- Description
- Intended User
- Features
- Restrictions
- User Interface Mocks
  - Screen 1 - Browsing new photos
  - Screen 2 - Browse favorite photos
  - Screen 3 - Browsing own photos
  - Screen 4 - Photo detail
  - Screen 5 - Photo Upload (Add Photo Activity)
  - Widget
- Key Considerations
  - Persisting Data
  - Used Libraries
- Required Tasks
  - Project Setup
  - Create Database and Content Provider
  - REST Client Interface
  - Local Account and SyncAdapter
  - Implement UI for Each Activity and Fragment
  - OAuth User Authentication
  - Add or edit a Photo
  - New Photos Widget
  - Analytics implementation
- Optional Tasks
  - Swipe Refresh for New Photos
  - Location Picker for a photo without location
  - Notification for new Photos

GitHub Username: h3u

YAUC - Your Android Unsplash Client

Description

Your Android Unsplash Client assists your interaction with https://unsplash.com/ (Unsplash | High Resolution Photos) on Android.
Like Unsplashs web application, YAUC is similar and simple to use. For the first version it's limited to browse and upload photos using the REST API of unsplash.

Intended User

This app mainly targets all registered users of unsplash that use an android device (ideally with a quite good camera).

Features

- Browsing of new, favorite and own photos organized in tabs
- Detail view with share and like actions
- Detail information dialog with data about photo and location
- Upload of own photos

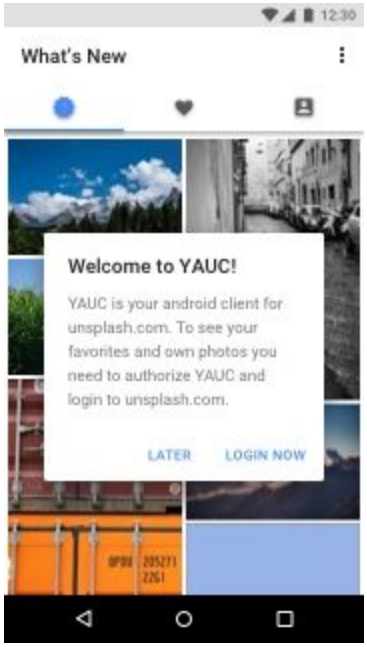
Restrictions

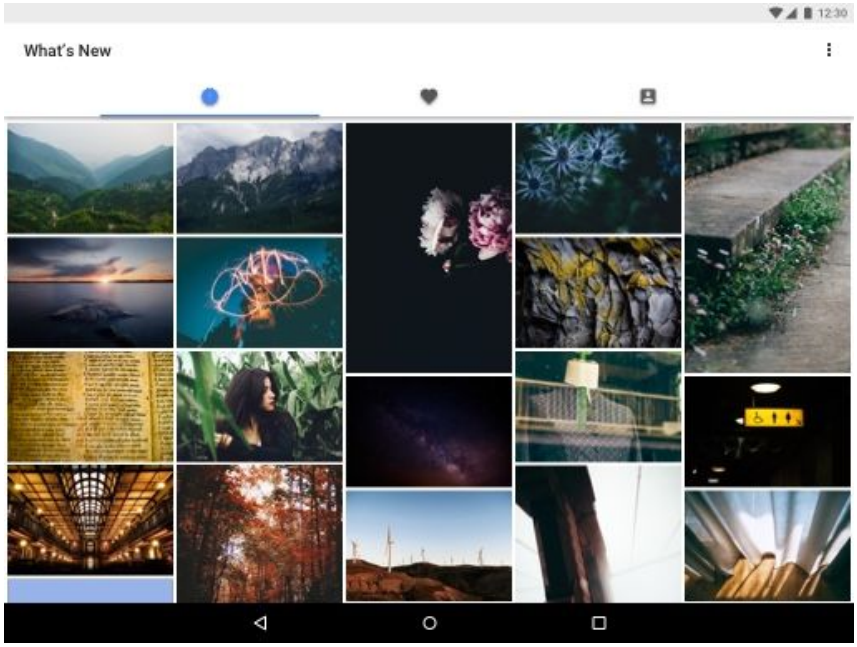
Unauthenticated users are not able to upload a photo, have an empty tab for favorites and also for own photos. Unsplash has the rule to limit photo uploads for a period. Users can upload 10 photos within 10 days. This limit also exist when using the API / YAUC.
In some cases the user can be faced with a rate limiting for the API usage.

User Interface Mocks

Screen 1 - Browsing new photos

This is the activity that will be launched at start and displays a staggered grid of new photos (organized in tabs with favorite and own photos) which can be scrolled vertically with a collapsing tool bar.





Tapping one image tile starts the detail activity (see below).

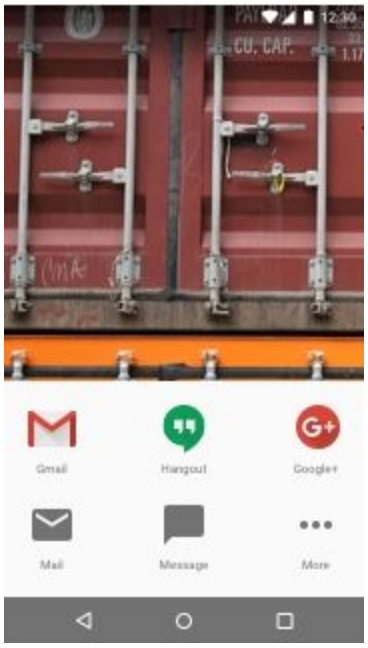
Screen 2 - Browse favorite photos

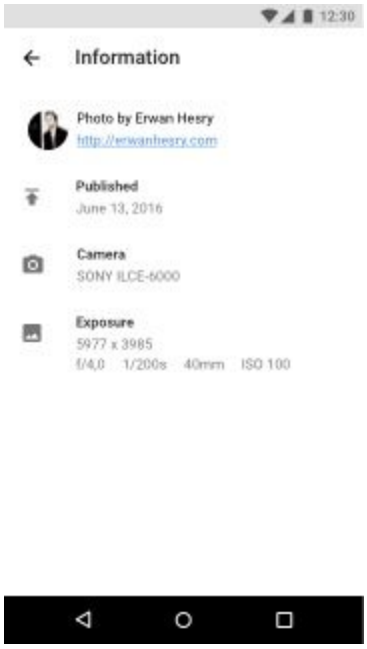
Favorite photos will be displayed in a list with full width to be more immersive displayed.
Maybe this can be irritating to the user when swiping to new or own photos. If this is the case a staggered grid will be the fallback.

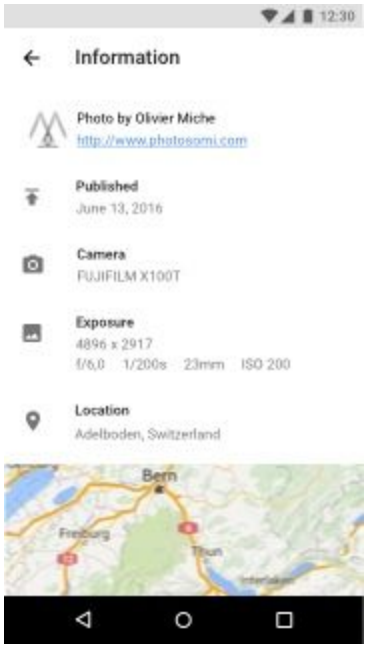
Screen 3 - Browsing own photos

A staggered grid is used to show the users own photos. A floating action button opens a bottom sheet to pick a picture from the gallery or to take a photo.

Screen 4 - Photo detail

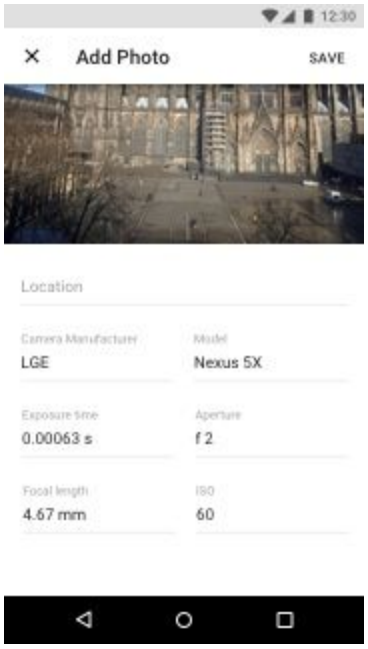






The detail view has transparent bars and actions for sharing, like and open additional informations for the picture.

Screen 5 - Photo Upload (Add Photo Activity)



After taking or choosing a picture when the user comes back to the app, a full screen dialog is shown. Most of the text fields should be pre-filled from exif data of image.

Widget

App offers a widget that shows number of unseen photos since last app visit.

Key Considerations

Persisting Data

All photo data will be stored in a database (SQLite) using a Content Provider for abstraction, encapsulation and synchronisation. Authentication with OAuth (Access-/Refresh-Token) can be stored in *SharedPreferences*.

Used Libraries

Schematic - for building the Content provider to write less code and generate more.

Retrofit - for encapsulation of the whole HTTP traffic
To simplify the REST API usage with an interface and annotations in a comfortable manner Retrofit is my choice.

Glide - for efficient image loading, caching and display
Good experiences with Glide when implementing Popular Movies let me choose this lib.

Butterknife - to bind view elements with annotations

Android Support Library - for backward compatibility and some components

- AppCompat
- Design
- RecyclerView

Google Analytics - to know what users use and do.

Google Maps - to display the photo location.

Required Tasks

Project Setup

- Create new empty project
- initialize git and its remote on Github
- add all dependencies to gradle that are identified so far

Create Database and Content Provider

With the help of schematic create the database and column classes and the provider with tables for Photos, Users and Favorites (User <-> Photos).

REST Client Interface

A service interface and the response objects are needed.
Following endpoints of api.unpslash.com will be used:

- GET /photos
- GET /photos/:id
- POST /photos
- PUT /photos/:id
- GET /me
- GET /users/:username/likes
- POST /photos/:id/like
- DELETE /photos/:id/like

Local Account and SyncAdapter

Create a local *Account* using *AccountManager* and build a *SyncAdapter*. New Photos should be fetched periodically once a day or immediately when requested (e.g. with swipe refresh). If the oauth token needs to be refreshed, the synchronisation must be done accordingly.

Implement UI for Each Activity and Fragment

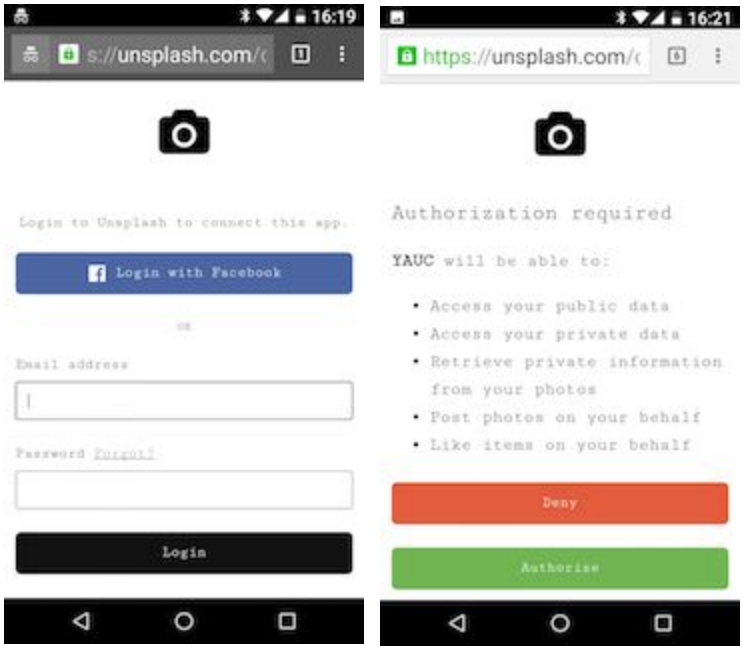
Following Activities and Fragments are planned:

- BrowseActivity (Main) with Fragments for
    - New Pictures (Screen 1)
    - Favorites (Screen 2)
    - My Photos (Screen 3)
- DetailActivity
- InformationActivity (Screen 4)
- Dialog to add/edit photo (Screen 5)

After building this add a transition from Browse to Detail Activity when tapping on one image.

OAuth User Authentication

Add an implicit intent that leads the user to unsplash to authorize the YAUC app.
After this request an access and refresh token and store them in *SharedPreferences*.
Add or enhance a request interceptor to populate the header with authorization with the given access token.
After that find out the lifetime of the access token and if needed add a refresh of auth token to the update in SyncAdapter and adapt the periodic updates interval.



Add or edit a Photo

First there must be built a bottom sheet with intents for taking or picking a photo. When these actions come back to the *Activity* it should start a *Dialog* that displays the photo and its meta data in *EditText* fields mostly pre-filled. Saving this performs insert/update to the database and the upload to unsplash in a background task. The User comes back where he started (My

Photos) and the view will be updated when the upload has been done (a *Snackbar* or *Toast* can inform about this).

New Photos Widget

The widget has a height of one row and minimum size of 2 columns and can be expanded to a maximum of 4 columns. Create Provider, IntentService and Widget Layout. Widget displays the number of inserted images in last synchronization.

Analytics implementation

Configure screens, session and userId tracking and define and implement events.

Optional Tasks

These tasks can be built when there will be time left before due date.

Swipe Refresh for New Photos

With a pull down on tab of new photos a sync should be started.

Location Picker for a photo without location

Instead of entering a location add a picker/live search when adding adding/editing a photo.

Notification for new Photos

After an automatic synchronization a notification can display number of new photos.