UNIVERSITÀ
DEGLI STUDI
FIRENZE

DISIT
DISTRIBUTED SYSTEMS
AND INTERNET
TECHNOLOGIES LAB

Cultural City Android Application

Università Degli Studi Di Firenze

Master Degree in Computer Engineering

Security and Knowledge Management course **Prof. Bellini Pierfrancesco**

Anno Accademico 2018-2019

App Overview

This app was developed by two students of master's degree in Computer Engineering at the University of Florence, Collini Enrico and Di Martino Andrea.

This project has been assigned as a Security and Knowledge Management's course exam, taught by professor Bellini Pierfrancesco.

Thanks to km4City's Ontology resources and Google Maps API, throw this app, you can find places of cultural interest in the city of Florence through an interactive application.

1 - Paper Structure

This paper is a detailed description about the **Cultural City** Android Application with also specific code implementation details.

The first part describes the **software architecture**, provides an **UML diagram** for the Java class developed and shows the development environment used: Android Studio.

Afterwards it presents in detail the **Data Management Module** with specific focus on the **Km4City Ontology** of the Disit Lab of the university.

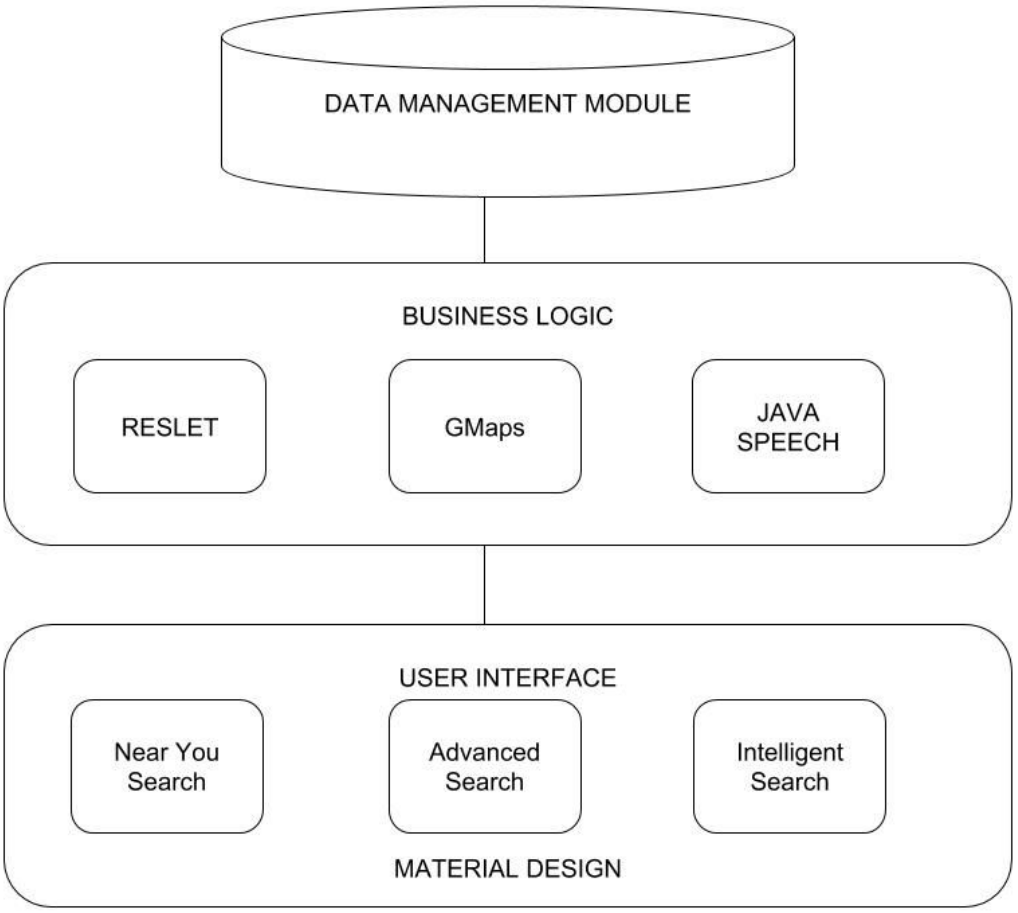
The **Business Logic** module is then detailed explaining the tools and libraries used:

- **Restlet and API table**
- **Gmaps**
- **Gson**

The next part details the **User Interface** with the presentation of the design used (**Material Design**) and then shows some **screens** of the application and a **video overview** of the Intelligent Search.

The last part proposes some **Conclusion** and some **future developments.**

2 - Software Architecture



The software architecture implemented is based on a three layer schema:

- **Data Management Module:** As the name suggests this module is about the datas of the application. The datas has been taken from the **Km4city Ontology** of the Disit Lab of the Università Degli Studi di Firenze.

- **Business Logic:** In this module there are the main tools and libraries used in the application: the **Relset** module that allowed us to connect to the ontology through API calls, the **Gmaps** library that we used for the application map of Florence which is the main element of our application,

and the **Java Speech** that enables the application to read the description of a specific cultural point

- **User Interface:** the user interface is based on the Material Design. In this module are also described the app functionalities in detail:

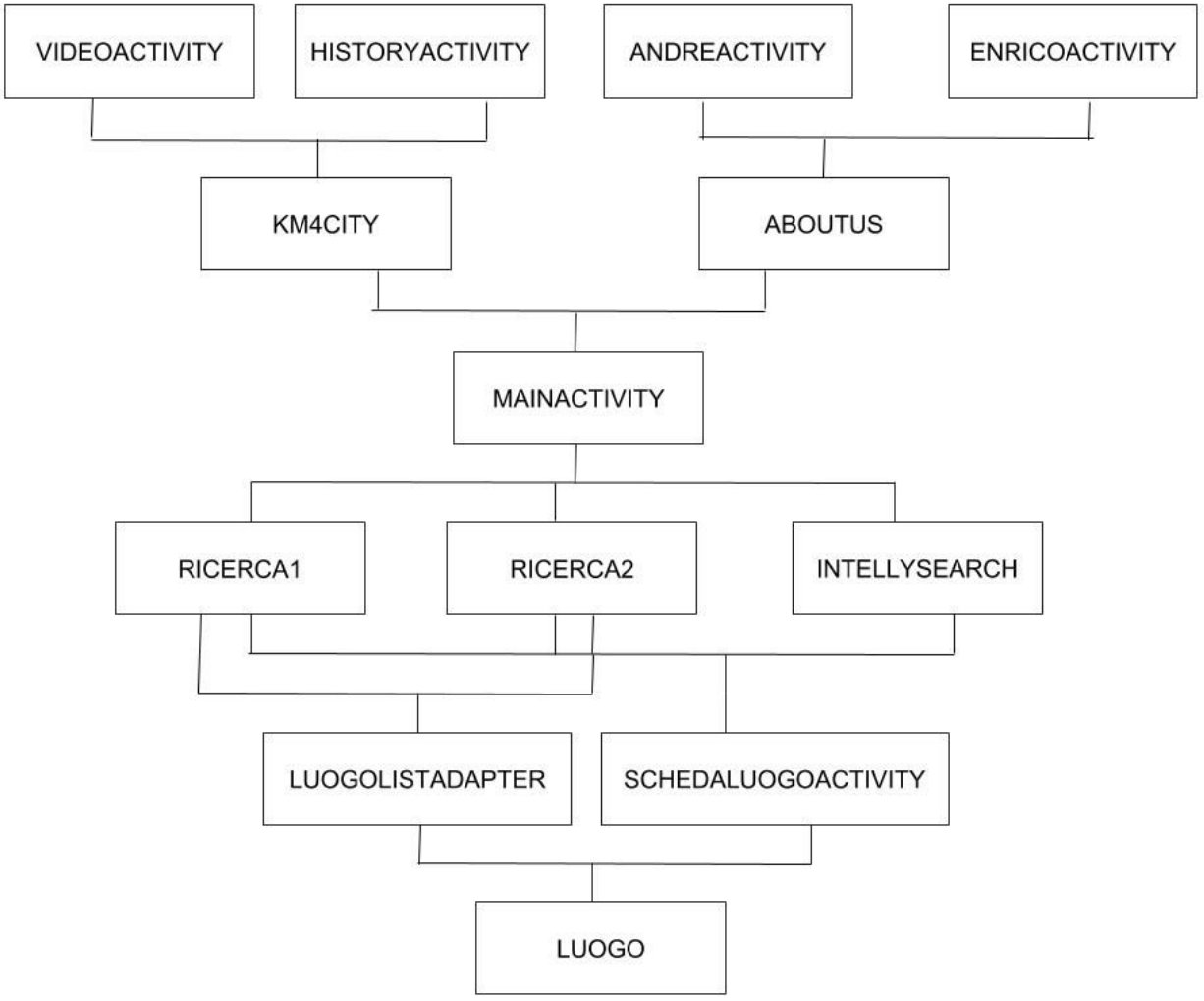
  - **Near You Search** This service finds all the cultural point of interests in the near by and shows it in a list with the icon of the specific category. If you are interested in a specific element you can click it to see the point card. In this one are included all the information included the description and th category. There are also two buttons: one brings you to the map where you can see its position, the other instead explains the description of the specific point of interest.

  - **Advanced Search** This functionality allows you to find all the cultural point of interest of a specific place throw the search bar where you can insert the address. You can also choose the radius of influence (100-1000 meters) and the category. Once the request is made the application will show you the list of all the resulting cultural points as in the Near You Search.

  - **Intelligent Search**The intelligent Search is the main feature of this application. It consists in an interactive map that tracks your position and displays all the cultural points in the near by. If there are some it will automatically explains to you the description of the closer cultural point. Assuming that you are walking in the center of Florence as soon as your position change so that you are closer to a new cultural point the application will automatically elaborate this and will explain to you the description of the new cultural point.

2.1 -

2.2 - UML Class Diagram



The **Mainactivity** Java Class is the one responsible for the front page of the application. It allows the user to navigate inside our application through the use of buttons. There connects this activity to:

- **Ricereca1**: this class represents the Near You Search. It uses the user position to show through the **Luogolistadapter** class a list of the cultural poitns in the nearby. The user selecting a specific element of the list is linked to the specific **SchedaluogoActivity.**

- **Ricerca2**: this class represents the Advanced Search. Through some input fields the user can insert the address of specific streets of Florence of the name of a Cultural point. After selecting a radius (from 100m to 1km) the user is linked to the Luogolistadapter view with the results for the search, and as in Ricerca1 when an element is selected the application shows its specific scheda.

- **Intellysearch**: This class is very important for the application. This is based on an interactive **map** on which are some markers representing the cultural points in a given radius of 400 meters. The user can simply walk around the city with this application opened and when it is near some cultural point the application will talk about the description of the specific cultural point.

- **Km4city:** This class contains a description of Km4city and links this activity page to the **history** page and to the **video** of this project.

- **Aboutus:** This activity contains a description about us and then through two imagebuttons links the user to our page with some further informations.

2.3 - Android Studio

In this project we choose, to construct our app, the most famous IDE for android applications: Android Studio.

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components

- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations

- Support for building Android Wear apps

- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine

- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

-

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "Java 7 language features and a subset of Java 8 language features that vary by platform version.
External projects backport some Java 9 features.
While IntelliJ that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

3 - DATA MANAGEMENT MODULE

3.1 - Km4City

The motto:

"Km4City platform is a Smart City tools for implementing the city vision, monitoring the city evolution, diffusely providing new services for improving quality of life of city users, for the city economic grow; stimulating city users; since an attractive city is a 'city that produces' in which users are happy and proud. "

Km4City is a knowledge base and a research line of the Disit Lab of the Università Degli Studi Di Firenze. It started in 2013 with a generic name: Smart City Ontology and it was named Km4city later.

The major data sources are:

- Open Data of Florence
- Lamma Weather
- Cultural Activities
- Schools
- Commercial Activities

For the development of our application the datas more important are the Cultural ones. We obtained the information needed exploring the results of the API calls through the portal https://www.km4city.org/swagger/external/ .

The tool (Restlet) we used to connect and get all the json datas is described in the next section.

API Call	API TYPE		JSON RESULT
Search given lat lon with all the categories	GET	http://servicemap.disit.org/WebAppGrafo/api/v1/?selection=43.772895;11.255235&categories=Museum;Botanical_and_zoological_gardens;Churches;Cultural_centre;Cultural_sites;Historical_buildings;Library;Monument_location;Photographic_activities;Squares;Theatre;&maxResults=0&maxDists=2&lang=it&format=json";	{ "Services":{ "fullCount": 689, "type": "FeatureCollection", "features": [{ "geometry": {"type": "Point","coordinates":[11.24899959564209,43.77519989013672]}, "type": "Feature", "properties": { "name": "COMPLESSO_MONUMENTALE_DI_SANTA_MARIA_NOVELLA", "tipo": "Luogo_monumento", "typeLabel": "Luogo monumento", "serviceType": "CulturalActivity_Monument_location", "distance": "0.0444", "serviceUri": "http://www.disit.org/km4city/resource/a421aa403641c0f1f4571047fbb6ab09", "photoThumbs": [], "multimedia": "" }, "id": 1 }, { "geometry": {"type": "Point","coordinates":[11.24870014190674,43.77519989013672]}, "type": "Feature", "properties": { "name": "Biblioteca Domenicana", "tipo": "Biblioteca", "typeLabel": "Biblioteca", "serviceType": "CulturalActivity_Library", "distance": "0.0505", "serviceUri": "http://www.disit.org/km4city/resource/1f765950114b7d2a25d451d3b73f29e8", "photoThumbs": [],

			"multimedia": ""
			},
			"id": 2
			},
			{
			"geometry": {"type":
			"Point","coordinates":[11.249400
			13885498,43.77460098266602]},
			"type": "Feature",
			"properties": {
			"name": "BASILICA DI SANTA MARIA
			NOVELLA",
			"tipo": "Museo",
			"typeLabel": "Museo",
			"serviceType":
			"CulturalActivity_Museum",
			"distance": "0.1155",
			"serviceUri":
			"http://www.disit.org/km4city/re
			source/9b2c978c164186fc23d74a1c9
			f922dd3",
			"photoThumbs": [],
			"multimedia": ""
			},

More informations about a place passing its uri found in the results of the above API call	GET	http://servicemap.disit.org/WebAppGrafo/api/v1/?serviceUri=http://www.disit.org/km4city/resource/6f5f9921c9eb19c969e75aa1b981554f	{ "Service":
			{"type":
			"FeatureCollection",
			"features": [
			
			{ "geometry": {
			"type": "Point",
			"coordinates": [
			11.2556, 43.7728]
			},
			"type": "Feature",
			"properties": {
			"name": "Campanile di
			Giotto",
			"typeLabel": "Museum",
			"serviceType":
			"CulturalActivity_Museum",
			"phone": "",
			"fax": "",
			"website": "",
			"province": "FI",
			"city": "FIRENZE",

```
    "cap": "50122",
    "email": "",
    "linkDBpedia":
["http://it.dbpedia.org/resource/Campanile_di_Giotto"],
    "note": "",
    "description": "84 meters in height and about 15 meters wide, the Giotto s Bell Tower is the most eloquent testimony of fourteenth-century Florentine Gothic architecture which, though with a vertical momentum, does not abandon the principle of solidity. Faced with white, red and green marbles like the Cathedral, the majestic square-base bell tower is considered the most beautiful in Italy. Begun by Giotto in 1334, it was continued by Andrea Pisano, who completed the first two levels respecting Giotto s project. The bell tower was completed in 1359 by Francesco Talenti, whose merits include the Sienese style double mullioned windows and the large triple lancet windows, thus making the construction elegantly Gothic, though maintaining the classical composition of the whole.",
    "description2": "",
```

			"multimedia": "http://www.florenceheritage.it/mobileApp/immagini/ficard/219.jpg", "serviceUri": "http://www.disit.org/km4city/resource/6f5f9921c9eb19c969e75aa1b981554f", "address": "PIAZZA DEL DUOMO", "civic": "18-20", "wktGeometry": "POINT (11.25562 43.772814)", "photos": [], "photoThumbs": [], "photoOrigs": [], "avgStars": 0.0, "starsCount": 0, "comments": []}, "id": 1 }
APi call to get the lat and long passing an address of florence or the name of a cultural point	GET	https://servicemap.disit.org/WebAppGrafo/api/v1/location/?search=Duomo&Firenze&maxResults=1	{"type": "FeatureCollection", "count": 694, "features": [{ "geometry": { "type": "Point", "coordinates": [11.882455,43.466927] }, "type": "Feature", "properties": { "serviceUri": "http://www.disit.org/km4city/resource/84dc5b1c64ecb275e78800c51667c8ee", "serviceType": "CulturalActivity_Library", "name": "ARCHIVIO CAPITOLARE DEL DUOMO", "city": "AREZZO", "score": "11.755986", "id": 1}}]}

4 - BUSINESS LOGIC

4.1 - RESTLET

Restlet is a high-level Http Client Framework written in Java.
It is used in our application to get the datas from the Km4city ontology through the API get requests.

In the following page there are some code snippets that describes the way the connection is made and how the results are managed.

```
public class GetUriDatas extends AsyncTask<String, Void, String> {
    private int response_code;

    protected String doInBackground(String... params) {
        String jsonResponse = null;
        ClientResource cr;
        String URI = "https://servicemap.disit.org/WebAppGrafo/api/v1/location/?search=" + indirizzo + "&maxResults=1";
        cr = new ClientResource(URI);
        try {
            jsonResponse = cr.get().getText();
            response_code = cr.getStatus().getCode();
        } catch (ResourceException | IOException e) {
            jsonResponse = " Error caused by resourceException: " + cr.getStatus().getCode() + " - " + cr.getStatus().getDescription() + " - ";
            response_code = cr.getStatus().getCode();
        }
        //System.out.println(jsonResponse);
        return jsonResponse;
    }
```

```
protected void onPostExecute(String res) {
    if (res != null) {
        if (response_code == 404) {
            Toast.makeText(getApplicationContext(), text: "Nessun risultato", Toast.LENGTH_LONG).show();
        } else if (response_code == 1000) {
            Toast.makeText(getApplicationContext(), text: "Errore di rete", Toast.LENGTH_LONG).show();
        } else {
            try {

                JSONObject joo = new JSONObject(res);
                JSONArray jFeature = joo.getJSONArray( name: "features");
                for (int i = 0; i < jFeature.length(); i++) {
                    JSONObject object = jFeature.getJSONObject(i);
                    JSONObject geo = object.getJSONObject("geometry");
                    JSONArray coo = geo.getJSONArray( name: "coordinates");
                    Object latitude= coo.get(0);
                    Object longitude = coo.get(1);
                    Double lat = (Double) latitude;
                    Double lon = (Double) longitude;
                    latSearch = lat.toString();
                    lonSearch = lon.toString();
                    Toast.makeText( context: Ricerca2.this, text: "latitudine e longitudine trovate: " + latSearch + lonSearch, Toast.LENGTH_SHORT).show();

                }

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

4.2 - GOOGLE MAPS

Google Maps is a web mapping service developed by Google. It offers satellite imagery, aerial photography, street maps, 360° panoramic views of streets (Street View), real-time traffic conditions (Google Traffic), and route planning for traveling by foot, car, bicycle and air (in beta), or public transportation.

In this project the google maps API was used to enable the visualization of every cultural sites on the map with a simple marker. User make a request to the km4city API, the JSON result is converted in informations which are represented on the map powered by google maps API.

We can observe the process of representation on a map thanks to the following code:

```
public class MarkerLuogo extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_marker_luogo);
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync( onMapReadyCallback: this);


    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        String latitudine = getIntent().getExtras().getString( key: "latitudine");
        String longitudine = getIntent().getExtras().getString( key: "longitudine");
        String nomeLuogo = getIntent().getExtras().getString( key: "nome");
        double lat = Double.parseDouble(latitudine);
        double lon = Double.parseDouble(longitudine);
        LatLng luog = new LatLng(lon,lat);
        mMap.addMarker(new MarkerOptions().position(luog).title(nomeLuogo));
        mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(luog,  v: 17.0f));



    }
```

In this Google Maps Activity we represent all the cultural places that are the result of the request to the km4City API. This activity is used by the activity Ricerca1 that find all the cultural places around the current position of the user in a range of 600 meters.

4.3 - GSON

Gson (also known as Google Gson) is an open-source Java library to serialize and deserialize Java objects to (and from) JSON.

Gson can handle collections, generic types and nested classes (including inner classes, this can not be done by default though)

When deserializing, Gson navigates the type tree of the object being deserialized. This results in ignoring extra fields present in the JSON input.

User can write a custom serializer and/or deserializer so that they can control the whole process and even (de)serialize instances of classes for which the source code is not accessible.

User can write an InstanceCreator which allows them to deserialize instances of classes without a defined no-args constructor.

Gson is highly customizable, you can specify:

- Compact/pretty printing (whether you want compact or readable output)

- How to handle null object fields - by default they are not present in the output

- Rules of what fields are intended to be excluded from (de)serialization

- How to convert Java field names

From Java API

com.google.gson

Class Gson

- java.lang.Object
    - com.google.gson.Gson

```
public final class Gson
extends Object
```

This is the main class for using Gson. Gson is typically used by first constructing a Gson instance and then invoking toJson(Object) or fromJson(String, Class) methods on it. Gson instances are Thread-safe so you can reuse them freely across multiple threads.
You can create a Gson instance by invoking new Gson() if the default configuration is all you need. You can also use GsonBuilder to build a Gson instance with various configuration options such as versioning support, pretty printing, custom JsonSerializers, JsonDeserializers, and InstanceCreators.

In our project we use this API to get the response from the km4City websites. In the activity called Ricerca1, for example, we made a restelet request to km4city websites and we got a json response. After this we extracted the information contained in the json answer and inserted in an array of "luoghi":

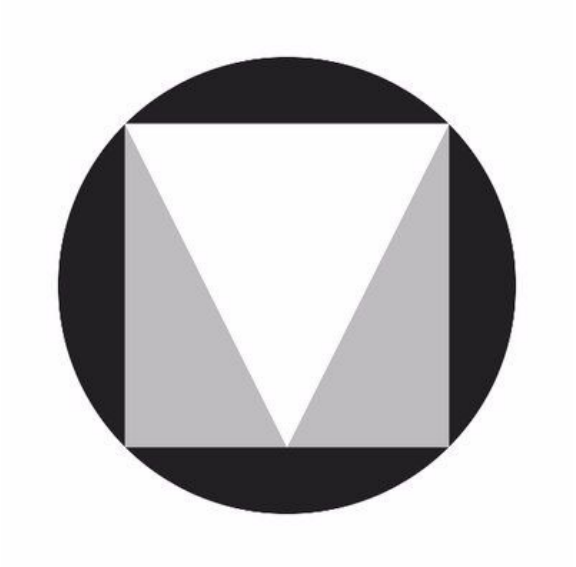
```
            JSONObject joo = new JSONObject(res);
            JSONObject jService = joo.getJSONObject("Services");
            JSONArray jFeature = jService.getJSONArray( name: "features");
            for (int i = 0; i < jFeature.length(); i++) {
                JSONObject object = jFeature.getJSONObject(i);

                JSONObject geo = object.getJSONObject("geometry");
                JSONArray coo = geo.getJSONArray( name: "coordinates");
                Object latitude= coo.get(0);
                Object longitude = coo.get(1);
                Double lat = (Double) latitude;
                Double lon = (Double) longitude;

                JSONObject prop = object.getJSONObject("properties");
                String nomeLuogo = prop.getString( name: "name");
                String tipoLuogo = prop.getString( name: "tipo");
                String distanzaLuogo = prop.getString( name: "distance");
                String uriLuogo = prop.getString( name: "serviceUri");
                String multimediaLuogo = prop.getString( name: "multimedia");



                Luogo luog = new Luogo(nomeLuogo, lat, lon, tipoLuogo, distanzaLuogo, uriLuogo, multimediaLuogo);
                luoghi.add(luog);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

```
//adapter = new ListAdapter(Ricerca1.this, luoghi);
//listView.setAdapter(adapter);
//listView.setVisibility(View.VISIBLE);

ArrayList<Luogo> locations = new ArrayList<~>(luoghi.size());
for (int i = 0; i<luoghi.size(); i++){
   locations.add(i,luoghi.get(i));
}
LuogoListAdapter adapter = new LuogoListAdapter( context: Ricerca1.this, R.layout.adapter_view_layout, locations);

final ListView mylist = (ListView) findViewById(R.id.listView1);
//final ArrayAdapter<String> adp = new  ArrayAdapter(Ricerca1.this, R.layout.adapter_view_layout,nomi);
mylist.setAdapter(adapter);
```

5 - CLIENT INTERFACE

5.1 - MATERIAL DESIGN

MATERIAL DESIGN



Material Design (codenamed Quantum Paper) is a design language that Google developed in 2014. Expanding on the "card" motifs that debuted in Google Now, Material Design uses more grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows.

Google announced Material Design on June 25, 2014, at the 2014 Google I/O conference.

Designer Matías Duarte explained that, "unlike real paper, our digital material can expand and reform intelligently. Material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch." Google states that their new design language is based on paper and ink but implementation takes place in an advanced manner. Material Design can be used in all supported versions of Android, or in API Level 21 (Android 5.0) and newer (or for older via the v7 appcompat library), which is used on virtually all Android devices manufactured after 2009. Material Design will

gradually be extended throughout Google's array of web and mobile products, providing a consistent experience across all platforms and applications. Google has also released application programming interfaces (APIs) for third-party developers to incorporate the design language into their applications.The main purpose of material design is creation of new visual language that combines principles of good design with technical and scientific innovation.

In this project we construct a kind of API based on a series of XML file like this:

```
<?xml version="1.0" encoding="utf-8"?>
<shape android:shape="oval" xmlns:android="http://schemas.android.com/apk/res/android">
<solid android:color="@color/materialgreen"></solid>
</shape>
```

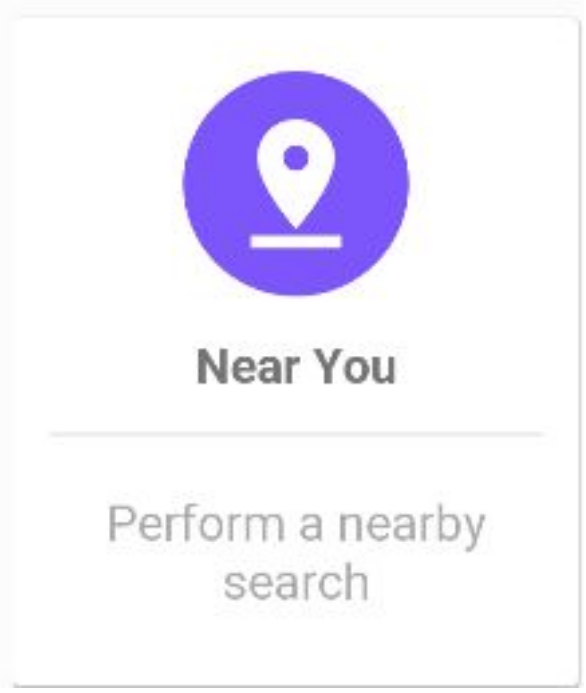
Thanks to this oval colored shapes and some google default icon we have built our Material Design API. The colors we used are those suggested by google.

An example is this:

```
<android.support.v7.widget.CardView
  android:id="@+id/nearYou"
  android:foreground="?android:attr/selectableItemBackground"
  android:clickable="true"
  android:layout_width="160dp"
  android:layout_height="190dp"
  android:layout_margin="10dp">
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">
    <ImageView
      android:layout_width="64dp"
```

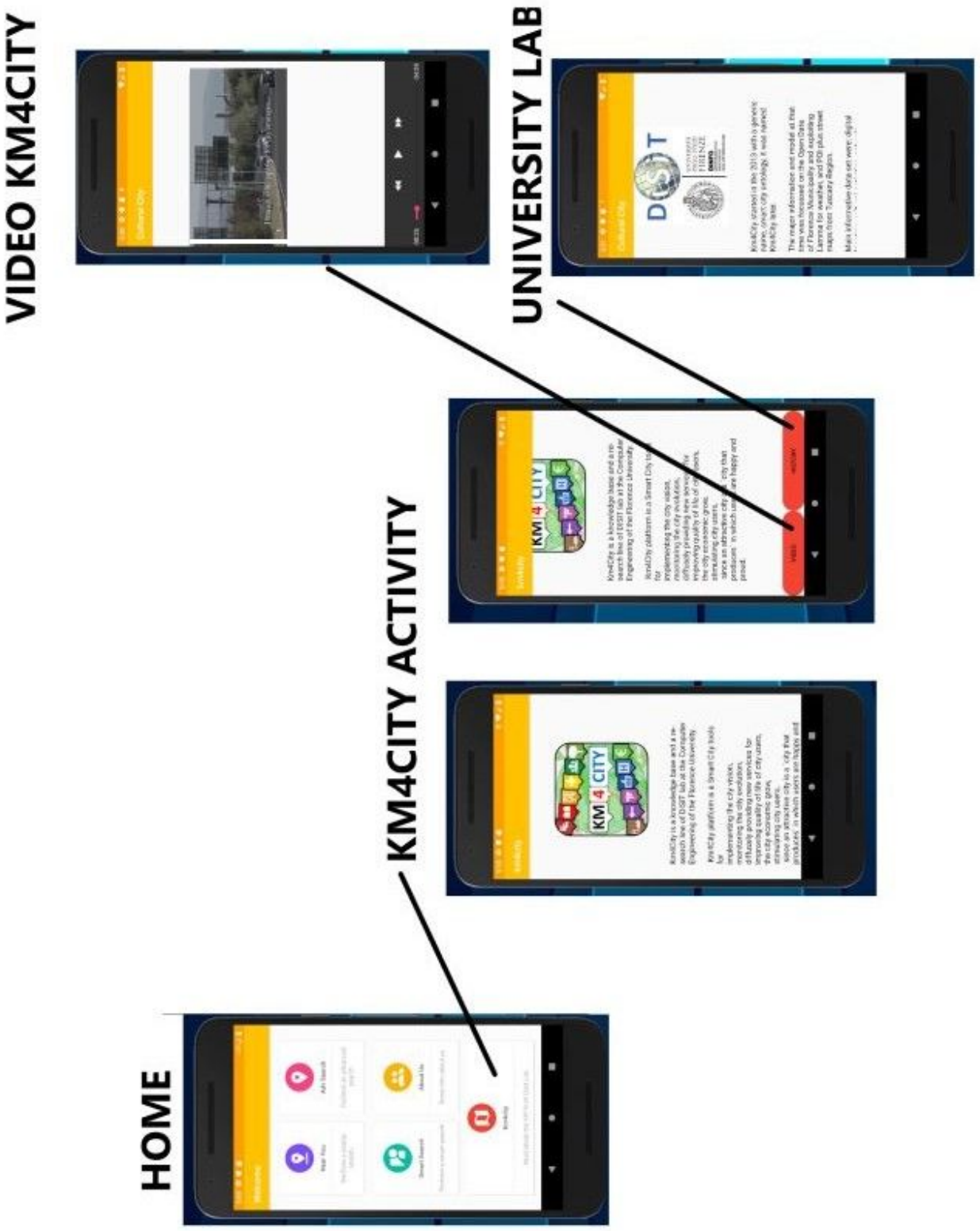
```
        android:layout_height="64dp"
        android:background="@drawable/cerclebackgroundpurple"
        android:src="@drawable/ic_pin_drop_black_24dp"
        android:padding="10dp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:layout_marginTop="10dp"
        android:text="Near You"/>
    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:background="@color/lightgray"
        android:layout_margin="10dp"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Perform a nearby search"
        android:padding="5dp"
        android:textColor="@android:color/darker_gray"/>
```

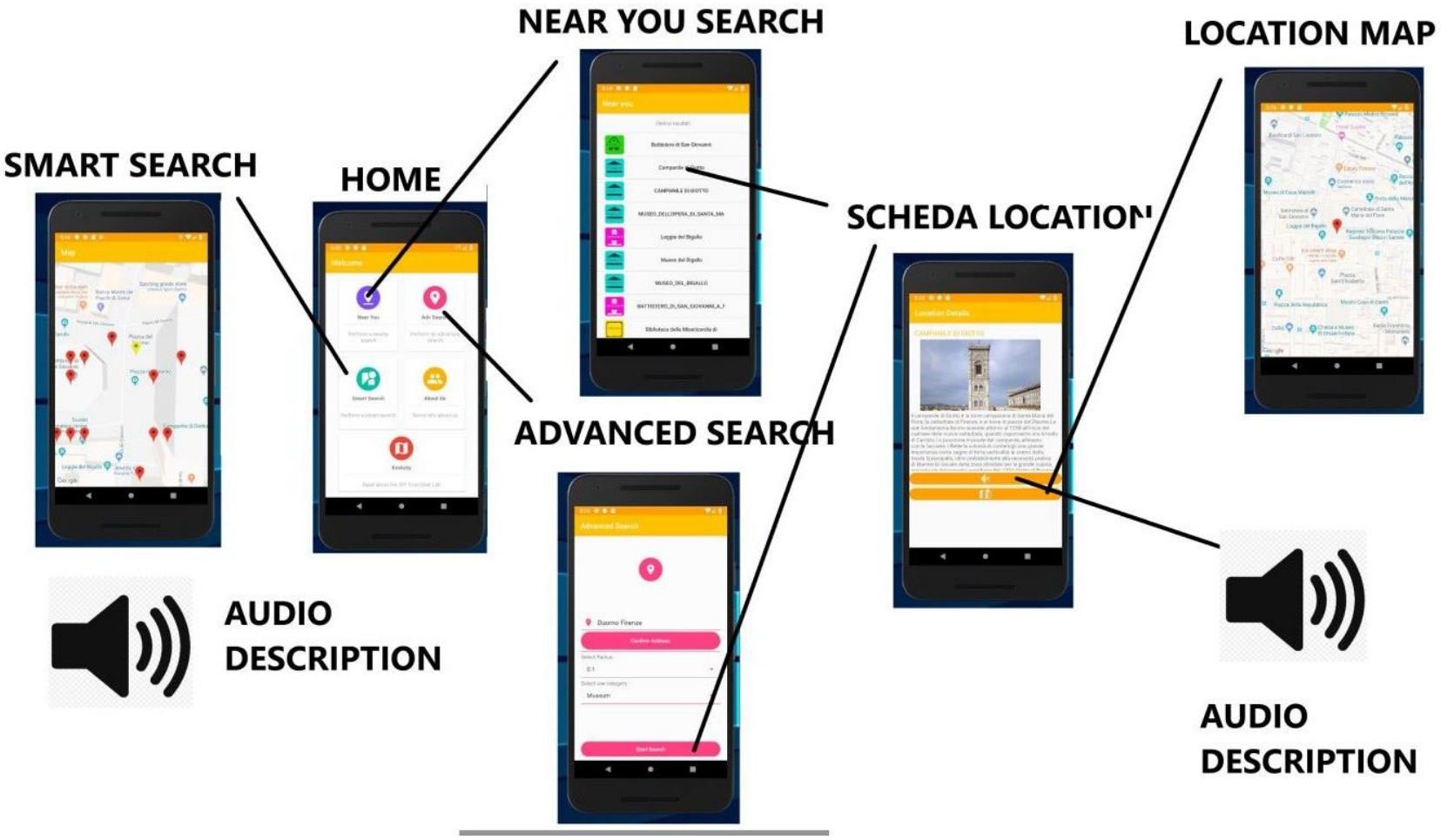
And the result is this:

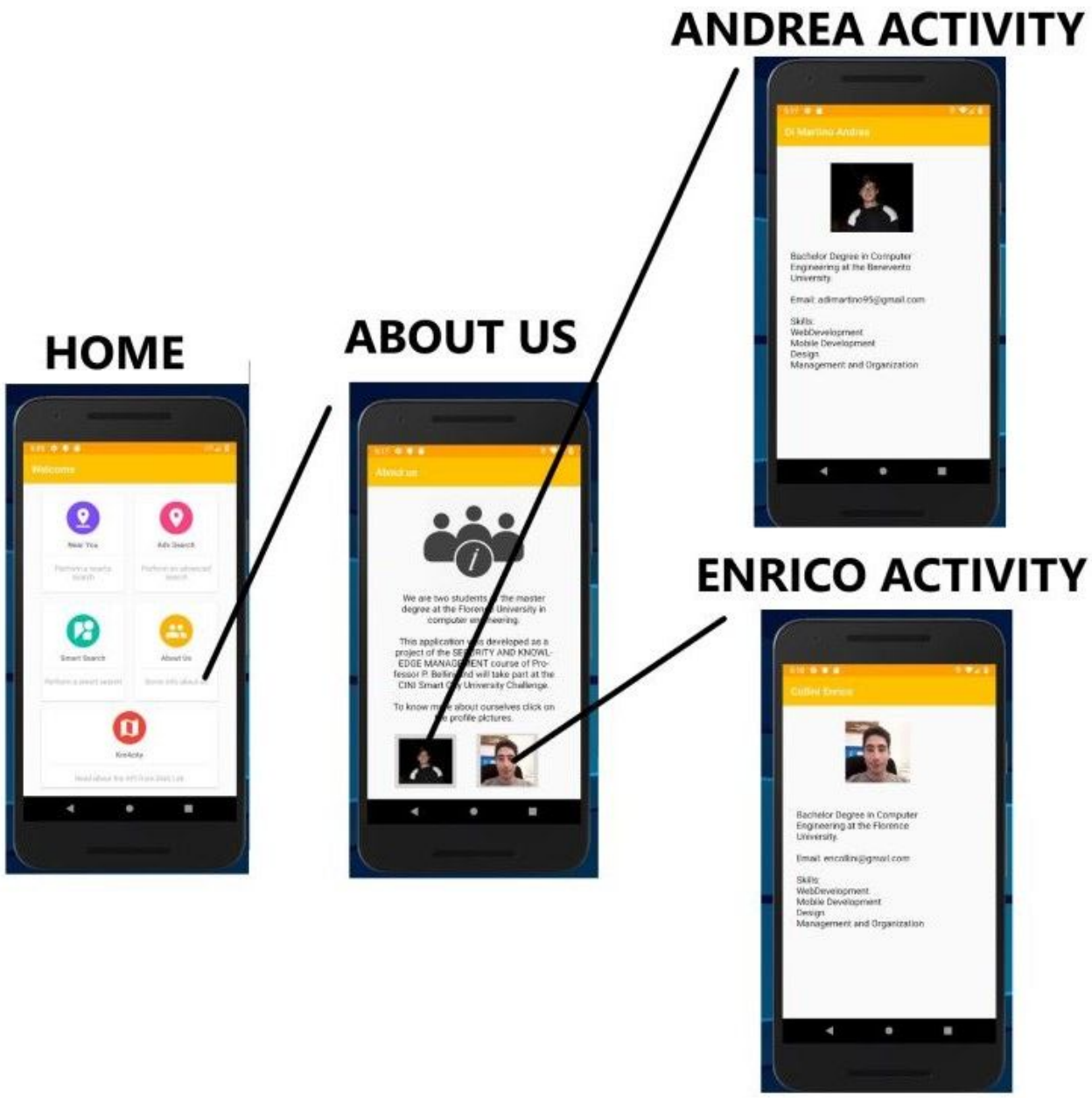


5.2 - SCREENSHOTS AND VIDEO

Demonstrative Video: https://youtu.be/8O8HQLpzlGI




NEAR YOU SEARCH
LOCATION MAP
SMART SEARCH
HOME
SCHEDA LOCATION
ADVANCED SEARCH
AUDIO
DESCRIPTION
AUDIO
DESCRIPTION


ANDREA ACTIVITY
HOME
ABOUT US
ENRICO ACTIVITY
Bachelor Degree in Computer Engineering at the Benevento University.
Email: adimartino95@gmail.com
Skills:
WebDevelopment
Mobile Development
Design
Management and Organization
Bachelor Degree in Computer Engineering at the Florence University.
Email: encollini@gmail.com
Skills:
WebDevelopment
Mobile Development
Design
Management and Organization

6 - CONCLUSIONS AND FUTURE DEVELOPMENTS