Abstract

The aim of our app is to collect in a unique web portal all of the opinions about objects, places, events and other interesting items that users want to evaluate. Whereas standard online systems are specific and limited to few topics, the app we developed involves many different categories. It allows you to share feedbacks, experiences, and doubts with the other system users. Existing solutions do not consider users' reliability and credibility. On the other hand, our app wants to give importance to evaluations written by our most appreciated users, which give us a useful and earnest contribution. The system is built to avoid a misleading usage of reviews for commercial purposes, making the reviews as reliable as possible. The most important feature is that the user can find any type of review with incredible speed.

Related Work

Although other systems with similar functionalities already exist, we strongly believe that none of them allows you to give and receive reviews on a broad range of services. Moreover, existing systems are not able to calculate an evaluation based on users' reliability, making a review as realistic as possible. Shops and online sellers such as Amazon, eBay, TripAdvisor, etc., always focus on the products they sell and are vulnerable to falsified reviews.

Usage model

Once you have installed the app, you will find yourself on the log-in page, which allows you to log in and enter. To log in you need to submit name, email, and password into the appropriate fields. The account is activated by submitting a univocal code received by mail. You can access by 3 different alternatives: by credentials, by Facebook or as an anonymous user. This special user cannot interact with the whole system but can only access to it as a visitor. The home page shows links to the most visited items, to the most recently viewed items and to the user's favorites. The bar on the top of the page is visible in every window. Therefore, it is always possible to search for an item in the system and, moreover, there is another pop-up menu on the left of the page. This is useful to rapidly reach the following pages: profile, notifications, one's reviews, favorite users and items, 'write a review', settings, exit and customer service. To review a new item which does not already exist in the system you first need to create it and store its information into the memory of the system. You can show an item by selecting it from past searches and the home page. An item page contains pictures, its global scoring, category, description and a list of reviews. It is also possible to access the questions and answers page, to start the procedure to add a review or add that specific item to the user's favorite. By clicking on a review, you can show another page, where you can rate it positively or negatively according to your judgment. Finally, every user has a personal profile page, which can be reached by clicking on the link present in the name of each user. On this page is shown the user level, his/her scoring trend on a line chart and a profile picture. Every user can edit his/her personal profile by adding information and changing his/her profile picture.

Architecture design

The app absolutely needs an Internet connection to work. Although we tried to limit the amount of data traffic needed to use the app, a digital connection is essential to the working of the app, as the sharing of information among users is based on a client/server approach. We assume that user knows how to basically use a smartphone. The main components are the application and the remote server, which communicate using HTTPS protocol for standard communications and GCM service for notifications. The server program is written in Python (about 2000 rows) and mainly composed of the central database. Other tasks, such as photos compression, data presentation, and others are managed by the smartphone (more than 10000 rows). The server is managed by SSH, as it was rent by

DigitalOcean. Moreover, another database has been implemented. This is stored on the smartphone with the aim of manage photos caching.

Implementation

As far as the initial planning is concerned, the smartphone side has been completely implemented. The only missing features are the possibility of reporting pictures and commenting on answers. Nevertheless, the server is completely working. To create the applications, we used the IDE AndroidStudio, while the server's side was written by using a simple text editor. Besides standard libraries, we used the following: CircleImageView", "GSON", "FacebookSDK", "jjoe64-graphview", "Firebase", "Swipelayout". On the server side, we instead used: "BaseHTTPServer", "cgi", "ssl", "urlparse", "hashlib", "json", "mail", "Levenshtein", "datetime", "urllib2", "urllib", "sqlite3".
Among the most interesting parts of code we underline the SuperActivity class, from which we inherit all of the activities, ImageViewLoading to manage the photos downloaded from the server, CircleImageViewSpecial which creates a special animation on a CircleImageView, and generally, some utility classes such as Utility, UtenteLogged e Images. Finally, the system used to calculate users' reliability is particularly relevant.

Evaluation

During the testing phase, several friends of ours were involved. We asked them to download the app and use it and to give us some feedback. To test the app on smartphones we used real smartphones, whereas we used an emulator to test the app on tablet devices. This allowed us to find out flaws and problems and to improve the quality of the graphic interface. Furthermore, we decided to make the design easier and essential, for instance, by making the size of the buttons greater with clearer information. We also tried to improve the system reactivity by adding alerts and feedback for each action. When the app is answering something to the server it may take some time. To notify the user that the app is still working we create some functions to show a message and a progress bar.

Limitations

This implementation limits the user, as he/her cannot modify a review, comment on other users' answers, and insert additional pictures to existing items. At the moment it is not possible to look for a specific user or filter reviews.

Member contributions

Luca Di Liello has developed the server side and also taken care of the project coordination, the preparation of the utility and the writing of a small part of Activity. Mohamed Dounnani has written a large part of Activity and has implemented some features such as notifications, autocomplete, translation, menu, camera usage and writing of emails. Enrico Campagnaro has mainly taken care of the graphic parts such as photo upload and presentation, layout editing (which depends on the specific user) and the creation of styles and drawable to make the graphic interface more enjoyable.
Written rows:

Luca Di Liello 5000 Java code and 2300 Python code and a minimal part of layout.

Enrico Campagnaro: 3000 Java code and 70% layout

Mohamed Dounnani: 3000 Java code and 30% layout

Lessons learned

In a future implementation, we should give more importance to the graphic interface, making it enjoyable and nice to see from the beginning of the work. Moreover, it's important to use external libraries where possible and a clear Grandle management to avoid troubles with the compatibility of the following libraries:

References:

- Facebook Developer: https://developers.facebook.com/docs/android
- CircleImageView: https://github.com/hdodenhof/CircleImageView
- GSON: https://github.com/google/gson
- GraphView: http://www.android-graphview.org
- Firebase: https://firebase.google.com
- GCM: https://developers.google.com/cloud-messaging/
- SwipeLayout: https://github.com/daimajia/AndroidSwipeLayout

BaseHTTPServer, cgi, ssl, urlparse, hashlib, json, mail, levenshtein, sqlite3: https://docs.python.org/2/library/basehttpserver.html