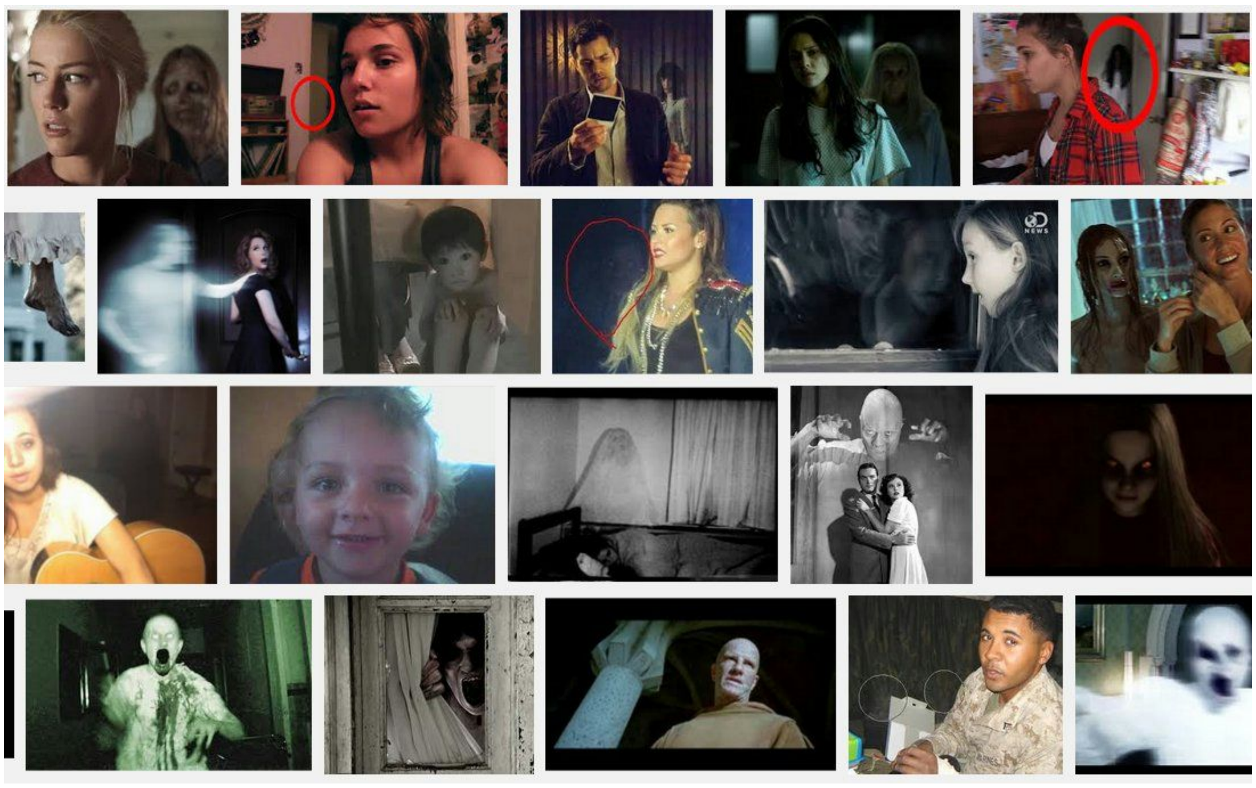
THE ANDROID CAPSTONE PROJECT 2015

GhostMySelfie App

DAILY SELFIE WITH CONCURRENT IMAGE PROCESSING ESPECIFICATION



Introduction

This app enables users to take pictures of themselves - selfies - over an extended period of time. It periodically reminds the user to take a selfie and presents the selfies in a list that makes it easy to see how the user has changed over time. This extended implementation also allows users to process their selfies to add filters **and also it presents an aleatory ghost overlaying the selfie, currently there is only six different ghost**. The image processing is done via a remote web service. In addition, all interactions between the device and the remote web service will be done concurrently in the background to ensure that the UI thread on the device is not interrupted.

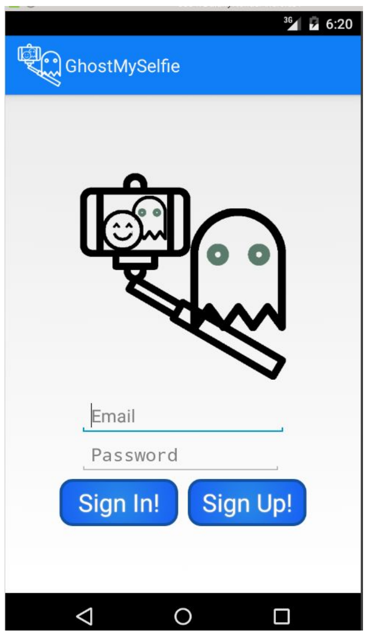
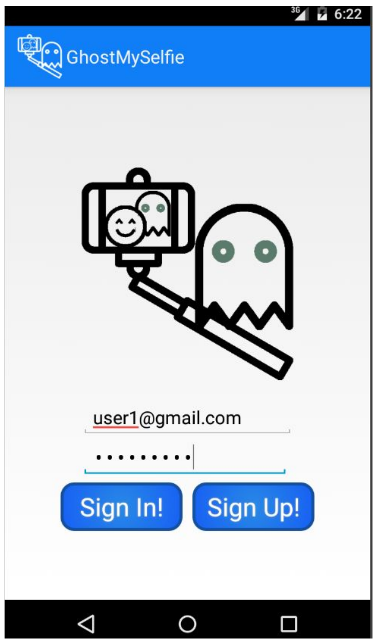
For register a user for first time, the user is registered with other hidden user that has permissions to register users. The Selfies of a user are saved in the server, so the user can see its own selfies even if is logged in other device.

How do I will implement basic project requirements.

Below I explain how my project implementation will meet each requirement listed in the rubric. Also there is a table that shows classes and methods related to each requirement.

App supports multiple users via individual user accounts.

Every user can login using basic authentication for first time and the user **(in this case is the email)** and password will travel over SSL/TLS protocol to the Server with OAUTH 2 and saved in the server for other logging-ins.

 6:20 GhostMySelfie Email Password Sign In! Sign Up!	 6:22 GhostMySelfie user1@gmail.com ••••••••• Sign In! Sign Up!
Figure 1. Sign In to Login	**Figure 2. Sign Up to register.**

User password will never saved in the device only the current logged user name or in this case **email as user**.

App contains at least one user facing function available only to authenticated users

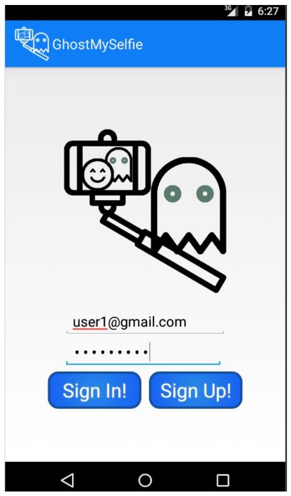
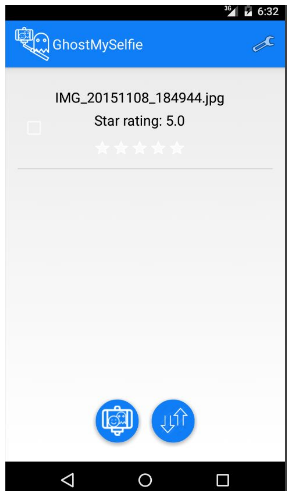
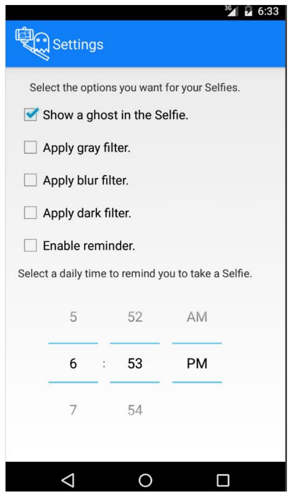
With Oauth 2 the server will filter not authenticated users, if the user is correctly authenticated, the server will process the images that the user send.

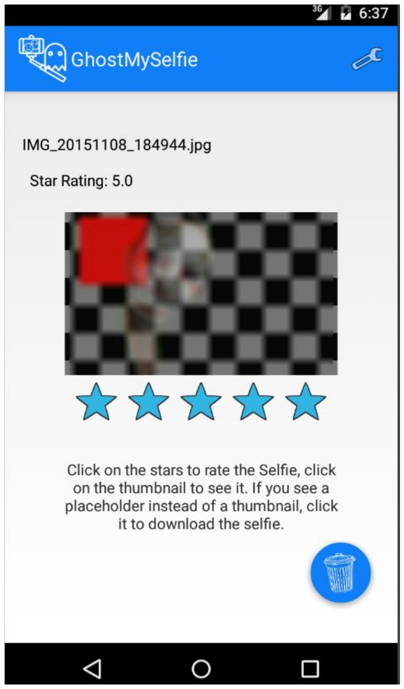
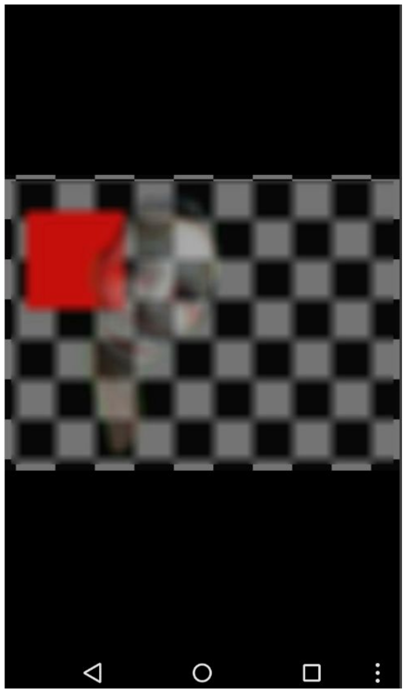
There is more than one function available to authenticated users, one of them is to rate a selfie.

App comprises at least 1 instance of each of at least 2 of the following 4 fundamental Android components:

- ### Activity

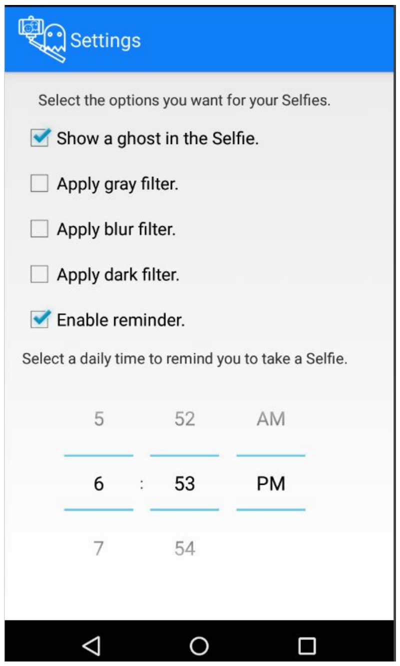
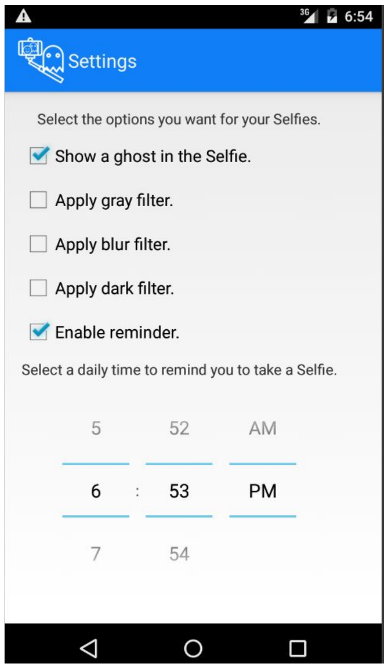
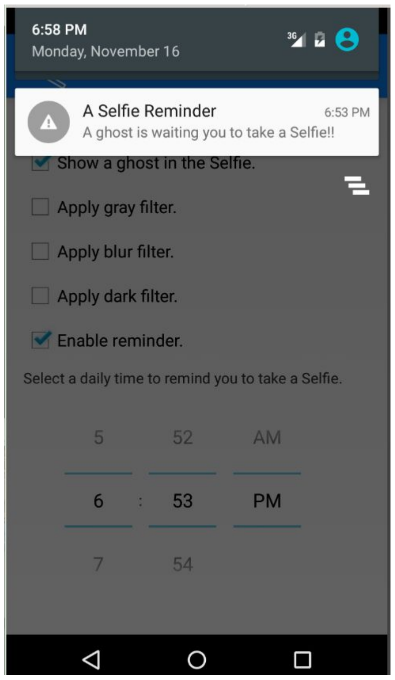
There are four activities GhostMySelfieDetailActivity, GhostMySelfieListActivity, LoginScreenActivity and SettingsActivity with an extra activity to show the selfie in fullscreen.

 user1@gmail.com ••••••••••	 IMG_20151108_184944.jpg Star rating: 5.0	 Select the options you want for your Selfies. ☑ Show a ghost in the Selfie. ☐ Apply gray filter. ☐ Apply blur filter. ☐ Apply dark filter. ☐ Enable reminder. Select a daily time to remind you to take a Selfie. 6 : 53 PM
Figure 3. Activity 1 for Sign In or Sign Up.	Figure 4. Activity 2 for take a Selfie or Modify Selfie(s).	Figure 5. Activity 3 for set filters or notifications.

 IMG_20151108_184944.jpg Star Rating: 5.0 Click on the stars to rate the Selfie, click on the thumbnail to see it. If you see a placeholder instead of a thumbnail, click it to download the selfie.	
Figure 6. Activity 4 for rate a Selfie and see the details of the selected Selfie.	Figure 7. Activity 5 for show the Selfie in fullscreen.

- **BroadcastReceiver**

The app use an Alarm class that extends the BroadcastReceiver to get notifications to take a selfie in certain time intervals with some scary sound. The app has options to stop notifications and the scary sound for notifications. Also after operations that process images to the server, it will be broadcasted an intent to send the result of the request to the calling activity.

 Settings Select the options you want for your Selfies. ☑ Show a ghost in the Selfie. ☐ Apply gray filter. ☐ Apply blur filter. ☐ Apply dark filter. ☑ Enable reminder. Select a daily time to remind you to take a Selfie. 6 : 53 PM	 6:54 Settings Select the options you want for your Selfies. ☑ Show a ghost in the Selfie. ☐ Apply gray filter. ☐ Apply blur filter. ☐ Apply dark filter. ☑ Enable reminder. Select a daily time to remind you to take a Selfie. 6 : 53 PM	 6:58 PM Monday, November 16 A Selfie Reminder 6:53 PM A ghost is waiting you to take a Selfie!! ☑ Show a ghost in the Selfie. ☐ Apply gray filter. ☐ Apply blur filter. ☐ Apply dark filter. ☑ Enable reminder. Select a daily time to remind you to take a Selfie. 6 : 53 PM
Figure 7. Activitating an alarm notification at 6:53 PM.	Figure 8. Alarm is triggered with a scary sound.	Figure 9. Notification.

- **Service**

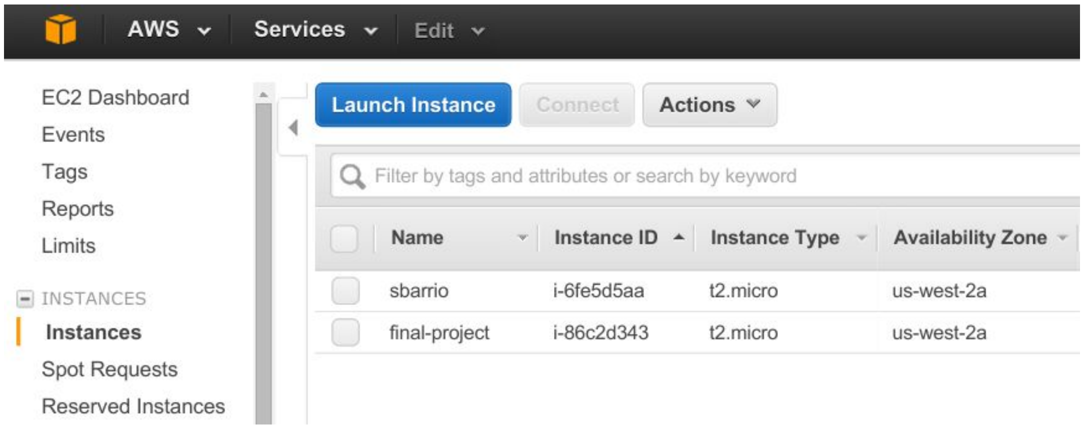
The app has services that runs in background and executes various requests to the server. After the operation, it broadcasts the intent to send the result of the request to the calling activity.

- **ContentProvider**

In the Model layer the app use ContentProvider and Resolver to save data to sqlite database.

App interacts with at least one remotely-hosted Java Spring-based service

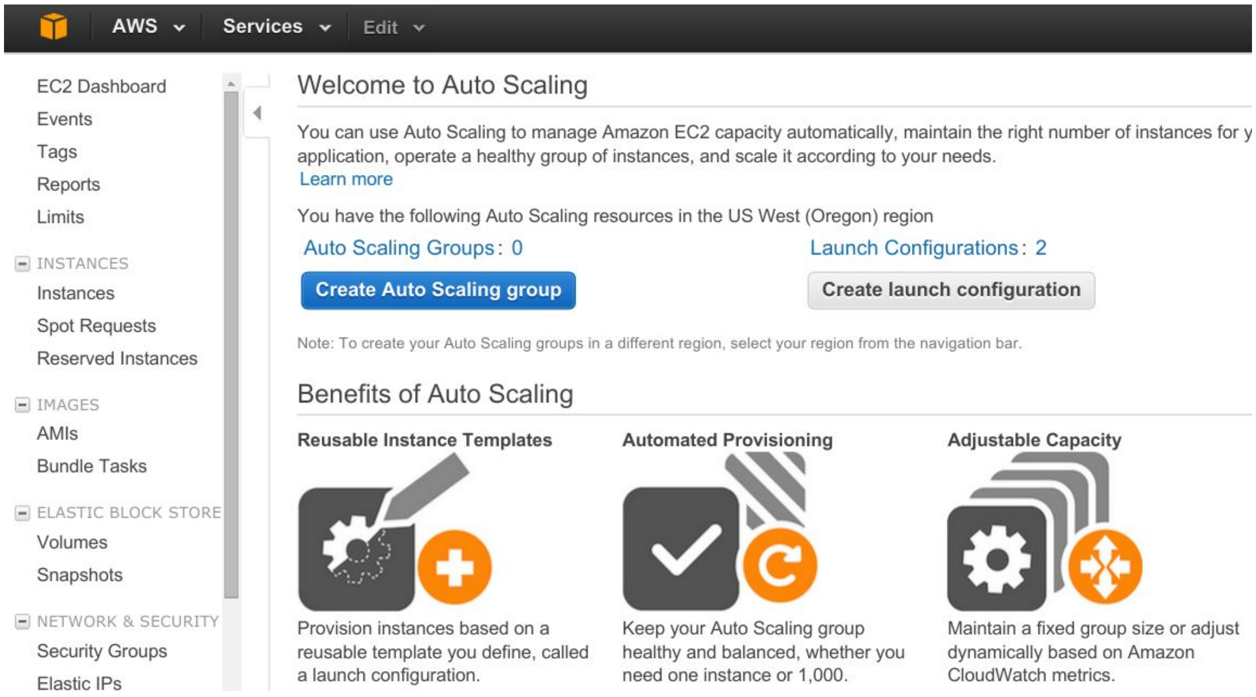
The app use **Amazon** Elastic Compute Cloud (**Amazon EC2**) in Ubuntu Server with MySQL Database, that is a web service that provides resizable compute capacity in the cloud.



Name	Instance ID	Instance Type	Availability Zone
sbarrio	i-6fe5d5aa	t2.micro	us-west-2a
final-project	i-86c2d343	t2.micro	us-west-2a

Figure 1.5

The service is hosted in an instance that can be putted in a Auto Scaling Group to manage Amazon EC2 capacity automatically, maintaining the right number of instances for the application, operate a healthy group of instances, and scale it when the app needs it.



Welcome to Auto Scaling

You can use Auto Scaling to manage Amazon EC2 capacity automatically, maintain the right number of instances for y application, operate a healthy group of instances, and scale it according to your needs.
Learn more

You have the following Auto Scaling resources in the US West (Oregon) region

Auto Scaling Groups: 0

Launch Configurations: 2

Note: To create your Auto Scaling groups in a different region, select your region from the navigation bar.

Benefits of Auto Scaling

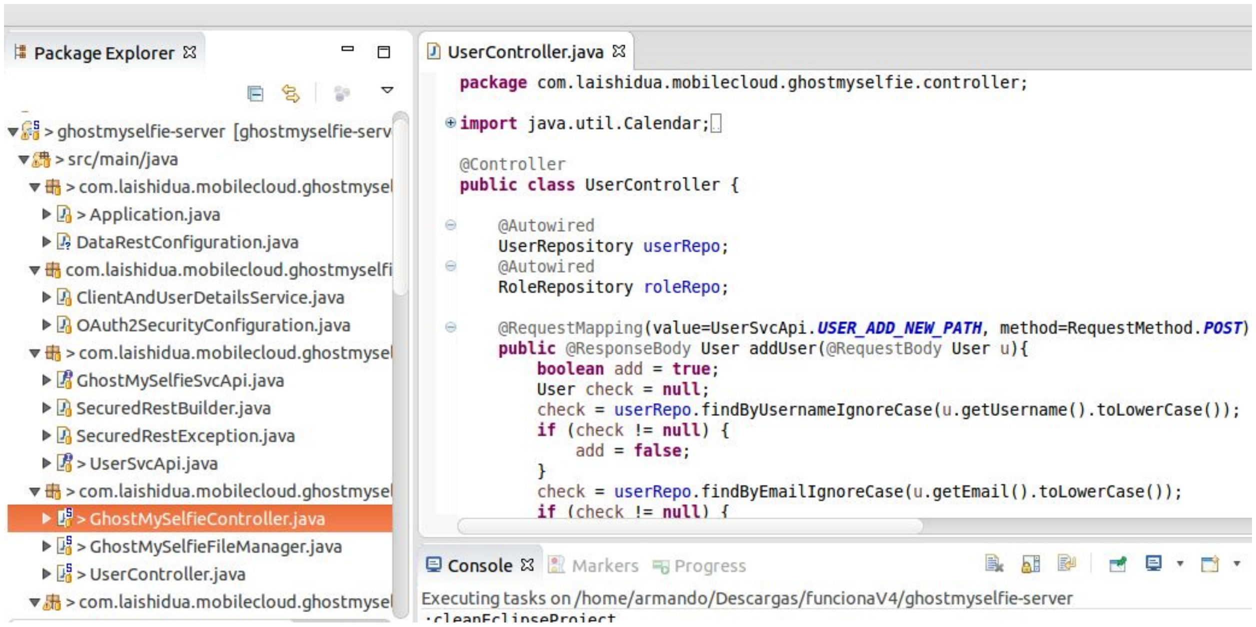
Reusable Instance Templates
Provision instances based on a reusable template you define, called a launch configuration.

Automated Provisioning
Keep your Auto Scaling group healthy and balanced, whether you need one instance or 1,000.

Adjustable Capacity
Maintain a fixed group size or adjust dynamically based on Amazon CloudWatch metrics.

Figure 1.6

I prefered it over the Amazon beans talk because I like to configure all the servers by myself.



```
Package Explorer
> ghostmyselfie-server [ghostmyselfie-serv
  > src/main/java
    > com.laishidua.mobilecloud.ghostmysel
      > Application.java
      DataRestConfiguration.java
    com.laishidua.mobilecloud.ghostmyselfi
      ClientAndUserDetailsService.java
      OAuth2SecurityConfiguration.java
    > com.laishidua.mobilecloud.ghostmysel
      GhostMySelfieSvcApi.java
      SecuredRestBuilder.java
      SecuredRestException.java
      > UserSvcApi.java
    > com.laishidua.mobilecloud.ghostmysel
      > GhostMySelfieController.java
      > GhostMySelfieFileManager.java
      > UserController.java
    > com.laishidua.mobilecloud.ghostmysel
```

```java
package com.laishidua.mobilecloud.ghostmyselfie.controller;

import java.util.Calendar;

@Controller
public class UserController {

    @Autowired
    UserRepository userRepo;
    @Autowired
    RoleRepository roleRepo;

    @RequestMapping(value=UserSvcApi.USER_ADD_NEW_PATH, method=RequestMethod.POST)
    public @ResponseBody User addUser(@RequestBody User u){
        boolean add = true;
        User check = null;
        check = userRepo.findByUsernameIgnoreCase(u.getUsername().toLowerCase());
        if (check != null) {
            add = false;
        }
        check = userRepo.findByEmailIgnoreCase(u.getEmail().toLowerCase());
        if (check != null) {
```

```
Executing tasks on /home/armando/Descargas/funcionaV4/ghostmyselfie-server
:cleanEclipseProject
```

App interacts over the network via HTTP/HTTPS.

To build a Self Signed certificate I execute in bash console:

%JAVA_HOME%/bin/keytool -genkey -alias tomcat -keyalg RSA -keystore keystore

Writing the DNS name when its needed.

I put the keystore in:

%Tomcat_Home%

Then I configure the file:

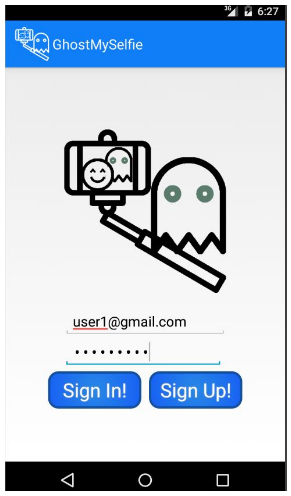
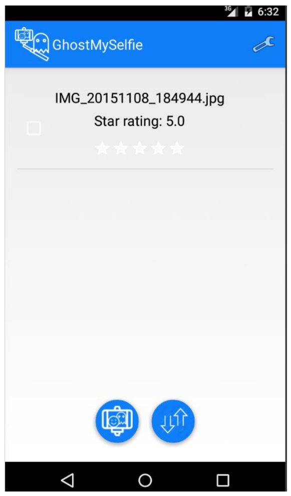
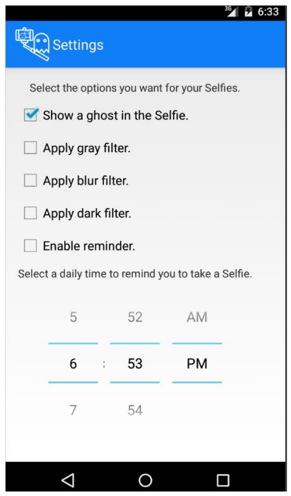
%Tomcat_Home%/conf/server.xml adding:

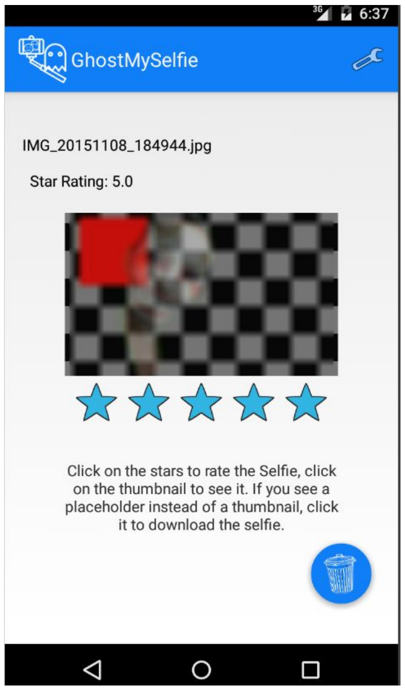
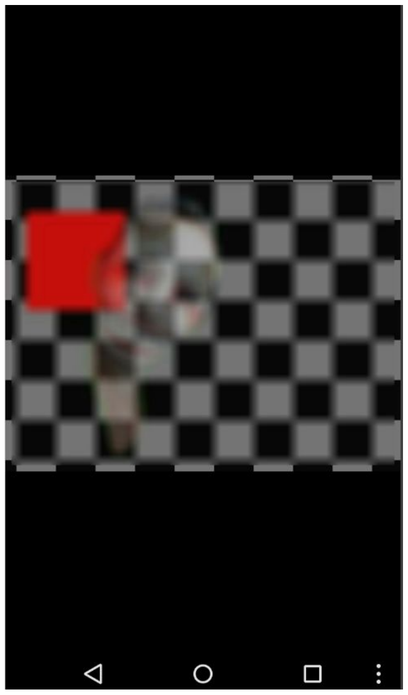
```
<Connector
        protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" maxThreads="200"
```

```
scheme="https" secure="true" SSLEnabled="true"

keystoreFile="keystore" keystorePass="<The keystore password>"

clientAuth="false" sslProtocol="TLS"/>
```

App allows users to navigate between 3 or more user interface screens at runtime

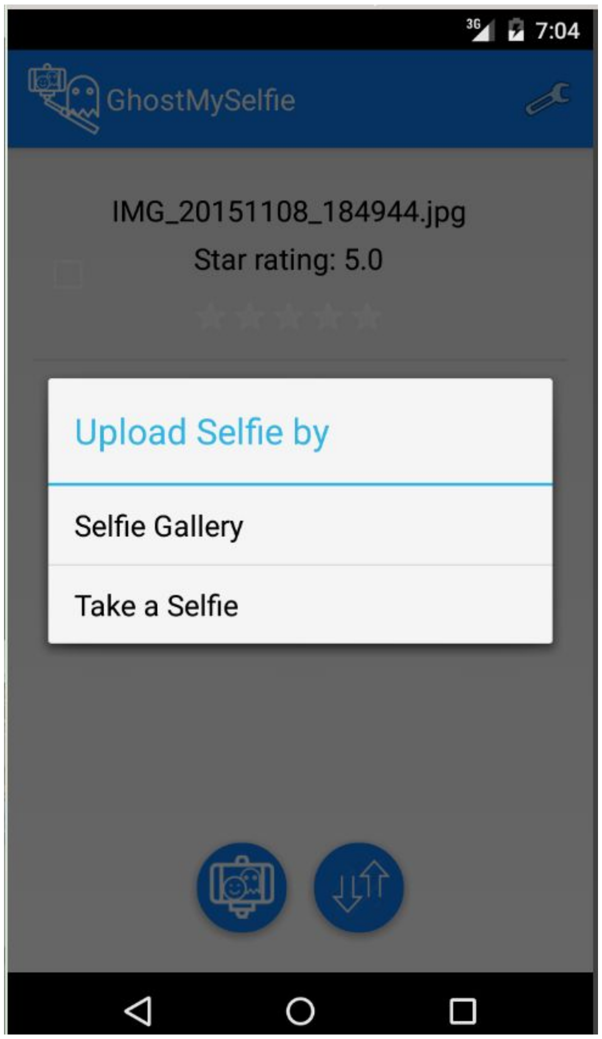
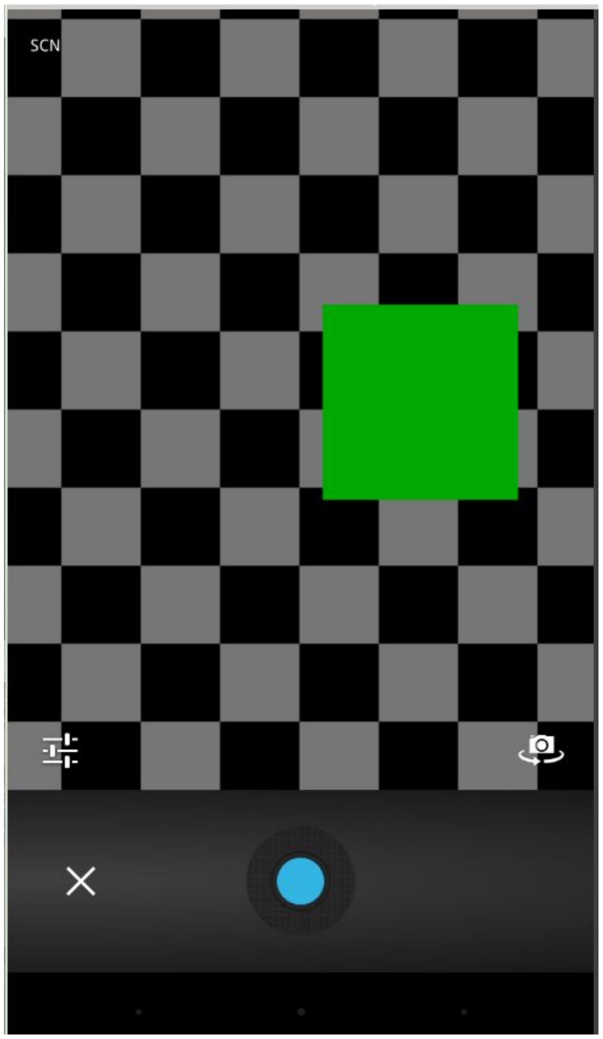
The user as shows in Activity part above, can navigate in firstime runtime over 4 screens.

 6:27 GhostMySelfie user1@gmail.com Sign In! Sign Up!	 IMG_20151108_184944.jpg Star rating: 5.0	 Select the options you want for your Selfies. [✓] Show a ghost in the Selfie. [ ] Apply gray filter. [ ] Apply blur filter. [ ] Apply dark filter. [ ] Enable reminder. Select a daily time to remind you to take a Selfie. 6 : 53 PM
Figure 10. Screen 1 for Sign In or Sign Up.	Figure 11. Screen 2 for take a Selfie or Modify Selfie(s).	Figure 12. Screen 3 for set filters or notifications.

 GhostMySelfie IMG_20151108_184944.jpg Star Rating: 5.0 Click on the stars to rate the Selfie, click on the thumbnail to see it. If you see a placeholder instead of a thumbnail, click it to download the selfie.	
Figure 13. Screen 4 for rate a Selfie and see the details of the selected Selfie.	Figure 14. Screen 5 for show the Selfie in fullscreen.

App uses at least one advanced capability or API from the following list (covered in the MoCCA Specialization): multimedia capture, multimedia playback, touch gestures, sensors, animation.**

The app use ACTION_IMAGE_CAPTURE from MediaStore Class as Multimedia Capture to take a Selfie. Standard Intent action is sent to have the camera application capture an image and return it.

 7:04 GhostMySelfie IMG_20151108_184944.jpg Star rating: 5.0 Upload Selfie by Selfie Gallery Take a Selfie	
Figure 15. Taking a Selfie.	Figure 16. Open camera to take a Selfie.

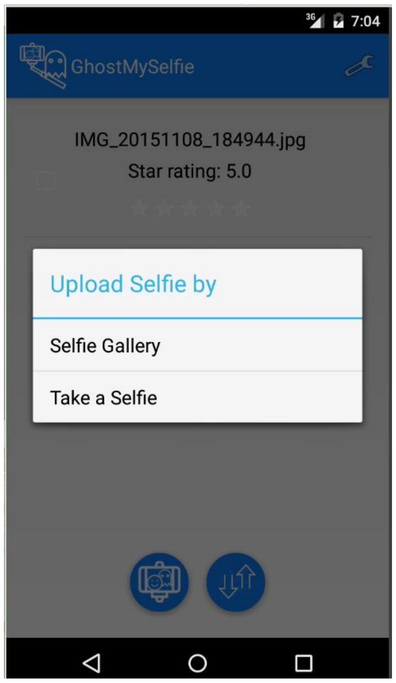
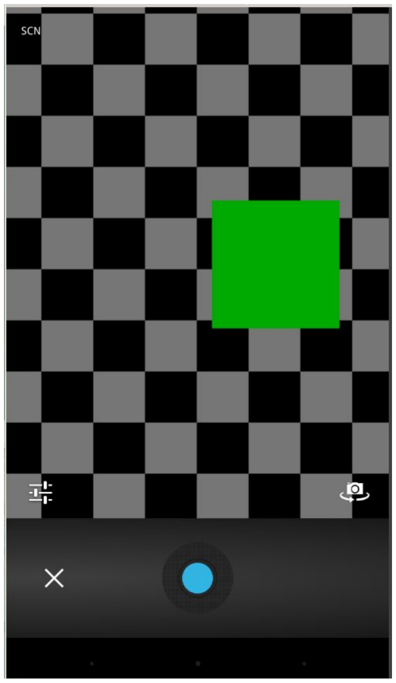
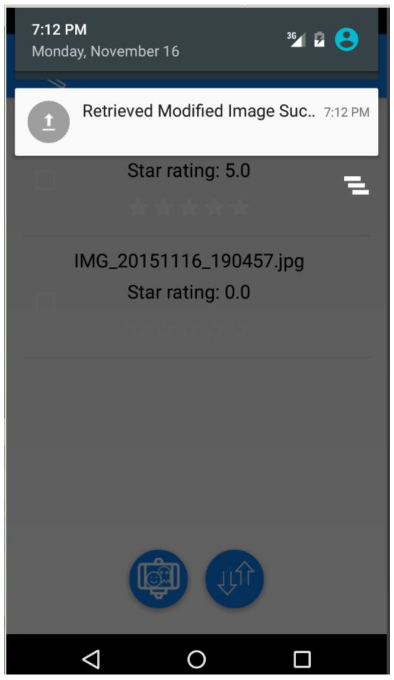
App supports at least one operation that is performed off the UI Thread in one or more background Threads of Thread pool.

The app use a mediator class that mediate communication between the Service and the local storage on the Android device. The methods in the mediator class block, so they are called from the background thread.

Functional Description and App Requirement.

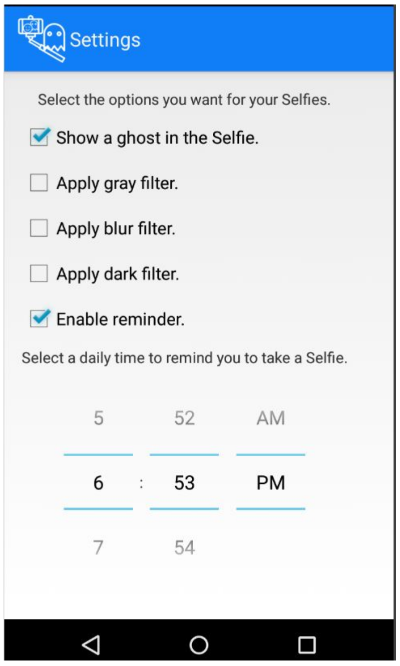
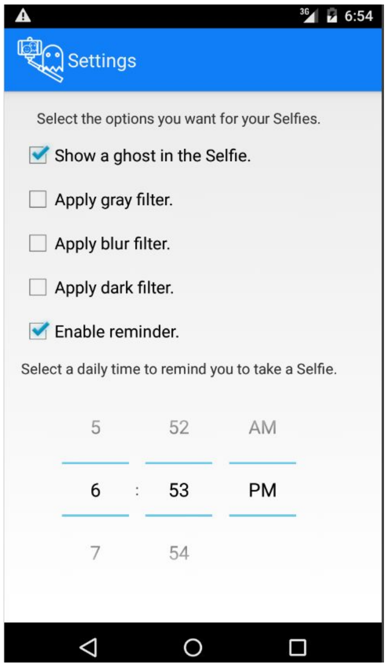
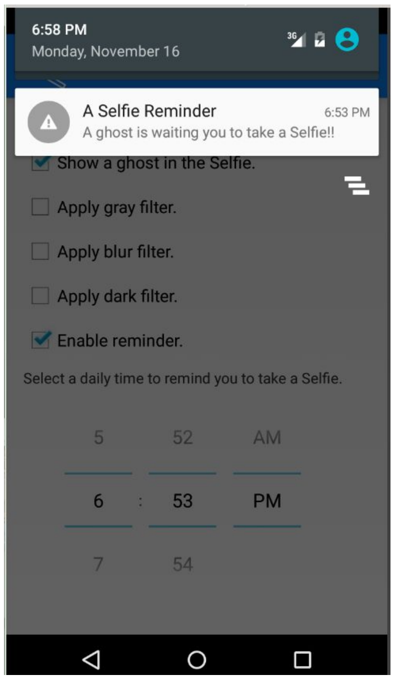
App allows user to take and save a Selfie.

The app use ACTION_IMAGE_CAPTURE from MediaStore Class as Multimedia Capture to take a Selfie. Standard Intent action is sent to have the camera application capture an image and return it.

 7:04 GhostMySelfie IMG_20151108_184944.jpg Star rating: 5.0 Upload Selfie by Selfie Gallery Take a Selfie		 7:12 PM Monday, November 16 Retrieved Modified Image Suc.. 7:12 PM Star rating: 5.0 IMG_20151116_190457.jpg Star rating: 0.0
Figure 17. Taking a Selfie.	Figure 18. Open camera to take a Selfie.	Figure 19. Retrieve the selfie from server.

App reminds user to take a Selfie.

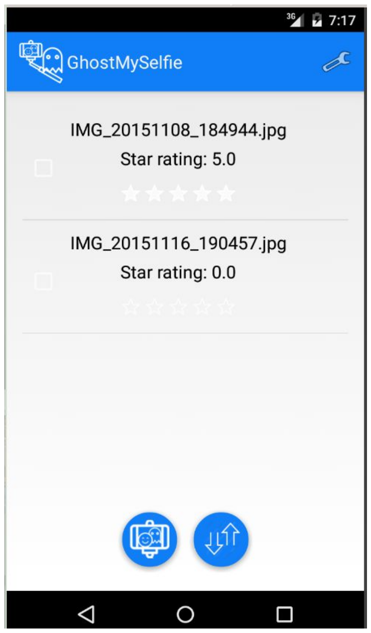
The app use an class that extends the BroadcastReceiver to get notifications to take a selfie in certain time intervals with some scary sound.

 Settings Select the options you want for your Selfies. ✓ Show a ghost in the Selfie. ☐ Apply gray filter. ☐ Apply blur filter. ☐ Apply dark filter. ✓ Enable reminder. Select a daily time to remind you to take a Selfie. 5 52 AM 6 : 53 PM 7 54	 6:54 Settings Select the options you want for your Selfies. ✓ Show a ghost in the Selfie. ☐ Apply gray filter. ☐ Apply blur filter. ☐ Apply dark filter. ✓ Enable reminder. Select a daily time to remind you to take a Selfie. 5 52 AM 6 : 53 PM 7 54	 6:58 PM Monday, November 16 A Selfie Reminder 6:53 PM A ghost is waiting you to take a Selfie!! ✓ Show a ghost in the Selfie. ☐ Apply gray filter. ☐ Apply blur filter. ☐ Apply dark filter. ✓ Enable reminder. Select a daily time to remind you to take a Selfie. 5 52 AM 6 : 53 PM 7 54
Figure 20. Activitating an alarm notification at 6:53 PM.	Figure 21. Alarm is triggered with a scary sound.	Figure 22.App reminds user to take a Selfie everyday at 6:53 PM.

App allows user to view saved Selfies in a List View.

App will use android.widget.ListView that contains a list of Videos available from the Selfie Service. This listView object get reference to the ListView created in the XML for displaying the results.

App contains an Adapter class that show the view for each Selfie's meta-data in a ListView. The Adapter Class has a method used by the ListView to "get" the "view" for each row of data in the ListView.



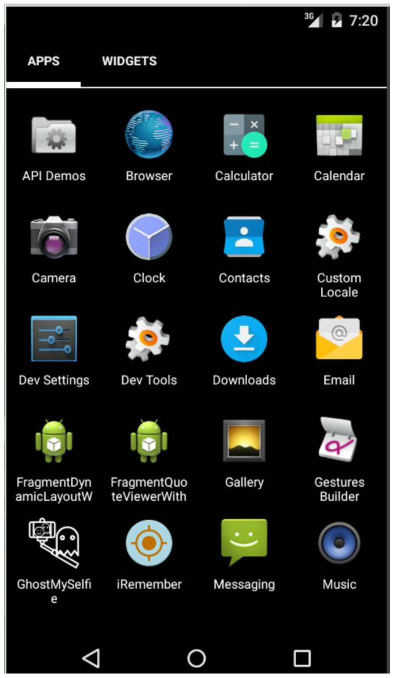
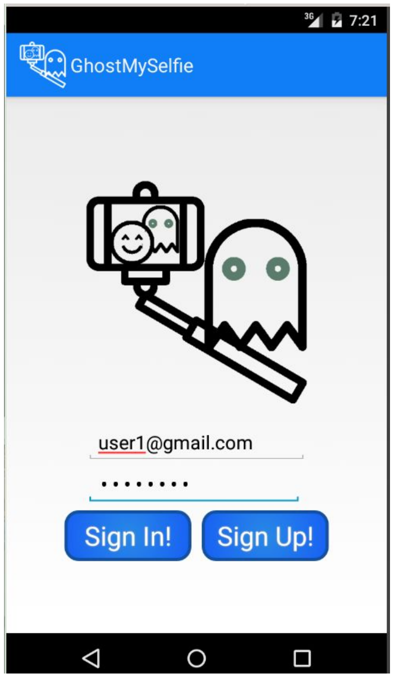
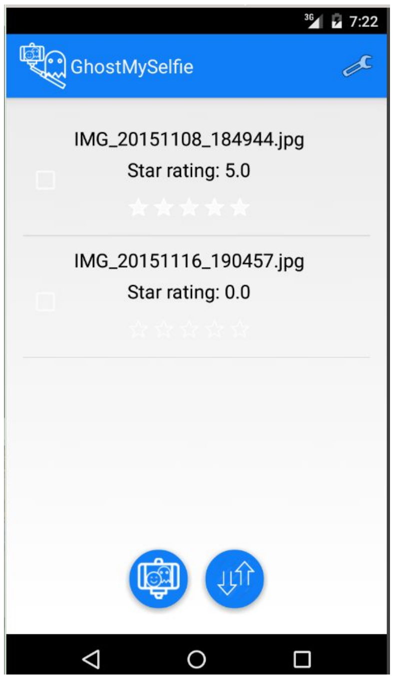
IMG_20151108_184944.jpg
Star rating: 5.0

IMG_20151116_190457.jpg
Star rating: 0.0

Figure 23. View saved selfies in List view.

If the User closes and then reopens the app, the user has access to all saved Selfies.

The app save all path from the processed images that cames from the server in a SQLite database. For this the user has to get internet connection. The Selfies of each user logged in are saved in the server, and once the user is logged in, only his or her selfies details will show as a list. The same happen if the user login using another device.

 7:20 APPS WIDGETS API Demos Browser Calculator Calendar Camera Clock Contacts Custom Locale Dev Settings Dev Tools Downloads Email FragmentDynamicLayoutW FragmentQuoteViewerWith Gallery Gestures Builder GhostMySelfie iRemember Messaging Music	 7:21 GhostMySelfie user1@gmail.com ••••••••• Sign In! Sign Up!	 IMG_20151108_184944.jpg Star rating: 5.0 IMG_20151116_190457.jpg Star rating: 0.0
Figure 24. Closing the App.	Figure 25. Opening the App.	Figure 26. Selfies of the user are saved and showed in the list.

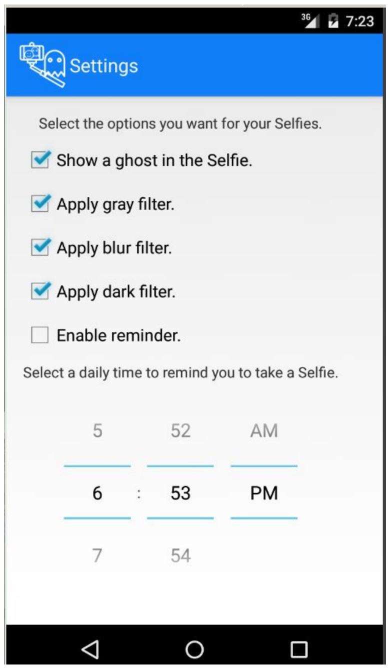
The App allows the User to select one or more Selfies, select one or more graphic effects, and apply the selected effects to the selected Selfies, and then view the processed Selfies in a ListView.

The Settings Menu has the next options :

- Show a ghost in the Selfie: If you want a face or body of a ghost image overlaying an image Selfie.
- Apply gray filter: Grays out an image.
- Apply blur filter: Simple blur effect to an image.
- Apply dark filter: Night effect to an image.

If user selects one or more than one Filter and click the button with the arrows in the list activity, the app will send the filters as an array to the server with the Image from the rows that are selected from the list of selfies, the Server will process each Image first overlaying a ghost to the Image using Chroma Key composition and then applying the effects one by one to the image.

App use Retrofit Rest client to communicate between the device and the remote web service using JSON for automatic parsing and Multipart data.



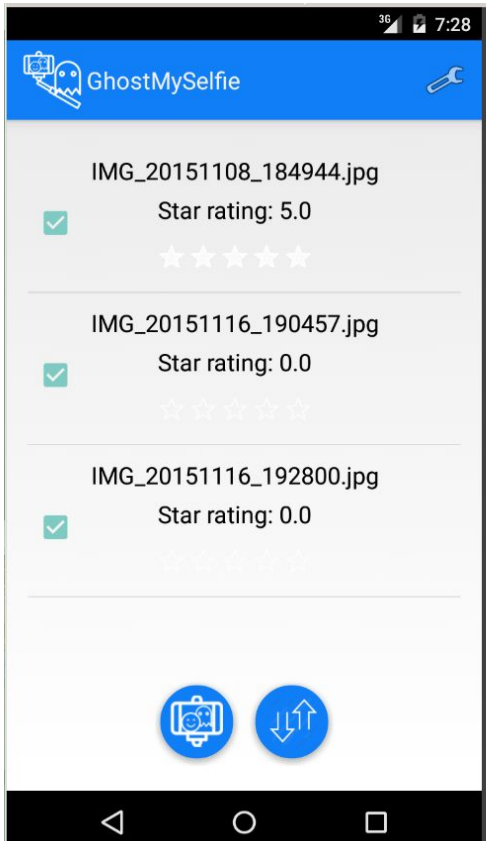
Select the options you want for your Selfies.

- [✓] Show a ghost in the Selfie.
- [✓] Apply gray filter.
- [✓] Apply blur filter.
- [✓] Apply dark filter.
- [ ] Enable reminder.

Select a daily time to remind you to take a Selfie.

Figure 27. Selecting one or more filters.

The user can select one or more Selfies to apply the effects.



IMG_20151108_184944.jpg
Star rating: 5.0

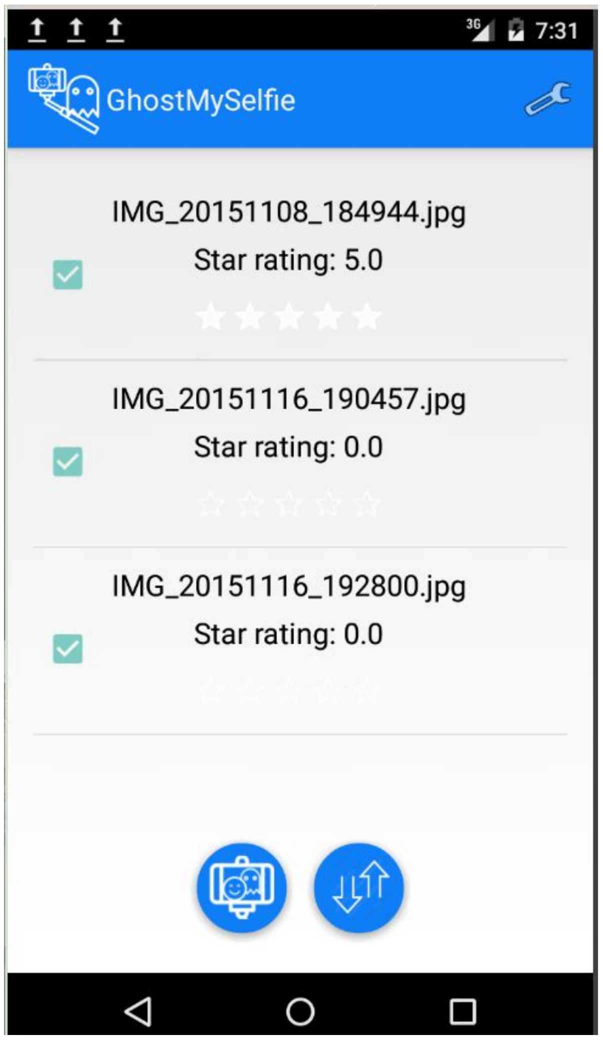
IMG_20151116_190457.jpg
Star rating: 0.0

IMG_20151116_192800.jpg
Star rating: 0.0

Figure 28. Selecting one or more selfies.

Some part of the operations corresponding to Requirement 5 (Applying Graphics Effects to Process Images) must be executed concurrently in a remote web service. Once processed, the images are returned to the device, where they are displayed in a ListView.

After user select filters, selfies and click the arrows button in the list Activity, the intent service runs in background and executes various request to the server to process the images at the same time, for each request to the server a **Connector** will create a number of request processing **threads using Tomcat options**. After each concurrent operation, it broadcasts the intent to send the image results of the request to the calling activity replacing each image in the device if already exist.



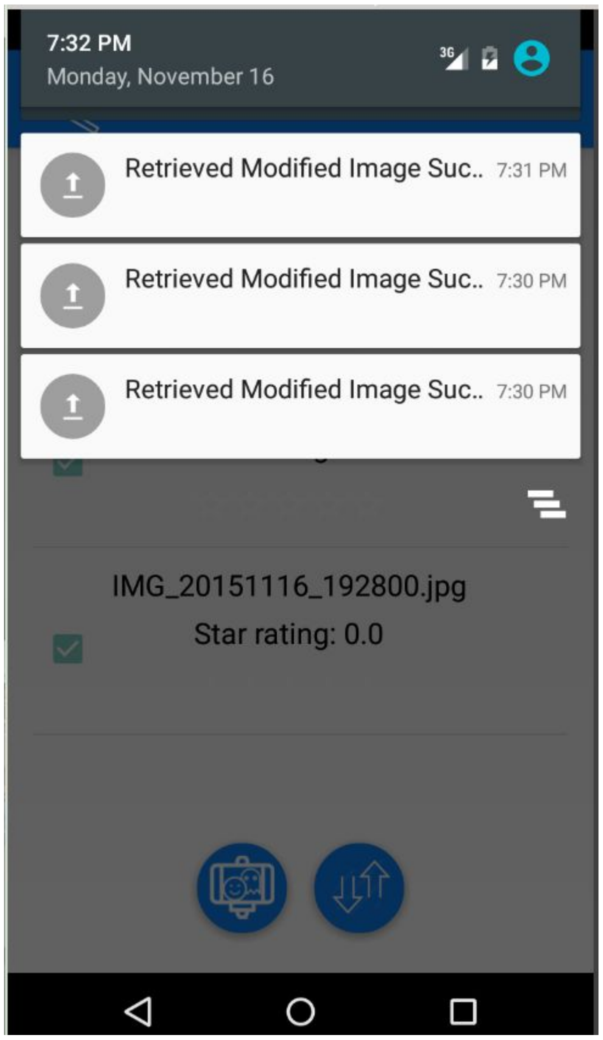
7:31

GhostMySelfie

IMG_20151108_184944.jpg
Star rating: 5.0

IMG_20151116_190457.jpg
Star rating: 0.0

IMG_20151116_192800.jpg
Star rating: 0.0



7:32 PM
Monday, November 16

Retrieved Modified Image Suc.. 7:31 PM

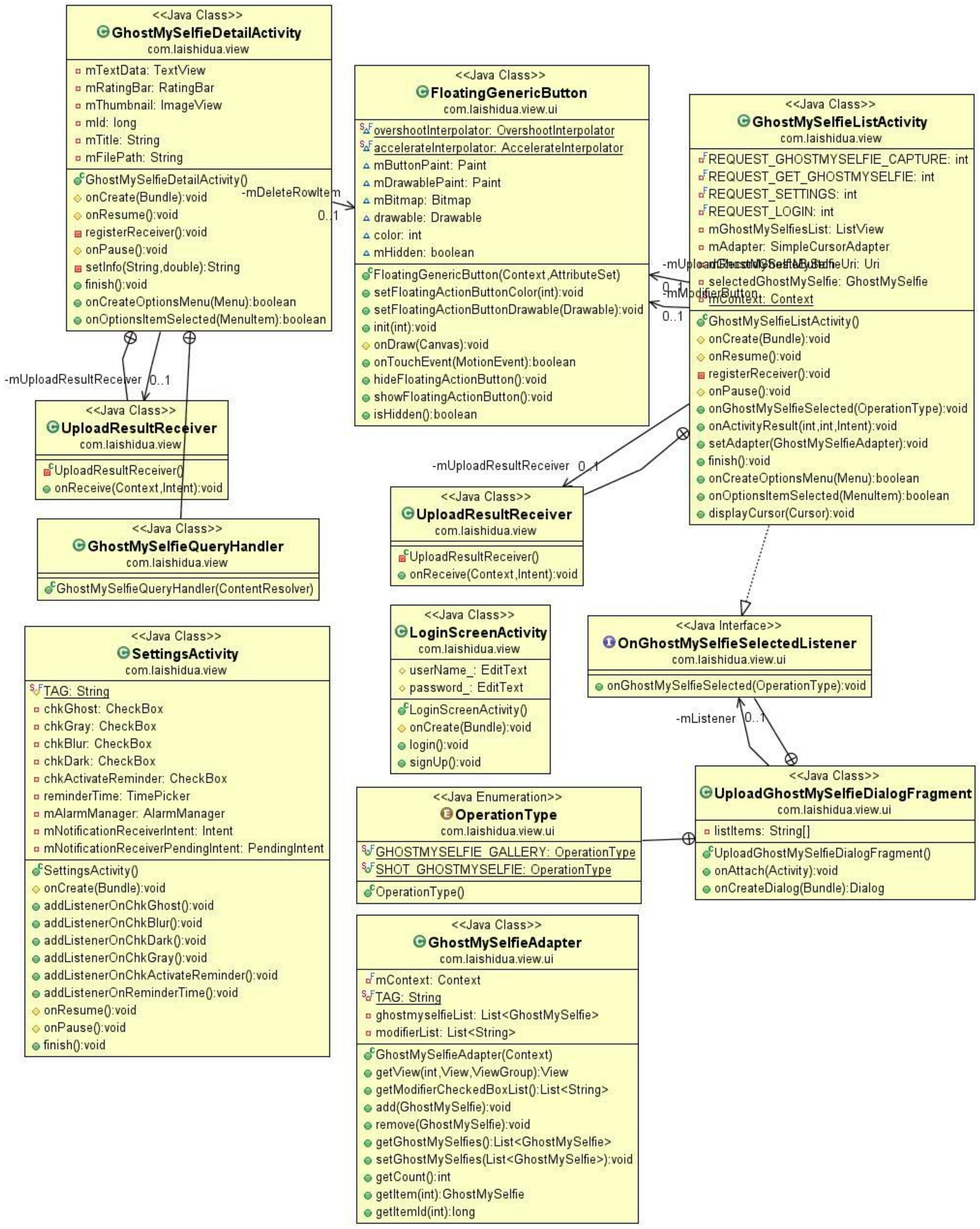
Retrieved Modified Image Suc.. 7:30 PM

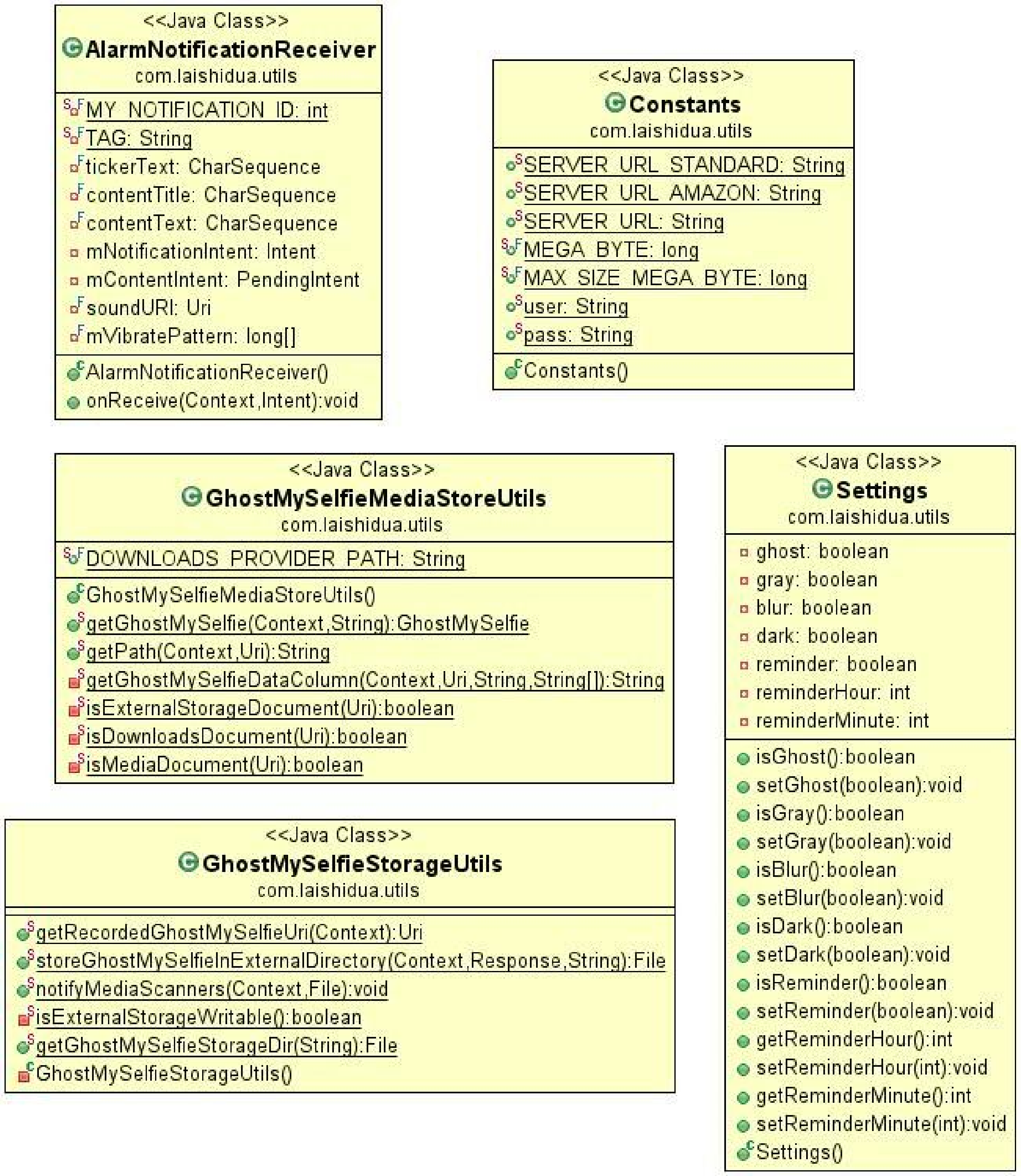
Retrieved Modified Image Suc.. 7:30 PM

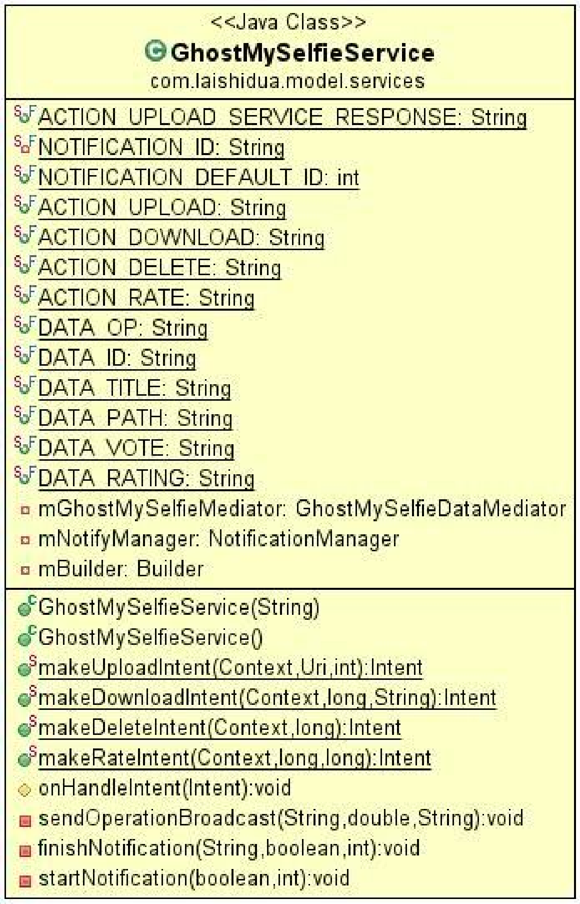
IMG_20151116_192800.jpg
Star rating: 0.0

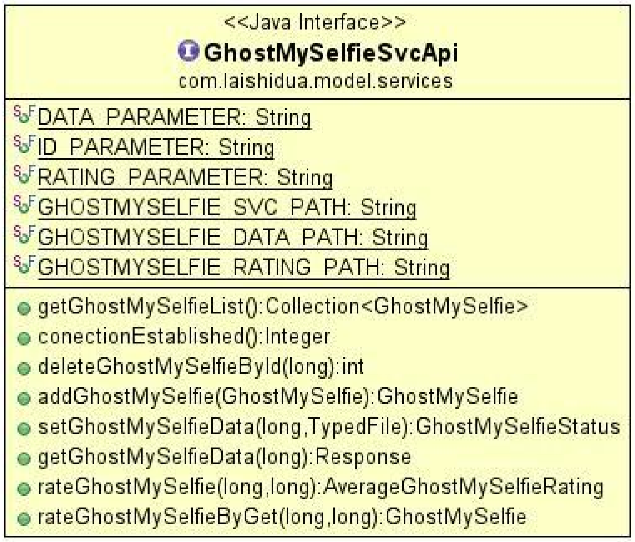
Figure 29. Concurrent Execution.

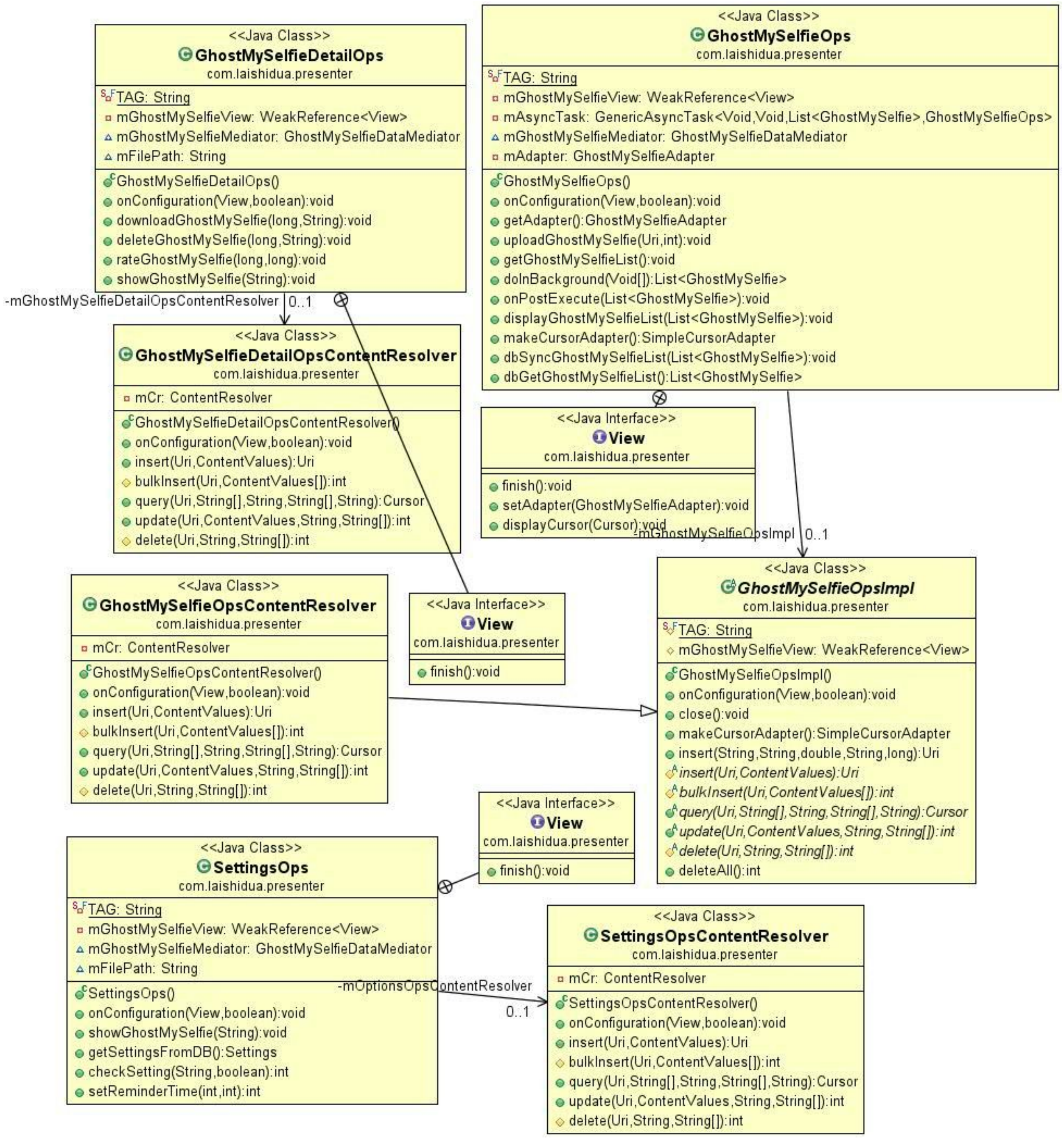
CLIENT UML

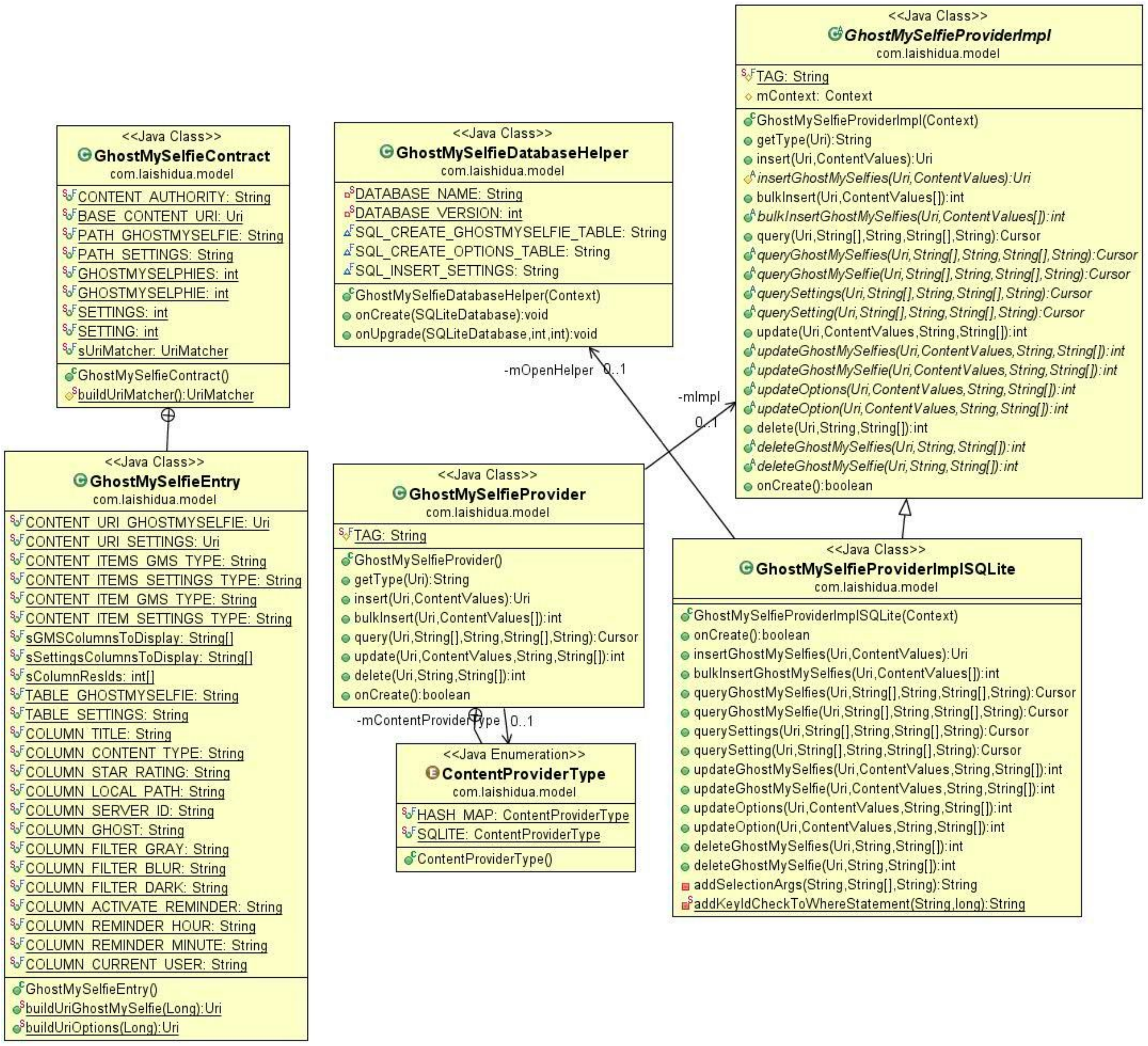

<<Java Class>>
GhostMySelfieDetailActivity
com.laishidua.view
mTextData: TextView
mRatingBar: RatingBar
mThumbnail: ImageView
mId: long
mTitle: String
mFilePath: String
GhostMySelfieDetailActivity()
onCreate(Bundle):void
onResume():void
registerReceiver():void
onPause():void
setInfo(String,double):String
finish():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
-mDeleteRowItem
0..1
<<Java Class>>
FloatingGenericButton
com.laishidua.view.ui
overshootInterpolator: OvershootInterpolator
accelerateInterpolator: AccelerateInterpolator
mButtonPaint: Paint
mDrawablePaint: Paint
mBitmap: Bitmap
drawable: Drawable
color: int
mHidden: boolean
FloatingGenericButton(Context,AttributeSet)
setFloatingActionButtonColor(int):void
setFloatingActionButtonDrawable(Drawable):void
init(int):void
onDraw(Canvas):void
onTouchEvent(MotionEvent):boolean
hideFloatingActionButton():void
showFloatingActionButton():void
isHidden():boolean
<<Java Class>>
GhostMySelfieListActivity
com.laishidua.view
REQUEST_GHOSTMYSELFIE_CAPTURE: int
REQUEST_GET_GHOSTMYSELFIE: int
REQUEST_SETTINGS: int
REQUEST_LOGIN: int
mGhostMySelfiesList: ListView
mAdapter: SimpleCursorAdapter
mCurrentGhostMySelfieUri: Uri
selectedGhostMySelfie: GhostMySelfie
mContext: Context
-mUploadGhostMySelfieButton
0..1
-mModifierButton
0..1
GhostMySelfieListActivity()
onCreate(Bundle):void
onResume():void
registerReceiver():void
onPause():void
onGhostMySelfieSelected(OperationType):void
onActivityResult(int,int,Intent):void
setAdapter(GhostMySelfieAdapter):void
finish():void
onCreateOptionsMenu(Menu):boolean
onOptionsItemSelected(MenuItem):boolean
displayCursor(Cursor):void
-mUploadResultReceiver
0..1
<<Java Class>>
UploadResultReceiver
com.laishidua.view
UploadResultReceiver()
onReceive(Context,Intent):void
<<Java Class>>
GhostMySelfieQueryHandler
com.laishidua.view
GhostMySelfieQueryHandler(ContentResolver)
-mUploadResultReceiver
0..1
<<Java Class>>
UploadResultReceiver
com.laishidua.view
UploadResultReceiver()
onReceive(Context,Intent):void
<<Java Class>>
SettingsActivity
com.laishidua.view
TAG: String
chkGhost: CheckBox
chkGray: CheckBox
chkBlur: CheckBox
chkDark: CheckBox
chkActivateReminder: CheckBox
reminderTime: TimePicker
mAlarmManager: AlarmManager
mNotificationReceiverIntent: Intent
mNotificationReceiverPendingIntent: PendingIntent
SettingsActivity()
onCreate(Bundle):void
addListenerOnChkGhost():void
addListenerOnChkBlur():void
addListenerOnChkDark():void
addListenerOnChkGray():void
addListenerOnChkActivateReminder():void
addListenerOnReminderTime():void
onResume():void
onPause():void
finish():void
<<Java Class>>
LoginScreenActivity
com.laishidua.view
userName_: EditText
password_: EditText
LoginScreenActivity()
onCreate(Bundle):void
login():void
signUp():void
<<Java Interface>>
OnGhostMySelfieSelectedListener
com.laishidua.view.ui
onGhostMySelfieSelected(OperationType):void
-mListener
0..1
<<Java Enumeration>>
OperationType
com.laishidua.view.ui
GHOSTMYSELFIE_GALLERY: OperationType
SHOT_GHOSTMYSELFIE: OperationType
OperationType()
<<Java Class>>
UploadGhostMySelfieDialogFragment
com.laishidua.view.ui
listItems: String[]
UploadGhostMySelfieDialogFragment()
onAttach(Activity):void
onCreateDialog(Bundle):Dialog
<<Java Class>>
GhostMySelfieAdapter
com.laishidua.view.ui
mContext: Context
TAG: String
ghostmyselfieList: List<GhostMySelfie>
modifierList: List<String>
GhostMySelfieAdapter(Context)
getView(int,View,ViewGroup):View
getModifierCheckedBoxList():List<String>
add(GhostMySelfie):void
remove(GhostMySelfie):void
getGhostMySelfies():List<GhostMySelfie>
setGhostMySelfies(List<GhostMySelfie>):void
getCount():int
getItem(int):GhostMySelfie
getItemId(int):long


<<Java Class>>
AlarmNotificationReceiver
com.laishidua.utils
MY_NOTIFICATION_ID: int
TAG: String
tickerText: CharSequence
contentTitle: CharSequence
contentText: CharSequence
mNotificationIntent: Intent
mContentIntent: PendingIntent
soundURI: Uri
mVibratePattern: long[]
AlarmNotificationReceiver()
onReceive(Context,Intent):void
<<Java Class>>
Constants
com.laishidua.utils
SERVER_URL_STANDARD: String
SERVER_URL_AMAZON: String
SERVER_URL: String
MEGA_BYTE: long
MAX_SIZE_MEGA_BYTE: long
user: String
pass: String
Constants()
<<Java Class>>
GhostMySelfieMediaStoreUtils
com.laishidua.utils
DOWNLOADS_PROVIDER_PATH: String
GhostMySelfieMediaStoreUtils()
getGhostMySelfie(Context,String):GhostMySelfie
getPath(Context,Uri):String
getGhostMySelfieDataColumn(Context,Uri,String,String[]):String
isExternalStorageDocument(Uri):boolean
isDownloadsDocument(Uri):boolean
isMediaDocument(Uri):boolean
<<Java Class>>
Settings
com.laishidua.utils
ghost: boolean
gray: boolean
blur: boolean
dark: boolean
reminder: boolean
reminderHour: int
reminderMinute: int
isGhost():boolean
setGhost(boolean):void
isGray():boolean
setGray(boolean):void
isBlur():boolean
setBlur(boolean):void
isDark():boolean
setDark(boolean):void
isReminder():boolean
setReminder(boolean):void
getReminderHour():int
setReminderHour(int):void
getReminderMinute():int
setReminderMinute(int):void
Settings()
<<Java Class>>
GhostMySelfieStorageUtils
com.laishidua.utils
getRecordedGhostMySelfieUri(Context):Uri
storeGhostMySelfieInExternalDirectory(Context,Response,String):File
notifyMediaScanners(Context,File):void
isExternalStorageWritable():boolean
getGhostMySelfieStorageDir(String):File
GhostMySelfieStorageUtils()

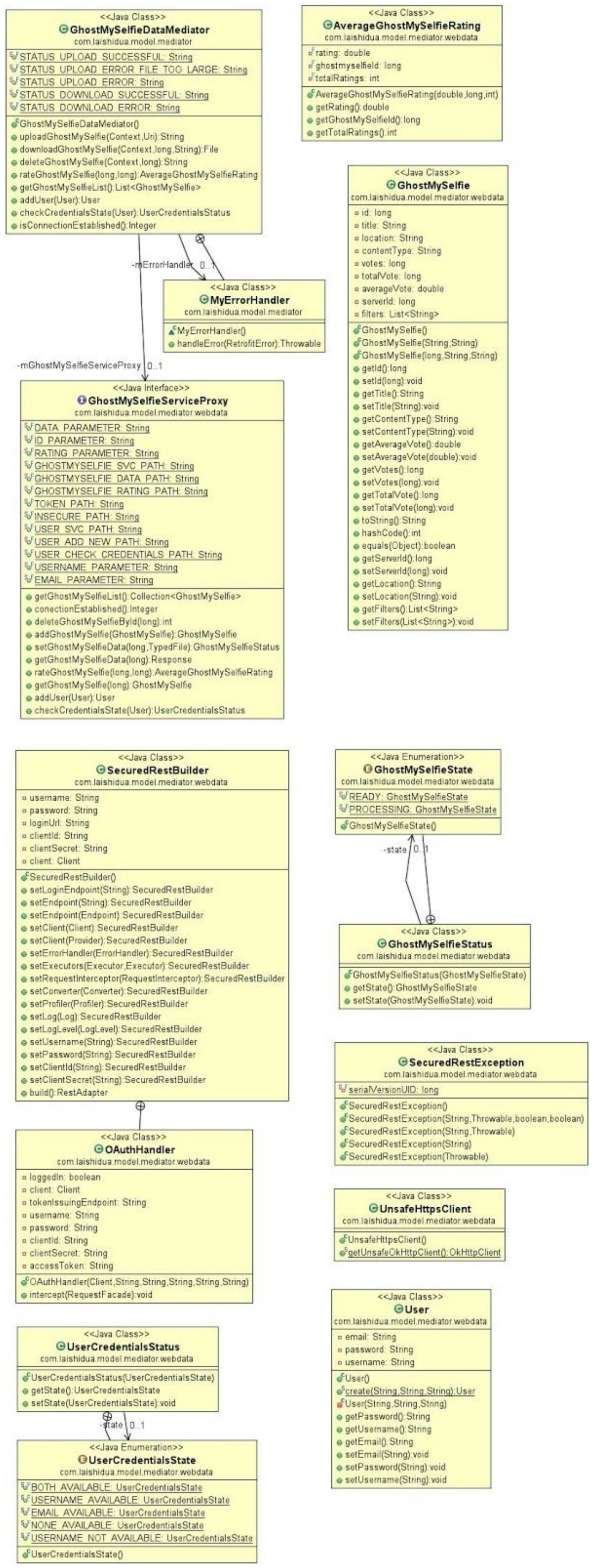

<<Java Class>>
GhostMySelfieService
com.laishidua.model.services
ACTION_UPLOAD_SERVICE_RESPONSE: String
NOTIFICATION_ID: String
NOTIFICATION_DEFAULT_ID: int
ACTION_UPLOAD: String
ACTION_DOWNLOAD: String
ACTION_DELETE: String
ACTION_RATE: String
DATA_OP: String
DATA_ID: String
DATA_TITLE: String
DATA_PATH: String
DATA_VOTE: String
DATA_RATING: String
mGhostMySelfieMediator: GhostMySelfieDataMediator
mNotifyManager: NotificationManager
mBuilder: Builder
GhostMySelfieService(String)
GhostMySelfieService()
makeUploadIntent(Context,Uri,int):Intent
makeDownloadIntent(Context,long,String):Intent
makeDeleteIntent(Context,long):Intent
makeRateIntent(Context,long,long):Intent
onHandleIntent(Intent):void
sendOperationBroadcast(String,double,String):void
finishNotification(String,boolean,int):void
startNotification(boolean,int):void


<<Java Interface>>
GhostMySelfieSvcApi
com.laishidua.model.services
DATA_PARAMETER: String
ID_PARAMETER: String
RATING_PARAMETER: String
GHOSTMYSELFIE_SVC_PATH: String
GHOSTMYSELFIE_DATA_PATH: String
GHOSTMYSELFIE_RATING_PATH: String
getGhostMySelfieList():Collection<GhostMySelfie>
conectionEstablished():Integer
deleteGhostMySelfieById(long):int
addGhostMySelfie(GhostMySelfie):GhostMySelfie
setGhostMySelfieData(long,TypedFile):GhostMySelfieStatus
getGhostMySelfieData(long):Response
rateGhostMySelfie(long,long):AverageGhostMySelfieRating
rateGhostMySelfieByGet(long,long):GhostMySelfie

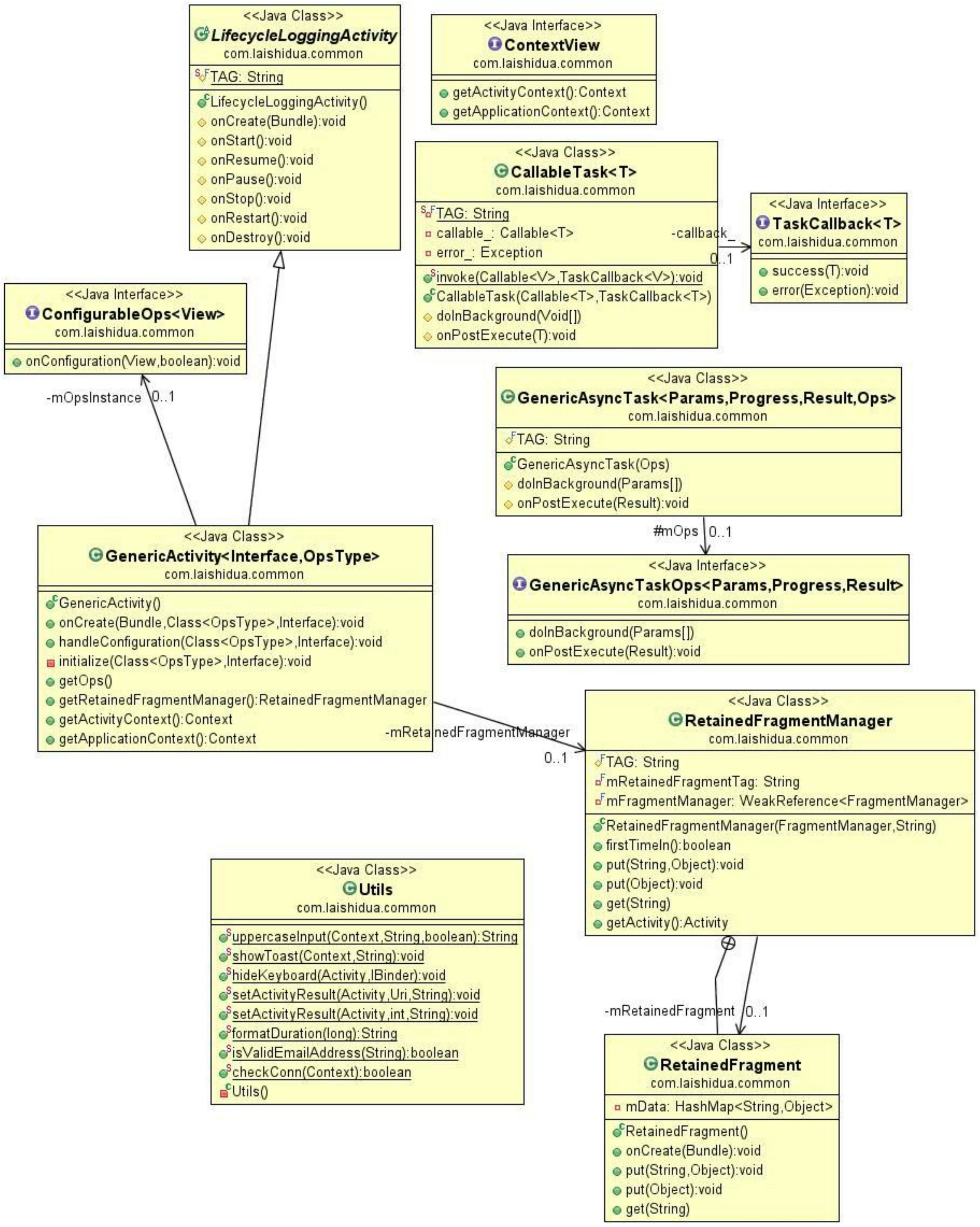

<<Java Class>>
GhostMySelfieDetailOps
com.laishidua.presenter
TAG: String
mGhostMySelfieView: WeakReference<View>
mGhostMySelfieMediator: GhostMySelfieDataMediator
mFilePath: String
GhostMySelfieDetailOps()
onConfiguration(View,boolean):void
downloadGhostMySelfie(long,String):void
deleteGhostMySelfie(long,String):void
rateGhostMySelfie(long,long):void
showGhostMySelfie(String):void
<<Java Class>>
GhostMySelfieOps
com.laishidua.presenter
TAG: String
mGhostMySelfieView: WeakReference<View>
mAsyncTask: GenericAsyncTask<Void,Void,List<GhostMySelfie>,GhostMySelfieOps>
mGhostMySelfieMediator: GhostMySelfieDataMediator
mAdapter: GhostMySelfieAdapter
GhostMySelfieOps()
onConfiguration(View,boolean):void
getAdapter():GhostMySelfieAdapter
uploadGhostMySelfie(Uri,int):void
getGhostMySelfieList():void
doInBackground(Void[]):List<GhostMySelfie>
onPostExecute(List<GhostMySelfie>):void
displayGhostMySelfieList(List<GhostMySelfie>):void
makeCursorAdapter():SimpleCursorAdapter
dbSyncGhostMySelfieList(List<GhostMySelfie>):void
dbGetGhostMySelfieList():List<GhostMySelfie>
-mGhostMySelfieDetailOpsContentResolver
0..1
<<Java Class>>
GhostMySelfieDetailOpsContentResolver
com.laishidua.presenter
mCr: ContentResolver
GhostMySelfieDetailOpsContentResolver()
onConfiguration(View,boolean):void
insert(Uri,ContentValues):Uri
bulkInsert(Uri,ContentValues[]):int
query(Uri,String[],String,String[],String):Cursor
update(Uri,ContentValues,String,String[]):int
delete(Uri,String,String[]):int
<<Java Interface>>
View
com.laishidua.presenter
finish():void
setAdapter(GhostMySelfieAdapter):void
displayCursor(Cursor):void
-mGhostMySelfieOpsImpl
0..1
<<Java Class>>
GhostMySelfieOpsContentResolver
com.laishidua.presenter
mCr: ContentResolver
GhostMySelfieOpsContentResolver()
onConfiguration(View,boolean):void
insert(Uri,ContentValues):Uri
bulkInsert(Uri,ContentValues[]):int
query(Uri,String[],String,String[],String):Cursor
update(Uri,ContentValues,String,String[]):int
delete(Uri,String,String[]):int
<<Java Interface>>
View
com.laishidua.presenter
finish():void
<<Java Class>>
GhostMySelfieOpsImpl
com.laishidua.presenter
TAG: String
mGhostMySelfieView: WeakReference<View>
GhostMySelfieOpsImpl()
onConfiguration(View,boolean):void
close():void
makeCursorAdapter():SimpleCursorAdapter
insert(String,String,double,String,long):Uri
insert(Uri,ContentValues):Uri
bulkInsert(Uri,ContentValues[]):int
query(Uri,String[],String,String[],String):Cursor
update(Uri,ContentValues,String,String[]):int
delete(Uri,String,String[]):int
deleteAll():int
<<Java Interface>>
View
com.laishidua.presenter
finish():void
<<Java Class>>
SettingsOps
com.laishidua.presenter
TAG: String
mGhostMySelfieView: WeakReference<View>
mGhostMySelfieMediator: GhostMySelfieDataMediator
mFilePath: String
SettingsOps()
onConfiguration(View,boolean):void
showGhostMySelfie(String):void
getSettingsFromDB():Settings
checkSetting(String,boolean):int
setReminderTime(int,int):int
-mOptionsOpsContentResolver
0..1
<<Java Class>>
SettingsOpsContentResolver
com.laishidua.presenter
mCr: ContentResolver
SettingsOpsContentResolver()
onConfiguration(View,boolean):void
insert(Uri,ContentValues):Uri
bulkInsert(Uri,ContentValues[]):int
query(Uri,String[],String,String[],String):Cursor
update(Uri,ContentValues,String,String[]):int
delete(Uri,String,String[]):int



<<Java Class>>
GhostMySelfieContract
com.laishidua.model
CONTENT_AUTHORITY: String
BASE_CONTENT_URI: Uri
PATH_GHOSTMYSELFIE: String
PATH_SETTINGS: String
GHOSTMYSELPHIES: int
GHOSTMYSELPHIE: int
SETTINGS: int
SETTING: int
sUriMatcher: UriMatcher
GhostMySelfieContract()
buildUriMatcher():UriMatcher
<<Java Class>>
GhostMySelfieEntry
com.laishidua.model
CONTENT_URI_GHOSTMYSELFIE: Uri
CONTENT_URI_SETTINGS: Uri
CONTENT_ITEMS_GMS_TYPE: String
CONTENT_ITEMS_SETTINGS_TYPE: String
CONTENT_ITEM_GMS_TYPE: String
CONTENT_ITEM_SETTINGS_TYPE: String
sGMSColumnsToDisplay: String[]
sSettingsColumnsToDisplay: String[]
sColumnResIds: int[]
TABLE_GHOSTMYSELFIE: String
TABLE_SETTINGS: String
COLUMN_TITLE: String
COLUMN_CONTENT_TYPE: String
COLUMN_STAR_RATING: String
COLUMN_LOCAL_PATH: String
COLUMN_SERVER_ID: String
COLUMN_GHOST: String
COLUMN_FILTER_GRAY: String
COLUMN_FILTER_BLUR: String
COLUMN_FILTER_DARK: String
COLUMN_ACTIVATE_REMINDER: String
COLUMN_REMINDER_HOUR: String
COLUMN_REMINDER_MINUTE: String
COLUMN_CURRENT_USER: String
GhostMySelfieEntry()
buildUriGhostMySelfie(Long):Uri
buildUriOptions(Long):Uri
<<Java Class>>
GhostMySelfieDatabaseHelper
com.laishidua.model
DATABASE_NAME: String
DATABASE_VERSION: int
SQL_CREATE_GHOSTMYSELFIE_TABLE: String
SQL_CREATE_OPTIONS_TABLE: String
SQL_INSERT_SETTINGS: String
GhostMySelfieDatabaseHelper(Context)
onCreate(SQLiteDatabase):void
onUpgrade(SQLiteDatabase,int,int):void
-mOpenHelper
0..1
-mImpl
0..1
<<Java Class>>
GhostMySelfieProvider
com.laishidua.model
TAG: String
GhostMySelfieProvider()
getType(Uri):String
insert(Uri,ContentValues):Uri
bulkInsert(Uri,ContentValues[]):int
query(Uri,String[],String,String[],String):Cursor
update(Uri,ContentValues,String,String[]):int
delete(Uri,String,String[]):int
onCreate():boolean
-mContentProviderType
0..1
<<Java Enumeration>>
ContentProviderType
com.laishidua.model
HASH_MAP: ContentProviderType
SQLITE: ContentProviderType
ContentProviderType()
<<Java Class>>
GhostMySelfieProviderImpl
com.laishidua.model
TAG: String
mContext: Context
GhostMySelfieProviderImpl(Context)
getType(Uri):String
insert(Uri,ContentValues):Uri
insertGhostMySelfies(Uri,ContentValues):Uri
bulkInsert(Uri,ContentValues[]):int
bulkInsertGhostMySelfies(Uri,ContentValues[]):int
query(Uri,String[],String,String[],String):Cursor
queryGhostMySelfies(Uri,String[],String,String[],String):Cursor
queryGhostMySelfie(Uri,String[],String,String[],String):Cursor
querySettings(Uri,String[],String,String[],String):Cursor
querySetting(Uri,String[],String,String[],String):Cursor
update(Uri,ContentValues,String,String[]):int
updateGhostMySelfies(Uri,ContentValues,String,String[]):int
updateGhostMySelfie(Uri,ContentValues,String,String[]):int
updateOptions(Uri,ContentValues,String,String[]):int
updateOption(Uri,ContentValues,String,String[]):int
delete(Uri,String,String[]):int
deleteGhostMySelfies(Uri,String,String[]):int
deleteGhostMySelfie(Uri,String,String[]):int
onCreate():boolean
<<Java Class>>
GhostMySelfieProviderImplSQLite
com.laishidua.model
GhostMySelfieProviderImplSQLite(Context)
onCreate():boolean
insertGhostMySelfies(Uri,ContentValues):Uri
bulkInsertGhostMySelfies(Uri,ContentValues[]):int
queryGhostMySelfies(Uri,String[],String,String[],String):Cursor
queryGhostMySelfie(Uri,String[],String,String[],String):Cursor
querySettings(Uri,String[],String,String[],String):Cursor
querySetting(Uri,String[],String,String[],String):Cursor
updateGhostMySelfies(Uri,ContentValues,String,String[]):int
updateGhostMySelfie(Uri,ContentValues,String,String[]):int
updateOptions(Uri,ContentValues,String,String[]):int
updateOption(Uri,ContentValues,String,String[]):int
deleteGhostMySelfies(Uri,String,String[]):int
deleteGhostMySelfie(Uri,String,String[]):int
addSelectionArgs(String,String[],String):String
addKeyIdCheckToWhereStatement(String,long):String

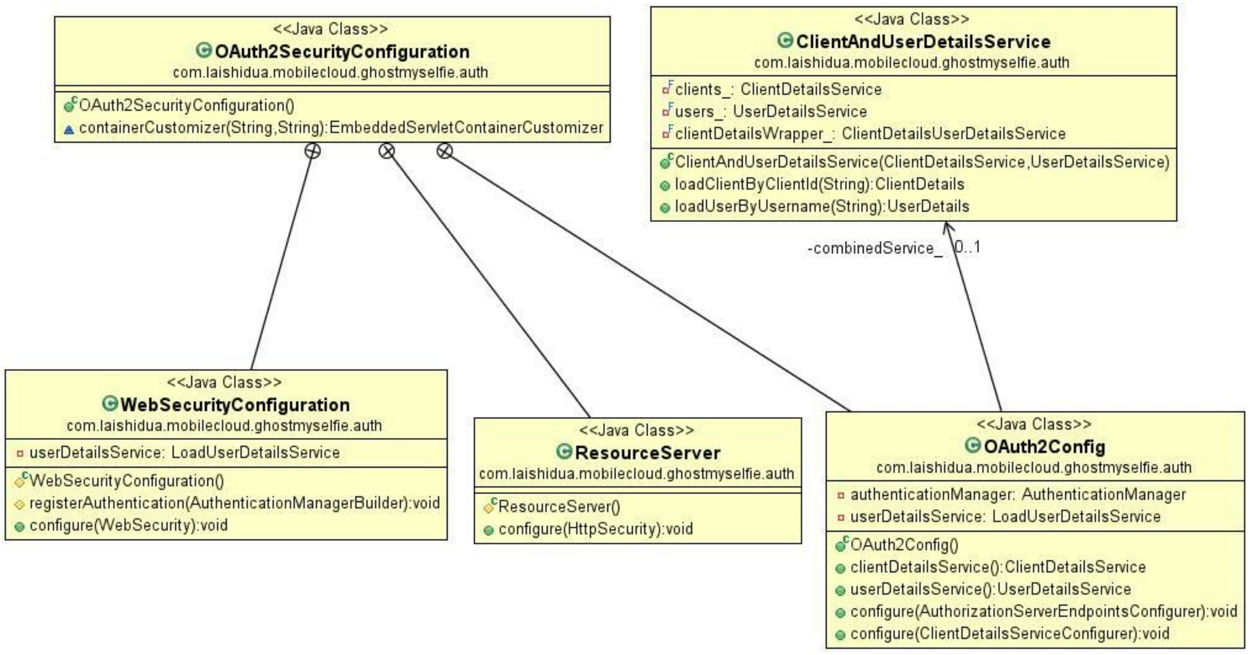


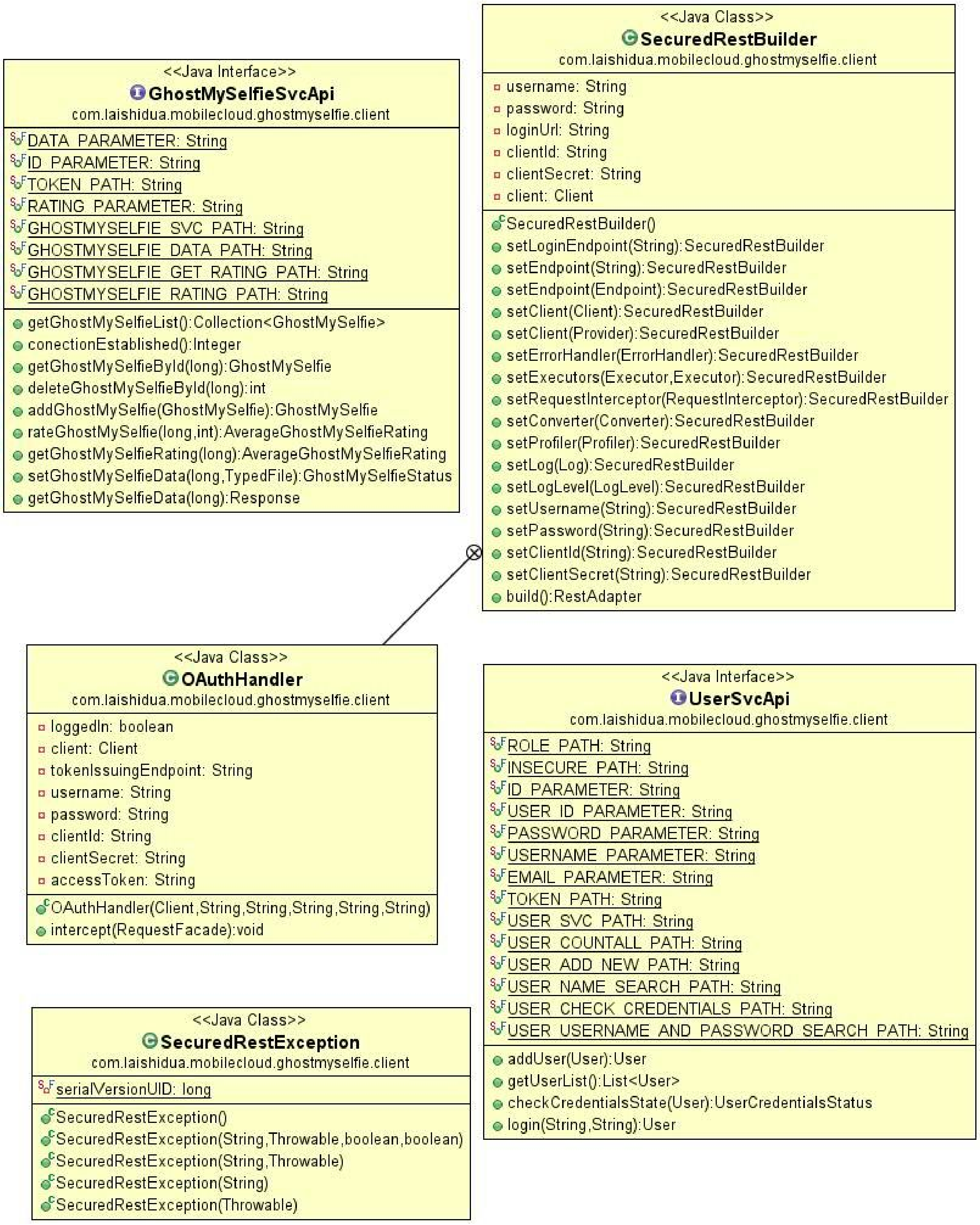
<<Java Class>>
GhostMySelfieDataMediator
com.laishidua.model.mediator
STATUS_UPLOAD_SUCCESSFUL: String
STATUS_UPLOAD_ERROR_FILE_TOO_LARGE: String
STATUS_UPLOAD_ERROR: String
STATUS_DOWNLOAD_SUCCESSFUL: String
STATUS_DOWNLOAD_ERROR: String
GhostMySelfieDataMediator()
uploadGhostMySelfie(Context,Uri):String
downloadGhostMySelfie(Context,long,String):File
deleteGhostMySelfie(Context,long):String
rateGhostMySelfie(long,long):AverageGhostMySelfieRating
getGhostMySelfieList():List<GhostMySelfie>
addUser(User):User
checkCredentialsState(User):UserCredentialsStatus
isConnectionEstablished():Integer
-mErrorHandler 0..1
<<Java Class>>
MyErrorHandler
com.laishidua.model.mediator
MyErrorHandler()
handleError(RetrofitError):Throwable
-mGhostMySelfieServiceProxy 0..1
<<Java Interface>>
GhostMySelfieServiceProxy
com.laishidua.model.mediator.webdata
DATA_PARAMETER: String
ID_PARAMETER: String
RATING_PARAMETER: String
GHOSTMYSELFIE_SVC_PATH: String
GHOSTMYSELFIE_DATA_PATH: String
GHOSTMYSELFIE_RATING_PATH: String
TOKEN_PATH: String
INSECURE_PATH: String
USER_SVC_PATH: String
USER_ADD_NEW_PATH: String
USER_CHECK_CREDENTIALS_PATH: String
USERNAME_PARAMETER: String
EMAIL_PARAMETER: String
getGhostMySelfieList():Collection<GhostMySelfie>
conectionEstablished():Integer
deleteGhostMySelfieById(long):int
addGhostMySelfie(GhostMySelfie):GhostMySelfie
setGhostMySelfieData(long,TypedFile):GhostMySelfieStatus
getGhostMySelfieData(long):Response
rateGhostMySelfie(long,long):AverageGhostMySelfieRating
getGhostMySelfie(long):GhostMySelfie
addUser(User):User
checkCredentialsState(User):UserCredentialsStatus
<<Java Class>>
AverageGhostMySelfieRating
com.laishidua.model.mediator.webdata
rating: double
ghostmyselfield: long
totalRatings: int
AverageGhostMySelfieRating(double,long,int)
getRating():double
getGhostMySelfield():long
getTotalRatings():int
<<Java Class>>
GhostMySelfie
com.laishidua.model.mediator.webdata
id: long
title: String
location: String
contentType: String
votes: long
totalVote: long
averageVote: double
serverId: long
filters: List<String>
GhostMySelfie()
GhostMySelfie(String,String)
GhostMySelfie(long,String,String)
getId():long
setId(long):void
getTitle():String
setTitle(String):void
getContentType():String
setContentType(String):void
getAverageVote():double
setAverageVote(double):void
getVotes():long
setVotes(long):void
getTotalVote():long
setTotalVote(long):void
toString():String
hashCode():int
equals(Object):boolean
getServerId():long
setServerId(long):void
getLocation():String
setLocation(String):void
getFilters():List<String>
setFilters(List<String>):void
<<Java Class>>
SecuredRestBuilder
com.laishidua.model.mediator.webdata
username: String
password: String
loginUrl: String
clientId: String
clientSecret: String
client: Client
SecuredRestBuilder()
setLoginEndpoint(String):SecuredRestBuilder
setEndpoint(String):SecuredRestBuilder
setEndpoint(Endpoint):SecuredRestBuilder
setClient(Client):SecuredRestBuilder
setClient(Provider):SecuredRestBuilder
setErrorHandler(ErrorHandler):SecuredRestBuilder
setExecutors(Executor,Executor):SecuredRestBuilder
setRequestInterceptor(RequestInterceptor):SecuredRestBuilder
setConverter(Converter):SecuredRestBuilder
setProfiler(Profiler):SecuredRestBuilder
setLog(Log):SecuredRestBuilder
setLogLevel(LogLevel):SecuredRestBuilder
setUsername(String):SecuredRestBuilder
setPassword(String):SecuredRestBuilder
setClientId(String):SecuredRestBuilder
setClientSecret(String):SecuredRestBuilder
build():RestAdapter
<<Java Class>>
OAuthHandler
com.laishidua.model.mediator.webdata
loggedIn: boolean
client: Client
tokenIssuingEndpoint: String
username: String
password: String
clientId: String
clientSecret: String
accessToken: String
OAuthHandler(Client,String,String,String,String,String)
intercept(RequestFacade):void
<<Java Class>>
UserCredentialsStatus
com.laishidua.model.mediator.webdata
UserCredentialsStatus(UserCredentialsState)
getState():UserCredentialsState
setState(UserCredentialsState):void
-state 0..1
<<Java Enumeration>>
UserCredentialsState
com.laishidua.model.mediator.webdata
BOTH_AVAILABLE: UserCredentialsState
USERNAME_AVAILABLE: UserCredentialsState
EMAIL_AVAILABLE: UserCredentialsState
NONE_AVAILABLE: UserCredentialsState
USERNAME_NOT_AVAILABLE: UserCredentialsState
UserCredentialsState()
<<Java Enumeration>>
GhostMySelfieState
com.laishidua.model.mediator.webdata
READY: GhostMySelfieState
PROCESSING: GhostMySelfieState
GhostMySelfieState()
-state 0..1
<<Java Class>>
GhostMySelfieStatus
com.laishidua.model.mediator.webdata
GhostMySelfieStatus(GhostMySelfieState)
getState():GhostMySelfieState
setState(GhostMySelfieState):void
<<Java Class>>
SecuredRestException
com.laishidua.model.mediator.webdata
serialVersionUID: long
SecuredRestException()
SecuredRestException(String,Throwable,boolean,boolean)
SecuredRestException(String,Throwable)
SecuredRestException(String)
SecuredRestException(Throwable)
<<Java Class>>
UnsafeHttpsClient
com.laishidua.model.mediator.webdata
UnsafeHttpsClient()
getUnsafeOkHttpClient():OkHttpClient
<<Java Class>>
User
com.laishidua.model.mediator.webdata
email: String
password: String
username: String
User()
create(String,String,String):User
User(String,String,String)
getPassword():String
getUsername():String
getEmail():String
setEmail(String):void
setPassword(String):void
setUsername(String):void

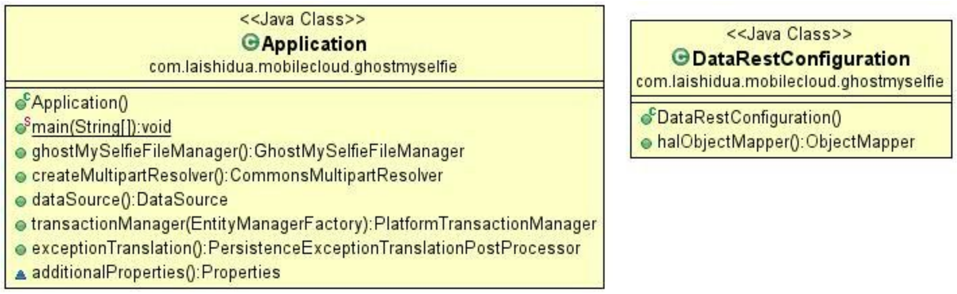


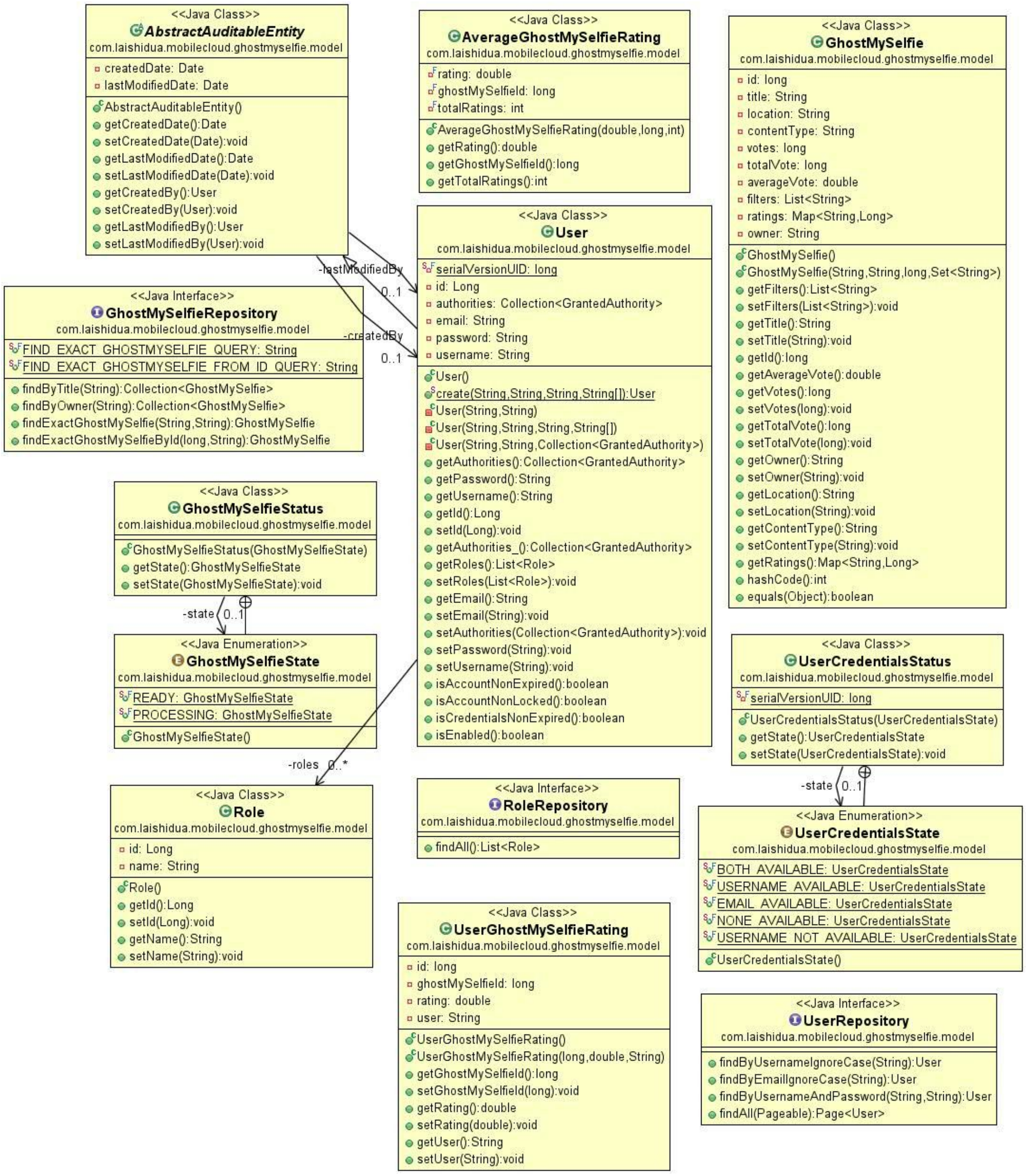
<<Java Class>>
LifecycleLoggingActivity
com.laishidua.common
TAG: String
LifecycleLoggingActivity()
onCreate(Bundle):void
onStart():void
onResume():void
onPause():void
onStop():void
onRestart():void
onDestroy():void
<<Java Interface>>
ContextView
com.laishidua.common
getActivityContext():Context
getApplicationContext():Context
<<Java Class>>
CallableTask<T>
com.laishidua.common
TAG: String
callable_: Callable<T>
error_: Exception
invoke(Callable<V>,TaskCallback<V>):void
CallableTask(Callable<T>,TaskCallback<T>)
doInBackground(Void[])
onPostExecute(T):void
-callback_
0..1
<<Java Interface>>
TaskCallback<T>
com.laishidua.common
success(T):void
error(Exception):void
<<Java Interface>>
ConfigurableOps<View>
com.laishidua.common
onConfiguration(View,boolean):void
-mOpsInstance
0..1
<<Java Class>>
GenericAsyncTask<Params,Progress,Result,Ops>
com.laishidua.common
TAG: String
GenericAsyncTask(Ops)
doInBackground(Params[])
onPostExecute(Result):void
#mOps
0..1
<<Java Interface>>
GenericAsyncTaskOps<Params,Progress,Result>
com.laishidua.common
doInBackground(Params[])
onPostExecute(Result):void
<<Java Class>>
GenericActivity<Interface,OpsType>
com.laishidua.common
GenericActivity()
onCreate(Bundle,Class<OpsType>,Interface):void
handleConfiguration(Class<OpsType>,Interface):void
initialize(Class<OpsType>,Interface):void
getOps()
getRetainedFragmentManager():RetainedFragmentManager
getActivityContext():Context
getApplicationContext():Context
-mRetainedFragmentManager
0..1
<<Java Class>>
RetainedFragmentManager
com.laishidua.common
TAG: String
mRetainedFragmentTag: String
mFragmentManager: WeakReference<FragmentManager>
RetainedFragmentManager(FragmentManager,String)
firstTimeIn():boolean
put(String,Object):void
put(Object):void
get(String)
getActivity():Activity
<<Java Class>>
Utils
com.laishidua.common
uppercaseInput(Context,String,boolean):String
showToast(Context,String):void
hideKeyboard(Activity,IBinder):void
setActivityResult(Activity,Uri,String):void
setActivityResult(Activity,int,String):void
formatDuration(long):String
isValidEmailAddress(String):boolean
checkConn(Context):boolean
Utils()
-mRetainedFragment
0..1
<<Java Class>>
RetainedFragment
com.laishidua.common
mData: HashMap<String,Object>
RetainedFragment()
onCreate(Bundle):void
put(String,Object):void
put(Object):void
get(String)

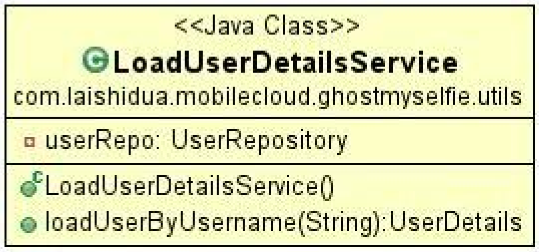
SERVER UML

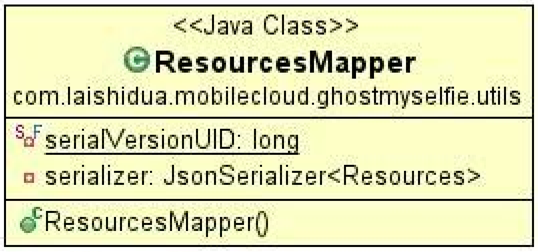

<<Java Class>>
OAuth2SecurityConfiguration
com.laishidua.mobilecloud.ghostmyselfie.auth
OAuth2SecurityConfiguration()
containerCustomizer(String,String):EmbeddedServletContainerCustomizer
<<Java Class>>
ClientAndUserDetailsService
com.laishidua.mobilecloud.ghostmyselfie.auth
clients_: ClientDetailsService
users_: UserDetailsService
clientDetailsWrapper_: ClientDetailsUserDetailsService
ClientAndUserDetailsService(ClientDetailsService,UserDetailsService)
loadClientByClientId(String):ClientDetails
loadUserByUsername(String):UserDetails
-combinedService_
0..1
<<Java Class>>
WebSecurityConfiguration
com.laishidua.mobilecloud.ghostmyselfie.auth
userDetailsService: LoadUserDetailsService
WebSecurityConfiguration()
registerAuthentication(AuthenticationManagerBuilder):void
configure(WebSecurity):void
<<Java Class>>
ResourceServer
com.laishidua.mobilecloud.ghostmyselfie.auth
ResourceServer()
configure(HttpSecurity):void
<<Java Class>>
OAuth2Config
com.laishidua.mobilecloud.ghostmyselfie.auth
authenticationManager: AuthenticationManager
userDetailsService: LoadUserDetailsService
OAuth2Config()
clientDetailsService():ClientDetailsService
userDetailsService():UserDetailsService
configure(AuthorizationServerEndpointsConfigurer):void
configure(ClientDetailsServiceConfigurer):void


<<Java Interface>>
GhostMySelfieSvcApi
com.laishidua.mobilecloud.ghostmyselfie.client
DATA_PARAMETER: String
ID_PARAMETER: String
TOKEN_PATH: String
RATING_PARAMETER: String
GHOSTMYSELFIE_SVC_PATH: String
GHOSTMYSELFIE_DATA_PATH: String
GHOSTMYSELFIE_GET_RATING_PATH: String
GHOSTMYSELFIE_RATING_PATH: String
getGhostMySelfieList():Collection<GhostMySelfie>
conectionEstablished():Integer
getGhostMySelfieById(long):GhostMySelfie
deleteGhostMySelfieById(long):int
addGhostMySelfie(GhostMySelfie):GhostMySelfie
rateGhostMySelfie(long,int):AverageGhostMySelfieRating
getGhostMySelfieRating(long):AverageGhostMySelfieRating
setGhostMySelfieData(long,TypedFile):GhostMySelfieStatus
getGhostMySelfieData(long):Response
<<Java Class>>
SecuredRestBuilder
com.laishidua.mobilecloud.ghostmyselfie.client
username: String
password: String
loginUrl: String
clientId: String
clientSecret: String
client: Client
SecuredRestBuilder()
setLoginEndpoint(String):SecuredRestBuilder
setEndpoint(String):SecuredRestBuilder
setEndpoint(Endpoint):SecuredRestBuilder
setClient(Client):SecuredRestBuilder
setClient(Provider):SecuredRestBuilder
setErrorHandler(ErrorHandler):SecuredRestBuilder
setExecutors(Executor,Executor):SecuredRestBuilder
setRequestInterceptor(RequestInterceptor):SecuredRestBuilder
setConverter(Converter):SecuredRestBuilder
setProfiler(Profiler):SecuredRestBuilder
setLog(Log):SecuredRestBuilder
setLogLevel(LogLevel):SecuredRestBuilder
setUsername(String):SecuredRestBuilder
setPassword(String):SecuredRestBuilder
setClientId(String):SecuredRestBuilder
setClientSecret(String):SecuredRestBuilder
build():RestAdapter
<<Java Class>>
OAuthHandler
com.laishidua.mobilecloud.ghostmyselfie.client
loggedIn: boolean
client: Client
tokenIssuingEndpoint: String
username: String
password: String
clientId: String
clientSecret: String
accessToken: String
OAuthHandler(Client,String,String,String,String,String)
intercept(RequestFacade):void
<<Java Interface>>
UserSvcApi
com.laishidua.mobilecloud.ghostmyselfie.client
ROLE_PATH: String
INSECURE_PATH: String
ID_PARAMETER: String
USER_ID_PARAMETER: String
PASSWORD_PARAMETER: String
USERNAME_PARAMETER: String
EMAIL_PARAMETER: String
TOKEN_PATH: String
USER_SVC_PATH: String
USER_COUNTALL_PATH: String
USER_ADD_NEW_PATH: String
USER_NAME_SEARCH_PATH: String
USER_CHECK_CREDENTIALS_PATH: String
USER_USERNAME_AND_PASSWORD_SEARCH_PATH: String
addUser(User):User
getUserList():List<User>
checkCredentialsState(User):UserCredentialsStatus
login(String,String):User
<<Java Class>>
SecuredRestException
com.laishidua.mobilecloud.ghostmyselfie.client
serialVersionUID: long
SecuredRestException()
SecuredRestException(String,Throwable,boolean,boolean)
SecuredRestException(String,Throwable)
SecuredRestException(String)
SecuredRestException(Throwable)


<<Java Class>>
Application
com.laishidua.mobilecloud.ghostmyselfie
Application()
main(String[]):void
ghostMySelfieFileManager():GhostMySelfieFileManager
createMultipartResolver():CommonsMultipartResolver
dataSource():DataSource
transactionManager(EntityManagerFactory):PlatformTransactionManager
exceptionTranslation():PersistenceExceptionTranslationPostProcessor
additionalProperties():Properties
<<Java Class>>
DataRestConfiguration
com.laishidua.mobilecloud.ghostmyselfie
DataRestConfiguration()
halObjectMapper():ObjectMapper


<<Java Class>>
AbstractAuditableEntity
com.laishidua.mobilecloud.ghostmyselfie.model
createdDate: Date
lastModifiedDate: Date
AbstractAuditableEntity()
getCreatedDate():Date
setCreatedDate(Date):void
getLastModifiedDate():Date
setLastModifiedDate(Date):void
getCreatedBy():User
setCreatedBy(User):void
getLastModifiedBy():User
setLastModifiedBy(User):void
-lastModifiedBy
0..1
-createdBy
0..1
<<Java Class>>
AverageGhostMySelfieRating
com.laishidua.mobilecloud.ghostmyselfie.model
rating: double
ghostMySelfield: long
totalRatings: int
AverageGhostMySelfieRating(double,long,int)
getRating():double
getGhostMySelfield():long
getTotalRatings():int
<<Java Class>>
GhostMySelfie
com.laishidua.mobilecloud.ghostmyselfie.model
id: long
title: String
location: String
contentType: String
votes: long
totalVote: long
averageVote: double
filters: List<String>
ratings: Map<String,Long>
owner: String
GhostMySelfie()
GhostMySelfie(String,String,long,Set<String>)
getFilters():List<String>
setFilters(List<String>):void
getTitle():String
setTitle(String):void
getId():long
getAverageVote():double
getVotes():long
setVotes(long):void
getTotalVote():long
setTotalVote(long):void
getOwner():String
setOwner(String):void
getLocation():String
setLocation(String):void
getContentType():String
setContentType(String):void
getRatings():Map<String,Long>
hashCode():int
equals(Object):boolean
<<Java Interface>>
GhostMySelfieRepository
com.laishidua.mobilecloud.ghostmyselfie.model
FIND_EXACT_GHOSTMYSELFIE_QUERY: String
FIND_EXACT_GHOSTMYSELFIE_FROM_ID_QUERY: String
findByTitle(String):Collection<GhostMySelfie>
findByOwner(String):Collection<GhostMySelfie>
findExactGhostMySelfie(String,String):GhostMySelfie
findExactGhostMySelfieById(long,String):GhostMySelfie
<<Java Class>>
User
com.laishidua.mobilecloud.ghostmyselfie.model
serialVersionUID: long
id: Long
authorities: Collection<GrantedAuthority>
email: String
password: String
username: String
User()
create(String,String,String,String[]):User
User(String,String)
User(String,String,String,String[])
User(String,String,Collection<GrantedAuthority>)
getAuthorities():Collection<GrantedAuthority>
getPassword():String
getUsername():String
getId():Long
setId(Long):void
getAuthorities_():Collection<GrantedAuthority>
getRoles():List<Role>
setRoles(List<Role>):void
getEmail():String
setEmail(String):void
setAuthorities(Collection<GrantedAuthority>):void
setPassword(String):void
setUsername(String):void
isAccountNonExpired():boolean
isAccountNonLocked():boolean
isCredentialsNonExpired():boolean
isEnabled():boolean
<<Java Class>>
GhostMySelfieStatus
com.laishidua.mobilecloud.ghostmyselfie.model
GhostMySelfieStatus(GhostMySelfieState)
getState():GhostMySelfieState
setState(GhostMySelfieState):void
-state
0..1
<<Java Enumeration>>
GhostMySelfieState
com.laishidua.mobilecloud.ghostmyselfie.model
READY: GhostMySelfieState
PROCESSING: GhostMySelfieState
GhostMySelfieState()
-roles
0..*
<<Java Class>>
Role
com.laishidua.mobilecloud.ghostmyselfie.model
id: Long
name: String
Role()
getId():Long
setId(Long):void
getName():String
setName(String):void
<<Java Class>>
UserCredentialsStatus
com.laishidua.mobilecloud.ghostmyselfie.model
serialVersionUID: long
UserCredentialsStatus(UserCredentialsState)
getState():UserCredentialsState
setState(UserCredentialsState):void
-state
0..1
<<Java Enumeration>>
UserCredentialsState
com.laishidua.mobilecloud.ghostmyselfie.model
BOTH_AVAILABLE: UserCredentialsState
USERNAME_AVAILABLE: UserCredentialsState
EMAIL_AVAILABLE: UserCredentialsState
NONE_AVAILABLE: UserCredentialsState
USERNAME_NOT_AVAILABLE: UserCredentialsState
UserCredentialsState()
<<Java Interface>>
RoleRepository
com.laishidua.mobilecloud.ghostmyselfie.model
findAll():List<Role>
<<Java Class>>
UserGhostMySelfieRating
com.laishidua.mobilecloud.ghostmyselfie.model
id: long
ghostMySelfield: long
rating: double
user: String
UserGhostMySelfieRating()
UserGhostMySelfieRating(long,double,String)
getGhostMySelfield():long
setGhostMySelfield(long):void
getRating():double
setRating(double):void
getUser():String
setUser(String):void
<<Java Interface>>
UserRepository
com.laishidua.mobilecloud.ghostmyselfie.model
findByUsernameIgnoreCase(String):User
findByEmailIgnoreCase(String):User
findByUsernameAndPassword(String,String):User
findAll(Pageable):Page<User>


<<Java Class>>
LoadUserDetailsService
com.laishidua.mobilecloud.ghostmyselfie.utils
userRepo: UserRepository
LoadUserDetailsService()
loadUserByUsername(String):UserDetails


<<Java Class>>
ResourcesMapper
com.laishidua.mobilecloud.ghostmyselfie.utils
serialVersionUID: long
serializer: JsonSerializer<Resources>
ResourcesMapper()