Bulletin of the *Transilvania* University of Braşov • Vol. 8 (57) No. 2 - 2015
Series I: Engineering Sciences

THE DEVELOPMENT OF AN INFORMATION SYSTEM FOR TOURISTS USING THE ANDROID PLATFORM (I)

D. KRISTALY[1] A.C. MANEA[1]
S.A. MORARU[1] C.L. CRISTOIU[1]

Abstract: *The interest for mobile applications has grown increasingly in the recent years and this thanks to the easiness with which their users can be informed in a very short time. The mobile application to inform tourists is used on the smart mobile terminals presenting the sights of a city, providing information about each point of interest and information on how the user can reach it. This paper presents a software system that is addressed, primarily, to tourists coming to visit the city of Braşov, but it can be used equally well by the inhabitants of Brasov who have not yet discovered all the attractions of the city.*

Key words: *location, map, server, tourist, route.*

1. Introduction

The access to information in the shortest time possible is today a necessity. The idea of developing applications for mobile devices that rely on wireless technology has made many developers create a wide variety of applications useful to people.

An important service in the use of these applications is the Wi-Fi that allows connecting to the Internet network.

Beside the applications to access accounts remotely (Gmail, Facebook, Yahoo Mail), applications that allow finding the latest news, media viewer applications and much more, mobile devices are very useful simply because they can also be used as GPS systems.

The real-time location on the map is a very important thing. Beside the much easier orientation in unfamiliar places, the concept of location can become crucial in an unfortunate event as for example an accident or getting lost, some applications allowing coordinates to be sent directly from the competent authorities.

The application gives the user the possibility to create an account based on a password and an email address, email address that can be used later for password recovery.

The user has access to 36 sights, the application making available to him/her some minimal information about that location together with the visitation program and a phone number, a photo gallery with images of the location, but also a list of opinions given by tourists who have already visited that location.

An important feature of the application is the ability of the user to track in any moment the list of public means of transport

[1] Dept. of Automation and Information Technology, *Transilvania* University of Braşov.

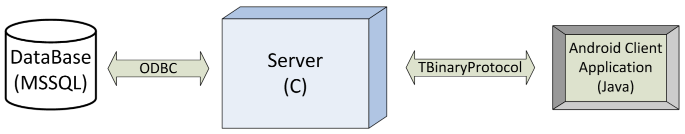

Fig. 1. *Client-server model*

that he can use to travel to a certain sight, or at least to get close to it.

The user can also view the routes plotted on Google Maps [14], [15].

The routes can mean:

a) the route from the user to the nearest bus stop when accessing the screen “Buses”;

b) the bus route or routes, when the distance to a certain sight must be covered travelling by more than one means of transport;

c) the route to the final bus stop (the place the user must get off) to the target sight (when the means of transport does not reach that point).

The user can add friends who use this application, friends whom he may locate on Google Maps if they have enabled the option: “visible on the map”.

With the help of the application, the user can view a route on the map between the user and a specific friend [10]. Also, the application enables messaging between friends.

The user has the option to be visible or not on the map, but also a configuration panel is available from any screen of the application.

2. The Architecture of the Software System

The software has been designed on the client-server model [7], as in the example presented for the Figure 1.

The information system consists of three components: database, server and the client application [4].

The latter sends requests to the server based on a communication protocol.

Then the server does selection operations, deletes, inserts and updates, depending on the request received from the client, on the data found in the database.

The communication between the server and the client is done based on a request-response mechanism [6], while the connection between the server and the database is carried through the ODBC library [16], [19], [20] (Figure 2).

The tables of the database are:

A. The table *Users*

The *Users* Table will contain data related to the users who will create an account in the application [5] and contains the following fields:

- *'idUser'* is of the *bigint* type and stores a unique identification ID for each user, being at the same time the primary key, containing the option Identity Specification set to *Yes* for the ID to be automatically incremented, at the same time eliminating the possibility that two or more users have the same ID;
- *'numeUser'* is of the *varchar* (50) type and stores the name the user identifies with, compared to the other users on the client application;
- *'email'* is of the *varchar* (50) type and stores an email address that the user uses besides the password in order to connect to the customer application.

Also based on this email address, the user can retrieve his/her password by sending an email to the email address in question by the computer system at the user's request.

In case an email address is already used it cannot be used again by another user:

- *'password'* is of the *varchar* (50) type

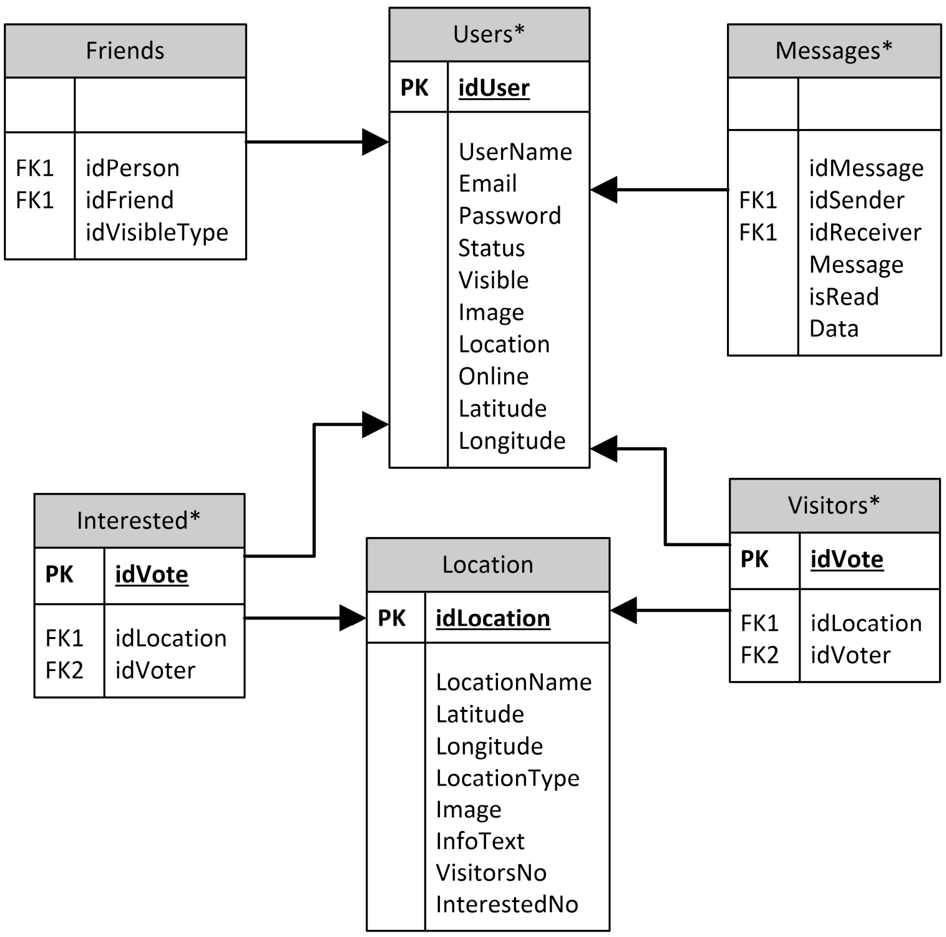

Fig. 2. *The relational chart of the database*

and it stores the password that the user uses to connect to the client application. It can be recovered by means of the email password;

- *'status'* is of the *varchar* (100) type and it stores the user's activity at a certain time, for example: "I am at the Black Church". The status of a user is visible to all that user's friends;
- *'image'* is of the *varchar* (50) type and it stores the path to the user's picture on the profile screen;
- *'location'* is of the *varchar* (50) type and it stores the user's current location;
- *'online'* is of the *varchar* (50) type and it receives True value if the user can communicate via chat and False if not;
- *'latitude'* is of the *float* type and it stores the user's current latitude. It is updated with the help of an Android [3] service running on the client, when the user changes position [18];
- *'longitude'* is of the *float* type and it stores the user's current longitude, the update being made similarly to that of latitude.

B. The *Location* Table

The *Location* Table will store all the data about the sights and it is composed of the following fields:

- *'idLocation'* is of the *int* type and it stores a unique identification ID for every sight in part, being at the same time a primary key and having the option Identity Specification set to No;

- *'LocationName'* is of the *varchar* (50) type and it stores the names of the sights that the user can visit;
- *'latitude'* is of the *float* type and it stores the latitude of each location;
- *'longitude'* is of the *float* type and it stores the longitude of each location;
- *'Location Type*' is of the *varchar* (50) type and it is used for filtering the sights for the client;
- *'VisitatorsNo'* is of the *int* type and it stores the number of people who visited a certain sight;
- *'InterestedNo'* is of the *int* type and it represents the number of people wishing to visit a certain sight;
- *'phone*' is of the *varchar* (50) type and it stores the phone number to which the user can call to receive information in relation to the targeted sight;
- *'program*' is of *varchar* (100) type and it represents the hours between which the user can go to visit some sights.

C. The *Messages* Table

The *Messages* Table stores the data needed in order to relate the users and it contains the following fields:

- 'id_Message' is of the *int* type and it represents the ID on which the identification of every message is done;
- 'id_Sender' is of the *bigint* type and it stores the ID of the user sending a message to another user being in his/her friends list;
- 'id_Receiver' is of the *bigint* type and it stores the ID of the user receiving the message;
- 'Message' is of the *varchar* (50) type and it represents the field that stores in the encrypted version the message itself;
- 'isRead' is of the *int* type and it gets value 1 if the message was read and 0 if it has not yet been read by the recipient user;
- 'Data' is of the *time* type and it stores the time when the message was received.

D. The *Visitors* Table

The *Visitors* Table stores the data used as in case of the "*Interested*" Table in order to make a statistic of each location:

- 'idVote' is of the *int* type and it stores a unique identification ID for each vote;
- 'idLocation' is of the *int* type and it stores the ID of the sight that the user has already visited;
- 'idVoter' is of the *bigint* type and it stores the ID of the user who voted.

E. The *Interested* Table

The *Interested* Table stores data used in making a statistic of each location in part:

- 'idVote' is of *int* type and it stores a unique identification ID for each vote;
- 'idLocation' is of the *int* type and it stores the ID of the sight that the user wishes to visit;
- 'idVoter' is the *bigint* type and it stores the ID of the user who voted.

F. The *Friends* Table

The *Friends* Table maintains the communication between the users, containing the following fields:

- 'idPerson' is of the *bigint* type that will store the ID of the user who is logged and adds friends to the friends list;
- 'idFriend' is of the *bigint* type and it stores the ID of the user added as a friend by another user.

The Thrift Service is an IDL or a binary communication protocol used to define and create services for numerous languages. Typically IDL's are used for remote procedure calls (RPC), providing a bridge between two different systems. The Thrift service is used in order to generate a Java code and code C on the basis of certain structures [13]. In addition to these structures, another service is created (*MyService*) that will contain the database query methods required.

The server supports a class header *MyService.h* where we find an abstract class *MyServiceIf* which contains the methods in the *DateBase.thrift* file generated as C virtual functions. Following the generation of a Java code, the client will contain one class for each Java structure.

The body Methods within the *DateBase.thrift* file can be found in an interface located in *MyService.java.* Their implementation is done in the class *NonblockingClient.java* [1].

The server supports methods to connect and query the database. This is implemented in C programming language by means of the development environment Microsoft Visual Studio 2010.

The client is implemented using the Java programming language [2], in the Eclipse KEPLER development environment.

When a method is appealed to, which sends a requirement to the server, a new socket is created that will connect to the server based on an IP address and a port [9].

Then the communication protocol is defined based on the socket created and the desired method is appealed [12].

After obtaining the desired data, the connection is closed.

3. Conclusions

The application developed can be of real help to people who want to visit the city of Braşov and the facilities offered are able to provide valuable information on the optimal routes that a tourist can follow to arrive soon and safely to the targeted sights [22].

The server designed in the C programming language allows quick access to the data stored in the database using ODBC [8], [20].

The client application has been optimized to run on different resolutions. Also, it can be used on mobile devices having different screen diagonal.

The application reached the goals set, allowing the users to identify on its basis what means of transportation to use in order to get to the desired location, but also to see or ask the opinions of other users who have already visited some sights [17].

The project can be improved, as far as the means of transport which will be equipped with GPS devices and the communications services proper by giving the possibility to track in real time on the map, the route of the means of transport [11], [21].

This is an important issue because the vehicle can pass either earlier or later than the scheduled time and such a facility would help tourists in planning their visits.

References

1. Bâscă, O.: *Baze de date* (*Date Bases*). Bucureşti. Bic All S.A., 1997.
2. Frăsinaru, C.: *Curs practic de Java* (*Practical Course for Java*). Electronic edition. 2006.
3. Gargenta, M.: *Learning Android.* O'Reilly. 2011.
4. Lauren, D., Shane, C.: *Android Application Development Second Edition.* In: SAMS Publishing, 2012.
5. Laurent, S.St.: *XML A Primer Second Edition.* M&T Books. 1999.
6. Meier, R.: *Professional Android 2 Application Development.* Wrox Professional.
7. Meier, R.: *Professional Android™ Application Development.* Wiley Publishing. 2009.
8. Mickey, M., Hamilton, D.: *Programming Windows NT 4.* SAMS Publishing. 1996.
9. *** 2.5 Creative Commons Attribution: *Getting Started | Android Developers.* Available: http://developer.android.com/ training/index.html. Accessed: 26.08. 2015.

10. *** Android-er: *A simple example using Google Maps Android API v2*. Available: http://android-er.blogspot.ro/2012/12/a-simple-example-using-google-maps.html. Accessed: 06.08.2015.
11. *** Android Asset Studio. 2010. Availabel: http://romannurik.github.io/AndroidAssetStudio. Accessed: 09.09.2015.
12. *** Android App Patterns. Available: http://www.android-app-patterns.com/. Accessed: 09.09.2015.
13. *** Brian, F.A.: *Thrift for Windows and MSVC 2010*. Available: http://www.brianfosterallen.com/thrift-for-windows-and-msvc-2010/. Accessed: 16.08.2015.
14. *** Developers, Google: *Google Maps Android API v2 - Google Developers*. Available: https://developers.google.com/maps/documentation/android/start?hl=ro#install_and_configure_the_google_play_services_sdk. Accessed: 05.08.2015.
15. *** Developers, Google: *The Google Directions API - Google Maps API Web Services - Google Developers*. Available: https://developers.google.com/maps/documentation/directions/#TravelModes. Accessed: 09.09.2015.
16. *** Limited, Easysoft: *ODBC from C Tutorial Part 1*. Available: http://www.easysoft.com/developer/languages/c/odbc_tutorial.html. Accessed: 01.09.2015.
17. *** Mayani, P.: *Android - GridView example*. Available: http://www.technotalkative.com/android-gridview-example/. Accessed: 06.08.2015.
18. *** McKenzie, D.: *Designing for Android|Smashing Magazine*. Available: http://www.smashingmagazine.com/2011/06/30/designing-for-android/. Accessed: 09.09.2015.
19. *** Network, Microsoft Developer: *ODBC API Implementation Details*. Available: http://msdn.microsoft.com/en-us/library/ms131675(v=sql.105). aspx. Accesed: 01.08.2015.
20. *** *Open Database Connectivity - 'ODBC'*. Available: http://www.anaesthetist.com/mnm/sql/odbc.htm. Accessed: 06.08.2015.
21. *** Tamada, R.: *Android Sliding Menu using Navigation Drawer*. Available: http://www.androidhive.info/2013/11/android-sliding-menu-using-navigation-drawer/. Accessed: 09.09.2015.
22. *** Teta Infoimpex SRL. Brașov. *Obiective turistice*. Available: http://www.welcometoromania.ro/Brasov/Brasov_Lista_Obiective_r.htm. Accessed: 05.08.2015.