Clio

(Initially named DailyShot)

Members:

Samuel Paz Mendes

Pedro Henrique Ulmi

Wesley Rafael Dos Santos Sousa

Clio is a new social media app currently implemented only for android. It allows users to share pictures with whomever is using the app. Our initial idea was to allow users to post only a single picture per day, but after getting some feedback from close friends, we changed our minds and now the user can post as many pictures as they want. In addition, like all social media, we can earn some money over advertisement. We are planning to implement ads in a less intrusive way than nowadays' other social media apps, keeping track of how many times users see ads and receiving some feedback from them.

- Application Structure

The back-end consists in a server that saves the user information and all the pictures taken and sent to the server by the users. The front-end is fairly simple. We have an activity for login and another one for the user interactions. The latter is based on an library we found on github called MaterialTabs. With that library, we can have a single activity and handle all the user interactions in 3 different fragments (one for the album, one for the home screen and the other one for the camera). This implementation was easier, because we didn't have to handle changes between different activities (except for the activity for login). However, this has a downside. The application was doing too much work in its main thread to process all the bitmaps and the server connections. Even after creating new threads, the app's performance dropped a bit.

- Design

We have decided to make a friendly user interface that is based on tabs. The tabs are the following: album, home and camera. Users can easily navigate through them by using simple touch gestures.

The camera is opened within a fragment inside the camera tab. This lets the user have a better experience using the application because it's not necessary to launch a new camera activity.

The album photo quality was reduced in order to keep the application smooth when it is on the main screen. Another design choice we've taken in order to improve the app performance was to keep only the neighbour tabs alive.

During the development of the application we had to choose which kind of structure we would use to display the pictures on the home screen. We decided to use a swipe layout instead of a recyclerView of custom cards because the users might not rate the pictures any longer or be influenced to change their votes based on the following pictures due to the ability to go back to older ones.

The login screen was the last feature that we added to the application, it is still not completely functional, but it gives another perspective to the app.

The logo was developed in order to be versatile and to be able to use in many different contexts and scenarios. For example, on the picture on the login screen and on the application top logo. It can also be easily used on a animation due to its rounded design.

- ## Design Decisions

We decided to create an library for all the server-side operations. We called this library "serverlib" and it also contains classes used by both server and application, for example: "Message" and "MessageType".

The library serverlib consist of two folders: server (all the classes used exclusively by the server) and shared (all the classes used both by the server and application).

We decided that in the application side all the support classes for the connection must be located in the folder “connection”.

- ## Implementation Difficulties

As already mentioned, our app’s performance dropped a bit because we were doing almost everything in one activity. This happened because of all the bitmap rendering in the fragments (fragments are kinda slow too) and also because of the time that the app takes to connect to the server. We had to make some changes in order to increase performance. First, we created new threads to handle tasks properly. Then, we limited the album fragment to show only the last six pictures taken by the user. We also had to decrease the quality of the pictures shown on the cards of the home screen fragment and after that we noticed the smoothness of the cards increased.

We also struggled to integrate the server and the application, especially when trying to debug the app and find glitches. Another server-related issue was that Android requires that all the connection (with the server) in the application side be done in another thread different than the main thread. That implied in some difficulties because we wanted to change a view using the connection thread (that is inside the application side), but that is impossible because Android only allow the main thread to do this type of operation in the view. We had to create some type of synchronization between the connection thread and the main thread, and this synchronization made the application slower.

- External Sources:

https://github.com/siyamed/android-shape-imageview (rounded imageView in the profile)

https://github.com/neokree/MaterialTabs (main activity tabs)

https://github.com/Diolor/Swipecards (cards to show pictures in home screen)

http://mrbool.com/implementing-gridview-in-android/27725 (tutorial for album GridView)

http://stackoverflow.com/questions/21833181/arrayadapter-text-and-image (BaseAdapter)