DAILY PHOTO

Mobile Cloud Computing with Android Specialization

I decided to modify the name of the application because there are so many uses for the app with such functionality. For example, you can take not selfies but pictures of your kitten and see how he is growing up. Another use is capturing the growth of plants. That is why I think that “Daily Photo” is a better name for my project.

I tried to write mainly about features that are already implemented.

General description

The Concurrent Daily Selfie application is an extension of one of the mini-projects in Professor Porter’s MOOCs, which 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 effects, such as blurring or charcoaling. The image processing is done via a remote web service. In addition, all interactions between the device and the remote web service should be done concurrently in the background to ensure that the UI thread on the device is not interrupted.

Basic Project Requirements

Multiple users and individual user accounts

Daily Photo have individual account for its users. When you launch an application the first screen is Login Screen, where you should enter server IP-address, your login and password. Application uses OAuth 2.0 to work with user accounts and it provides a secured connection with HTTPS. Currently it has only hardcoded users. Application uses EasyHttpClient to ignore certificates for testing purposes. This part of application has a client-server implementation from Security course as a basis. To make everything work correctly you need to add a keystore as a resource to your IDE.

Operations available only to authenticated users

Each photo entity has an owner property that is used for selecting and showing user only his or her own photos. Other photos that are stored on server and that have other owners will not be shown. User can’t pass login screen to see the content of application if he or she doesn’t have valid username and password. That means that all interactions require user to authenticate.

Activity, BroadcastReceiver, Service, ContentProvider usage

Application contains only one activity in which different fragments are shown. The class name is “MainActivity”. Application will contains Service to notify user that is it time to take a photo. The basis of Service is taken from Daily Selfie project from the second course of the Specialization.

Remote Server

The application connects to remote Spring-based server and uses HTTPS during its work. The communication is implemented using Retrofit library. The server basis is a combination of server implementation for Video Service application that was introduces in Communication, Spring and Security courses. You need to change a default server path in Login screen according to your environment.

Multiple user interface screens

User can navigate between multiple screens. Application contains only one activity and different screens are implemented with Fragments. Currently there are: Login Screen for signing in, Gallery Grid Screen with user photos. Also it will have a screen with full-size photo, a screen for working with filters and screen with settings to set up notifications and default server address.

Advanced capability

Obviously, there is a multimedia capture capability to take photos that will be loaded to the app gallery and remote server. I’m also planning to add custom animations to Gallery Grid loading: photos will appear like fading plus enlarging from very small to grid size.

Concurrency

Currently I am already using AsyncTask for login in, getting photo list from server, uploading and downloading photo file. AsyncTasks will also be used for sending photos to filtering. So that operations are working not in the UI Thread.

Basic Functional Requirements

Ability to take a photo

There is a button that makes an intent to Android basic Camera application. It receives photo in onActivityResult() and then adds it to gallery grid and sends to server.

Reminder

There is a Notification Service that will remind user to take a photo. There will be a message on Notification screen tapping on which will take to the application. The reminder will show the notification everyday or according to the set custom settings.

View photos in ListView

There is a Gallery screen where user can see all his or her photos. I decided to use not a ListView but a GridView. Prof. Douglas Schmidt approved that usage and confirmed that it follow that requirements. When user enter a login information, application checks this information as requests photo list from server. Then the Gallery Fragment is opened. When user takes a photo, it is added to Gallery automatically. Tapping on a photo will open it in larger size in separate Fragment. Prof. Douglas Schmidt approved using Fragment instead of the Activity that is mentioned on requirements page.

Persisting photos

If user closes the application that contained his or her photos and then opens it again he will need to login first. After that application will send a request to the server to get photo list. If there is no connection to the server or to the Internet in general, list of photos will be loaded from the device, for example, from SharedPreferences. If some photos are missing on device, they will be loaded from server.

Filters

Filtering will be implemented on the remote web service. User will need to press the button that means Select. After selecting photos that needed to be filtered, user will proceed to the Fragment screen where he or she can choose filters to apply. Photos will be sent to server for filtering using concurrent framework. I think to use AsyncTask here for each photo. I decided not to rely on some currently working services that provide filtering. I am planning to use ImageMagic or even implement my own filters since I have some experience in this. The easiest filters are: Black&White, GrayScale, Inverting colors. This filters are easy to implements using the most simple parallel algorithms. Once the result is obtained, photo will return as a response to application proper request and the regular photo in the gallery will be replaced with the new filtered one.

Implementation considerations

How will Reminders inform the User that it’s time to take a Selfie?

I created Notification Service that is an Android Service from Daily Selfie second source project. It sets an Alarm and sends notification to user.

How will you store the images on the device?

Images are stored as regular photos in public directory. Gallery don’t see this images because they have different format.

Will copies of both the original and filtered images be stored?

Original images will be replaced with filtered ones.

What image processing algorithms will you support?

I will implement black&white filter, grayscale, inverted colors and some others. Filters that I mentioned are very easy to implement and it is easy to launch them in parallel. I will do filtering on my server (I have just one server), it will be in Java.

What, if any, user preferences can the User set?

I don’t clearly understand what preferences are being talked about. I guess, it is about my settings screen that saves some default values to SharedPreferences.

How will concurrency requirements be implemented?

I use AsyncTask for each image to send it to server. AsyncTask will have galleryUpdate() method in onPostExecute() to show filtered images.

How will device communicate with the the remote web service?

I use Retrofit, Guava, HTTPS

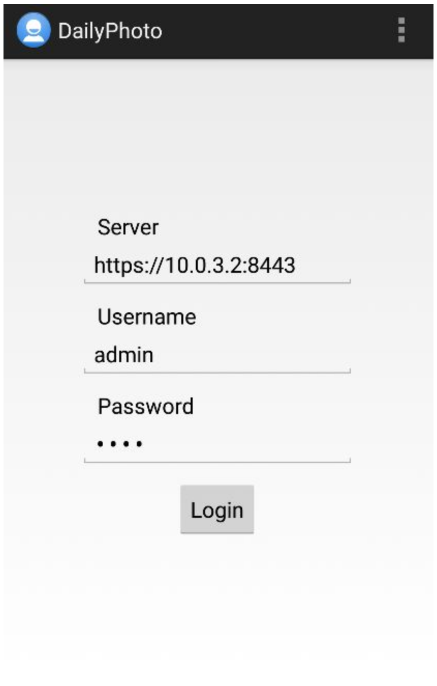
User Interface

There are 3 screenshots that shows what is already implemented.

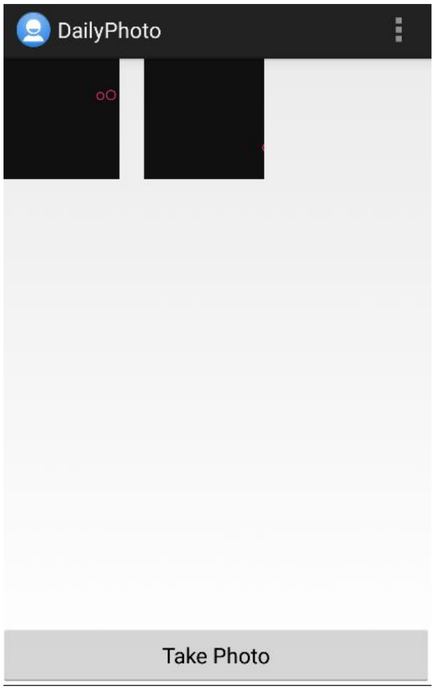
The First one is a login screen. User have to enter Server IP address, Username and Password. The IP address shown below is for Genymotion on Mac.

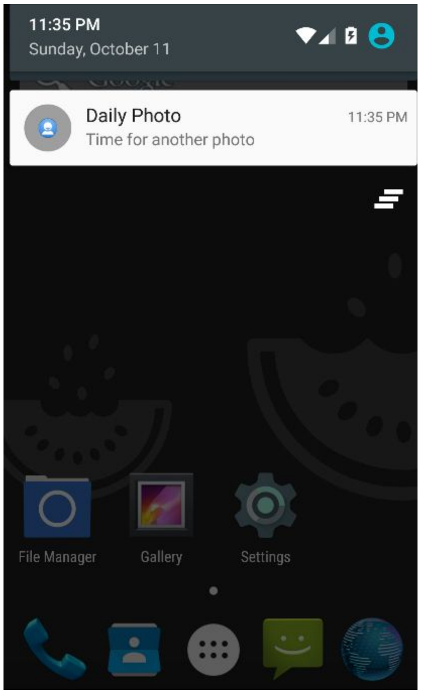
The second one is photo gallery grid. Here you can see that user have already made 2 photos. The grid has 3 columns. The Take Photo button will become floating material design button.

The third one shows notification that was sent by the application. The image will be different.



Server
https://10.0.3.2:8443
Username
admin
Password
••••





Daily Photo 11:35 PM
Time for another photo

Workflow

Login

1) Application is opened (or onBackPressed())
2) Login screen is shown. Gallery is reseted if onBackPressed.
3) User enters server, username, password and presses Login.
4) Application sends credentials to server:
    a) If login is correct:
        i) Server response with photo list data.
        ii) Client opens a grid gallery fragment and shows photos.
        iii) Client checks if some photos are missing on device:
            (1) If photo is missing, client will download it from server.
    b) If login is not correct:
        i) User receives Toast about error.
    c) No connection:
        i) User receives Toast about error.

Gallery

1) Gallery Grid Fragment is opened.
2) Image Adapter is set.
3) Image Adapter show images:
    a) Image Adapter opens images according to list sent by server
    b) Skip photos that are not found
    c) Scaled images are created to be shown in GridView
4) User Actions:
    a) User press Take Photo button
        i) Application send intent to Android camera application
        ii) Result is received from Camera:
            (1) If result is not OK:
                (a) do nothing
            (2) If result is OK:
                (a) Photo data is sent to server
                (b) Id received from server and used to upload photo file to server
                (c) Update photo grid
    b) User taps photo on grid
        i) Photo is opened in new Fragment, where it is shown in bigger size
    c) User taps select button
        i) User can cancel selection
        ii) User selects several photos and press filter button
            (1) Filter screen is opened

Filters

1) Filters screen is opened, pictures demonstrating filters are shown
2) User selects filters ne want to apply and press Done
3) Client starts asynctask for each image sending them to server
4) Server applies filters. It uses parallel algorithms to process images.
5) Server saves resulting image to his storage
6) Server sends result image to client
7) Client replaces old image with new one in the app and in file system

Photo

1) User taps on photo in gallery and single photo fragment is opened
2) If photo file is missing, app with request it from server
3) User press Back to go to the gallery

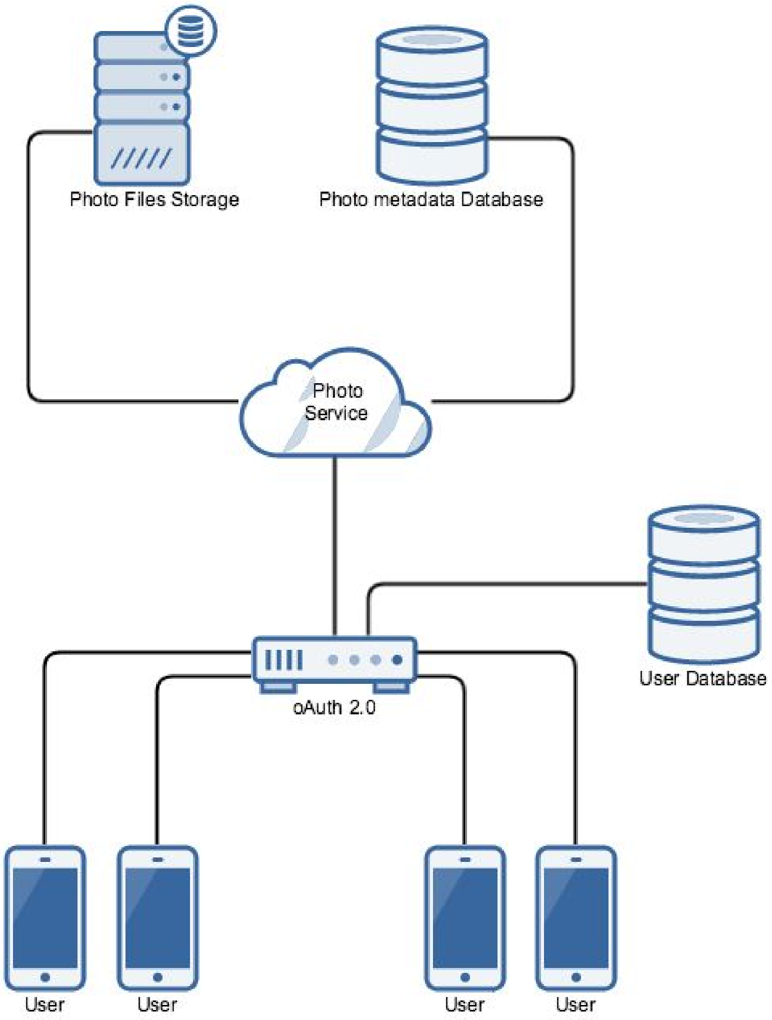
Settings

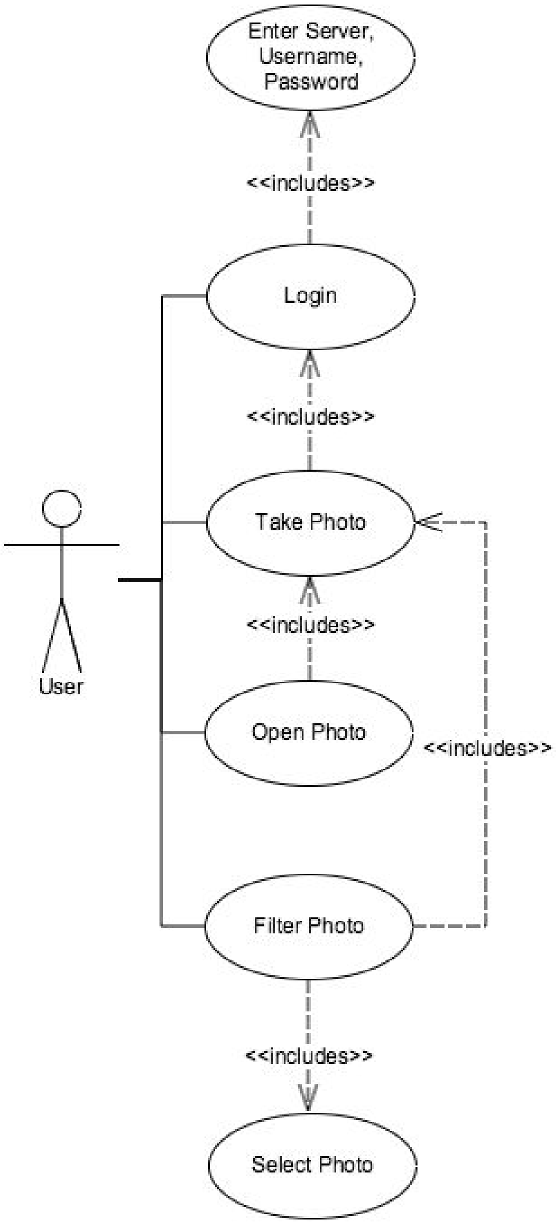
There will be the next options:

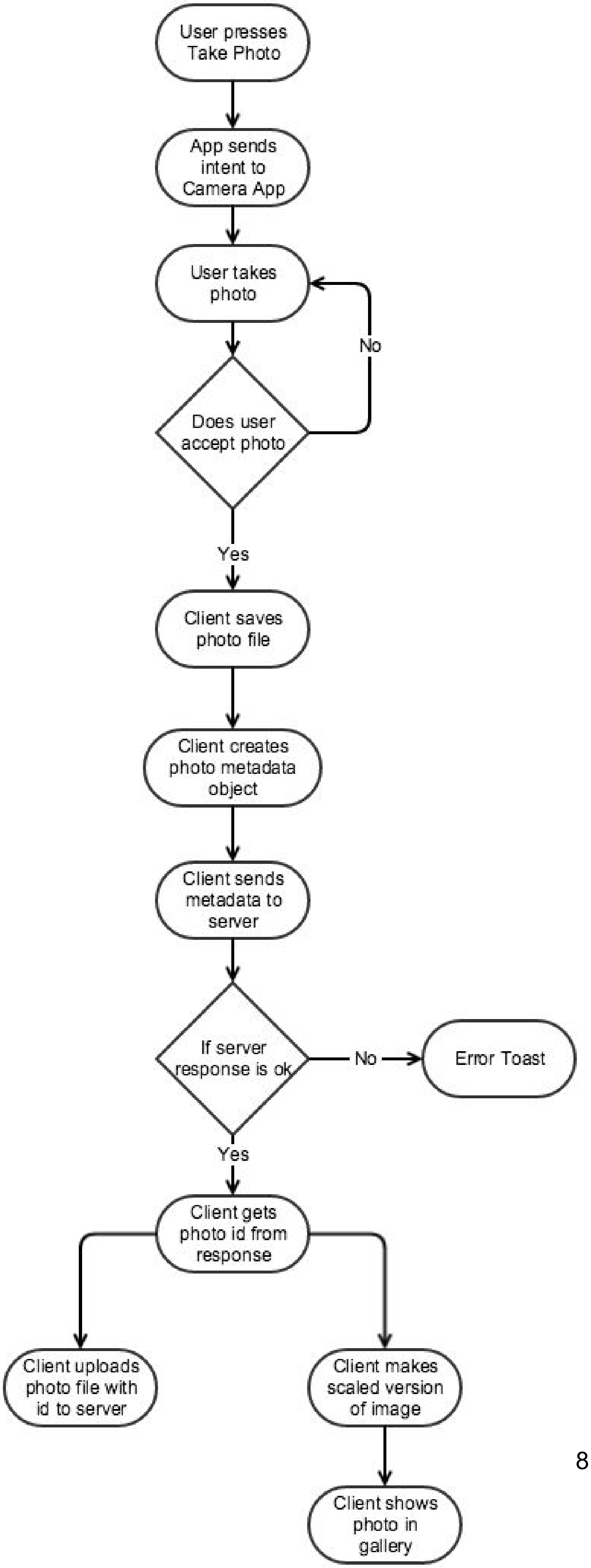
1) Default server, username, password parameters
2) Reminder time
3) Server synchronization options

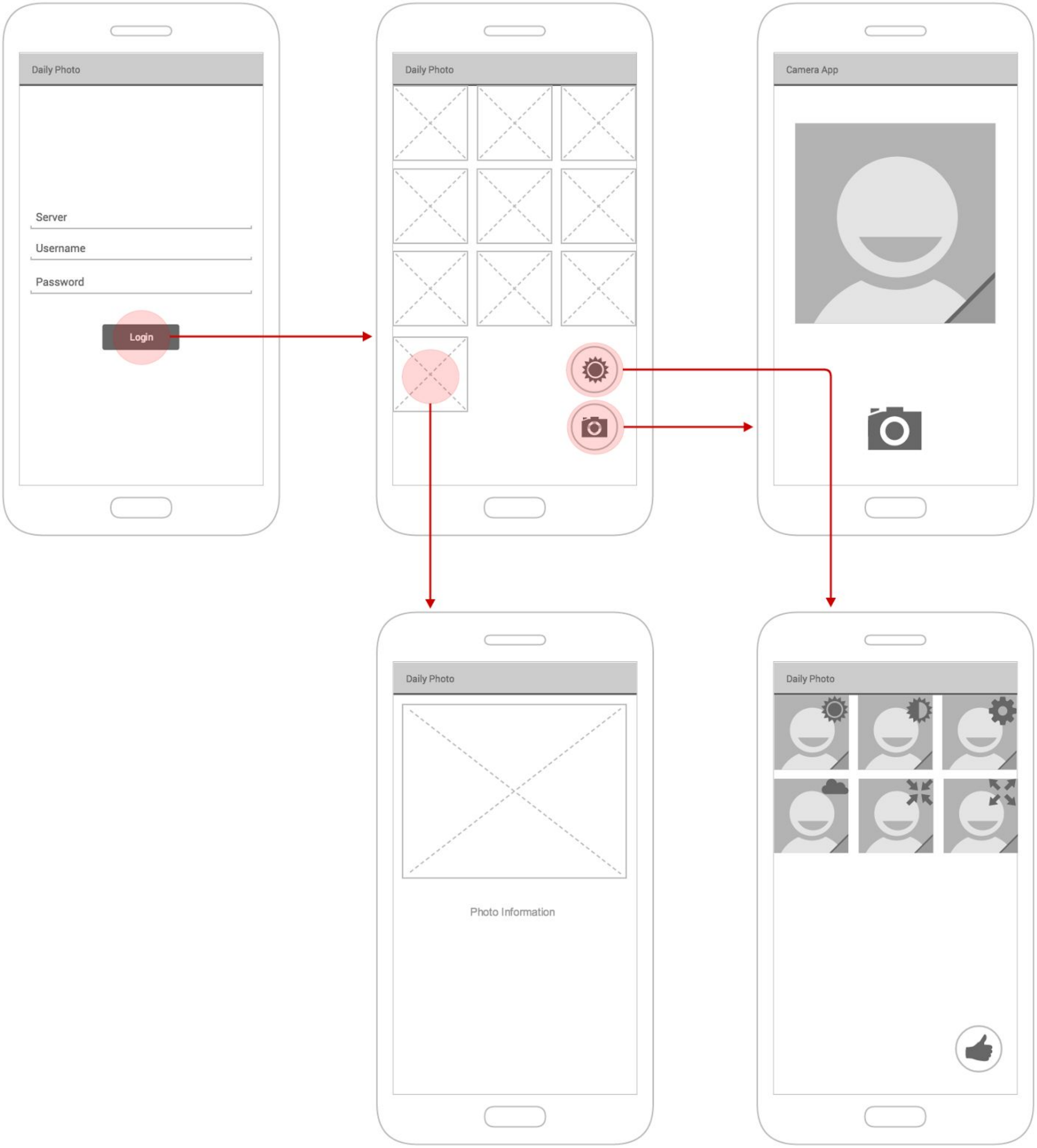
Additional materials

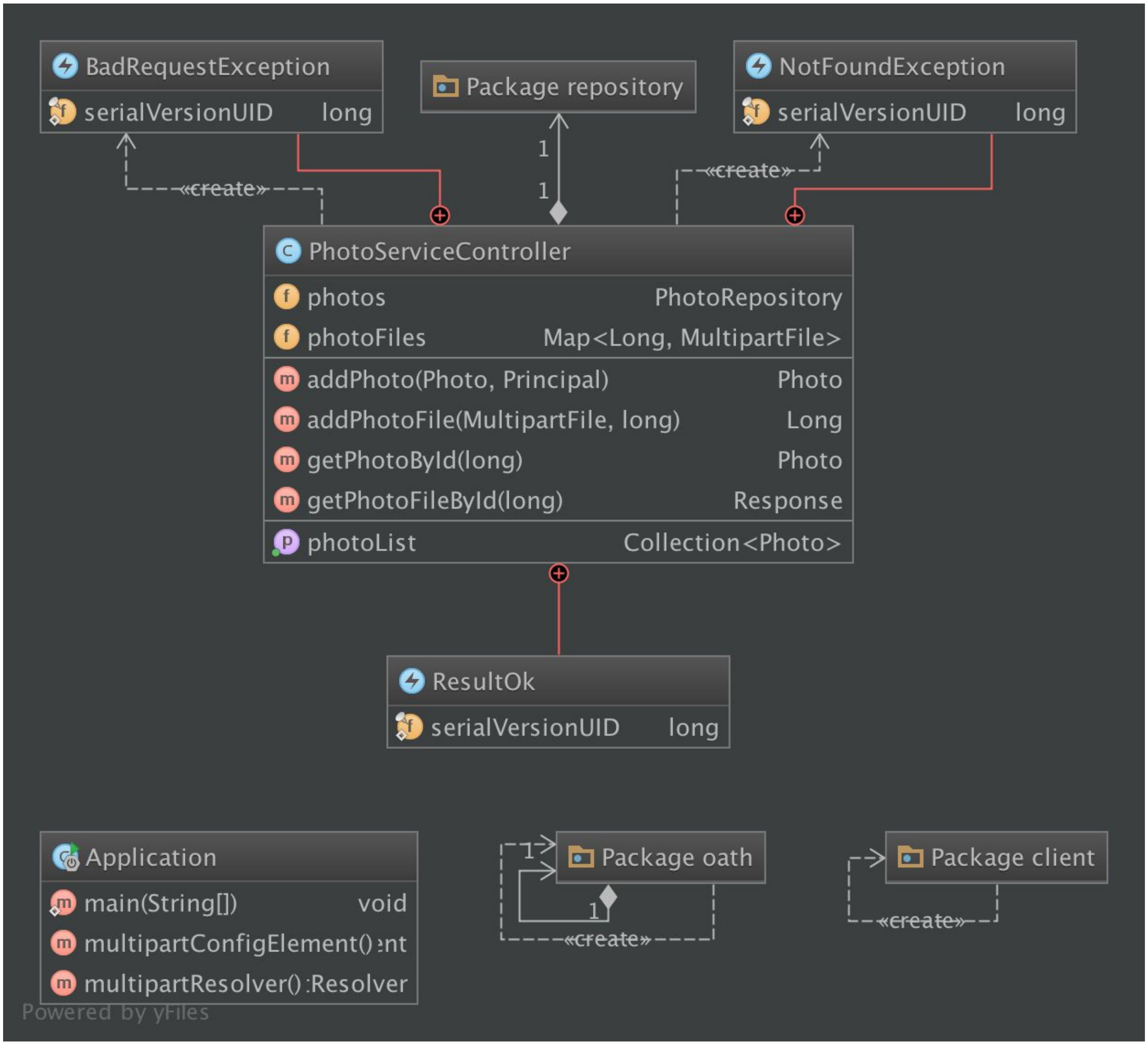
There are several diagrams that show application structure and workflow. UML class diagram was created automatically in IntelliJ based on my existing project. A picture with blue background is Trello board created on trello.com. And all other diagrams were created in gliffy.com.

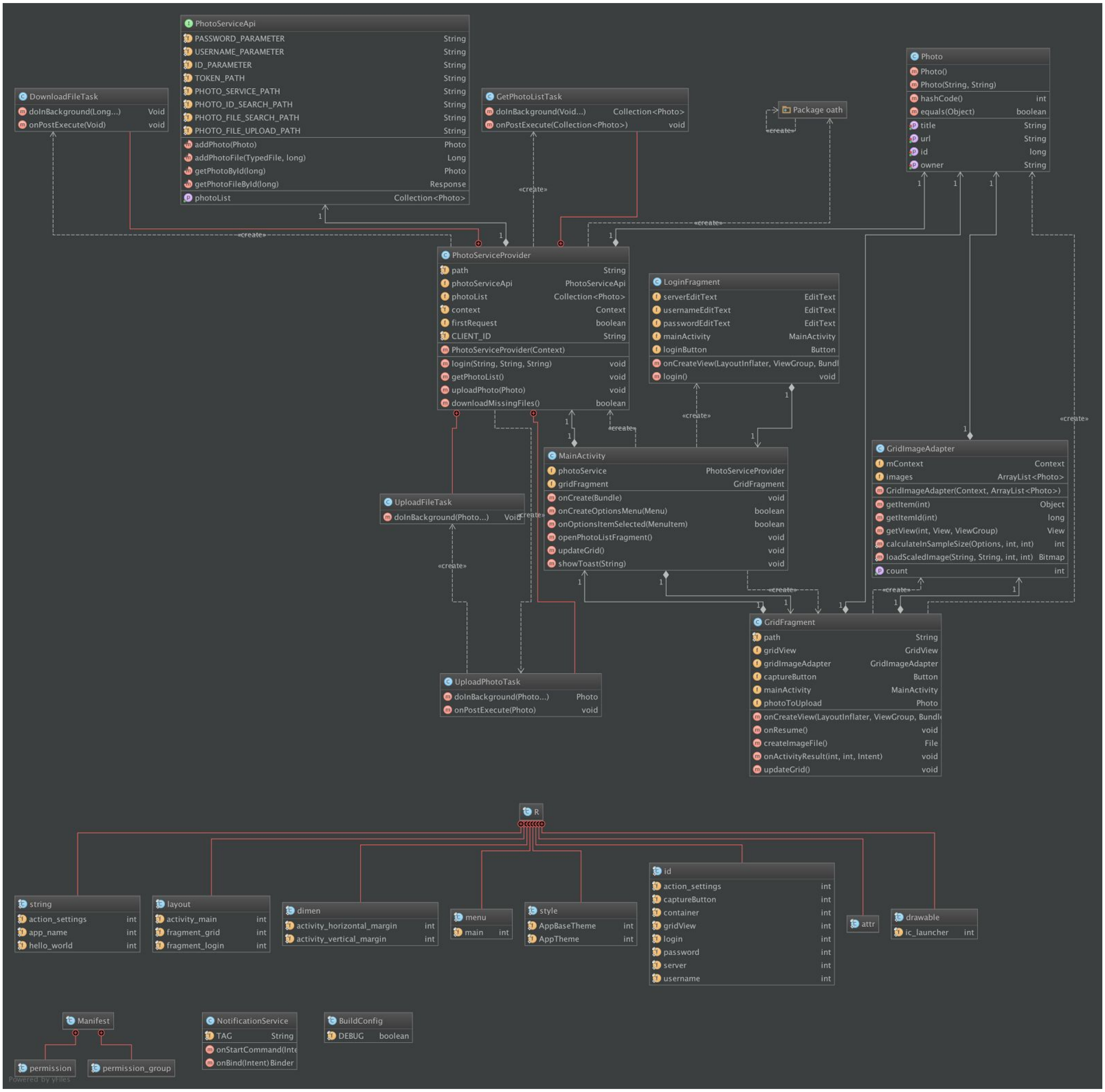



Enter Server, Username, Password
<<includes>>
Login
<<includes>>
Take Photo
<<includes>>
User
Open Photo
<<includes>>
Filter Photo
<<includes>>
Select Photo


User presses Take Photo
App sends intent to Camera App
User takes photo
No
Does user accept photo
Yes
Client saves photo file
Client creates photo metadata object
Client sends metadata to server
If server response is ok
No
Error Toast
Yes
Client gets photo id from response
Client uploads photo file with id to server
Client makes scaled version of image
Client shows photo in gallery


Daily Photo
Server
Username
Password
Login
Daily Photo
Camera App
Daily Photo
Photo Information
Daily Photo


BadRequestException
serialVersionUID long
Package repository
NotFoundException
serialVersionUID long
«create»
1
1
«create»
PhotoServiceController
photos PhotoRepository
photoFiles Map<Long, MultipartFile>
addPhoto(Photo, Principal) Photo
addPhotoFile(MultipartFile, long) Long
getPhotoById(long) Photo
getPhotoFileById(long) Response
photoList Collection<Photo>
ResultOk
serialVersionUID long
Application
main(String[]) void
multipartConfigElement() :nt
multipartResolver() :Resolver
Package oath
1
1
«create»
Package client
«create»
Powered by yFiles


PhotoServiceApi
PASSWORD_PARAMETER String
USERNAME_PARAMETER String
ID_PARAMETER String
TOKEN_PATH String
PHOTO_SERVICE_PATH String
PHOTO_ID_SEARCH_PATH String
PHOTO_FILE_SEARCH_PATH String
PHOTO_FILE_UPLOAD_PATH String
addPhoto(Photo) Photo
addPhotoFile(TypedFile, long) Long
getPhotoById(long) Photo
getPhotoFileById(long) Response
photoList Collection<Photo>
DownloadFileTask
doInBackground(Long...) Void
onPostExecute(Void) void
GetPhotoListTask
doInBackground(Void...) Collection<Photo>
onPostExecute(Collection<Photo>) void
Package oath
Photo
Photo()
Photo(String, String)
hashCode() int
equals(Object) boolean
title String
url String
id long
owner String
«create»
1
PhotoServiceProvider
path String
photoServiceApi PhotoServiceApi
photoList Collection<Photo>
context Context
firstRequest boolean
CLIENT_ID String
PhotoServiceProvider(Context)
login(String, String, String) void
getPhotoList() void
uploadPhoto(Photo) void
downloadMissingFiles() boolean
LoginFragment
serverEditText EditText
usernameEditText EditText
passwordEditText EditText
mainActivity MainActivity
loginButton Button
onCreateView(LayoutInflater, ViewGroup, Bundl
login() void
MainActivity
photoService PhotoServiceProvider
gridFragment GridFragment
onCreate(Bundle) void
onCreateOptionsMenu(Menu) boolean
onOptionsItemSelected(MenuItem) boolean
openPhotoListFragment() void
updateGrid() void
showToast(String) void
GridImageAdapter
mContext Context
images ArrayList<Photo>
GridImageAdapter(Context, ArrayList<Photo>)
getItem(int) Object
getItemId(int) long
getView(int, View, ViewGroup) View
calculateInSampleSize(Options, int, int) int
loadScaledImage(String, String, int, int) Bitmap
count int
UploadFileTask
doInBackground(Photo...) Void
GridFragment
path String
gridView GridView
gridImageAdapter GridImageAdapter
captureButton Button
mainActivity MainActivity
photoToUpload Photo
onCreateView(LayoutInflater, ViewGroup, Bundl
onResume() void
createImageFile() File
onActivityResult(int, int, Intent) void
updateGrid() void
UploadPhotoTask
doInBackground(Photo...) Photo
onPostExecute(Photo) void
R
string
action_settings int
app_name int
hello_world int
layout
activity_main int
fragment_grid int
fragment_login int
dimen
activity_horizontal_margin int
activity_vertical_margin int
menu
main int
style
AppBaseTheme int
AppTheme int
id
action_settings int
captureButton int
container int
gridView int
login int
password int
server int
username int
attr
drawable
ic_launcher int
Manifest
permission
permission_group
NotificationService
TAG String
onStartCommand(Inte
onBind(Intent) Binder
BuildConfig
DEBUG boolean
Powered by yFiles