Geo-tagged time capture

Damu Ding
Dept. information engineering and computer science
University of Trento, Italy
damu.ding@studenti.unitn.it

Abstract

Nowadays more and more people have the smartphone, they can use it to communicate with each other.
Sometimes, what they need is not only to take a photo, but also to attach the photo one small tag, like location, time, coordinates, etc. For instance, these information can be applied in many cases:

- **In the restaurant:**
  When somebody can't decide which dish to eat on the menu, just open the GPS and look at the photos from the others at this location, he/she will quickly know which one is typical in this restaurant, and which one is more ordered by the others.
- **In the tourism**
  For example, when somebody is in Piazza Duomo in Trento, he/she can look at all the pictures in the Piazza Duomo from the others, maybe he/she can find some missing place from the pictures catched by the other tourists.
- **In the hotel**
  How is the room in this hotel? The layout? Is there the swimming pool? How about the hotel nearby? So many hotels are arround me, which one is better for me ? Filter the related hotels' photos , look at the environment inside the hotel from others, now the problem is solved!
- **,etc…**

Now we're trying to develop an Android application to satisfy the user's need.
The following features can be realized by our application:

- Get your location
- Geo tagged and time capture for the local photos with system's camera / our customized compass-camera
- Filter the local photos with your current location
- One simple compass application
- Upload the photos which you took to the Cloud Server(all the related information included)

- Filter all the photos in the Cloud Server with your current location, what's more than the local filter, in this way you can know who uploaded these photos.

Objectives

We need to develop **a geo-tagged time capture application within Android** that not only can be applied by the local , but also can communicate with the Cloud Server.

- It's obvious that the users can use the GPS to get their location and look at all the photos from the others at this place(e.g., a restaurant from google map). Optionally, he/she can just filter the photos in the smartphone which he/she took with current location.
- In addition, the photos are in chronological order. It's like showing the more recent photos first.
- What's more, the users can choose if he/she wants to add the compass information from the camera. This information wiil be printed on the photo.
- For instance, the users can also upload their photos, for sure the photos will be stored by the database with the location and the time.

Achievements

Our main **achievements** are that we developped an Android application with following **features**:

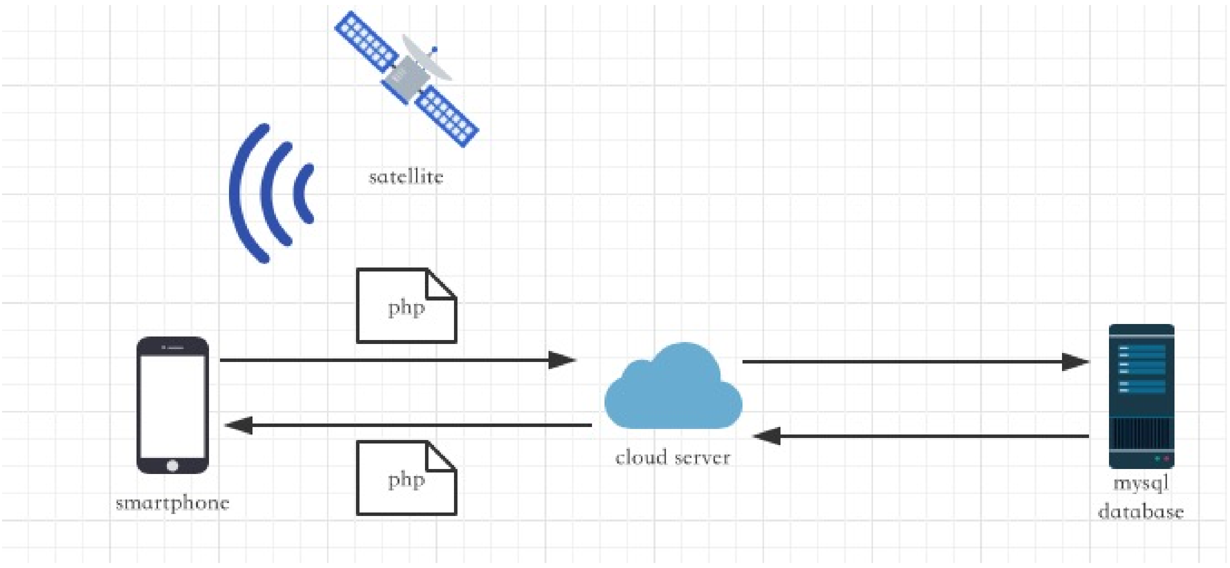
- Get your location
- Geo tagged and time capture for the local photos with system's camera or our customized compass-camera
- Filter the local photos with your current location
- One simple compass application
- Upload the photos which you took to the Cloud Server(all the related information included)
- Filter all the photos in the Cloud Server with your current location, what's more than the local filter, you can know who uploaded these photos.

To achieve these features, these basic knowledges are needed:

- **Java and xml in the Android Studio.**

Consider that there are many useful APIs applied by Google, and **Java** is the recommend way to design the Android application, we choose it as our main programming language. In addition, the **xml** is used to design the layout,menu, etc.

- **Database**
  Database is used to store all the needed information.
- **Deep understanding of Android API**
  Google gives us a lot of useful **APIs**, with them we can easily use it to develop our new application.
- **Smartphone sensor**
  The smartphone can be applied as a sensor, with this feature, we designed our simply **compass** application.
- **Cloud Server**
  One Cloud Server can help us to communicate with the smartphone .Moreover, the **MySQL** database with all the dates should be stored in that Cloud Server.
- **Communication between users and Cloud Server via php**
  Thanks to **php**, now we can connect the Android application request to the **MySQL** database, Login, Register, upload photos included.
- **Internet protocols**
  TCP/IP, **HTTP** and many other Internet protocols will be used during development of the application.



Brief description for the support

- Our application supports **at least API16**, which means that it can be used all the smartphone later than the system Android 4.1.1.
  It’s obvious that the smartphone pubblished 5 years ago(like sumsung S3) can also perfectly use our application.

- For the server part we used the **OpenShift** Cloud Server(https://openshift.redhat.com).
  OpenShift is Red Hat's Platform-as-a-Service (PaaS) that allows developers to quickly develop, host, and scale applications in a cloud environment. With OpenShift you have a choice of offerings, including online, on-premise, and open source project options.
- Let **php** be the deliverer
  Consider that transmitting the dates from smartphone to the Mysql database via JDBC is not **safe**(because people can see the database's username and password directly), php with Http transmitting protocol should be a good choice.
  In the most case, we can use php to encaspulate our dates(**POST**) to the database very safely. Otherwise, in order to improve the visiting velocity, for some filter case, like GOOGLE searching, we can use the **GET** way.
- **AsyncHttpClient** web framework
  The Async Http Client library's purpose is to allow Java applications to easily execute HTTP requests and asynchronously process the HTTP responses.
  The library also supports the WebSocket Protocol.
  The Async HTTP Client library is simple to use.
- **Picasso** framework
  Images add much-needed context and visual flair to Android applications.
  Picasso allows for hassle-free image loading in the application—often in one line of code!
- We choose the **MySQL** as our web database.
  MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open-source web application software stack.
- Instead for the local case in the Android application, we choose the lighter database: **SQLite**.
  It can be stored in the smartphone directly. If you rooted your smartphone, you can easily find it in the **/data/data/com.example.dingdamu.ding/databases/QuestionDatabase.db** folder.

Installation

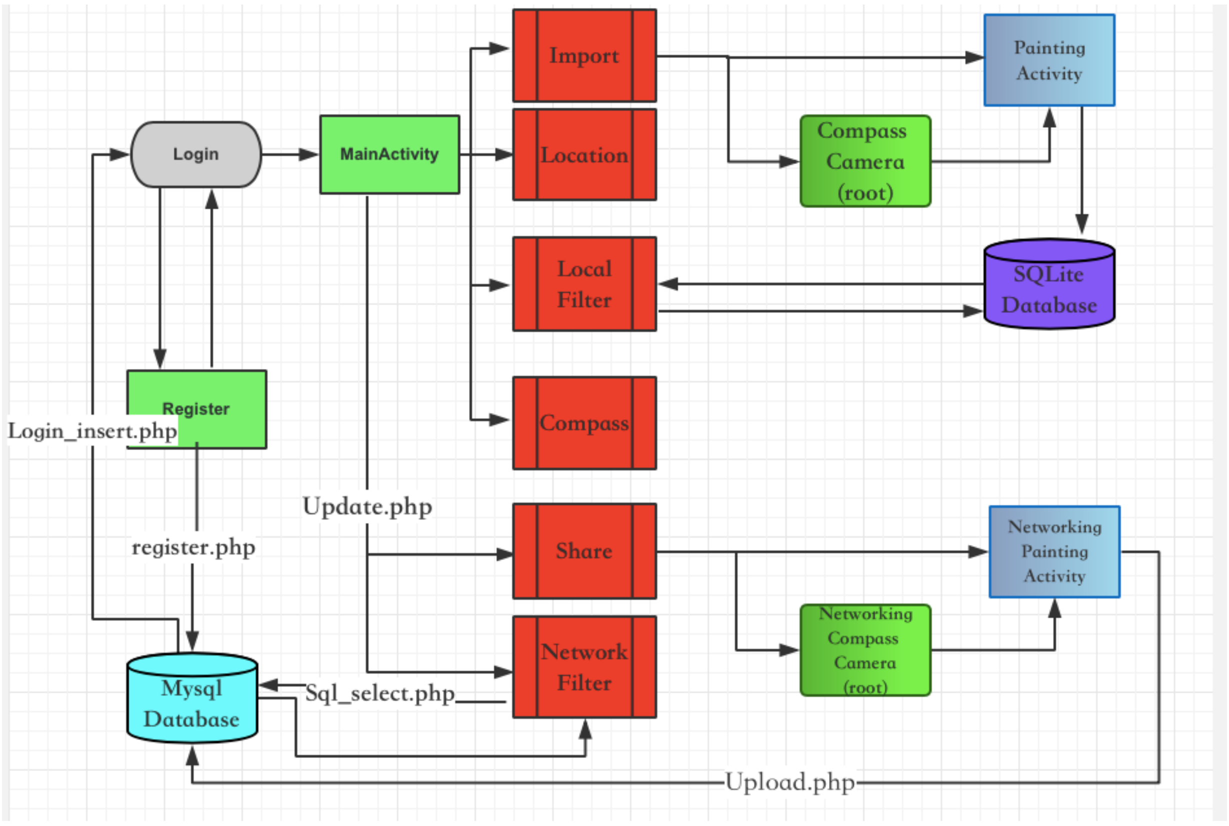
git clone https://github.com/DINGDAMU/Geo-tagged-time-capture.git

and build it with Android Studio

or simply install the .apk in the apk folder

Project Implementation

The **architecture** of the project is below:



The project can be divided into three parts:

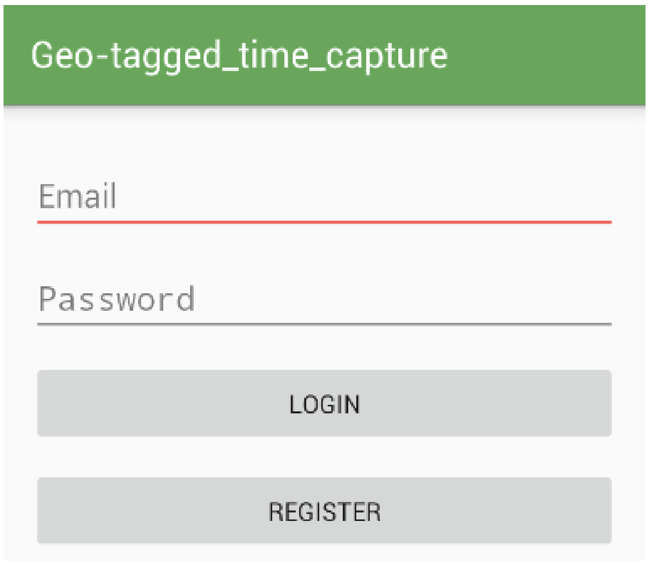
Android application, **php** server and **database**. In the server-client struct, android application is the client, php server is not only the server, but also plays a role like a “postman” to forward the dates to the database.

1.Android application part:

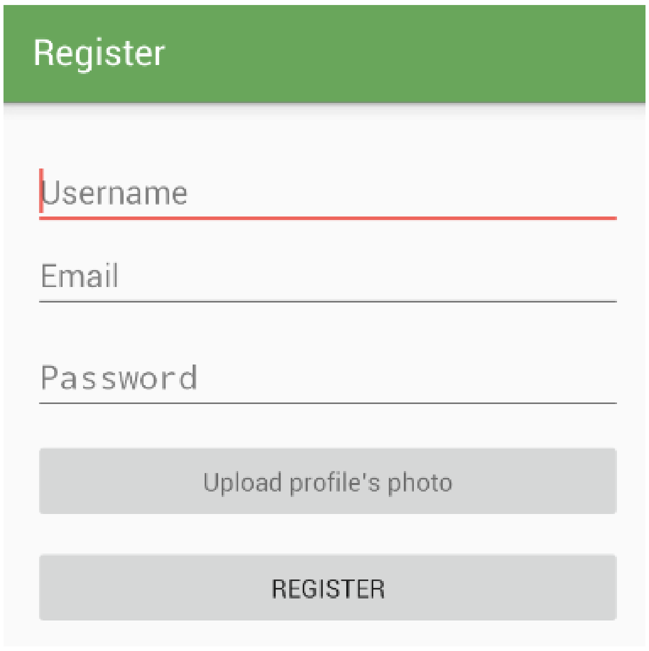
- **Brief introduction:**
  ○ As the flow chart above shown, the application begins with the **Login** screen. The new users can register their new account in the **Register** session.
  ○ In the **Register** session, everyone should digital at least 8 characters for the username, password and E-mail. For instance, a profile photo chosen from the gallery is needed.
  ○ After successfully login, the **MainActivity** comes out. It is separated into two parts: **Local** part and **Networking** part.
  ○ In the **Local** part, the users can take the photos tagged with location and time. In addition, they can filter them by current location.
  ○ Different with **Local** part, in the **Networking** part, the users can upload the photos to the web server, moreover, they can look at the photos from the others by the networking filter.

- **Login**

This is the **initial** screen of our application. The users which have registered can login here to jump into the **MainActivity**. Otherwise, the unregistered users should go to **Register** page first.
If you don't change your password or logout, next time you will login automatically.



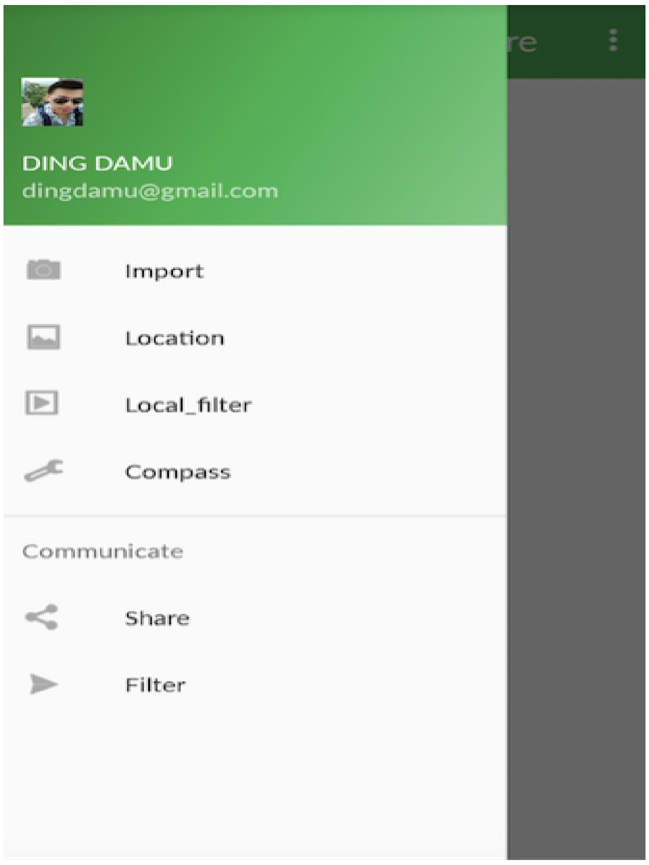
- **Register**



In this part the new users can register their new account here. In order to increase the **security**, the username, E-mail,password must be at least 8 characters.
For instance, the users should also upload one photo as their profile's photo. Once they receive "**Register success!**", the register is done.
Remember that the E-mail cannot be duplicated.

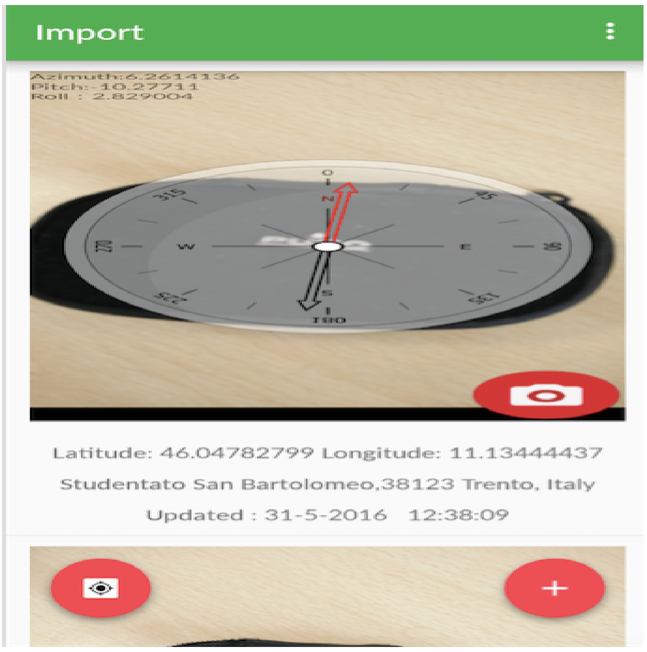
- **MainActivity**

In this page, on the top-right menu you can choose to **logout**.
After login, the username,email,profile's photo will be shown in the **nav_header** of the NavigationView.
The profile's photo is clickable to change the photo.
Due to the uploading velocity problem, we compress it into very low quality because it's really very small size(100*100);
The NavigationView helps you to jump to the following pages:
Local part:
○Import
○Location
○Local_filter
○Compass
Networking part:
○Share
○Network-filter



- **Import**

There are two **FloatingActionButtons** in this page. The left one is the Compass camera, the other one is the System's camera.
All the photos taken by these two cameras are displayed in the blank here. The size of them are 1000*1000.
In addition, the related **coordinates**, **address** ,**updated time** are tagged here.

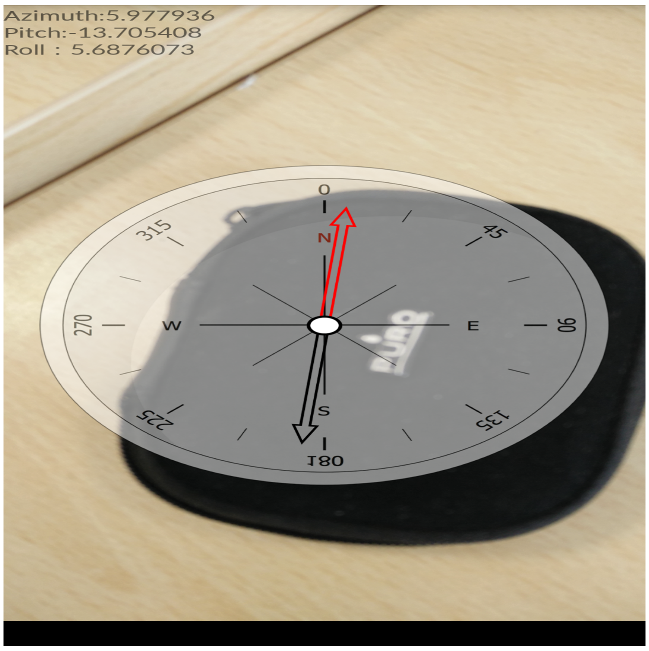


Latitude: 46.04782799 Longitude: 11.13444437
Studentato San Bartolomeo,38123 Trento, Italy
Updated : 31-5-2016 12:38:09

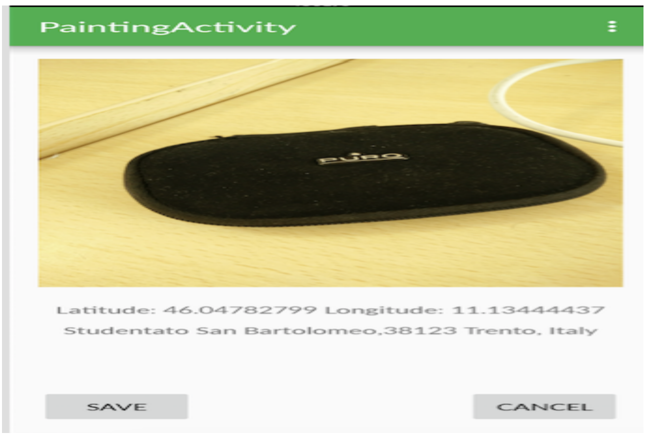
- **Compass camera**

Important: To use the Compass camera, the smartphone must be **rooted** first!

In the Compass camera, on the top-left you can find three parameters of the compass: **Azimuth**, **Pitch** and **Roll**.
In the center is the transparent compass,under which it's the camera preview.
In order to take the photo, you just need to click the screen and click the red camera icon.
It must be rooted because we can't use the button take the screenshot without the root-permission .


Azimuth:5.977936
Pitch:-13.705408
Roll : 5.6876073

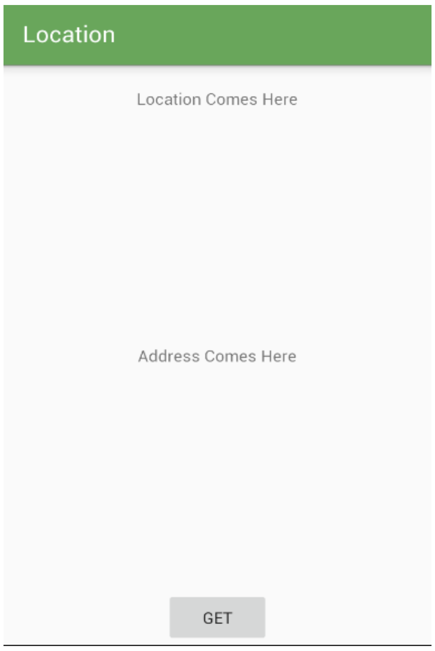
- **Painting Activity**



Latitude: 46.04782799 Longitude: 11.13444437
Studentato San Bartolomeo,38123 Trento, Italy

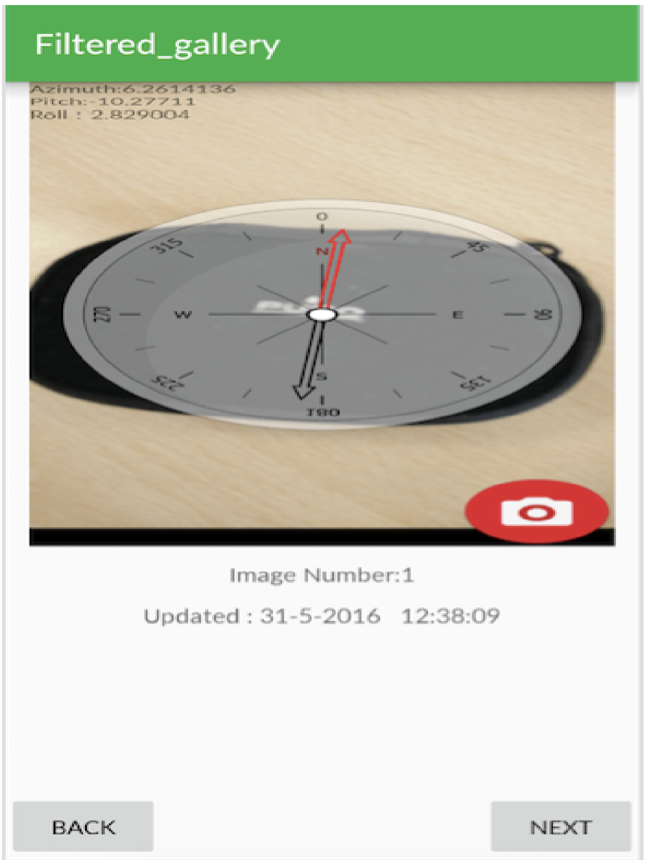
After taking the photo, both from the system's camera or Compass camera, the page will jump to the **Painting Activity**.
It displays your photo with coordinates and address.
The user can choose to save it or discard it.
Moreover, if you choose to save it , the updated time will also be tagged in the **Import** page.

- **Location**



Simple part, we used the Google GPS API to implement it.
Click ‘**GET**’ button to get your location.
In order to get your location **quickly** and **accurately**, please open your GPS, and would be better not to stay in the building.

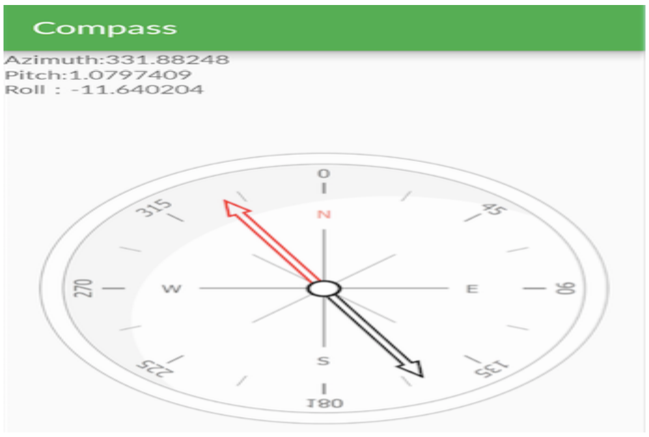
- **Local-filter**



The **Local-filter** can help the users to filter your tagged pictures with your location.
It displays all the pictures with your current location.
The “BACK” and “NEXT” button is used to help the users to look at the previous/next picture.
For instance, if arrive the last picture, the page will jump to the first picture. So does the first picture.

- **Compass**

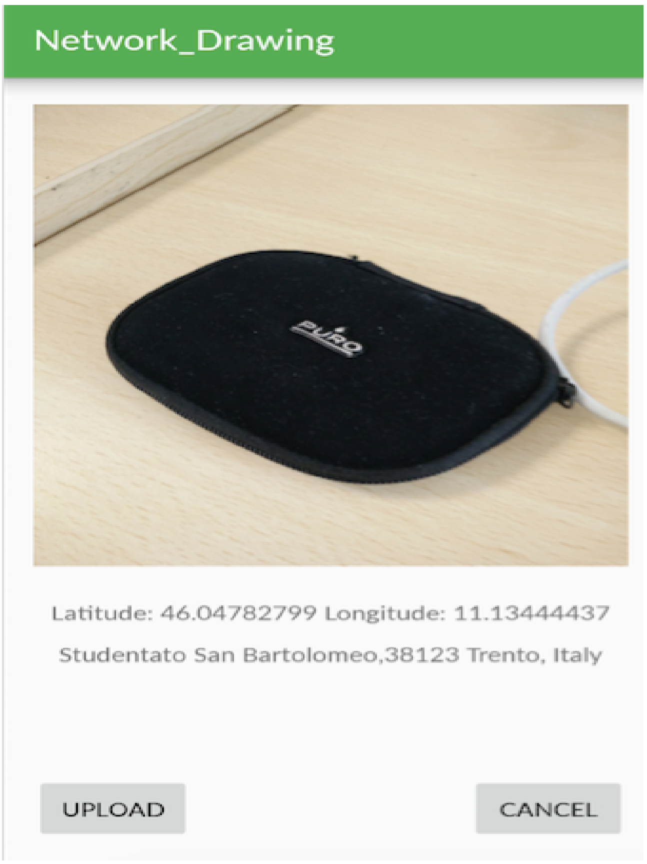
This is a simply compass application. It's based on the **sensor** API.
We add the dial and arrow as the ImageView.
Following the matrix calculated by SensorManager, we give the arrow the animation effect to tell us the temporal angle.



Azimuth:331.88248
Pitch:1.0797409
Roll : -11.640204

- **Share**
  In this page, the layout is the same as "**Import**" page, but both of them will jump to the "Network_painting_activity" page.

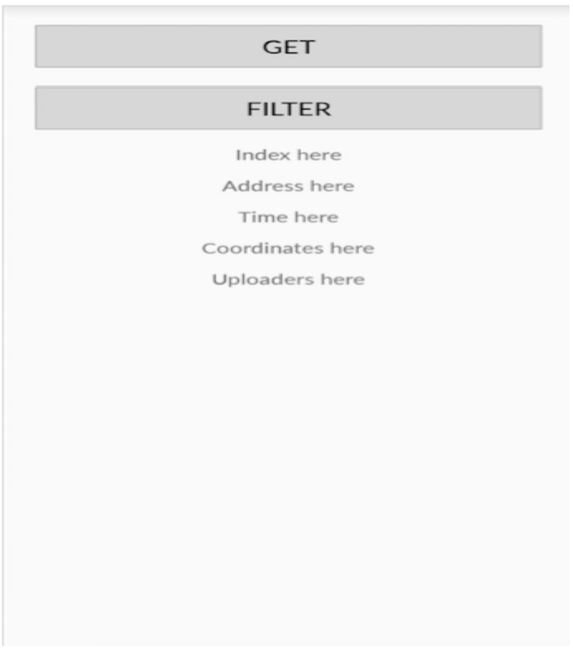
- **Network_painting_activity**



Latitude: 46.04782799 Longitude: 11.13444437
Studentato San Bartolomeo,38123 Trento, Italy

On this page is a little different with the "**Painting_Activity**" because the users can upload the photo to the **Cloud Server**. The remaining are the same, the photos with the related information.

- **Network-filter**



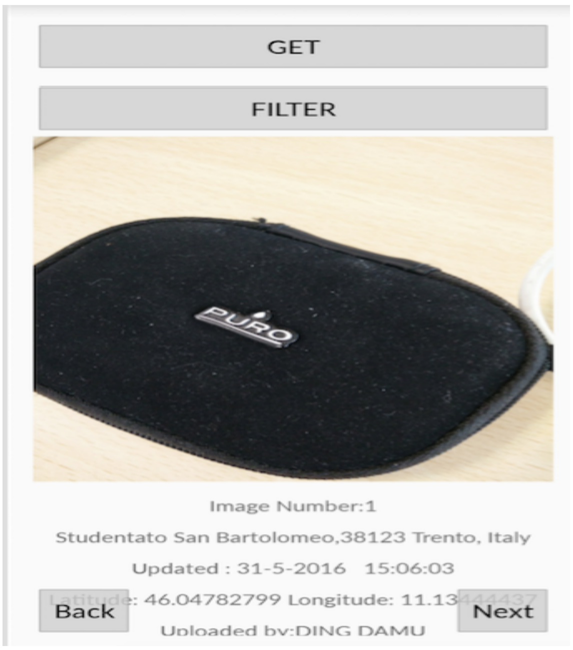


Image Number:1
Studentato San Bartolomeo,38123 Trento, Italy
Updated : 31-5-2016 15:06:03
Latitude: 46.04782799 Longitude: 11.13444437
Uploaded by:DING DAMU

Click “**GET**” button to get the location first. Then the “**FILTER**” button will help the users to filter all the photos in the **Cloud Server** with the current location.

For the Uploading photos, we **compress** them into 40% of the quality, so when the users filter the photos, all the related information shows first, it would take 30s to load the image.
We have to make a trading between the **quality** and **loading velocity**.

2.Cloud Server part:

We registered one web-server on https://openshift.redhat.com/ .
The domain name of the server is http://php-dingdamu.rhcloud.com.
It supports **php5.4**, **MySQL 5.5** and **phpMyAdmin 4.0**.
We manage it via **SSH/SFTP**.
All the php scripts we put them in the folder **/app-root/repo/**.

Name (*.php)	**Main functionality**
login_insert	Send the E-mail and password to the server to ask for the confirmation. Once confirmed, the server will reply and insert the username and profile’s photo to the smartphone.

register	Send the register request to the Server. If the E-mail is **duplicated**, the server will ask the users to retry another E-mail. Otherwise, the registration would be **accepted**.
sql_select	This helps the server to **filter** the MySQL database with the current location.
update	It's used to change the **profile's photo**. The profile's photo will be stored in the folder "**profile**" in the server.
upload	It allows to send the photo to the server. All the photos are in the "**upload**" folder.
mysql	Together with upload.php, it inserts all the related information of photo to the MySQL database.

The server and the smart phone are in communication via **AsyncHttpClient**.
The **Async Http Client** library's purpose is to allow Java applications to easily execute **HTTP requests** and asynchronously process the **HTTP responses** .
Respect to HttpURLconnetion and traditional HTTP(deprecated by Google now), it's really **simple** to use and **efficient**.

3.Database part:

We used two databases in our project: one is **SQLiter**, the other one is **MySQL**.

- **SQLite** is mainly used in the local part, and it can be stored in the smartphone directly.
  If you rooted your smartphone, you can easily find it in the **/data/data/com.example.dingdamu.ding/databases/QuestionDatabase** folder.
  In the SQLite databse, the table "**post**" stores all the information about local photos taken by our application: "**uri**","**coordinates**","**address**" and "**time**".
  The users can load the photos from the smartphone quickly, but the drawback is also very clear: **SQLite** can't share with other smartphone except you copy the db to another one.
  The android smartphone is an **ARM embedded system**, respect to the computer, its performance is very poor. For this reason, at this moment, we can't put the **MySQL** on it, but it supports the lighter database **SQLite** very well.
  So, we need to put the **MySQL** on the server, transmitting and receiving the dates via php. The capacity of the dates is very huge, **SQLite** can't support so high throughput.
  That's why we choose **MySQL** as our Netwroking part's database.

- **MySQL** is a popular choice of database for use in **web applications**, and is a central component of the widely used LAMP open-source web application software stack.
  There are two tables in our **MySQL** database.
  ○**login**
  In this table, it stores four keys: “**username**“,” **email**“, “**password**“, “**profile_url**“. All the login information are here.

  ○**information**
  There are “**username**”,”**url**“,”**coordinates**“,”**address**“,”**time**” in this table. This is the uploaded photo’s information.

The Picasso framework can help us to easily load the photos from both the local uri and the Networking url.

Conclusion

Consider that our original idea is to make a TripAdvisor-like application,
in our opinion, we can try to improve some performances of this application with following
:

- **Improve the accuracy of the GPS**
  In many cases, especially in the building, we can’t get the location from GPS.
  We can do some optimizations on it, for example, adopt one useful algorithm to find all the possible locations in one determined area, the users can choose accurate one from them.
  If the signal is really weak, just take the location of city or only the country.
- **Find the best trading between upload photo quality and loading velocity**
  The photo is very clear, but the loading velocity is very low.
  Otherwise, the loading velocity is high, but the quality of the photo is bad.
  An optimization should adopted due to this trading.
  We need to make the quality of the photo not bad, also the waiting time is acceptable.
- **Classification of the places**
  Some place is the hotel, and some place is the restaurant, somebody just wants to see the hotel, some other people just wants to see the restaurant.
  Due to this problem, we can extend these photos into different types.
- **Use the compass camera without the root**
  Not all the people are professional at smartphone. Many users don’t know how to root the smartphone, and they can’t use our Compass camera.
  Maybe we can try to find a better way to take the screenshot without the root permission.

- …etc

At last but not the least, to develop the Android application is a **long-term** process, during last two and a half months, we finished the main objectives of the project. However, it’s not finished, we are going to add more interesting features to make our application more like TripAdvisor.