ErasmusApp: A Location-Based Collaborative System for Erasmus Students

Karel Bruyneel and Benedita Malheiro

Abstract. This paper reports on the design and development of an Android-based context-aware system to support Erasmus students during their mobility in Porto. It enables: (*i*) guest users to create, rate and store personal points of interest (POI) in a private, local on board database; and (*ii*) authenticated users to upload and share POI as well as get and rate recommended POI from the shared central database. The system is a distributed client / server application. The server interacts with a central database that maintains the user profiles and the shared POI organized by category and rating. The Android GUI application works both as a standalone application and as a client module. In standalone mode, guest users have access to generic info, a map-based interface and a local database to store and retrieve personal POI. Upon successful authentication, users can, additionally, share POI as well as get and rate recommendations sorted by category, rating and distance-to-user.

1 Introduction

The main goal of this application is to support Erasmus students both upon arrival and throughout their stay in Porto. Initially, newcomers can feel isolated and lost, experience language difficulties or may need to find a place to stay, i.e., they require information about where to meet young people, accommodation, food or health treatments. In a later stage, when they develop a network of relationships and establish a set of preferred indoor and outdoor places, they are ready to share this information and, thus, make recommendations to fellow students.

The developed application intends to help overcoming the initial difficulties and promoting information sharing and student integration. The ErasmusApp has

Karel Bruyneel
Katholieke Hogeschool Sint-Lieven, Ghent, Belgium
e-mail: `karel.bruyneel@gmail.com`

Benedita Malheiro
School of Engineering, Polytechnic Institute of Porto and INESC TEC –
INESC Technology and Science (formerly INESC Porto), Porto, Portugal
e-mail: `mbm@isep.ipp.pt`

DOI: 10.1007/978-3-319-05440-7_4, © Springer International Publishing Switzerland 2014

two operation modes: standalone and distributed. The standalone mode, which is offered to guest users or offline registered users, provides a map-based interface to explore, find and store personal POI as well as basic information regarding accommodation, restaurants, hospitals, meeting places, the local Erasmus Students Network (ESN), etc. The distributed mode requires user authentication and offers, additionally, the possibility to classify, rate and share personal POI and to get recommendations per category ordered by rating and/or distance-to-user.

This paper is organized in six sections covering the introduction, the state of the art, the development environment, the ErasmusApp system, the tests and results, the conclusions and references.

2 Context-aware Mobile Applications

There are numerous context-aware mobile applications to help tourists finding historic sites, museums, movies, shows, restaurants, stores, recreational activities, etc. PC World magazine presents in [1] a distilled list of the essential Android applications for travellers. Those that inspire this work are hereby listed:

- **COMPASS** or COntext-aware Mobile Personal ASSistant is an application that provides a tourist with information and services based on his specific context and current goal [2].
- **MyMytilene** is a mobile tourist guide platform for the Municipality of Mytilene, Greece. MyMytilene addresses personalization in the context of allowing users to explicitly select touristic content to be included in a customized mobile application which is generated on the fly, adapting the application so as to meet the screen size and hardware constraints of the user’s mobile phone. The mobile application may be used in either offline or online modes [3].
- **Wikitude** is an augmented reality global travel guide that overlays Wikipedia and user-contributed content over the mobile device camera view, providing information on the user surroundings. Wikitude uses the on-board GPS, compass and movement sensors to match the user position in relation to the landmarks that the camera is pointing towards [4].
- **Hotels Near Me** uses the on-board GPS sensor to establish the user location and, then, consults a 60 000 records hotel database to find accommodation in the proximity. The user gets the ratings, address, phone number and user reviews for each hotel. Once he/she picks a hotel, the app shows a gallery of photos of the rooms and provides the price quotes for the duration of the stay. If the user actually books an accommodation, he/she receives a confirmation by e-mail. Hotels Near Me also lets the user browse for hotels prior to the trip by letting the user select 'elsewhere' from the main screen and then specifying the desired city [5].

- **Where To?** provides in real-time information on cheap gas, movies, shows, restaurants, traffic conditions, weather forecasts and news headlines [6] based on the user location.
- **Yelp** is a service, which includes an Android app, that provides location-based business information on top of real-time images from the phone's camera [7].
- **moreTourism** is a content-based and collaborative (hybrid) recommendation platform providing information about tourist resources depending on the user profile, location, schedule and the amount of time for visiting interest points isolated or combined in a route [8].
- **WebMD** is a location and preference-aware healthcare and fitness mobile application mostly for non-clinical use [9].
- **Cinemappy** is a location-based mobile application that provides enriched contextual movie recommendations. The current spatial and temporal user location is the basis for the contextualization and DBpedia, one of the best-known datasets publicly available in the Linked Open Data (LOD) project, is the semantic enrichment source [10].
- **Where** is a mobile application supported by a local search and recommendation portal that allows people build their own profiles and to find local events, read reviews and other business information, and write their own reviews [11].

All these context-aware mobile applications provide one or more categories of information related to the current user context. The recommendation categories range from health [9], tourism [2,4,3,5,8], business [4,6,7,11] to media content [10] and their sophistication varies from text, image, video or augmented reality suggestions. The user context information corresponds overwhelmingly to the user current spatial and temporal data.

The idea behind the Erasmus App is close to the "Where" or "Where To?" applications since it provides and allows the classification of location-based recommendations regarding user added locations or POI. However, the Erasmus App, which is the front-end of a dedicated location-based collaborative tool, does not include the remaining functionalities of these products since it focussed on creating a sense of community and a support network for Erasmus students.

3 Development Environment

The development of Android applications is a dynamic field involving several technologies and areas of expertise. Therefore, the first task of an Android developer is to select the set of inputs, languages, technologies, tools and API to use. Since the Android programming environment is Java-based, the distributed system (client and server modules) was fully developed in Java, using Eclipse as the Integrated Development Environment (IDE).

In terms of persistent data storage, the user local database relies on the local Android device SQLite database engine [12]. The central database is stored in a MySQL database server [13]. The communication with both SQLite and MySQL uses Structured Query Language (SQL) commands.

A dedicated object oriented protocol was created and used for the communication between the GUI application and the server. All exchanged messages extend from one class with undefined objects as attributes, allowing the definition of specific subclasses for the different message primitives to be exchanged.

To enable the installation of the application in as many devices as possible, the developed code only uses the Android application programming interface (API) 8 and the Google Android API for map views [14]. In particular, the Android API 8 is supported by Android devices with version 2.2.1 or higher [15].

To determine the user context, which in this case is the user location, both the GPS and the network providers are used [16]. The selection of the provider depends on whether the user is connected to the Internet or whether the user is indoors or outdoors. When displaying a map, the compass input is also used to show the North. The inputs of the accelerometer and the gyroscope have been studied and considered, but are not used in this version of the application.

4 ErasmusApp Location-Based Collaborative System

The ErasmusApp system is composed of the front-end mobile application, which includes a local database, and the back-end server module together with the central database. The system requires Internet access to be fully operational, i.e., to provide add, delete, edit, share and rate POI functionalities. Furthermore, the user can request a map view of all locations.

4.1 Architecture

Figure 1 shows the overall architecture that includes the Android GUI application together with the local database, the dedicated server module and the central database. The GUI application requires an Internet connection to interact with the dedicated server module, the Google Maps servers and all external sites used. Every module plays a well-defined role in the overall system:

- The Android application provides the graphical user interface (GUI), including, the Web and map views. The Web views correspond to the *Basic Erasmus info* Web links and the map views display the maps provided by Google Maps.
- The local device database contains a set of predefined locations necessary to provide the *Basic Erasmus info* as well as all personal user-defined POI.

- The server module processes the client requests and interacts with the database server using SQL. The client/server communication uses a dedicated object-oriented protocol.
- The central database contains all information about the registered users, shared POI, ratings, etc.

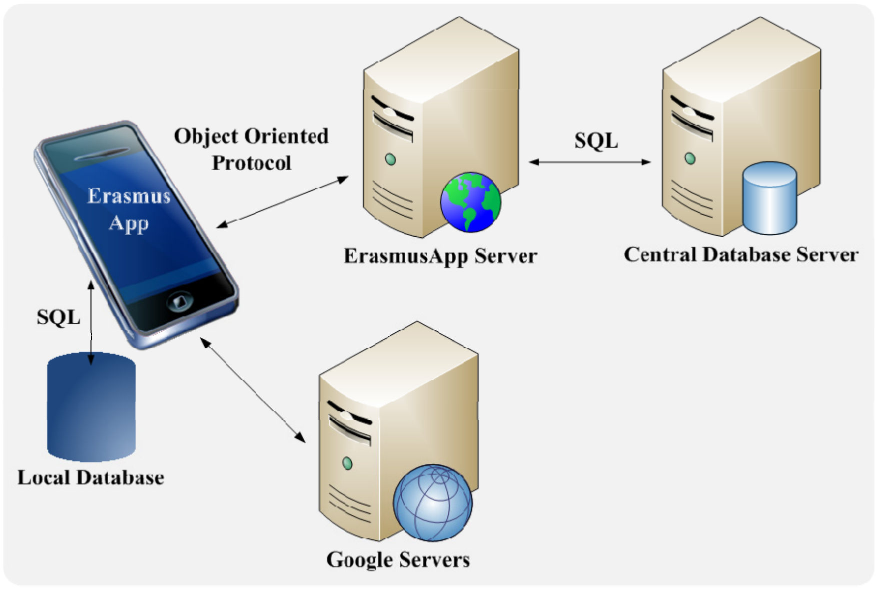


Fig. 1 ErasmusApp system architecture

4.1.1 Persistent Storage

The enhanced entity–relationship (EER) model of the MySQL central database in crow's foot notation is shown in Figure 2.

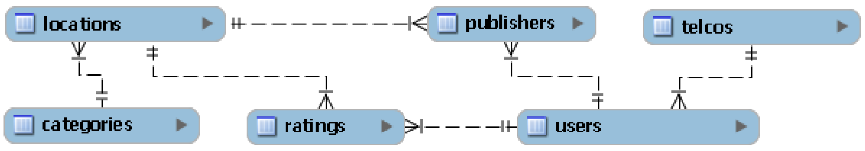


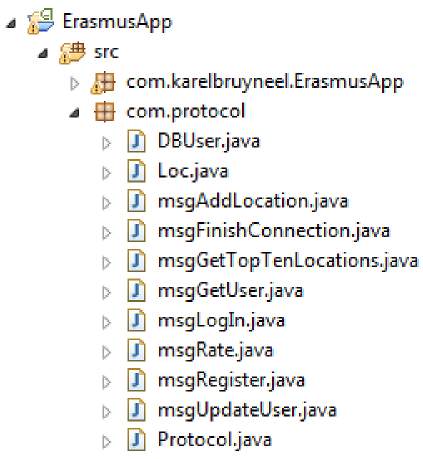
Fig. 2 Central database EER model

The main central database tables are the *locations* (POI) and the *users* tables, which are connected via the *ratings* and *publishers* tables. Additionally, every user subscribes a given telecommunications company (telco) and every location belongs to a category from the *telcos* and *categories* tables, respectively. The *publishers* table allows a location to have multiple publishers (users). The *ratings* table stores the location ratings, where multiple users can rate a single POI and a user can rate multiple POI. The *categories* table holds the eight pre-defined POI categories: hospitals, supermarkets, restaurants, hairdressers, outside & quiet, touristic, nightlife and others.

The on-device SQLite database, which stores the personal POI, holds a single *locations* table with id, name, description, latitude and longitude attributes.

4.1.2 Object Oriented Protocol

The application level protocol used for the communication between client and server modules contains several messages with contents depending on their purpose. Figure 3 shows the contents of the protocol package. This package provides the client-side and server-side modules with a common understanding of the exchanged messages and objects.



```
ErasmusApp
  src
    com.karelbruyneel.ErasmusApp
    com.protocol
      DBUser.java
      Loc.java
      msgAddLocation.java
      msgFinishConnection.java
      msgGetTopTenLocations.java
      msgGetUser.java
      msgLogIn.java
      msgRate.java
      msgRegister.java
      msgUpdateUser.java
      Protocol.java
```

Fig. 3 Protocol package

The package defines the *DBUser* and *Loc* objects which represent a user and a location (the object *Location* is already defined in the Google Android API). Furthermore, it specifies a class called *Protocol*, which is inherited by all protocol messages, with four generic objects that will be latter casted to specific object types in the message subclasses.

After messages are exchanged, the connection between client and server closes in order to minimize the cost of communication.

4.2 Distance-to-User

The equirectangular projection distance (in meters) between the user current position and a given POI is calculated using Equation (1) and Equation (2):

$$\phi_m = (lat_u + lat_{POI})/2 \tag{1}$$

$$d = R\sqrt{(lon_u \cos\phi_m - lon_{POI} \cos\phi_m)^2 + (lat_u - lat_{POI})^2} \tag{2}$$

where ϕ_m is the mean latitude angle, R is de radius of the Earth (in meters) and lon_u, lat_u, lon_{POI} and lat_{POI} represent the geodetic longitude and latitude coordinates of the user u and *POI*.

When the user requests recommendations, they are selected and ordered by rating (average number of stars) and distance-to-user (*d*). Therefore, the application recommends POI ordered by the quotient between the average rating and the distance-to-user (stars/m).

4.3 Functionalities

The ErasmusApp has a dual-mode operation: standalone (guest users) or client (registered users) application. In standalone mode, guest users have access to generic info, a map-based interface and to the local database to store and retrieve personal POI. In distributed mode, authenticated users can, additionally, share POI as well as get and rate recommendations sorted by category, rating and distance-to-user. The stand-alone mode is intended to minimize connection costs and for offline operation. The GUI application offers the following functionalities:

- In stand-alone mode to guest users:
  - *Map* shows the user location on a Google map.
  - *Basic Erasmus info* loads an initial set of POI, including essential locations such as the *Closest Hospitals*, Porto *ESN* and *Looking for a House?* information, as well as famous student meeting points like *Piolho*, a downtown coffee house, *Cais da Ribeira*, the old quay by the river, and *Matosinhos beach.* This option provides, apart from the regular Android activities pages, a Web view activity.
  - *Personal Locations* allows the user to: (*i*) add, edit and delete personal POI to and from the local database and (*ii*) display personal POI on a map.
- In distributed mode and in addition:
  - *Login* for user authentication and a registration button.
  - *Create/Edit profile* allows the user to register and modify his profile.
  - *Find Location* recommends the top ten shared POI per category. These recommendations can be ranked by distance-to-user, rating or both rating

and distance-to-user. The user, after receiving the top ten recommended locations, can go to the POI page to rate and view the location on a map.

– *Personal Locations* allows the user to: (*i*) add, edit and delete personal POI to the local database; (*ii*) display personal POI on a map; and (*iii*) categorise, rate and share the POI with fellow users (only rated categorised POI can be shared). When adding a new POI, its position can be determined automatically via the GPS sensor, the network provider or established by the user by clicking on the map.

5 Tests and Results

Tests were successfully conducted both with the Android device and the Eclipse emulator plug-in. Although the application runs faster on the device, the emulator was used to capture higher quality screenshots.

Map

The mapping functionality is widely used to display the user and POI over Google Maps. Figure 5 displays a tapping map of the user location and other POI.

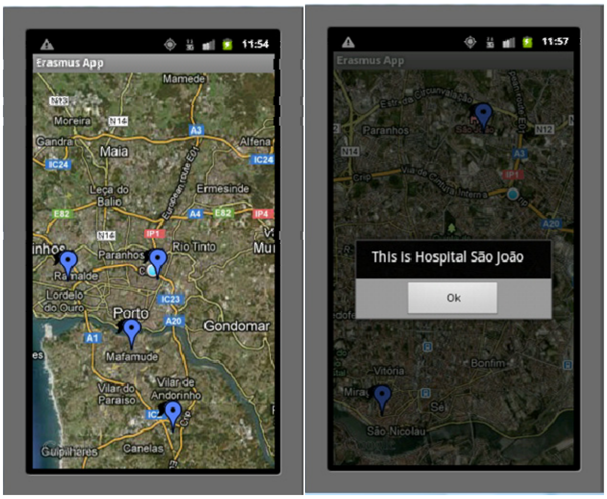
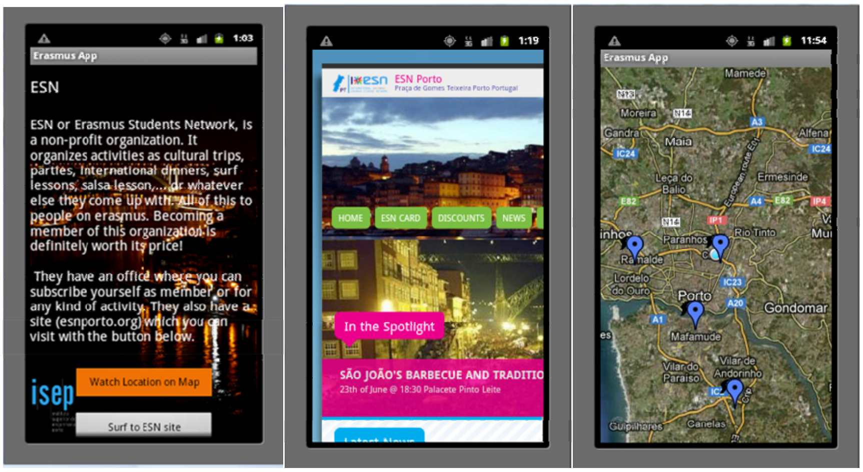


Fig. 4 Displaying the user location with a tapping map

Basic Erasmus info

The *Basic Erasmus info* loads a default set of POI, including hospitals, accommodation, Erasmus-related information and student meeting points. Figure 6 displays the ESN info, ESN site and a map with the user and the closest hospitals.



ESN

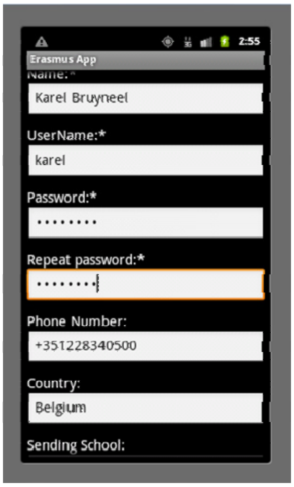
ESN or Erasmus Students Network, is a non-profit organization. It organizes activities as cultural trips, parties, international dinners, surf lessons, salsa lesson,... or whatever else they come up with. All of this to people on erasmus. Becoming a member of this organization is definitely worth its price!

They have an office where you can subscribe yourself as member or for any kind of activity. They also have a site (esnporto.org) which you can visit with the button below.

Fig. 5 Displaying basic information

Registration and Profile Editing

User registration and profile editing share the same screen as shown in Figure 7.



Name:*
Karel Bruyneel

UserName:*
karel

Password:*
••••••••

Repeat password:*
••••••••

Phone Number:
+351228340500

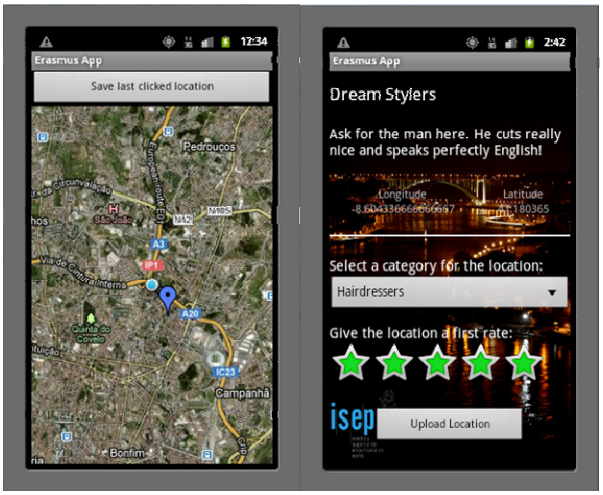
Country:
Belgium

Sending School:

Fig. 6 Registration/Profile editing

Personal Locations

Personal POI can be added using the current user location, specifying the geodetic coordinates or via a clickable map – Figure 8. When the user decides to share a personal POI, first has to attribute a category and a rating to the POI and, then, has to upload the POI to the central database – Figure 8. Upon success, the user gets a new page displaying the new shared POI together with the attributed rating.



Ask for the man here. He cuts really nice and speaks perfectly English!

Fig. 7 Adding a personal POI via a clickable map and sharing a personal POI

Find Locations

Find locations returns the ten best ranked shared locations per category. The ranking is based on the collaborative rating and/or on the distance-to-user. Figure 9 shows the result of a *Find locations* request for supermarkets ordered by rating together with a detailed view of the highest ranked supermarket. The detailed view presents the name, description, coordinates and the collaborative rating of the item. Additionally, the user can add/edit his personal rating, which is immediately incorporated into the collaborative rating. The button on the bottom of the page loads a map displaying this POI and the user location.

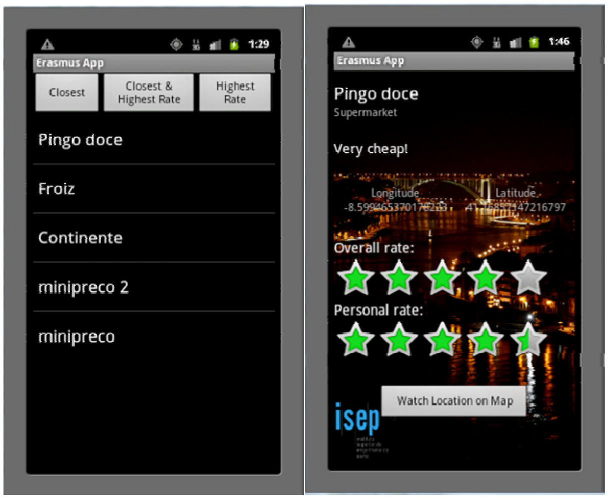


Fig. 8 Recommending shared supermarkets by rating

6 Conclusion

The application is intended to foster the integration of Erasmus students by collaboratively creating and rating a POI database for current and future usage. Users decide whether to keep or share their personal POI.

Achievements

The ErasmusApp system is a distributed application composed of a central database, a server module and a mobile map-based GUI application with a local database. The Android-based GUI application allows both standalone and client operation modes. Guest users can add, rate, edit, display on a map and delete personal POI that are stored in the local database as well as register. Authenticated users can edit the profile, share personal POI, by uploading them to the central database, and get recommendations by category ranked by rating, distance-to-user or the combination of both. Shared POI are not editable nor removable by regular users.

The communication between client and server and between server and database are both supported by TCP. While the first uses a dedicated protocol, the latter exchanges SQL commands. The user spatial context is established through the GPS and/or the network provider.

The tests conducted showed that the implemented functionalities work properly. The application can be easily ported for other cities than Porto since only the basic info needs to be substituted.

Future Work

The system has several limitations and new functionalities are currently being developed, including geo-referenced video sharing, semantic enrichment via LOD repositories and hybrid recommendations, i.e., that take into account both the user personal profile and the fellow users ratings. In terms of personal profile, it is being refined to accommodate both spatial and temporal context data, e.g., use the temporal context to make recommendations based on the period of the day, day of the week or season, and the past POI ratings by category. Another promising feature is to add a new accommodation main entry in the start menu, rather than in the basic Erasmus info section, and expand it to allow finding flats and flatmates.

More work has to be done on reporting invalid or duplicate locations. The server should look not only for identical names, which it does right now, but, also, for nearby identical category POI. In this case, the user should get a list of identical close POI for selection before confirming his own POI input. Additionally, the user should be able to report incorrect data, e.g., wrong addresses. This report would then notify the publisher of the fact and grant the publisher the permission to correct his input. At a later stage, the application could compute the credibility of the publishers and, thus, use it to compute the POI rating.

Finally, user assessment and feedback has to be performed to refine and improve the ErasmusApp.

Acknowledgments. This work was partially supported by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «FCOMP – 01-0124-FEDER-022701». The authors would like to thank their home institutions, ISEP and KAHOSL, and the EU Lifelong Learning Programme, in particular, the Erasmus student mobility programme.

References

1. PCWorld, 11 Essential Android Travel Apps (2010), http://www.pcworld.com/article/193523/11_essential_android_travel_apps.html (accessed in June 2012)
2. van Setten, M., Pokraev, S., Koolwaaij, J.: Context-aware recommendations in the mobile tourist application COMPASS. In: De Bra, P.M.E., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137, pp. 235–244. Springer, Heidelberg (2004)
3. Gavalas, D., Kenteris, M.: A web-based pervasive recommendation system for mobile tourist guides. Personal Ubiquitous Computing 15(7), 759–770 (2011), doi:10.1007/s00779-011-0389-x, ISSN 1617-4909
4. Wikitude, Wikitude - The World's leading Augmented Reality SDK (2012), http://www.wikitude.com/ (accessed in June 2012)
5. Bluemedialab.com, "Hotels Near Me" App. (2012), http://blumedialab.com/products/google-android-apps/Hotels-near-me-for-Android.htm (accessed in June 2012)
6. FutureTap GmbH, "Where To?" App. (2012), http://www.futuretap.com/apps/whereto-en/ (accessed in June 2012)
7. Yelp Inc., Yelp. (2012), http://www.yelp.com/ (accessed in June 2012)
8. Rey-López, M., Barragáns-Martínez, A.B., Peleteiro, A., Mikic-Fonte, F.A., Burguillo, J.C.: moreTourism: Mobile recommendations for tourism. In: 2011 IEEE International Conference on Consumer Electronics (ICCE), pp. 347–348 (2011), doi:10.1109/ICCE.2011.5722620, ISSN 2158-3994
9. Liu, C., Zhu, Q., Holroyd, K.A., Seng, E.K.: Status and trends of mobile-health applications for iOS devices: A developer's perspective. Journal of Systems and Software 84(11), 2022–2033 (2011), doi:10.1016/j.jss.2011.06.049, ISSN 0164-1212
10. Ostuni, V.C., Di Noia, T., Mirizzi, R., Romito, D., Di Sciascio, E.: Cinemappy: a Context-aware Mobile App for Movie Recommendations boosted by DBpedia. In: Proceedings of the International Workshop on Semantic Technologies Meet Recommender Systems & Big Data (SeRSy 2012), pp. 37–48 (2012), http://ceur-ws.org/Vol-919/paper4.pdf ISSN 1613-0073 (accessed in December 2012)
11. Where, "Where for Android" (2012), http://site.where.com/download-where/google-android/ (accessed in June 2012)
12. Vogel, L.: Android SQLite Database and Content Provider - Tutorial (2012), http://www.vogella.com/articles/AndroidSQLite/article.html (accessed in June 2012)
13. MySQL, The world's most popular open source database (2012), http://www.mysql.com/ (accessed in June 2012)

14. Volgel, L.: Location API and Google Map in Android (2012), `http://www.vogella.com/articles/AndroidLocationAPI/article html` (accessed in June 2012)
15. Android Developers, "Platform Versions" (2012), `http://developer.android.com/about/dashboards/index.html` (accessed in June 2012)
16. Android Developers, "Location Strategies" (2012), `http://developer.android.com/guide/topics/location/strategies.html` (accessed in June 2012)