		아니오	없음		

Member 테이블은 사용자의 정보를 저장하는 테이블이다.

ID 값을 기본키로 가지며 PW와 Name 값은 NULL값을 가질 수 없다.

ID 값을 통해 사용자를 구별하며 로그인, 회원가입, 반려견 등록, 찜 등과 같은 전반적인 기능에 관여한다.

## 11-2 Place테이블

#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
1	<b>Place_id</b> 	int(10)			아니오	없음		AUTO_INCREMENT
2	<b>P_name</b>	varchar(20)	utf8mb4_general_ci		아니오	없음		
3	<b>P_address</b>	varchar(50)	utf8mb4_general_ci		아니오	없음		
4	<b>P_Lat</b>	double(17,14)			예	NULL		
5	<b>P_Lng</b>	double(17,14)			예	NULL		
6	<b>p_cate</b>	varchar(10)	utf8mb4_general_ci		아니오	없음		
7	<b>p_grade</b>	int(2)			예	1		
8	<b>p_kg</b>	int(3)			예	NULL		
9	<b>p_only</b>	tinyint(1)			예	0		
10	<b>p_image</b>	varchar(256)	utf8mb4_general_ci		예	NULL		
11	<b>p_image2</b>	varchar(256)	utf8mb4_general_ci		예	NULL		
12	<b>p_sav</b>	tinyint(1)			예	0		

Place 테이블은 장소의 정보를 저장하는 테이블이다.

Place\_id 값을 기본키로 가지며 P\_name과 P\_address 값은 NULL값을 가질 수 없다.

Place\_id 값을 통해 레코드를 구별하며 Auto\_Increment로 지정되어 자동으로 1씩 증가하여 입력된다. 해당 테이블엔 차례로 위도, 경도, 카테고리, 단계, 제한 몸무게, 전용 여부, 대표 이미지, 세부 이미지, 맵건 기능 여부를 나타내는 칼럼들이 있다.

위도, 경도는 지도에 마커를 표시하는데 쓰이며 이미지는 장소를 리사이클러뷰 혹은 상세 페이지에서 나타낼 때 쓰인다. 그 외의 항목들은 장소의 제한 조건들이며 검색, 맞춤, 카테고리 별 항목에서 장소들을 분류하는데 쓰인다. boolean 값으로 쓰이는 경우에는 default 값을 지정하여 번거로움을 덜었다.

### 11-3 Dog테이블 -----


#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
1	<b>Dog_id</b> 	int(11)			아니오	없음		AUTO_INCREMENT
2	<b>ID</b> 	varchar(15)	utf8mb4_general_ci		아니오	없음		
3	<b>D_name</b>	varchar(10)	utf8mb4_general_ci		아니오	없음		
4	<b>D_kg</b>	float(3,1)			아니오	없음		
5	<b>D_sav</b>	tinyint(1)			예	0		
6	<b>D_sex</b>	tinyint(1)			예	1		
7	<b>D_age</b>	float(3,1)			예	NULL		
8	<b>D_feature</b>	varchar(100)	utf8mb4_general_ci		예	NULL		

Dog 테이블은 반려견의 정보를 저장하는 테이블이다.

Dog\_id 값을 기본키로 가지며 D\_name과 D\_kg 값은 NULL값을 가질 수 없다.

Dog\_id 값을 통해 레코드를 구별하며 Auto\_Increment로 지정되어 자동으로 1씩 증가하여 입력된다. 해당 테이블엔 차례로 반려견 보호자, 반려견 이름, 반려견 몸무게, 맹견 여부, 성별, 나이, 특징을 나타내는 칼럼들이 있다. ID 값을 외래키로 받아 와 반려견과 해당 보호자를 알 수 있다. Dog 테이블의 Dog\_id, ID 값을 제외한 모든 값은 사용자가 '강아지 등록 페이지'에서 입력할 수 있다. boolean 값으로 쓰이는 경우에는 default 값을 지정하여 번거로움을 덜었다.

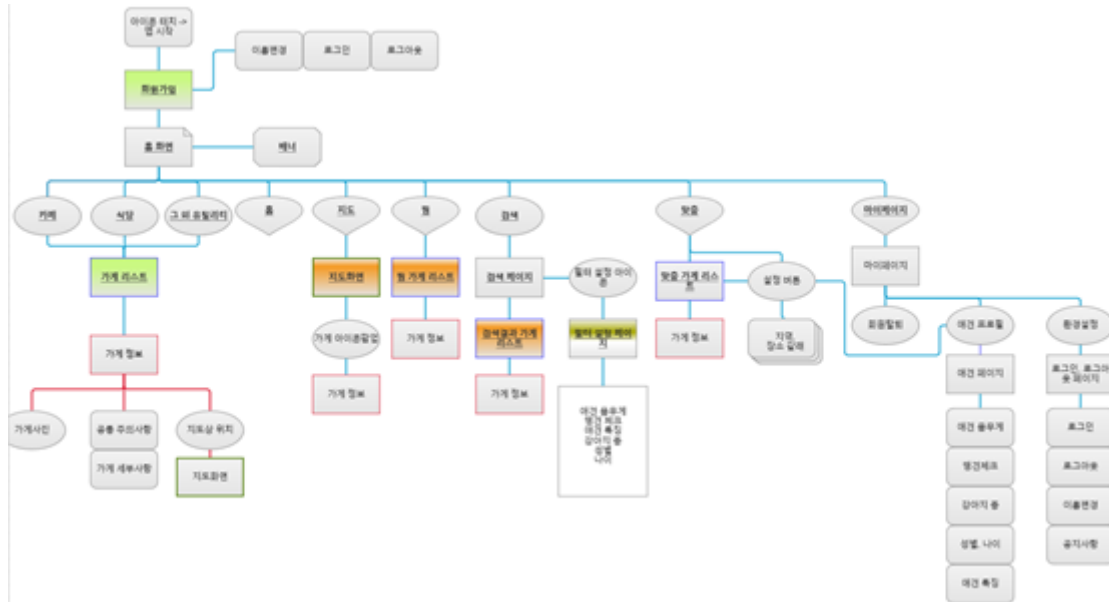
#### 11-4 Liked테이블 -----

#	이름	종류	데이터정렬방식	보기	Null	기본값	설명	추가
1	<b>L_ID</b> 	int(15)			아니오	없음		
2	<b>ID</b>	varchar(15)	utf8mb4_general_ci		아니오	없음		
3	<b>Place_id</b>	int(10)			아니오	없음		

Liked 테이블은 사용자가 '찜'한 장소의 정보를 저장하는 테이블이다.

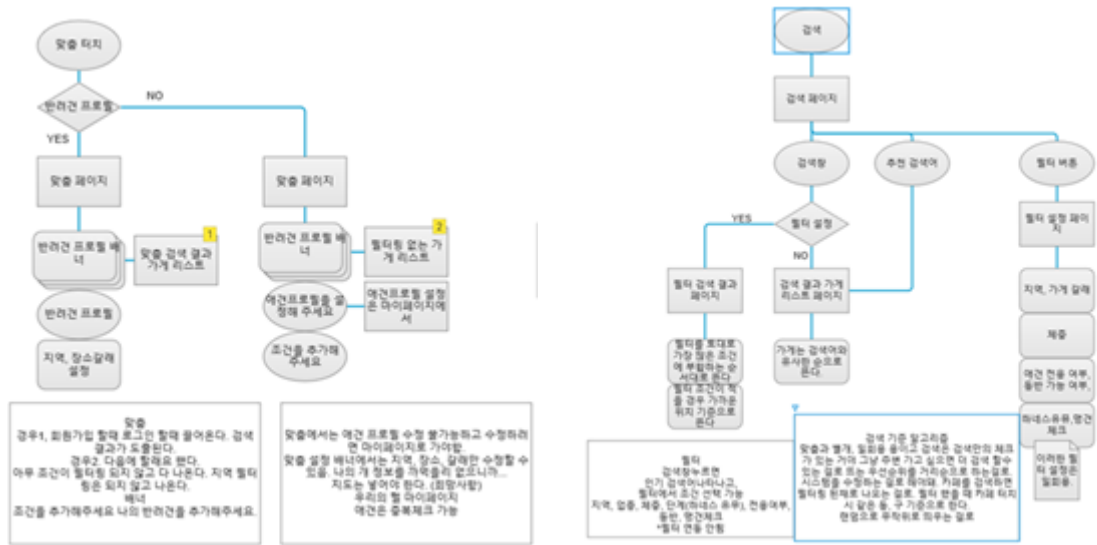
L\_ID 값을 기본키로 가진다. L\_ID 값을 통해 레코드를 구별하며 Auto\_Increment로 지정되어 자동으로 1씩 증가하여 입력된다. ID 값과 Place\_id 값을 외래키로 받아 와 사용자와 사용자가 찜한 장소를 알 수 있다. 찜 여부는 사용자가 장소 아이템의 '하트' 버튼을 통해 입력할 수 있다.

-----



1. 원시 코드를 통해 애플리케이션의 구조를 이해 (플로우 차트 참고)
2. 검증 기준을 설정
3. 각 경로를 구동시키는 테스트 데이터를 준비

## IV-2 테스트 방법



### ‘루프 테스트’ 방법 선택

#### - 연속 반복

:반복 구조가 서로 독립적이면 ‘단순 반복’

:반복 구조가 어느 한쪽이 포함된 관계라면 ‘중첩된 반복’

#### - 비구조화 반복

:구조적 반복 형태로 변경하여 테스트

상태 기반 테스트를 통해 같은 입력을 통해 같은 값을 보이는지 확인

통합 테스트는 연쇄식 통합을 채용한다.

- 특정 기능을 수행하는 모듈의 최소 단위로부터 시작

- 시스템 분리 개발에 용이하며 초기에 빠르게 시스템 구조를 세울 수 있음

## 2-1 기능 테스트-----

### - 회원가입

- 아이디/패스워드/re패스워드 모두 정상일 경우 “회원가입 완료”라는 메시지 출력
- 아이디/패스워드가 정상, re 패스워드가 비정상일 경우 “올바른 re 패스워드가 아닙니다.”라는 메시지 출력
- 아이디가 정상, 패스워드가 비정상 혹은 예외일때는 re패스워드의 정상/비정상 여부에 상관 없이 “옳지 않은 패스워드입니다”
- 아이디가 값의 명세를 만족하지 않거나 예외일 경우 “올바른 ID가 아닙니다.”라는 메시지 출력
- 아이디가 다른 사용자와 겹칠 경우 “이미 존재하는 ID입니다.”라는 메시지 출력

### - 로그인

- 아이디/패스워드가 정상일 경우 로그인이 완료된 채로 홈화면으로 이동
- 아이디는 정상, 패스워드는 해당 아이디와 매치되는 값이 아닌 경우 “옳지 않은 패스워드입니다” 라는 메시지 출력
- 존재하지 않는 아이디일 경우 “존재하지 않는 아이디”라는 메시지 출력

### -지도

- 등록된 장소들이 알맞은 위치에 팝업 표시가 되는가?
- 해당 장소의 팝업을 터치했을 때 해당 장소로의 길 찾기가 가능한가?
- 검색 조건을 입력하였을 때 조건에 맞는 장소만 표시되는가?

### -반려동물관리

- 사용자는 자신의 반려동물 정보를 추가, 삭제, 수정이 가능한가?

### -검색

- 카테고리를 선택했을 때 해당되는 카테고리의 장소만 표시되는가?
- 조건 필터 설정 후 검색할 시 해당 조건을 만족하는 장소만 표시되는가?

### -맞춤

- 카테고리를 선택했을 때 해당되는 카테고리의 장소만 표시되는가?
- 조건 필터 설정 후 검색할 시 해당 조건을 만족하는 장소만 표시되는가?
- 애견 프로필을 눌렀을 때 등록된 애견 조건이 알맞게 설정되는가?

### -찜

- 장소의 하트버튼을 누르면 해당 장소가 찜 목록에 저장되어 찜 항목에 출력되는가?
- 장소의 하트버튼을 취소하면 해당 장소가 찜 목록에서 삭제되어 찜 항목에 출력되지 않는가?

## 2-2 성능 테스트-----



- 데이터베이스에 구조도에 설계에 알맞은 값이 등록되는가?
- 반응 시간과 처리 속도가 3초 이내인가?
- 처리율이 1건당 3초 이내인가?
- 3대 이상의 기기 혹은 3명 이상의 사용자가 동시 접속 가능한가?
- Oreo 이상의 운영체제의 안드로이드 스마트폰에서 구동되는가?



## 2-3 UI 테스트 -----



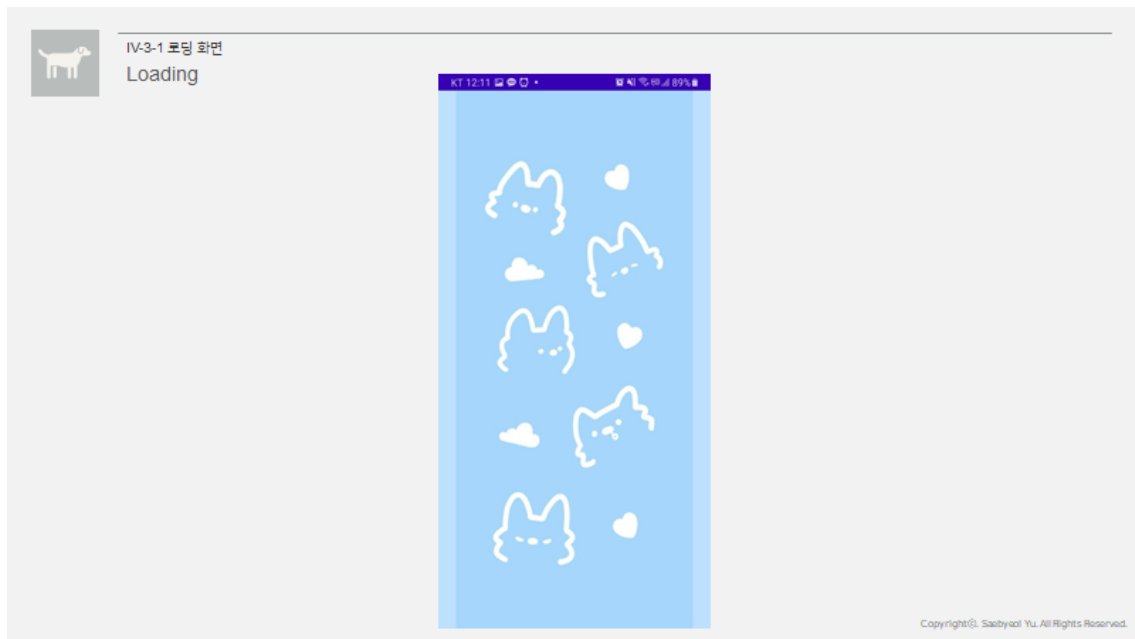
- 한 페이지에 나타나는 정보를 나타내는 텍스트 크기가 알맞은가?
- 사용자가 아무런 정보와 사전지식 없이 모든 기능을 활용할 수 있는가?

## 2-4 인수테스트 -----

- 시스템을 당장 사용할 수 있도록 모든 준비가 되어 있는가?
- 사용자의 환경에서 모든 기능, 비기능적 요구에 맞게 실행되는가?

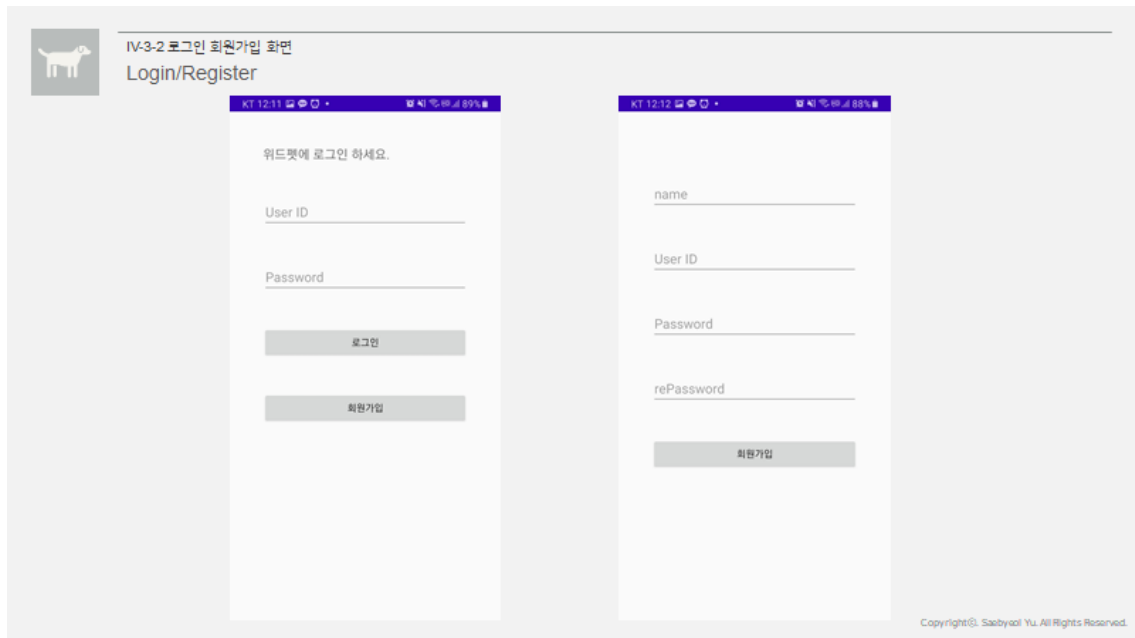
## IV-3 테스트 결과 -----

### 1. Loading



애플리케이션 실행 후 3초 뒤에 로그인/회원가입 페이지가 열린다.

### 2. Login/Resister



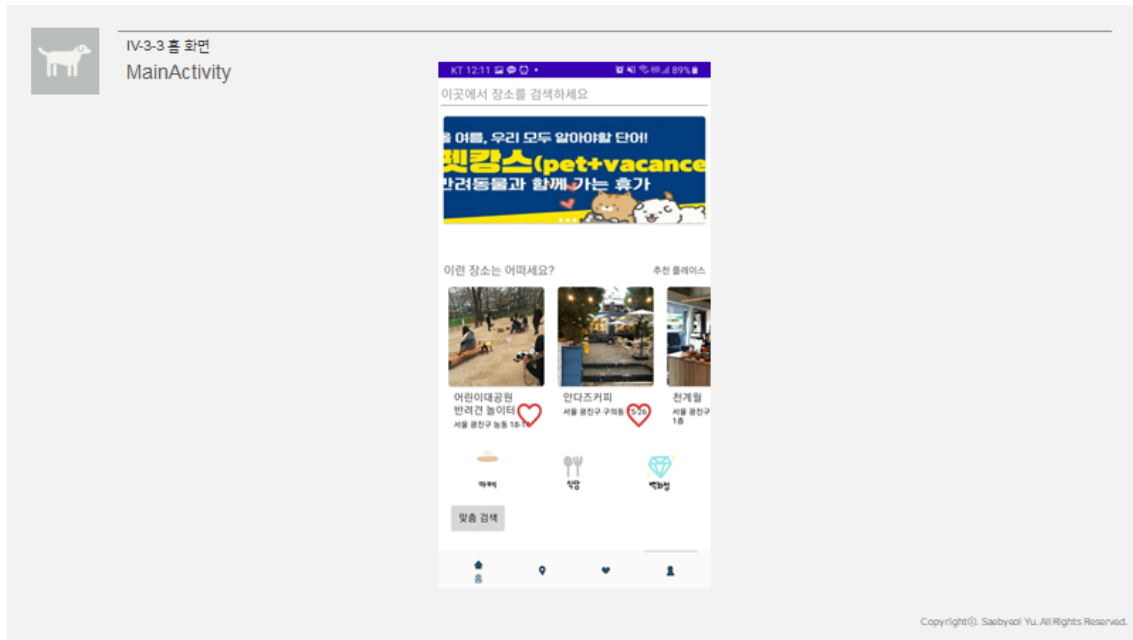
해당 페이지에서 회원가입 버튼을 눌러보면 이름, 아이디, 비밀번호, 비밀번호 확인의 입력을 통해 회원가입이 가능하다.

해당 유저의 정보는 데이터베이스에 실시간으로 반영된다.

가입을 성공적으로 마치게 되면 토스트 메시지와 함께 로그인 페이지로 넘어간다.

회원가입 시 입력한 정보로 로그인을 하면 홈 화면으로 이동한다.

### 3. MainActivity (배너, 추천장소, 카테고리별 목록)



홈 화면 상단에는 펫티켓 관련 사항들이 배너로 안내되어 있다.

추천 장소가 리사이클러뷰에 의해서 보여진다.

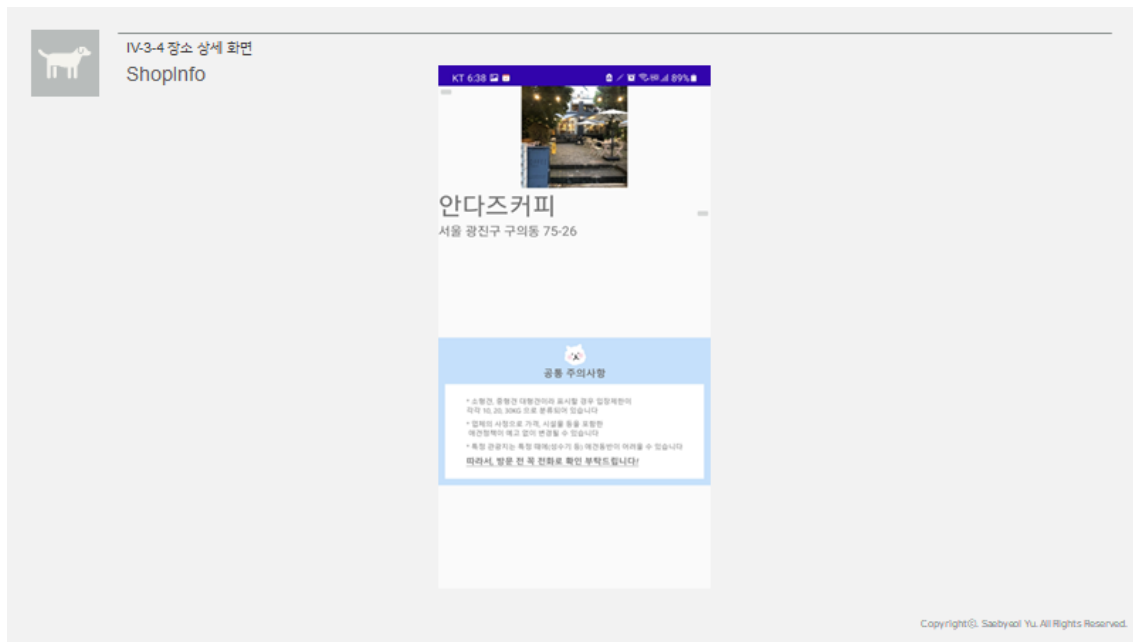
장소들의 카테고리는 카페, 식당, 백화점으로 분류되어 있으며, 각 버튼을 터치 시 해당 종류의 장소 확인이 가능하다.

각 결과 페이지에서 하트 버튼 터치를 통한 찜 등록이 가능하다. (하지만, 무결성 확인x)

해당 뷰를 터치 시 가게 상세 페이지로 이동한다.

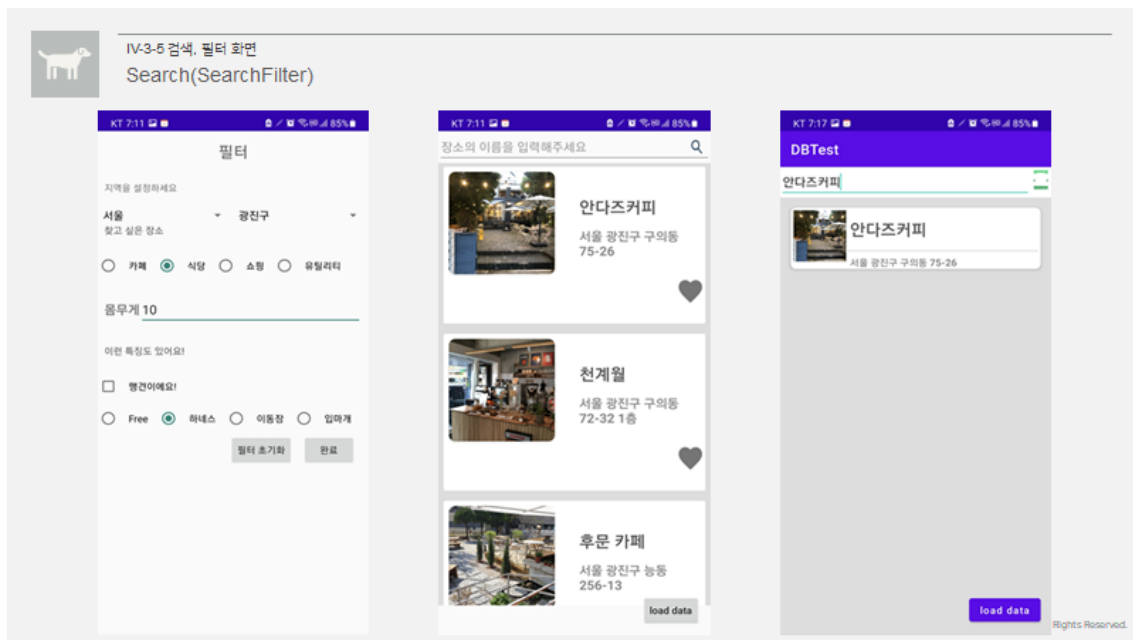
하단바의 홈, 지도, 찜, 마이페이지의 터치를 통해 각 액티비티로의 이동이 가능하다.

#### 4. ShopInfo



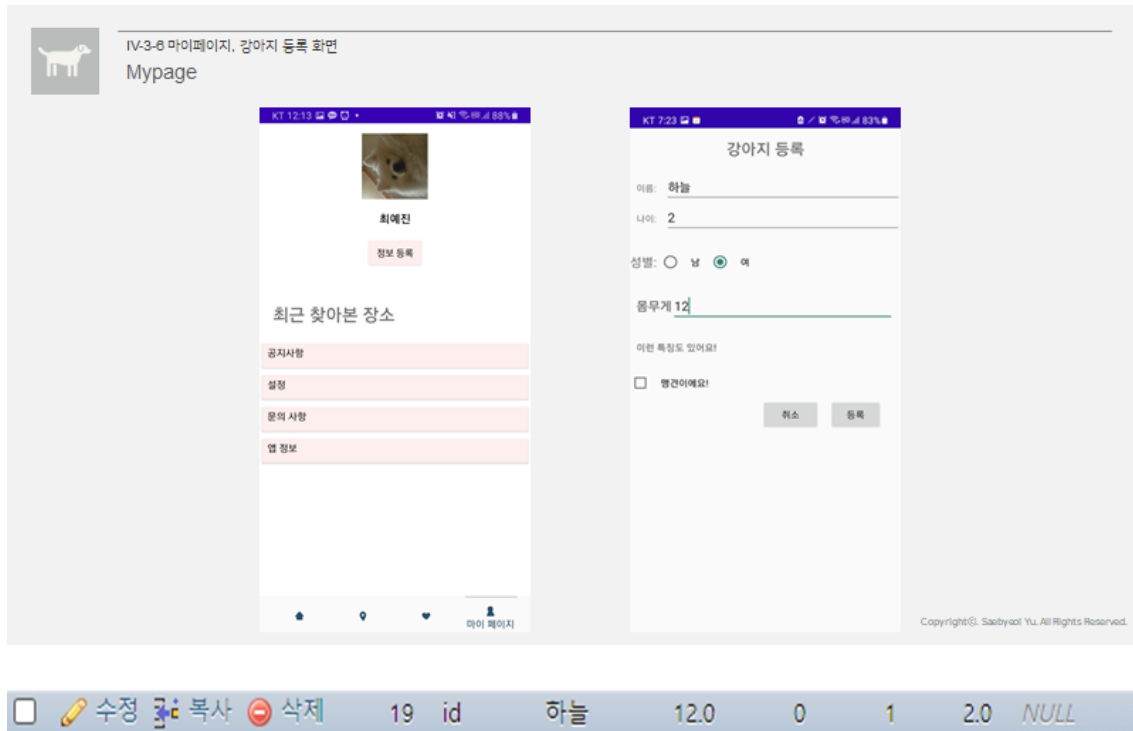
아이템 클릭 시 해당 가게의 이름, 위치, 사진의 정보 확인이 가능하다.

#### 5. Search (SearchFilter)



돋보기 버튼을 터치 시 검색 필터 적용이 가능하다.  
지역, 카테고리, 제한 몸무게, 맹견 출입 여부, 제약 조건의 필터링이 가능하다.  
설정된 필터에 따라 그에 해당하는 가게 검색이 가능하다.

## 6. Mypage (강아지 등록)

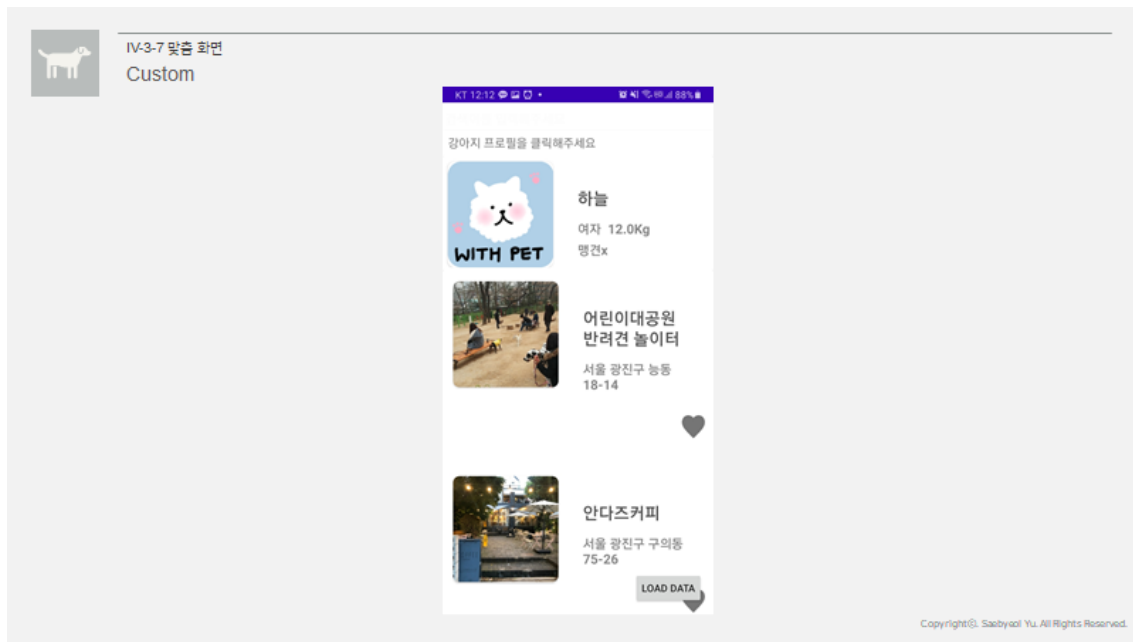


마이페이지에서 강아지 등록이 가능하다.

등록 버튼을 터치 시 강아지 등록 페이지로 이동하며, 강아지의 이름, 나이, 성별, 맹견 여부의 정보를 등록할 수 있다.

등록된 정보는 데이터베이스에 저장된다.

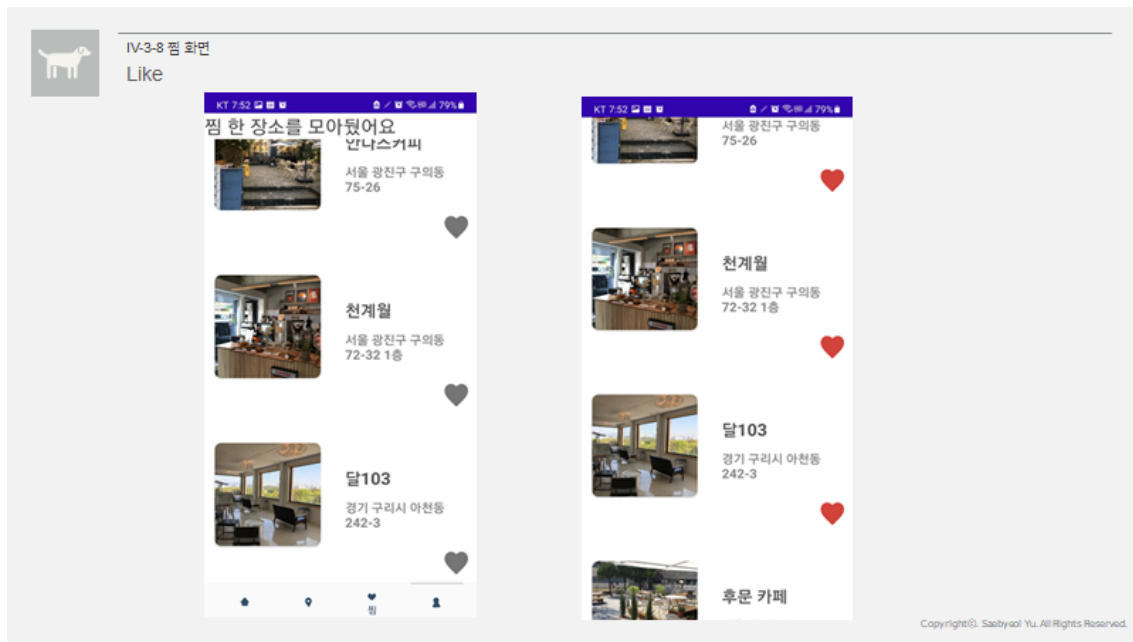
## 7. 맞춤



등록된 강아지 정보는 맞춤에서 사용할 수 있다.

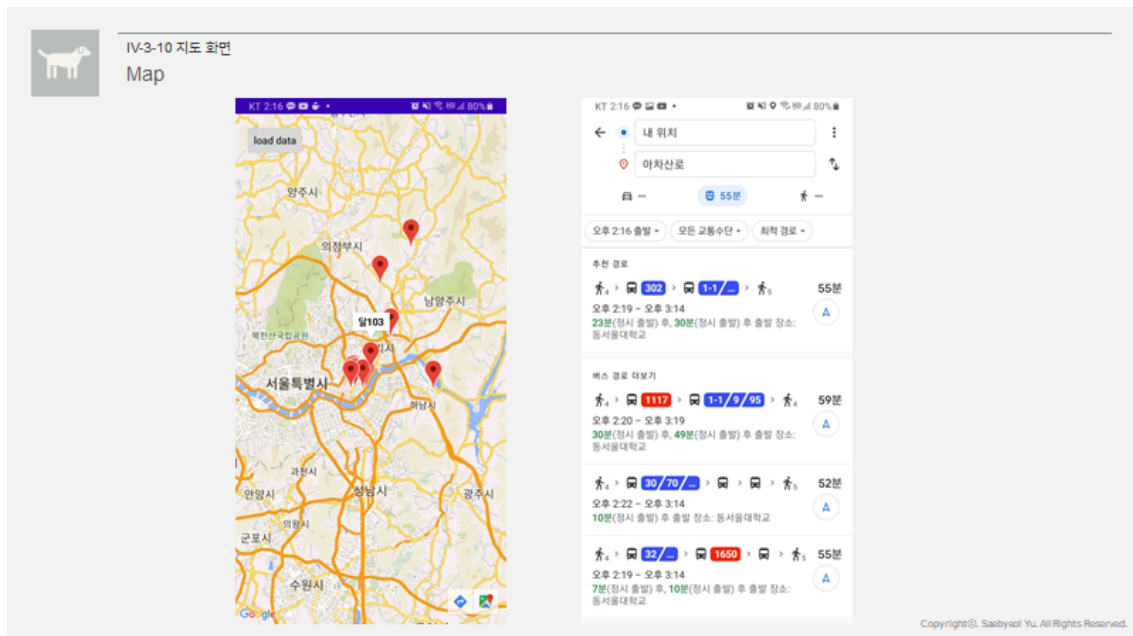
등록된 강아지 프로필을 선택하면 별도의 선택 없이 한 번에 조건이 설정되며, 그에 맞는 가게를 검색해준다.

## 8. Like



하트 버튼을 통해 찜으로 등록된 장소들의 목록이 나타난다. 그러나 무결성 확인에 있어 어려움이 있다.

## 9. Map



가게의 위치들을 지도를 통해 보여준다.  
지도에서 원하는 가게를 선택 시 가게까지의 길 찾기를 해준다.