NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

CultureSpot: an Android Application for navigation to Cultural Interest Areas, using Linked Open Data.

Adam H. Mahjoub

Supervisor:	**Manolis Koubarakis, Professor**
Co-supervisor:	**George Stamoulis, Ph.D. Candidate Uoa**

ATHENS

MARCH 2021

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

CultureSpot: Εφαρμογή Android πλοήγησης για σημεία ενδιαφέροντος χρησιμοποιώντας συνδεδεμένα δεδομένα.

Αδάμ Χ. Μαχζούμπ

Επιβλέπων: Μανόλης Κουμπαράκης, Καθηγητής

Συνεπιβλέπων: Γιώργος Σταμούλης, Υποψήφιος Διδάκτωρ ΕΚΠΑ

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2021

BSc THESIS

CultureSpot : Android Application for Areas of Interest using Linked Data.

Adam H. Mahjoub
S.N.: 1115201600099

SUPERVISOR: **Manolis Koubarakis, Professor**

CO-SUPERVISOR: **George Stamoulis, Ph.D.**
                                           Candidate Uoa

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

CultureSpot: Εφαρμογή Android για σημεία ενδιαφέροντος χρησιμοποιώντας συνδεδεμένα δεδομένα.

Αδάμ Χ. Μαχζούμπ
Α.Μ.: 1115201600099

ΕΠΙΒΛΕΠΩΝ: **Μανόλης Κουμπαράκης, Καθηγητής Γιώργος Σταμούλης,**

ΣΥΝΕΠΙΒΛΕΠΩΝ: **Γιώργος Σταμούλης, Υποψήφιος Διδάκτωρ ΕΚΠΑ**

ABSTRACT

Linked Data is structured data which is interlinked with other data so we can acquire useful information through queries. With the use of the android UI, this information can be easily accessible and shown to the user. The purpose of this thesis is to showcase the power of linked open data in building smartphone applications. Our application CultureSpot, brings together information about cultural areas from OpenStreetmap and DBpedia, and uses semantic web technologies to interlink and query these data sources.

CultureSpot was designed to assist tourists in navigating a city searching for areas of cultural interest which they would like to visit. Our goal was to develop a user-friendly application that requires no technological expertise and could be easily used by anyone who wanted assistance in finding her favourite cultural areas in a city. Through the interlinking of information for these cultural spots with resources from DBpedia, users can also learn more information for their selected cities without the need to use different applications or visit other web resources.

ΠΕΡΙΛΗΨΗ

Τα συνδεδεμένα δεδομένα είναι δομημένα δεδομένα που συνδέονται με άλλα δεδομένα, ώστε να μπορούμε να αποκτήσουμε χρήσιμες πληροφορίες μέσω ερωτημάτων. Με τη χρήση της διεπαφής Android, αυτές οι πληροφορίες μπορούν να είναι εύκολα προσβάσιμες και να εμφανίζονται στον χρήστη. Ο σκοπός αυτής της διατριβής είναι να δείξει τη δύναμη των συνδεδεμένων ανοικτών δεδομένων στην κατασκευή εφαρμογών smartphone. Η εφαρμογή μας, CultureSpot, συγκεντρώνει πληροφορίες για πολιτιστικές περιοχές από το OpenStreetmap και το DBpedia και χρησιμοποιεί σημασιολογικές τεχνολογίες Ιστού για τη διασύνδεση και την αναζήτηση αυτών των πηγών δεδομένων.

Το CultureSpot σχεδιάστηκε για να βοηθήσει τους τουρίστες να πλοηγηθούν σε μια πόλη, αναζητώντας περιοχές πολιτιστικού ενδιαφέροντος, που θα ήθελαν να επισκεφθούν. Στόχος μας ήταν να αναπτύξουμε μια φιλική προς το χρήστη εφαρμογή που δεν απαιτεί τεχνολογική γνώση και θα μπορούσε εύκολα να χρησιμοποιηθεί από όποιον χρειάζεται βοήθεια για να βρει τις αγαπημένες της πολιτιστικές περιοχές σε μια πόλη. Μέσω της διασύνδεσης πληροφοριών για αυτά τα πολιτιστικά σημεία, με πόρους από τη DBpedia, οι χρήστες μπορούν επίσης να μάθουν περισσότερες πληροφορίες για τις επιλεγμένες πόλεις τους χωρίς να χρειάζεται να χρησιμοποιούν διαφορετικές εφαρμογές ή να επισκεφθούν άλλους ιστότοπους.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Εφαρμογή android με συνδεδεμένα δεδομένα

ACKNOWLEDGMENTS

CONTENTS

LIST OF FIGURES

LIST OF IMAGES

PREFACE

This thesis is essential to the completion of my bachelor degree in the Faculty of Informatics and Telecommunications, National and Kapodistrian University of Athens.

Smartphones are becoming more and more essential in the last few years. As a passionate programmer, I chose a subject that will help me build something useful in a state-of-the-art user interface, such as Android.

I trust that the application developed will help all of the types of users, experienced and novice, whilst searching for information about places of interest. It is also my expectation that this thesis will aid future research in similar fields.

1. INTRODUCTION

It is common knowledge that the need for smartphone applications is becoming more and more essential, as they are the main point of information for all types of users.They offer a user-friendly environment, where one can get useful results in no time, regardless of being an experienced user or a user with no background.

This app was developed in order to provide information of several points of interest to people, who are not familiar with datasets, such as OpenStreetMap, and are not able to use query languages.

By taking into account these needs and limitations of most of the people, I was urged to develop an android application which will deliver results in the most readable and user-friendly way possible, without the need of any programmatically knowledge.

Our users are offered a well-defined and simple interface, they are able to navigate the app, without getting lost in the process. This simplicity is accomplished by defining a path from screen to screen which is not complex and can be easily traced back.

The main part of CultureSpot is its map. Most of the information, available in this application, is shown on it. It was a priority to show information in the simplest manner and this was achieved by using a dynamic map which interacts with the users, and offers them the results they want to see in no time.

Moreover, a user can personalize his search results by using the various tools offered in this application, which would help me to find the desired result faster. These tools vary from specifying a category to searching in a specific area, among others.

Finally, CultureSpot was developed with the mindset of having an application that was, first of all, useful to the public and most importantly extensible to future updates, which would make it an even better application that could easily be used by multiple users all around the world.

2. BACKGROUND AND RELATED WORK

In this section information, about the background of the technologies used in this application, will be provided.

2.1   Linked Data, RDF, SPARQL and OpenStreetMap

In this section of this thesis, Linked Data, RDF, SPARQL and will be analyzed and explained thoroughly. Firstly, Linked Data is a set of "rules" for data published on the Web, which provide a lot of information. RDF provides the foundation for publishing and linking this data, and it is now a high-quality standard for representing data that is read by software systems. SPARQL is the standard query language and protocol for Linked Open Data and RDF databases and has multiple extensions. All of the above are the basic part of the Semantic Web.

2.1.1 Linked Data

Linked data is a set of practices for publishing structured data on the Web.The main principles are the following :

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information.
4. Include links to other URIs. so that they can discover more things.

Linked data[1] is used to retrieve useful information about various things that are published on the Web, which are interlinked in order to search and find more and more useful information. The main goal is to achieve the Semantic Web, an extension of the current web in which information is easily understandable by programs, no matter what their original design was.

Linked Open Data is a powerful blend of Linked Data and Open Data: it is both linked and uses open sources. One notable example of an LOD set is DBpedia – a crowd-sourced community effort to extract structured information from Wikipedia and make it available on the Web.
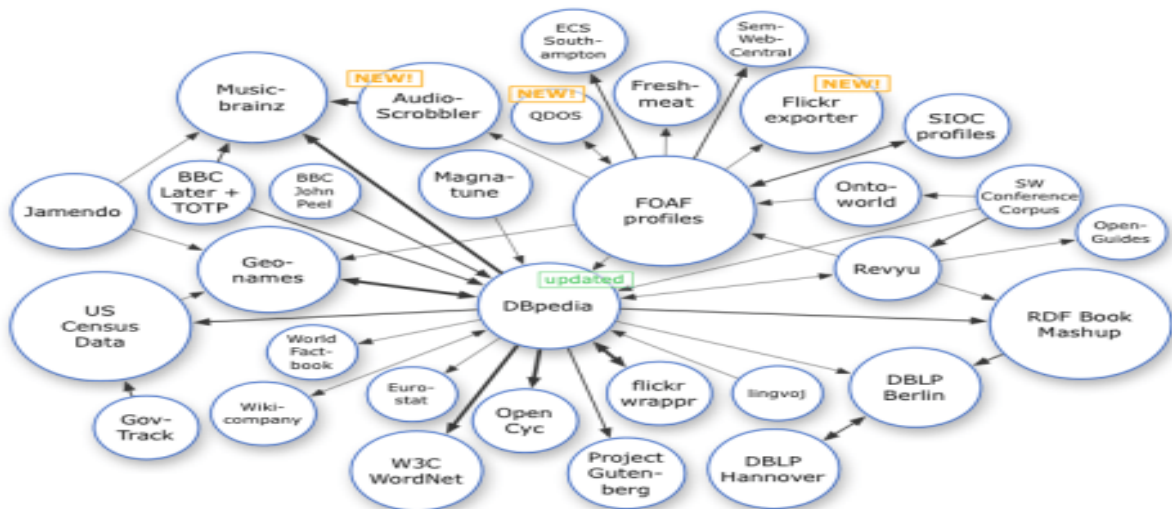

Figure 1 : Linked Data Example

---

[1] W3, Linked Data ; https://www.w3.org/wiki/LinkedData [Acessed 17/3/20]

2.1.2 RDF

RDF(Resource Description Framework) is a standard for data interchange, which was developed by W3C[1]. It is considered to be the easiest and most powerful standard by now. It was first designed as a part of the Semantic Web, but now it is used for representing high-quality linked data that is read and analyzed by various software systems.

RDF is known for its simplicity as a uniform structure to express any kind of information and all other formats of data can be converted to RDF data.RDF is built around the existing Web standards: XML and URL.
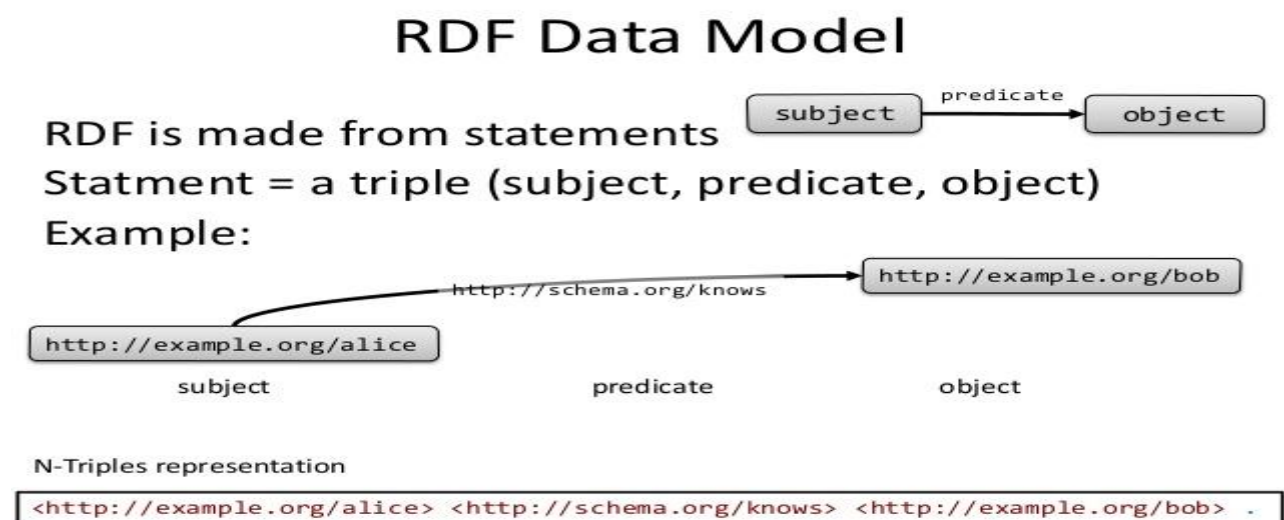
Figure 2 : RDF Statement Example

In the example the sentence : "Alice knows Bob" is expressed through the RDF structure.

1. http://example.org/alice (Alice) is the subject.

2. http://schema.org/knows (knows) is the predicate.

3. http://example.org/bob (Bob) is the object.

The predicate expresses the relationship between the subject and the object.

RDF statements state facts, relationships and data by linking resources of a different kind. With the help of an RDF statement, just about anything can be expressed by a uniform structure, consisting of three linked data pieces.

2.1.3 SPARQL

SPARQL is the standard query language and protocol for Linked Open Data and RDF databases,which was also designed by W3C.SPARQL queries can also be executed on databases that are not formatted on RDF standard,but are viewed so by a middleware.

SPARQL was designed to enable Linked Data for the Semantic Web[2]. That can be easily understood as queries on this language can work on multiple endpoints (data stores).

SPARQL has four types of queries. It can be used to:

1. ASK whether there is at least one match of the query pattern in the RDF graph data;

2. SELECT all or some of those matches in a tabular form (including aggregation, sampling and pagination through OFFSET and LIMIT);

3. CONSTRUCT an RDF graph by substituting the variables in these matches in a set of triple templates;

4. DESCRIBE the matches found by constructing a relevant RDF graph.

SPARQL has multiple extensions, one of which is GeoSPARQL for querying geospatial data, which is used in CultureSpot (e.g in order to show points of interest in a specific map area).

A SPARQL query[2] comprises, in order:

- Prefix declarations, for abbreviating URIs

- Dataset definition, stating what RDF graph(s) are being queried

- A result clause, identifying what information to return from the query

- The query pattern, specifying what to query for in the underlying dataset

- Query modifiers, slicing, ordering, and otherwise rearranging query results

prefix declarations   PREFIX foo: <http://example.com/resources/>

dataset definition   FROM...

result clause   SELECT...

query pattern   WHERE {}

query modifiers   ORDER BY …

---

[2] W3, SPARQL By Example ; https://www.w3.org/2009/Talks/0615-qbe/ [Accessed 22/3/20]

2.1.4 GeoSPARQL

GeoSPARQL [3] is a standard for representation and querying of geospatial linked data for the Semantic Web from the Open Geospatial Consortium. It is intended to provide an exchange basis for geospatial RDF data and querying with the SPARQL database query language.

In particular, GeoSPARQL provides three main assets :

1. a vocabulary to define features, geometries and their relationships
2. a set of spatial functions for use in queries
3. a set of query transformation rules

PREFIX geo: <http://www.opengis.net/ont/geosparql#>

PREFIX geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?what

WHERE {

  ?what geo:hasGeometry ?geometry .

  FILTER(geof:sfWithin(?geometry,

        "POLYGON((-77.089005   38.913574,-77.029953   38.913574,-77.029953 38.886321,-77.089005 38.886321,-77.089005 38.913574))"^^geo:wktLiteral))

}

In the example[3], shown above we search for objects that have a GeoSPARQL's defined geometry and are within a bounding box(a polygon).

In the CultureSpot application, two functions were used from GeoSPARQL, one to find areas of interest in a bounding box(the map) and one to find nearby places in a kilometer-based radius, around a selected area of interest.

---

[3] Wikipedia, GeoSPARQL https://en.wikipedia.org/wiki/OGC_GeoSPARQL [Accessed 30/3/20]
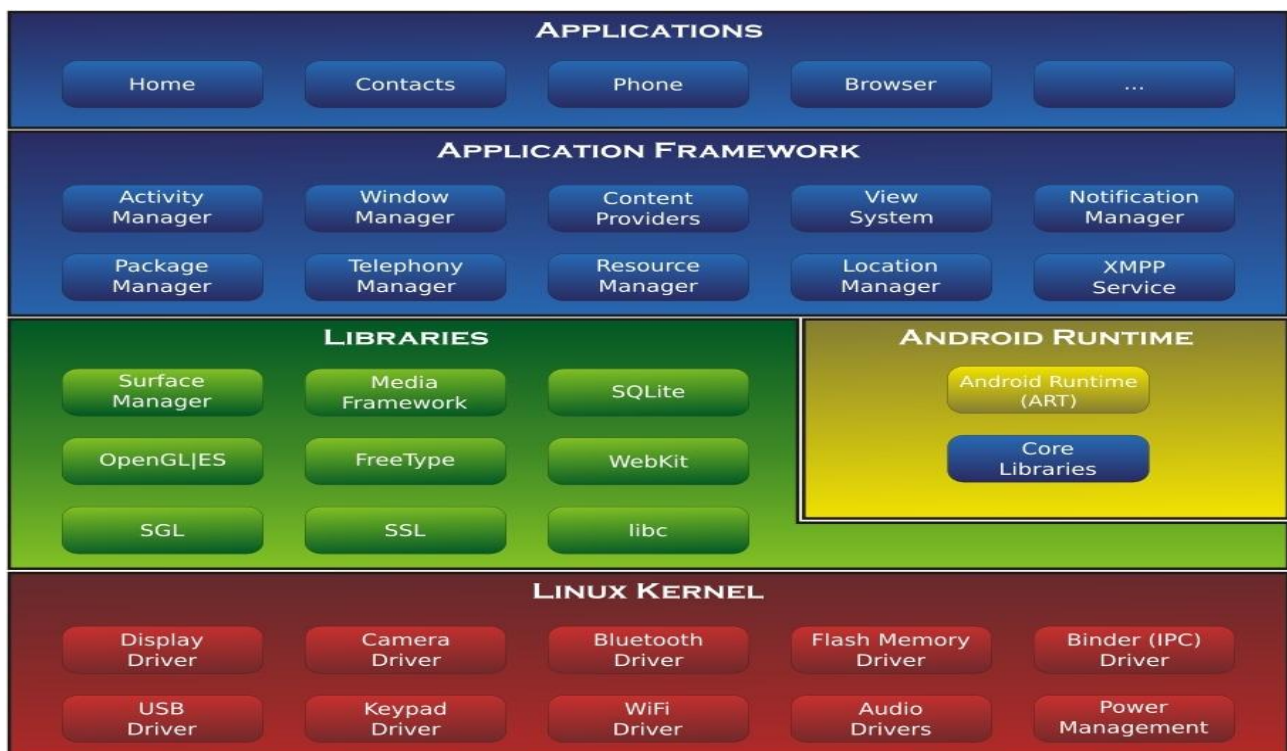
2.2   Android Development

Android[4] is a mobile operating system, based on a modified version of the Linux Kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones.

On top of the Linux kernel, there are the middleware, libraries and APIs written in C, and application software running on an application framework which includes Java-compatible libraries. Development of the Linux kernel continues independently of Android's other source code projects.

Android development is the process by which applications are created for Android devices. The most common languages used for this process are Java and Kotlin (CultureSpot was written in Java).

The Android software development kit(SDK)[5] is a set of development tools(e.g debugger,libraries etc),which is used by programming languages.

Finally, the most common way to develop an Android application is to use the Android Studio application suite, which comes with a lot of plugins that help novice and experienced Android Developers.

Figure 3 : Android System Architecture

[4]   Wikipedia,   Android(operating_system)   ;   https://en.wikipedia.org/wiki/Android_(operating_system) [Accessed 19/3/20]

[5] Wikipedia, Android software development ; https://en.wikipedia.org/wiki/Android_software_development [Accessed 18/3/20]

2.3   Datasets

Two datasets were used in order to retrieve as much information as needed for the application to be fully completed.These datasets are OpenStreetMap and DBpedia.

2.3.1 OpenStreetMap

OpenStreetMap[6] is built by a community of mappers that contribute and maintain data about various areas of interest. In our case these would be cultural areas.OpenStreetMap is open data: it is free to use it for any purpose as long as OpenStreetMap and its contributors are credited.

OpenStreetMap emphasizes local knowledge. Contributors use aerial imagery, GPS devices, and low-tech field maps to verify that OSM is accurate and up to date. OpenStreetMap is powered by open-source software from its map interface to the underlying data access API (a web service interface for reading and writing map data).

The Overpass API[7] is a read-only API that serves up custom selected parts of the OSM map data. It acts as a dataset over the web: the client sends a query to the API and gets back the data set that corresponds to the query.

Results format fetched from this API vary from XML to JSON.In this application we fetched JSON results which were later converted to CSV and then to RDF format by using the GeoTriples[8] application.This data was later saved to the GraphDB dataset, which will be analyzed in the sections below.

For each category of the places of interest we had to do a different query.For example for the museum category, we used this query :

node

  [tourism=museum]

({{bbox}});

 out;

, where the box is the bounding box with 4 corners(coordinates) and the "tourism=museum" is set, so the category is specified.

This will return a file with all the museums inside the bounding box, alongside their information(e.g their name, website, opening hours etc).

---

[6] OpenStreetMap, About OSM ; https://www.openstreetmap.org/about [Accessed 23/3/20]
[7] OpenStreetMap, Overpass API ; https://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide
[8] Github, GeoTriples ; https://github.com/LinkedEOData/GeoTriples [Accessed 22/3/20]

2.3.2 DBpedia

DBpedia[9] is a crowd-sourced community effort to extract structured content from the information created in various Wikimedia projects. This structured information resembles an open knowledge graph (OKG) which is available for everyone on the Web.

The DBpedia RDF Data Set is hosted and published using OpenLink Virtuoso. The Virtuoso infrastructure provides access to DBpedia's RDF data via a SPARQL endpoint, alongside HTTP support for any Web client's standard GETs for HTML or RDF representations of DBpedia resources.

DBpedia can be easily queried both with and without a deep knowledge of the DBpedia ontology. We can start building a query by searching on language-tagged literal values that are the objects of "rdfs:label" properties :

SELECT *

WHERE

  {

    ?place rdfs:label "Acropolis Museum"@en

  }

This will produce URIS, which their labels are "Acropolis Museum" as an English-formatted text[10].

Figure 4 : DBpedia SPARQL Endpoint

---

[9] DBpedia Wiki, DBpedia ; https://wiki.dbpedia.org/OnlineAccess [Acessed 22/3/20]

[10] Medium, Running Basic SPARQL Queries Against DBpedia ; https://medium.com/virtuoso-blog/dbpedia-basic-queries-bc1ac172cc09 [Accessed 22/3/20]

3. APPLICATION

In this section the reader will be provided with information about the architecture and the technologies used while developing CultureSpot.The role of the app, its UI and its features will also be shown.

3.1   Role of the App

CultureSpot is an android application that serves as an information "center" for points of interest currently(18/3/21) for the metropolitan area of Athens, but could be easily extended to the whole world with the correct data.

A user has to connect (through registration or signing in) in order to have access to the app. By using this app, a user can personalize his search, by searching categories, such as museums, and specific sub-area(e.g the center of Athens). He can also, of course, search for a specific name ang get a list of possible results, depending on what he typed. By clicking on a specific point of interest in the map he can see various information about a point of interest (e.g Acropolis Museum's wiki page). It is well coordinated with other apps, as he can press on telephone and be redirected to the dialer app or press on a link and be redirected to the browser app of his choice. Moreover, he can search for points of interest near his chosen one. It is also possible to get map directions of the driving route he has to follow in order to get to that specific point, from his current GPS location(app permission must be handed) or from another point of interest. There is, of course, the option to make this point a favorite one amongst others. Lastly, he can see his data such as his username,email etc and see his favorite points of interest, which are deletable in a personalized page.

In conclusion, CultureSpot can serve as a tourist information app where people can search for cultural centers to visit when they are on vacation.

3.2   Architecture
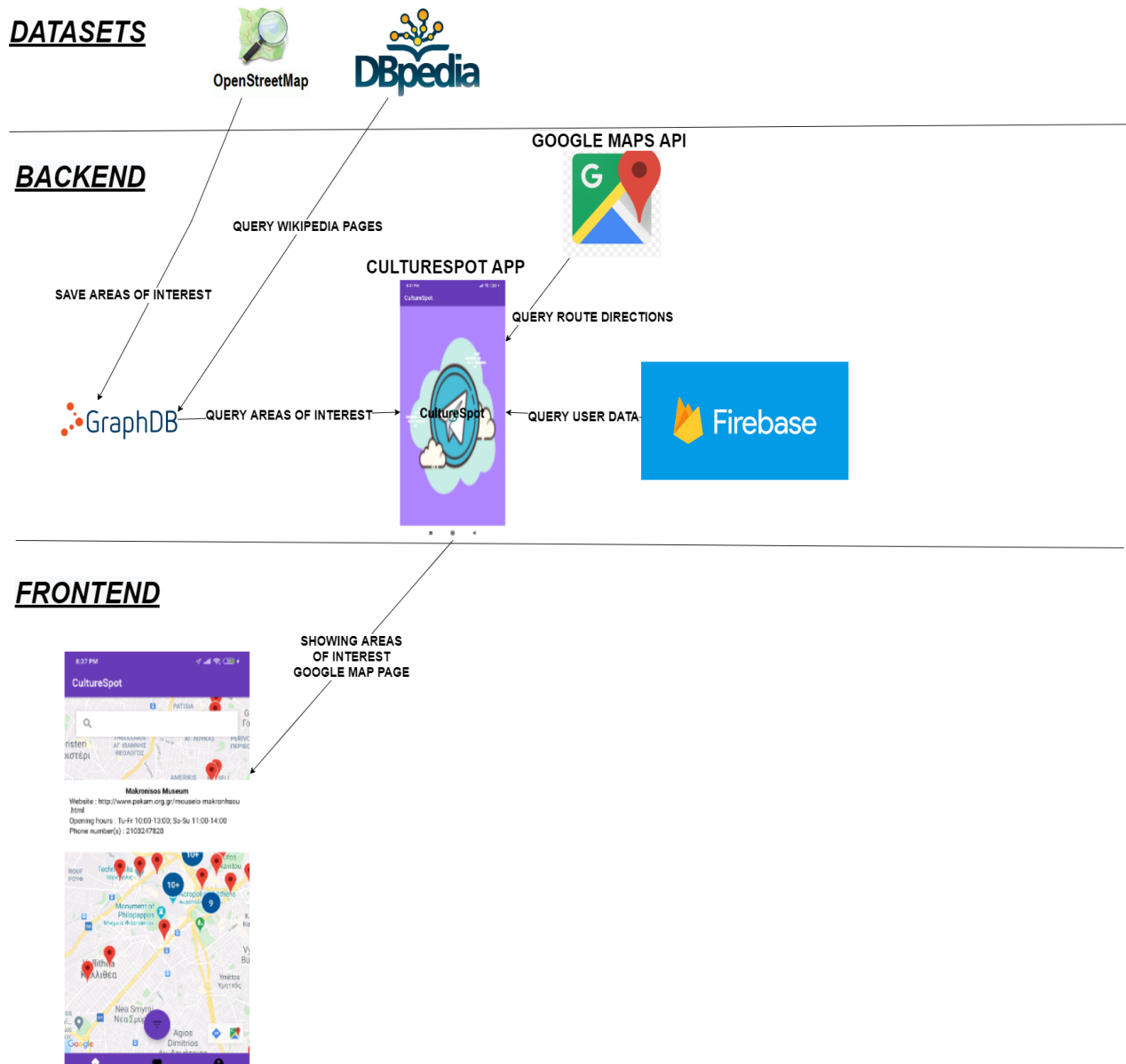
Figure 5 : CultureSpot Architecture

Main data for points of interest were taken from OSM, converted to RDF and uploaded to GraphDB. Using CultureSpot a query is sent to GraphDB for all or some points of interest depending on the parameters. The ones that do not have a wiki page are subject to another query to DBpedia in order to find theirs, if they do have. The results are shown to the Google Maps API.

In order to authenticate a user, register or even get his data there is a query to Google's Firebase database, then the user can login/register or logout and even see his personal data, which includes a dedicated page for his favorite points of interest.

Note: In figure 5,only the map results are shown for the frontend part.

3.3   Technologies

The app consists of five main technologies, Geotriples, GraphDB, Firebase, Android Studio and Google Maps API. All of them will be analyzed in the sections below

3.3.1 Geotriples

In order to transform the CSV files from OSM Overpass API to RDF,Geotriples was used.

GeoTriples allows the transformation of geospatial data stored in raw files (shapefiles, CSV, KML, XML, GML and GeoJSON) and spatially-enabled RDBMS (PostGIS and MonetDB) into RDF graphs using well-known vocabularies like GeoSPARQL and stSPARQL, but without being tightly coupled to a specific vocabulary.

Geotriples was used in a docker pack called KR-Suite-docker[11],which includes all the linked data tools developed by the KRR&A team of the National and Kapodistrian University of Athens.

3.3.2 GraphDB

All of the data acquired from the Geotriples app was stored in GraphDB[12], Ontotext's database for RDF data.

Data about the points of interest are saved in GraphDB, a database for RDF data. Whenever a user searches for specific points of interest a query to this database is sent in order to get the desired results (a HTTP GET query is also sent to DBpedia for some wiki page, but this is subject to change).

Furthermore,a user can save time by getting a list of recommended results depending on the name he typed.The search for every name is done once when the map is loaded so there is no need to query GraphDB everytime there's a change to the search boxes.
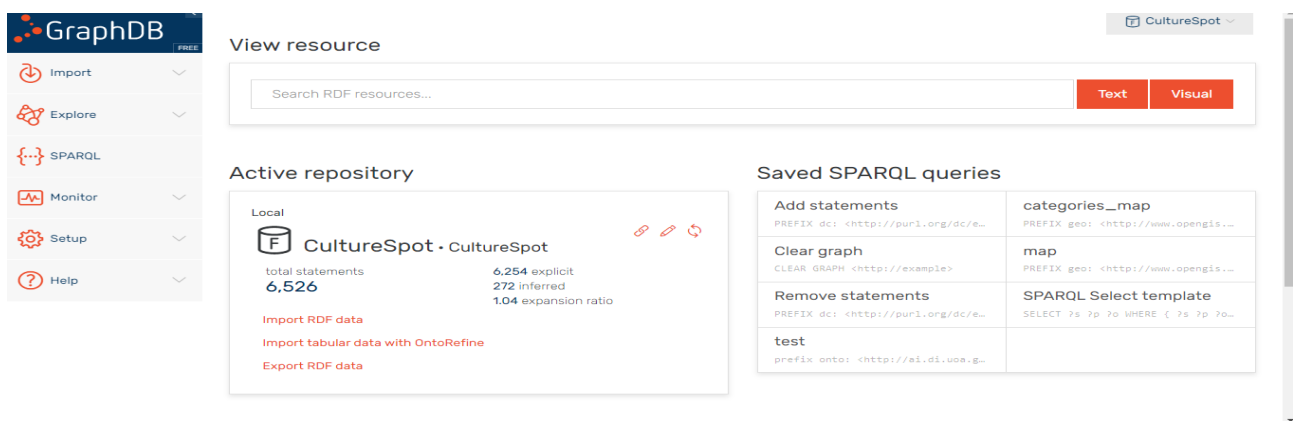

Figure 6 : GraphDB Dashboard

[11] Github, KR-Suite-docker ; https://github.com/GiorgosMandi/KR-Suite-docker [Accessed 22/3/20]
[12] Ontotext, GraphDB ; https://www.ontotext.com/products/graphdb/ [Accessed 22/3/20]

3.3.3 Firebase

Each user's data, such as his credentials and his favorite points of interest, are saved to Firebase[13], a Google database.Whenever a user logins a query is sent to Firebase to check his credentials and if they are correct and then all of his data is provided to the app.

Whenever a user wants to access his data a request is sent to Firebase with his id and all of his data, including his favorite areas of interest, are shown to him. The same applies whenever a user adds or deletes a favorite point of interest. Then the data is updated and re-uploaded to the database.

In the image below you can see the database of our app which in its current state (22/3/20) has only one document "Users",where every user's data is saved.In the Authentication we can also be shown the credentials of every user(email and encrypted password).
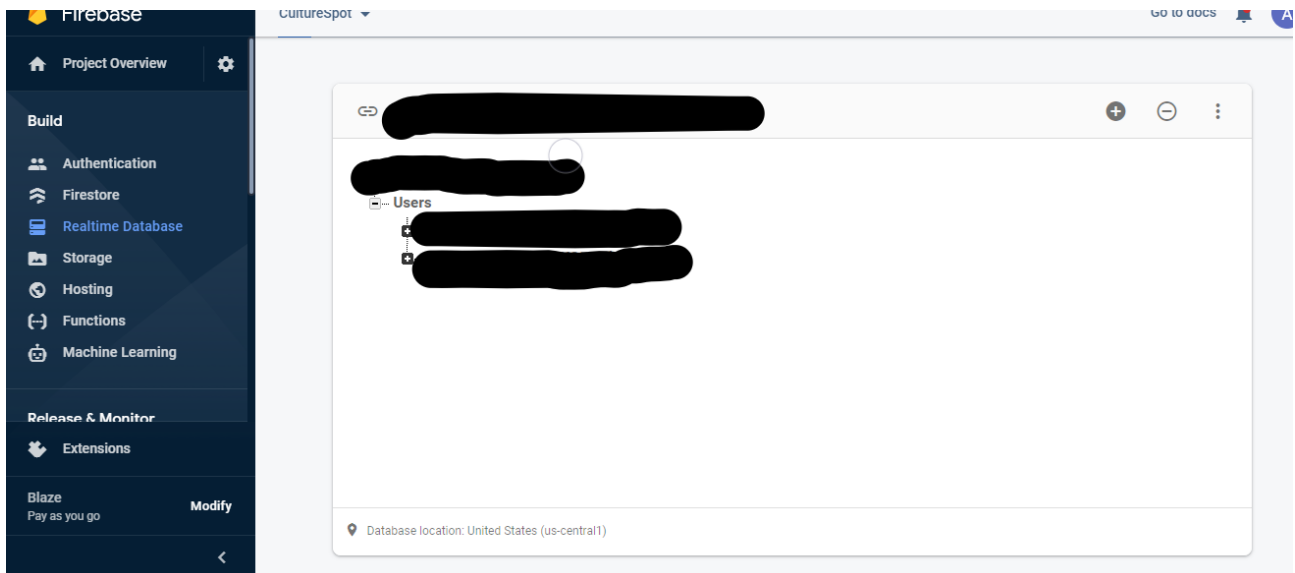

Figure 7 : Firebase Dashboard

13 Google, Firebase ; https://console.firebase.google.com/u/0/ [Accessed 22/3/20]

3.3.4 Google Maps

The main technology used is, of course, Google Maps API[14]. Every result retrieved from GraphDB is shown on this map.

Android applications are well-coordinated with this API as both are developed and maintained by the same company, Google. The functionalities are multiple and can vary a lot.In CultureSpot, marker clustering and Directions API was used in order to show the necessary information that was wanted.

The SDK supports the Java programming language and provides additional libraries and extensions for advanced features and programming techniques.

In order to use this API, a key must be generated (see Future Work section).

General Information about the areas of interest and their position is given by using the marker clustering API,via clusters,markers and info windows making the search on the map as convenient as possible.

The driving directions of a route between two points is shown by using the Directions API, via an HTTP-GET request with the right parameters.
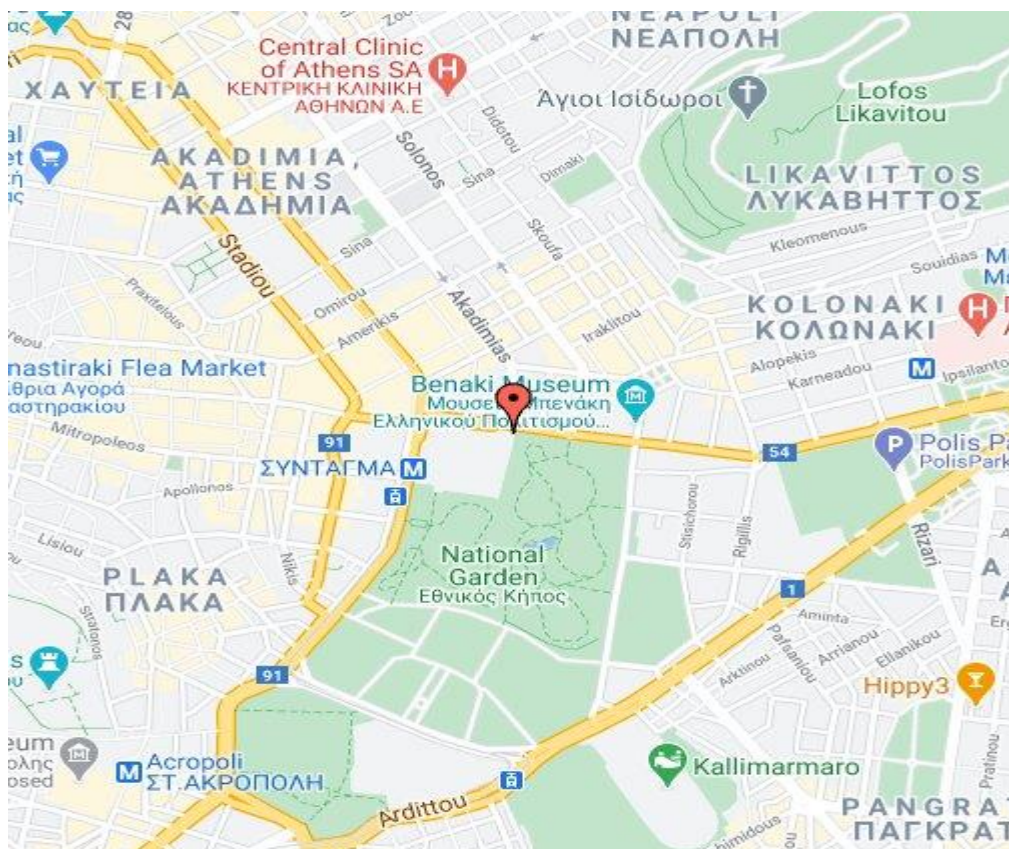

Figure 8: Google Maps Example

---

[14] Google, Google Maps Platform ; https://developers.google.com/maps [Accessed 22/3/20]

3.3.5 Android Studio

In order to write the code necessary for this project Android Studio IDE[15] was used.It is the most well-known IDE for android applications, as it comes with various tools that will make the development the easiest possible.

In order to create the screens of the applications Android Studio provides a Visual layout editor(the code for the layout is written in XML format), with multiple configurations that are available in the menu.

In the backend Java was used as the programming language.Using Android Studio's code autocomplete, we were able to write better code in a faster and more productive manner.

Among the various tools which were available, the Firebase and the Google Maps ones were the most useful as there was no need to write extra code.All of the features were available with the click of a button.

Finally, Android Studio comes with an emulator of Android smartphone to test applications, so there is no need to own an android smartphone.
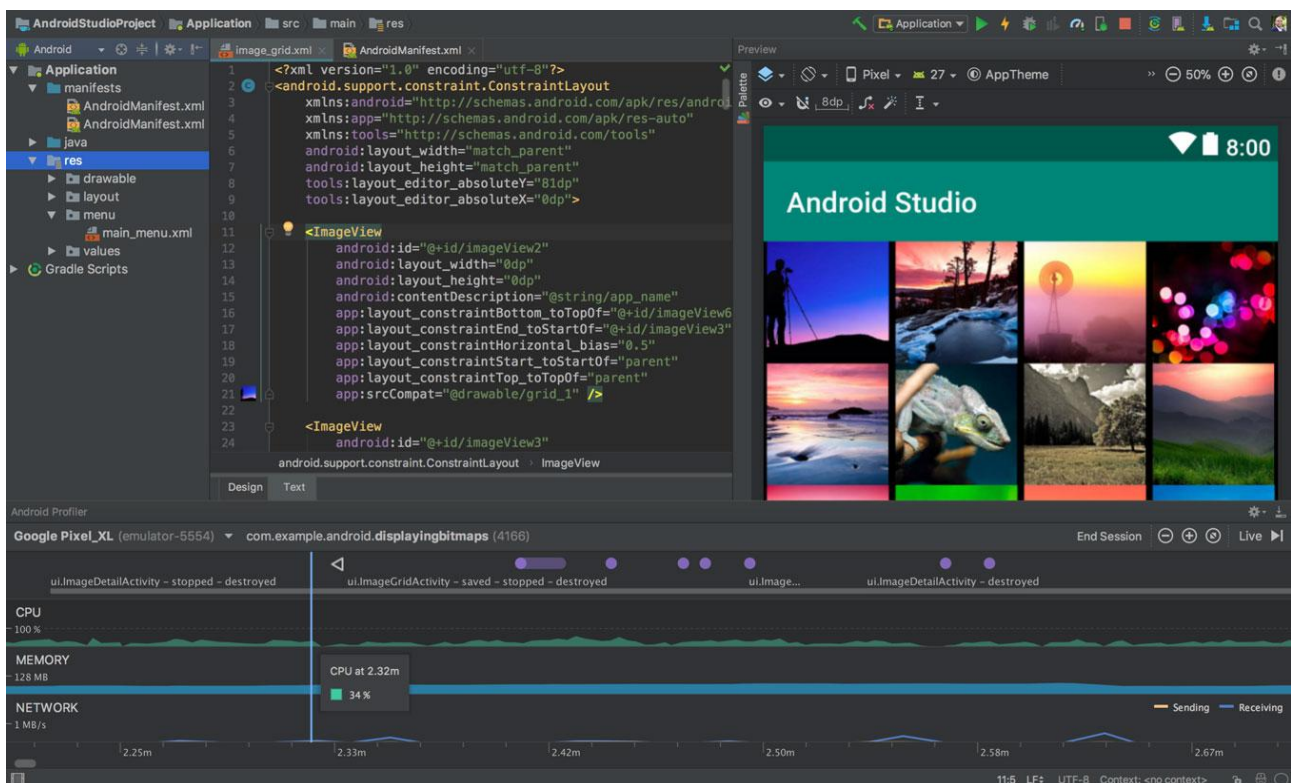

Figure 9 : Android Studio Homepage

---

[15] Android, Android Studio ; https://developer.android.com/studio [Accessed 22/3/20]

3.4   User Interface and Features

In this subsection we will analyze how the app works by showing the screens of the app as well as explaining them.

3.4.1 Splash Screen

Image 1 : Splash Screen

Whenever a user opens the app he is shown the app's splash screen, until it is figured out if he has already logged in or not.If he is logged in he is redirected to the map screen, else he is redirected to the login screen shown below.
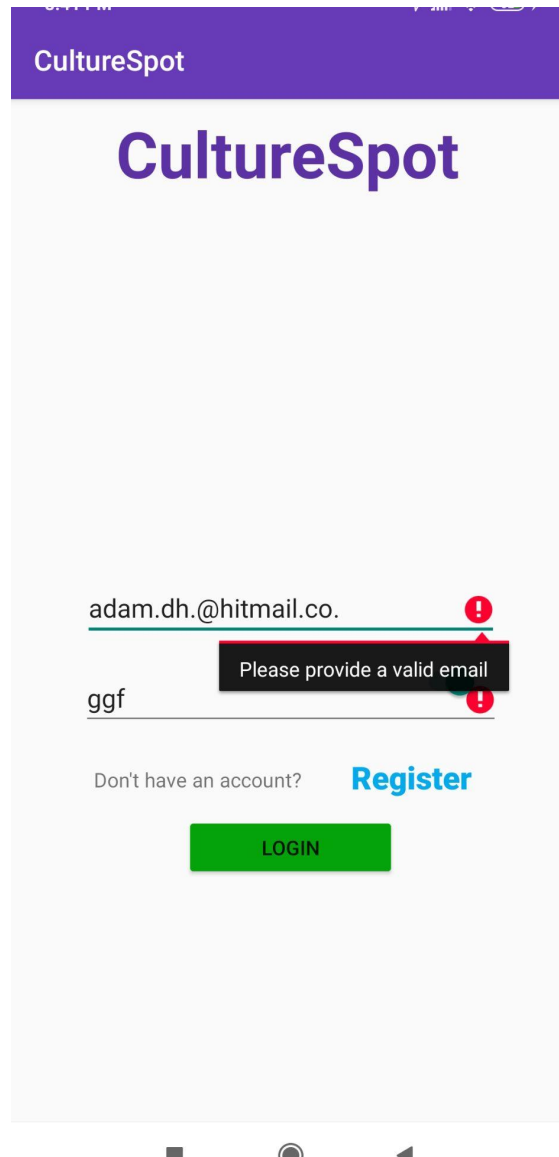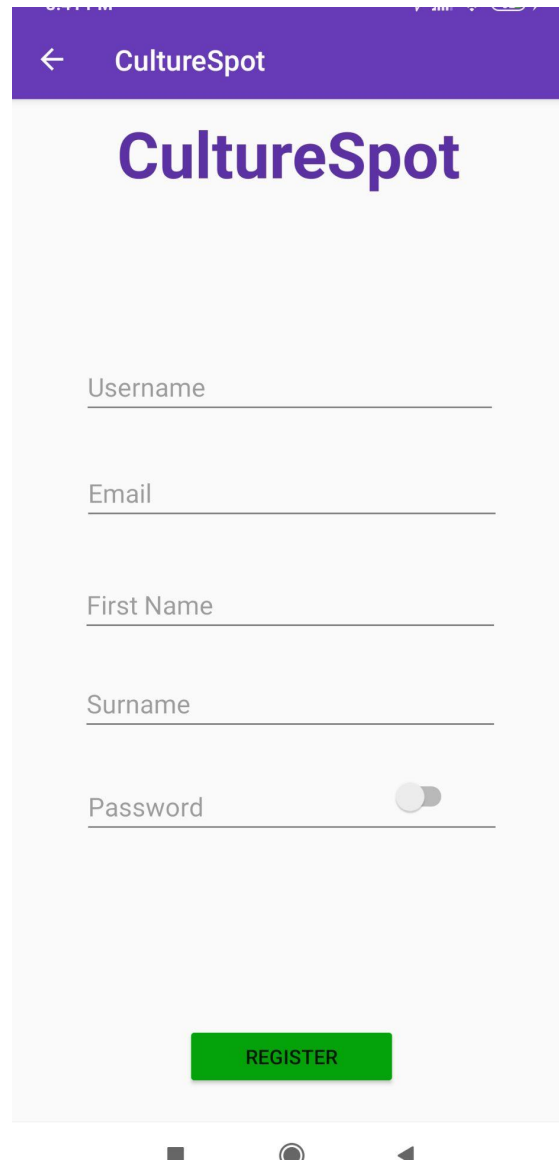
3.4.2 Login/Register Screens

Image 2 : Login Screen

If he is not logged in he is redirected to the login activity where has to type his credentials to use the application.He can also (un)show his password by clicking on the check button right of the screen.As we can see in these screenshots there's a checking mechanism for errors.If the user clicks on the Register text he is redirected to the register activity shown below.If he signs in successfully he is redirected to the map screen shown 2 images below.

Image 3 : Register Screen

This is the register screen where the user has to type his desired credentials.There is of course the "(un)show password" mechanism as the mechanism for checking errors in the typing fields, as in the login screen. By pressing the back button on top-left of the screen he will be redirected to the login activity.If he registers successfully he is redirected to the map screen shown below.

3.4.3 Map View
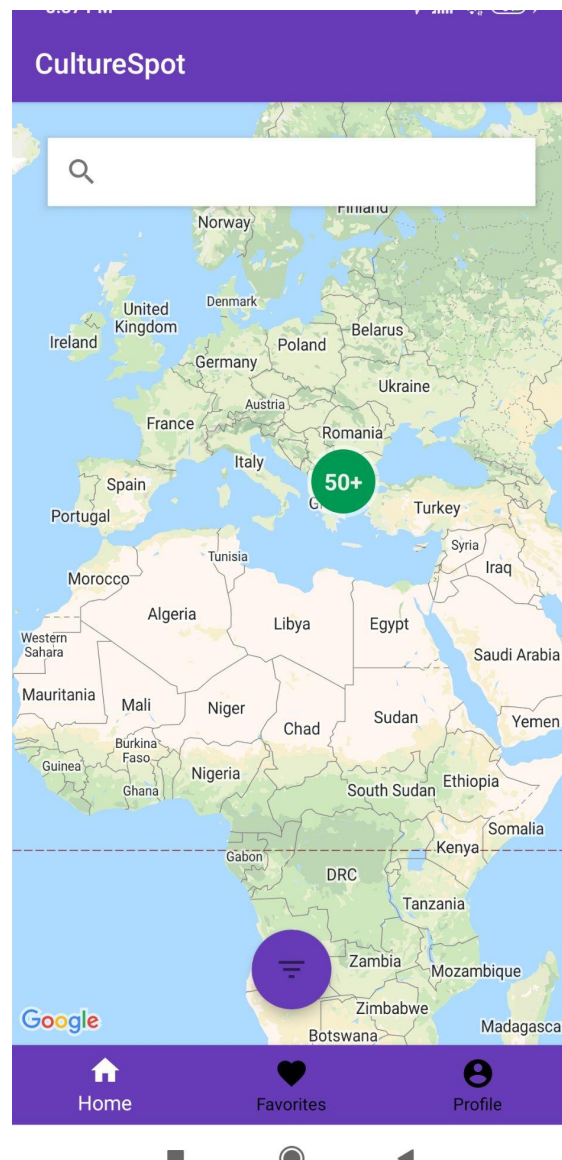
Image 4 : Default Map View

This is the default page of CultureSpot app.The user is shown all of the points of interest gathered on a cluster if he is zoomed out.By clicking on the floating button on the bottom of the screen he is shown the filters' section(Image 5). He can also search for a name and a list of names will be shown to him(Image 6). On the bottom navigation, by clicking on favorites the user can see his favorite points of interest shown below(Image 12) and by clicking on the profile, he can check his credentials(Image 13) and logout from the app and be redirected to the login page shown above.
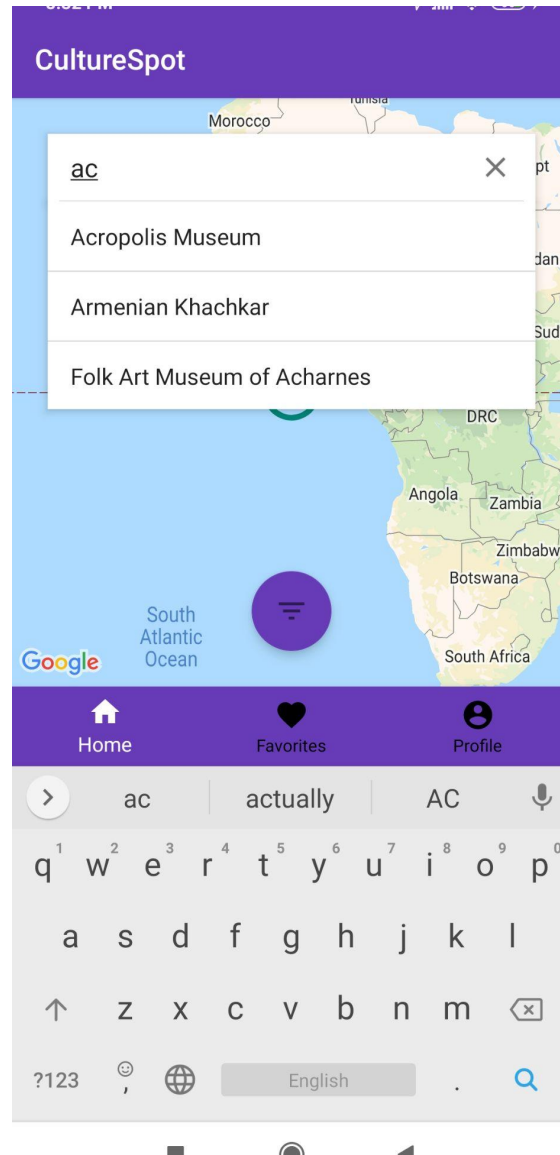
Category Filter
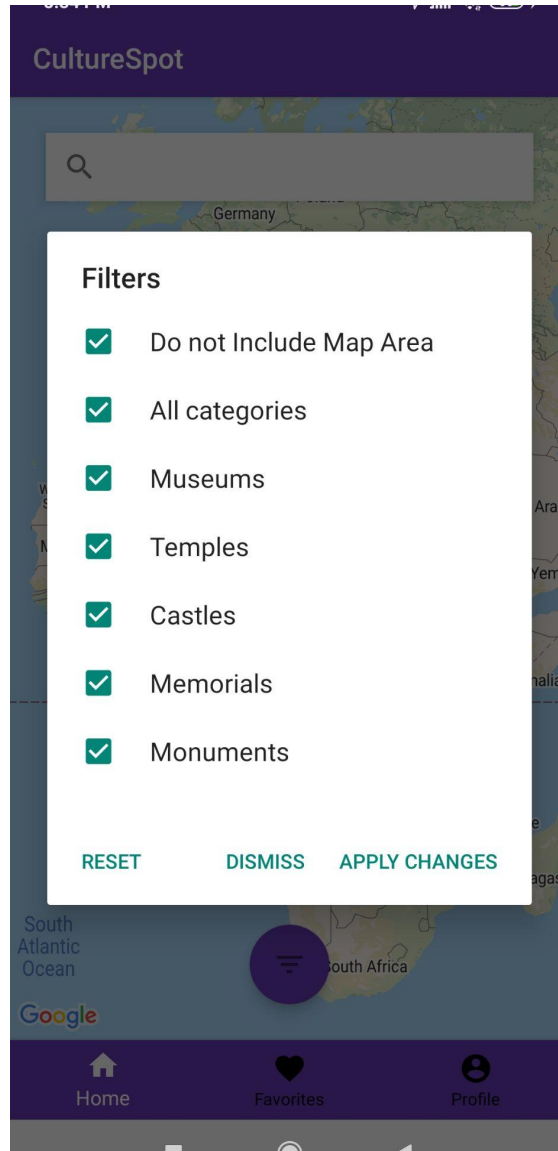
Image 5 : Category Filter Map View

This is the filters' section. Here the user has to choose at least one category,otherwise his options are not saved. He can also choose if the bounding box(the map) will be included as a parameter to his search.

Search

Image 6 : Search List Map View

This is the autocomplete list suggestion. Here the user is provided with a list of names that correspond to the text he typed. If he clicks on one of them, directions to that place from the users' location will be shown(Image 11).
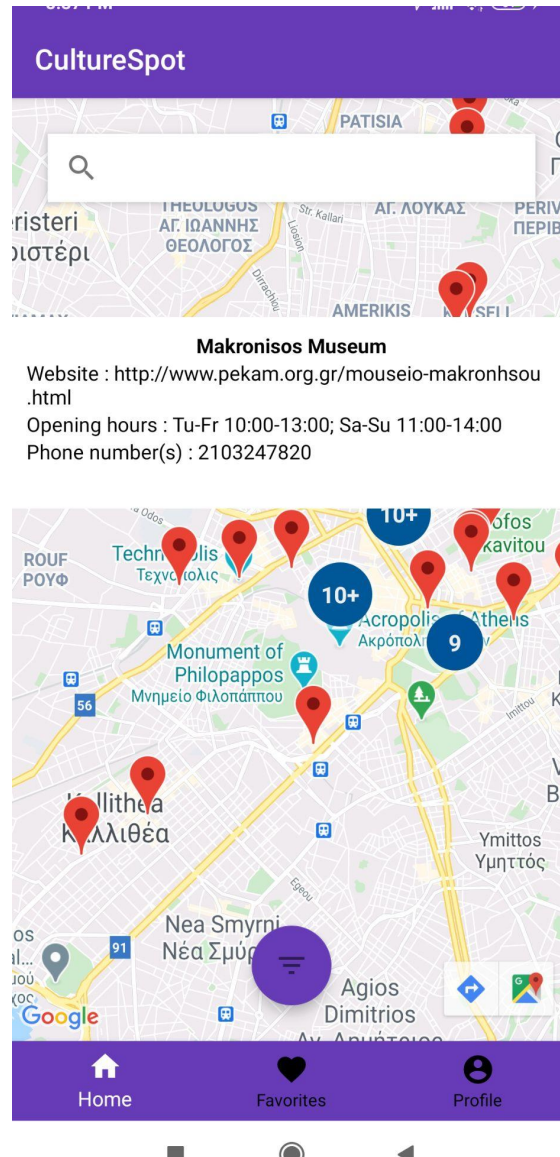
View Results and Metadata

Image 7 : Clusters and Markers Map View

This is the default map view if a user zooms in. If a marker is clicked an info window is shown with all of the information available for this point of interest. If the info window is clicked then we get a popup with information about this place(Image 8).
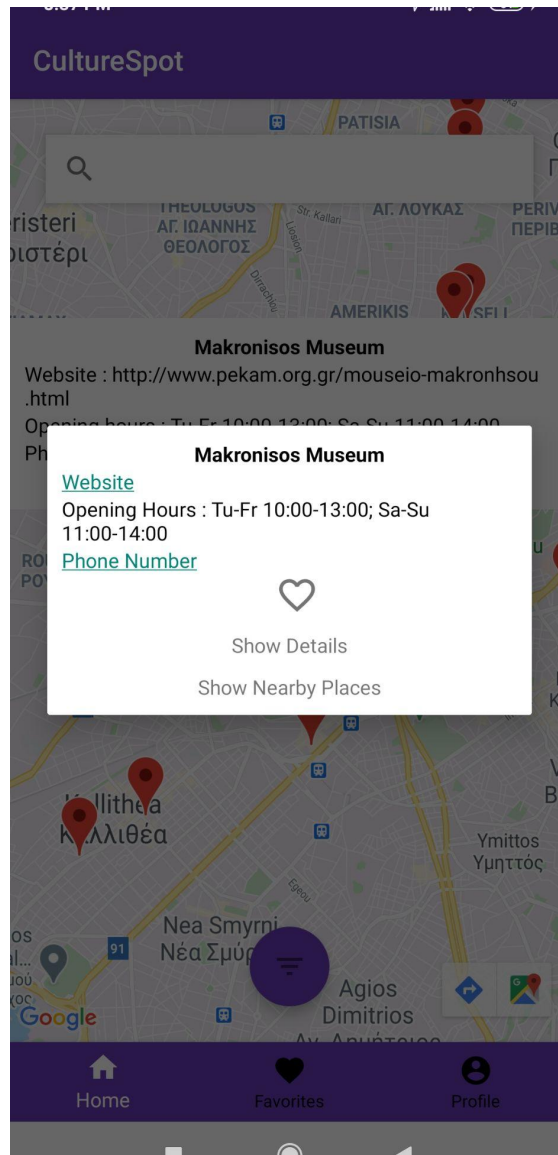
Image 8 :Popup Window of Marker Map View

Here the user can make the place of interest a favorite one(this will make the heart red,indicating that is a favorite point of interest. The heart would be already red if this place was a favorite one). He can click to show nearby places to show another popup(Image 9), he can click on show details which will have the same results as clicking a name from the search list mentioned above(Image 11).He can also click the links to redirect to another app (Image 10).

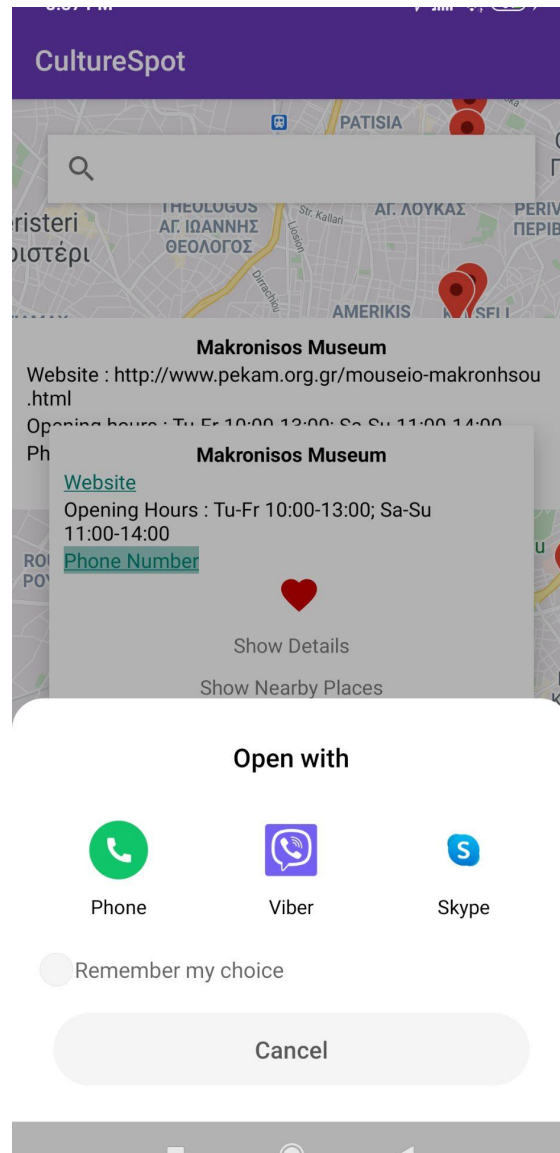**Search**
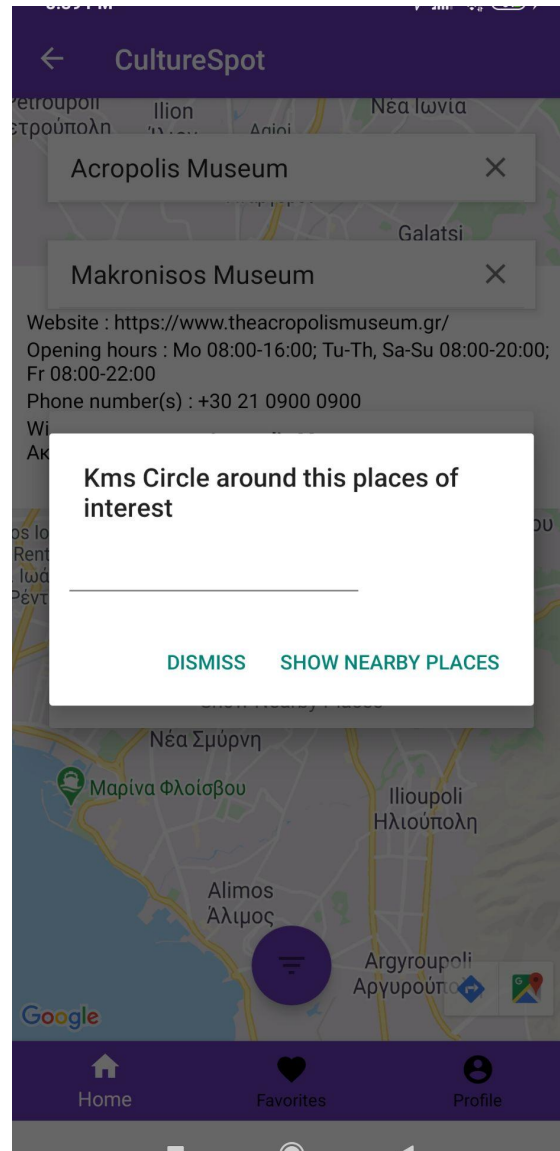
Image 9 :Search Nearby Map View

Here the user chooses the nearby places he wants to see by adding Kms radius of the chosen (it would be the same,as the clusters on Image 4 and Image 7).
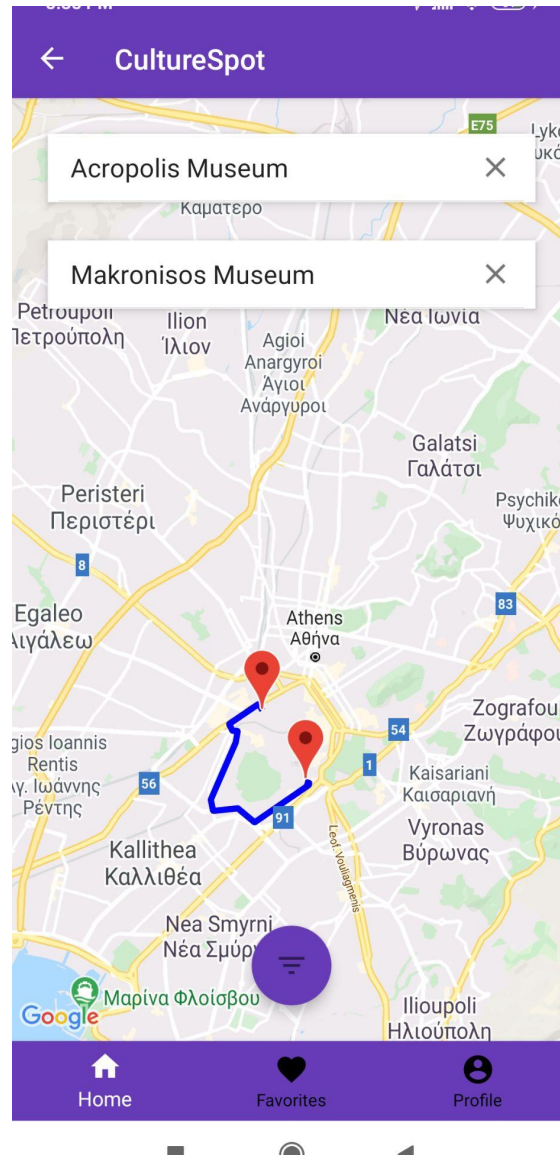
View Results and Metadata

Image 10 : Redirect to App Map View

If the user clicks for example on a telephone he is redirected to the dialer app(same for links to browsers).

Navigation

Image 11 : Route Directions between two Points of Interest Map View

Here we can see the route directions fetched from Google Maps Api.If the back button is pressed then we research for points of interest depending on the filters (it would be the same,as the clusters on Image 4 and Image 7).The default route shown is from the user's location to the place of interest selected. We can also search for two points of interest to get their route (My location included).The markers shown in this image have the same functionalities as the markers in the images above.

Note : A place must be clicked from the list(as in Image 6) in order to show the directions.

3.4.4 Favorites and User Profile Screens
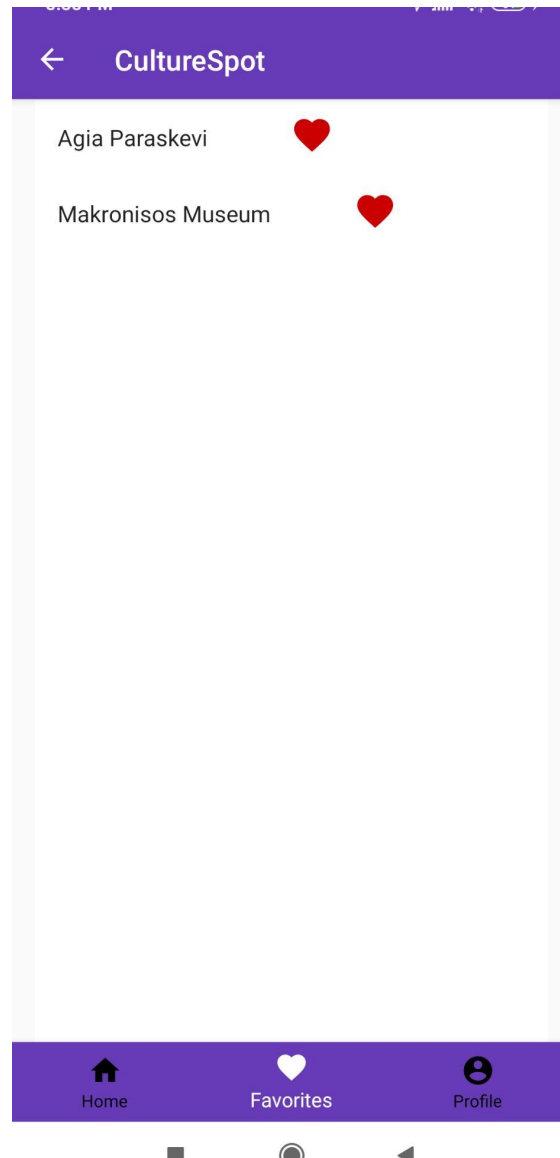
Image 12 : Favorites List Screen

Here we can see the Favorites list page where a user can see his favorite areas of interest.If he clicks on the heart then he deletes this place from the page.
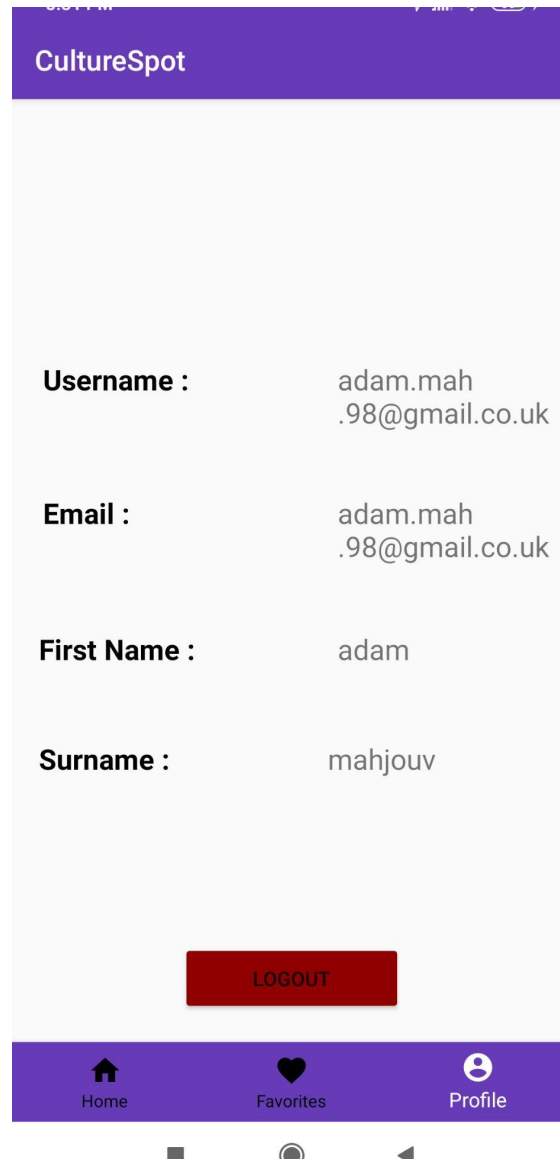
Image 13 : Profile Screen

Here we can see the profile page where the user can see his credentials and press logout
to be redirected to the login screen(Image 2).

4. FUTURE WORK

CultureSpot is a well structured application that has the base to create a more well-rounded app.There are some future extensions that could make the app better. These are mentioned below.

Our first goal is to bypass the DBpedia query by incorporating its data to GraphDB.This will minimize the backend time to query both DBpedia and GraphDB. If we take into consideration that from time to time it takes a lot of seconds(sometimes even minutes) to query DBpedia, then we can safely assume that the query time will be just 1 to 2 seconds.

Secondly, we could get written route directions, so the user can have a better experience when searching a route between two map points.This will of course help him navigate to an unknown city and it will decrease his time spent,because he will not have to use google maps application,but use CultureSpot from the beginning to the end.

Furthermore, a review section for points of interest, where multiple users can share their experience, must be created. This will help the users decide which cultural spots are worth visiting and will also make their opinion on a museum or monument valuable.

Finally, if a chat is created, users can have personal conversations that are not allowed in a review section. This will make the experience of our app much more personalized, in which users have the opportunity to exchange thoughts and experiences with one another.

I believe these four extensions could make the app even better and would make it a true choice for the market of tourist information applications.

5. APPENDICES

The code for the application,plus the information on how to install it are found in [here](#).

Instructions on how to use the application :

Download the CultureSpot folder and open it with Android Studio, which you have to download.

Connect Firebase with the project through Android Studio Tools.

Change the GraphDB_URL in MapFragment.java:89(file:line) to your GraphDB url found in the dashboard of the GraphDB application, which you have to download.

Put your Google API keys in MapFragment:841 google_maps_api:19(src/debug and src/release/res/values) AndroidManifest.xml:33 google-services.json:23.This key can be created in Google Cloud Console(google account is needed).

Load the rdf-ttl files to your GraphDb database(First Setup->Repositories->Create a new Repository and then Import->RDF to load the files).

Run the code from Android Studio.

ABBREVIATIONS - ACRONYMS

RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
SDK	Android software development kit
OSM	OpenStreetMap
UI	User Interface
XML	Extensible Markup Language
W3C	World Wide Web Consortium
CSV	Comma Separated Values
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
API	Application Program Interface
IDE	Integrated Development Environment
RDFS	Resource Description Framework Schema
OWL	Web Ontology Language

REFERENCES

[1] Nuno Lopes, Axel Polleres, Alexandre Passant, Stefan Decker, Stefan Bischof, et al.. *RDF and XML: Towards a unified query layer. Proc. W3C workshop on RDF next steps*, Jun 2010, Stanford, United States.

[2] Jorge Pérez, Marcelo Arenas, Claudio Gutierrez, *Semantics and Complexity of SPARQL*, ISWC 2006, Athens, Georgia.

[3] Robert Battlem Dave Kolas, *GeoSPARQL: Enabling a Geospatial Semantic Web*, *Semantic Web, vol. 3, no. 4*, pp. 355-370.