Team 02

WAT - Where Are They?

Fabio Dolci 167969
fabiodolci87@gmail.com
Mirko Nespoli 167966
mirkonespoli@gmail.com

Abstract

Sometimes when visiting a city it take place that someone stops to admire the landscape or just to photograph some detail. Often, when they want to resume the trip, happens that they don't find anymore other members of the group.
Nowadays almost everyone owns a smartphone with a mobile data plan so we just ask ourselves "Why don't provide to who is involved in a visit a simple and discreet tool, but at the same time powerful, able to track the movements of people, that could avoid problems as the one described above?".
To meet this need, we thought to realize a smartphone application, for Android platform, that provides the useful functionality of tracking, with in addition some useful tools. The purpose of the system is the real time tracking of group of person in order to let them retrieve the others in case of lost. If that happens, the application will guide automatically the user towards the group.
The leader instead could manage the whole activity and has a broader view on the situation of all the users. Moreover they can interact with the system sending messages and photos to the other members, sharing thoughts and pictures of interesting places or funny things. In addition, the leader of the group can manage the activity and invite all the users to reach him in order to tighten the team. Information about user's position and about the center of the group are shown on a simple and clean map.

1. Objectives

The main objective is the realization of an Android application that helps managing and enjoying trips and, in general, groups of people linked by a common objective.
First of all we had to understand the needs of the system that we were building in terms of trust, power consumptions and interactivity between users. We need a reliable systems, that is always reachable and that performs all the controls of the access. For these reasons, we have chosen an online server hosting with a SQL database used for storing data.
The user's position is acquired periodically by the smartphone and it is sent to the remote server that stores it in the database. All the geographical positions are used to understand where the group is located; this information could be exploited by someone that, for any reason, gets lost and wants to reach the others.
For a better organization of the functions, it was necessary to split the different functionalities between two types of person. In addition to the basic operations described above, a normal user can take a look at the map and see where is located with respect to the group, consult the itinerary or read/post comments about the trip and attach photos. The leader of the group, instead, is also in charge to draft a list of activity with the fundamental appointments of the day(if needed). Moreover, he could also see manage all the people joining the journey, knowing the position of every user.

Our project fits some of the arguments of the Multimedia Networking course: various aspects regarding Multimedia (deployment of images and its sending, GPS data) and networking (i.e. HTTP connections) are involved in this activity. For these reasons we could assert that our project is compliant with some of the topics covered during the course.

2. Achievements

The principal achievement of our project activity is the creation of an application fulfilling the objectives previously described, and a server with all the necessary php scripts along with the database.
All the predefined goals were achieved, including some more features not considered at the beginning.
The main objective of the application is the estimation of the position of the group in order to follow the shortest path to rejoin it. This is done thanks to the API provided by Google for the Android platform. All these operations should be thrifty regarding the battery consumption. To satisfy that requirement we exploit the Fused Location Provider that manages the underlying location technology in a smart way, and gives us the best one according to our needs. In our case we decided to use the "*LOCATION NETWORK*" provider. Android's Network Location Provider determines user position using cell tower and Wi-Fi signals, providing information for indoors and outdoors scenario, responds faster, and uses less battery power. Moreover it uses Assisted-GPS, if available. We choose the A-GPS system because it can address poor satellite signal conditions, downloading orbital information, like ephemerides and the almanac, by using data available from the network, either mobile or through a standard internet connection: this improve the startup performance for a fast localization. The update of the position is done every minute; the choice of this time interval is a compromise between accuracy on the localization of the single user and the reduction of the power consumption due to frequent location updates/exchanging of data with the server.
The second objective regards the whole exchange of data between the application and the server. All the connections are based on HTTP protocol. We have chosen this because we have only to transfer simply string of data from both side, and it's the easiest one. The application has to send a request with *POST* attribute plus the information necessary and the client respond with a status line and the associated data.
For the transfer of the picture, we still use a *POST* request, because the code automatically proceed to convert the image from the original type to Bitmap, serializing it. We also decided to use HTTP instead of FTP in order to let our packets flow over every system. In fact FTP traffic is not allowed on some public networks or blocked by firewalls.
We choose to store the image as a file on the database with respect to *Blob* type because it's faster, it can be ordered easily without losing information and, for large data sets, it's easier to control it.
Even for the communication of structured data (i.e. the list containing all the comments) we transfer all the information formatted with JSON, so it's only necessary to elaborate an HTTP *POST* request. JSON is an open standard that permits us to transmit data objects (attribute-value pairs) between the server and the application in a simple manner. At the client side, it is necessary to parse the content of the server reply to our query in order to retrieve the information needed.
Regarding the navigation activity, we have chosen the Google Maps Navigator using the relatives APIs. We have made this choice because they are present on every Android device, works very well and are easily accessible with few line of code. If another application of navigation is installed on the smartphone (i.e. Sygic) it is possible to choose it instead of Google Maps Navigator.
Even for the capture of an image, we have decided to use the integrated camera application of the smartphone.

Following is reported an explanation diagram of the architecture of the classes as implemented in our project.

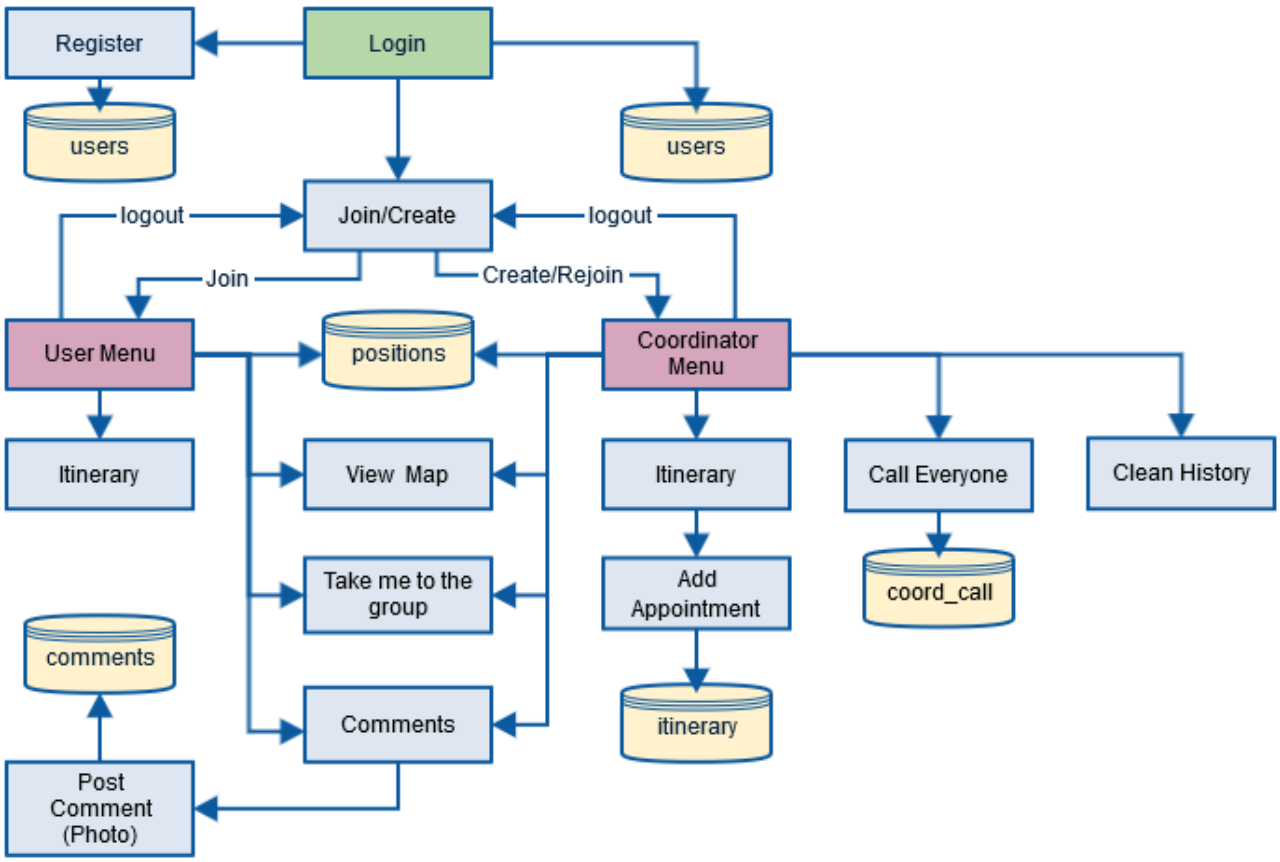

Figure 1 - Diagram of the application

We choose Android API available from version 4.1 (API 16) since those versions can be found on about 88.7% of the Android devices all over the world[1]. This choice was made to target our application to the largest number of devices possible, without losing functionalities. Moreover Android's source code is released by Google under open source licenses, so everyone can access it and learn the code. This results in a vast community of developers that supports and suggest solution for this platform.
For the server we have chosen to host it on an online free hosting company (Altervista) that provides us an Apache server with PHP and msqli installed. Altervista is a web platform where you can open a free host, with some limitation on the initial traffic. Here you can create a website with PHP, MySQL databases and HTTP access. The use of space is free, but the resources available, such as web space and monthly traffic, are initially limited at 3GB of storage, 20000 maximum query per hour (expandable to infinity free for one year) and 30 GB of traffic. We have choose this solution because is free, easily accessible and give us all the functionality necessary, like PHP and MySQL. Moreover, we have already a little experience on this system.

3. Project Implementation

At the beginning we need to choose which development environment use to implement the mobile application side of the project. After a brief analysis of the available tools, we choose Android Studio since it is the official IDE for Android.
As regards the server, we decided to host it on an online free hosting company running an Apache HTTP server. It seemed to be the best compromise for the initial version of our client-server application, as explained before.

APPLICATION DESCRIPTION

The application starts with a login/register screen. Here the new user should tap "*register*" and follow the guided procedure in order to add his username and password into the system. The one that is already archived can instead continue adding his information and tap the "*login*" button. The system checks the credentials connecting to the database, and if they are correct it leads to a new screen where the user can either choose if join to an existing trip or if he wants to start a new one becoming the leader of it. The coordinator of a group can also rejoin the tour entering in this section.
Here the application splits in two branches; the first one is for the normal user and permits him to perform some basics operations. The leader of the group instead has the access to more advanced tools. As soon as the main screen starts, the system initiate to collecting the position of the smartphone and send it to the server. At the same time a request for the barycenter of the group is performed. The user/leader menu is divided into tiles in order to have a clean and user-friendly layout. Each of these redirect to the specific activity linked to each button.

- **Login**

This is the first activity that appears when the app is opened. As the name suggests, it is possible to login a user to the system. The user is directly prompted to insert username and password. In case of a new user just click on the "*register*" button in order to proceed to the registration of a new user.

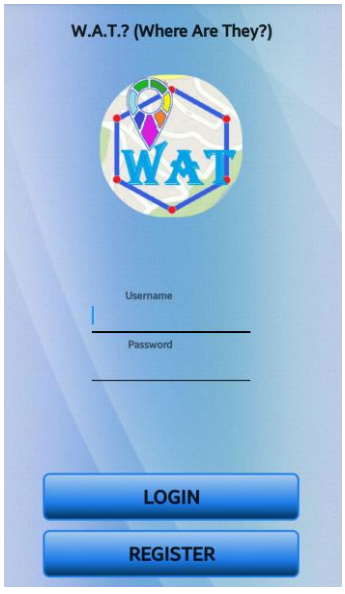

Figure 2 - Login

- **Register**

The register screen is similar to the previous one and let the new user to insert his username and password into the system. A check on the uniqueness of the credentials is done in order to avoid misunderstanding in managing the users.

- **Join/Create**

In the join/create activity is possible to choose either to create a new group or join an existing one. The leader of a group can rejoin it simply selecting "create a group".

- **User menu**

This is the main screen that present 5 tiles and the user can choose what to do. At the starting of this activity, the status of the location provider is checked. If the position service is not active, it will be requested to turn it on. The best choice among A-GPS and network localization is selected by the system in order to reduce at the minimum the battery consumption. Moreover, once the position of the user is available, it is sent to the server and the barycenter of the group is retrieved. Updates on the position are provided every minute. On this activity, there is even a "dummy" cycle that, every minute, check the

presence of a new comment or a global call of the coordinator. If so, it will appear a notification as explained later.

Figure 3- Coordinator Menu

- **View map**

This opens a fragment containing a map (from the Google Maps service) that displays the position of the user and the position of the barycenter of the group. A purple marker is used to show the location of the user; a green one instead is used to display the center of the group.

- **Take me to the group**

When the user is lost, it could tap this button and a navigator starts towards the group barycenter. An intent loads the navigator (standard Google Maps Navigation, or if it is present, another navigation tool) that leads the user to the position of the others.

- **Itinerary**

Here it's possible to see if useful appointments are set by the leader of the group.

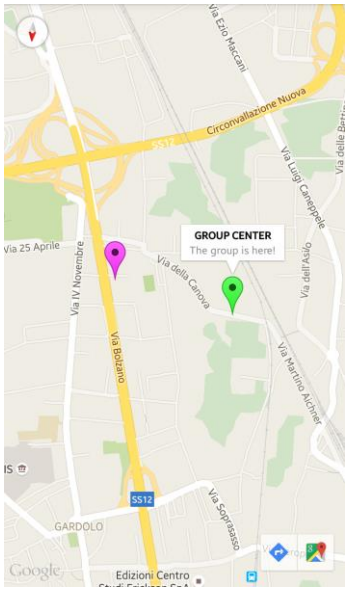

Figure 4 - View Map

- **Itinerary (coordinator side)**

0As for the user with the addition to insert new important meetings. In this case, a new activity will appear, with the possibility of insert a time with, the *TimePicker* model, and a brief description.

- **Comments**

A complete list of the comments regarding the group is show. If a photo is available, tapping the relative comment opens the image. A notification of new messages is displayed when somebody posts something on the group. On this activity, the user can "*post comment*" and insert a new message. If he wants to add a photo to the comment he could tap "*upload pic*" and choose either to take a new photo with the camera of the phone or load it from the gallery. In both case, before the transferring of the image, a message to approve the selection is showed, in order to correct eventual mistake.
In any case, one can decide only to post something, with a title and the relative message.

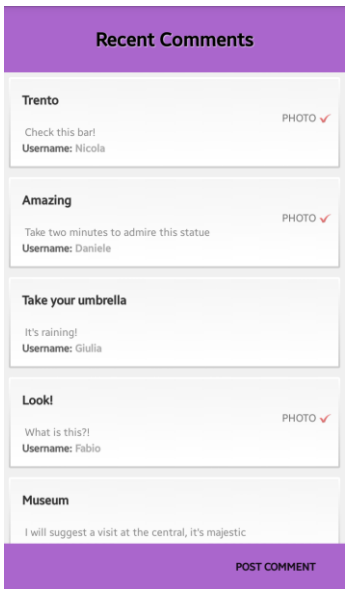

Figure 5 - Comments

- **List of users (only coordinator)**

Here the leader can see all the users joining the trip. Tapping on them, opens a map that shows the position of both.

- **Call everyone (only coordinator)**

Pushing this button triggers a notification on the phone of the other users that are in the group coordinated by him; this starts the navigation towards the leader of the group.

- **Clean history (only coordinator)**

When the coordinator taps this button calls a routine on the server that cleans the comments and itinerary tables respectively, of the group that he is managing.

In order to have a better interaction with the application, we have decided to put some notification in case of new contents.
The user will receive an alert if there is a new message or if the coordinator decide to unite the whole group. In the first case, pressing it will make open directly the comments activity in order to check the news. In the second one, instead, it will open the navigation towards the position of the coordinator to facilitate the reunion of the group.
For the notification, we have decided to make the application check every minute if there is something new, according to their purpose, instead of using push notification and Google Cloud Messaging. We have made this choice because this was one of the optional feature, and the implementation of the GCM will require a big change of the background; due to the fact that it was implemented while we were concluding the development of the application, we don't have the time and resources in order to implement it in a proper way.

SERVER DESCRIPTION

The server is based on the Apache HTTP Server Platform, with database client version: libmysql - 5.1.71 and with PHP extension of mysqli type. Altervista grants 3 GB of storage for the files necessary and the information saved in the database.
On the server, we will store all the php scripts, used in order to connect application and database, and the various picture uploaded by the user, in addition to the database data.
The server is divided into two directories, *imgupload* and *webservice.*
The first, include a folder with all the uploaded picture and a php script called *upload_image* that is used in order to store all the picture in the defined space.
Instead, *webservice* include all the main functionality and scripts necessary to handle all the information from/to the database; to have a better comprehension of the activity, a table is provided with a brief description of all the functions.

Name (*.php)	**Main functionality**
addCoordCall	Add to the relative table the information regarding an important meeting
addItinerary	Add an appointment to the table *"itinerary"*
addComment	Add a comment of a user
comments	Retrieve all the comments from the table and send it to the requester
config.inc	Include all the starting information necessary to establish the connection
coordName	Returns the username of the coordinators of a specific group
createGroup	Create a group with leader the coordinator who call this
delete	Used to erase all the comments and the itinerary of a specific group
groupLocation	Returns the position of the group

itinerary	Retrieve the whole itinerary and send it to the requester
listUser	Returns the list of all the people in a group
login	Permits the login of a registered person on the application
newComment	Check if there is a new comment in order to notify the users
register	Permits the registration of an user in the application
retrieveCoordCall	Check if there is a new global call in order to notify the user
updateCoordinates	Update the coordinates of a single user
verifyGroup	Check if the name of the group is available or not and eventually add it

Regarding the estimation of the position of the group, this is done in the script groupLocation.php. There, we decided to order all the users of the group by the latitude(longitude) criterion. Once ordered, a median position is chosen as reference. The underlying hypothesis is that the group is generally clustered with only some outliers (the user that are likely lost).
We decided to put all the computation complexity on the remote part of the system relaxing the needs and the computational load on the mobile side.

DATABASE DESCRIPTION

The database is hosted on the Altervista platform and the server type is MySQL version 5.1.71-community-log - MySQL Community Server (GPL). The connection between server and database are with UNIX socket and the charset is UTF-8 Unicode (utf8).
The database is composed by six tables, each one with a precise purpose and properties. The various table are not interconnected, in order to to reduce the complexity of the background.
In **comments** there are stored all the messages posted by the users. The main data, like username and title, are *varchar* of variable lengths; post_id is an *int* used as an incremental and univoque identification; PostTime is of type *TimeStamp* and is automatically filled.
Coordinators is used to have a connection between the username of the coordinators and the respective group that they decide to create.
Coord_call hold the information about the call that the coordinators execute in order to warn everybody of a meeting. Contain the name of the group, the latitude and longitude of the leader, stored as *float(15,10),* and a TimeStamp.
Itineraries include all the data about the activity of the group, incorporate the name of the group, a short description of the appointment and time posted as *Time.*
Positions is the core of the database, that include the unique position of all the users. For every person there are associated the activity that he's joining, the coordinates, a *bigint(20)* that show the last time of update (in seconds), and a *tinyint(1)* which indicates if he's the leader or not.
At last, **Users** include the username that every person choose and the password. During the registration phase, this information is coded with the SHA algorithm as a *text* type. The choice of this protocol was made because is easy to implement while maintaining a reasonable level of data encryption.
In the github directory /doc is provided a file called *database.pdf* that describe the whole architecture of our database in order to reconstruct it.

4. Conclusions

This application have a lot of potential and possible expansion, maintaining this original purpose or changing it.
The first main implementation can be the integration of the GCM exposed before. This will reduce the energy consumption, quantity of data transferred and increase the dynamicity of the product. In second instance, a better layout, more intuitive and engaging, that follows the Material Design, can be deployed. For this part, we don't have the basis in order to achieve it in the best and proper way, so we haven't done it so far. After, a better estimation of the barycenter of the group can increase the precision of the navigation. In order to have a better position, probably a clusterization of the user should work fine. At the end an upgrade on the server side and a better storage of the image can lead to an improvement of the overall performance.

Despite that, the basis of this application can be used to create new products that change the target of the systems.
For examples, it can be used in an emergency scenario where the coordinator knows the position of the all equipages and equalized them in order to optimize the time of arrival on an event. The application can even handle the transmission of important news about the patient in a fast and reliable way in order to perform the rescue at best.
Another possibility can be the deployment of a game based on the location of a random object. Using the real scenario, different groups can try to reach a target in the shortest way in order to conquer it, while avoiding or deliberately "attacks" the others.
Last but not least, the whole application can be reduce on the creation of meeting for a bunch of people, finding the best solution for all, considering the center of their actual position and using various POI in order to find a meeting place pleasant for all.

Github repository

https://github.com/fabiodolci87/WAT.git

References

[1] [https://developer.android.com/about/dashboards/index.html?utm_source=suzunone]