Potlatch Application

Final Submission

for Coursera Capstone project

November 30th 2014

Related Documentation:

- Related links and information: https://sites.google.com/site/potlatch4u/
- Source Code: https://github.com/potlatch4u/potlatch4u
- demo video: on youtube

Thank you for your time to read and evaluate this project !

1. Basic Project Requirement:

App supports multiple users via individual user accounts

User have to be registered with an OpenId-provider, initially will be supported:

- Google accounts
- Facebook accounts

Related artifacts: LoginActivity

2. Basic Project Requirement:

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

The following actions require authorization. If an unauthorized user chooses such an action, a login pop-up will appear. The actions are:

- add a gift/gift chain
- delete a (formerly added) gift
- mark/unmark a gift as touching

The buttons for “delete” and “mark as touching” will not be shown if the user is not authenticated. Instead a login button will be shown.

3. Basic Project Requirement:

App comprises at least 1 instance of each of at least 2 of the following 4 fundamental Android components: Activity, BroadcastReceiver, Service, ContentProvider

Currently there are 8 activities:

- GiftChainListActivity
- GiftListActivity
- TopGiftListActivity
- GiftViewActivity
- GiftChainViewActivity
- CreateGiftActivity
- SettingsActivity
- LoginActivity

and 2 services:

- PhotoUploadService
  for uploading of the pictures taken into the cloud
- RefreshCacheService
  to regulary (according to preference) download touched-counts

Activities and services are part of the *potlatch4u-Eclipse-project*.

4. Basic Project Requirement:

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

The backend application is implemented as a Spring-Boot-Application (*potlatchServer-Eclipse-project*) providing:

- REST-based services to add/search for/delete/update gifts and gift chains
  (implemented in class PotlatchSvc)
- authorization will take place using Springs OAuth2 support and OpenId

- images will later be stored on GoogleDrive (transparently for the user)

5. Basic Project Requirement:

App interacts over the network via HTTP

The Android-App will interact with the back end through the RefreshCacheService and the PotlatchSvsApi.

6. Basic Project Requirement:

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

All 7 activities (see 3. above) will use a different user interface screen. The navigation is shown in the navigation overview.

7. Basic Project Requirement:

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.

When posting a gift the user is given the option to take a photograph at that moment. This is part of the CreateGiftActivity.

8. Basic Project Requirement:

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

The PhotoUploadService (see 3.) will be performed in the background using a Thread pool.
The RefreshCacheService performs periodical sync tasks using the HaMeR-framework (Handlers/Messages/Runnables) with at 1 thread running off the UI thread.
Pictures of gifts from other users will be downloaded using the DownloadService in a Thread pool.

1. Functional Description and App Requirement:

App defines a Gift as a unit of data containing an image, a title, and optional accompanying text.

When a gift is created 3 pieces of information can be given (see Application-Screens.pdf):

- Title (mandatory)
- description (optional)
- image (mandatory, either select from android filesystem (e.g. gallery) or take a photograph)

On the Android device gift data is stored in the GiftData class, and on the server it it persisted using the class GiftBean.

2. Functional Description and App Requirement:

A User can create a Gift by taking a picture (or optionally by selecting an image already stored on the device), entering a title, and optionally typing in accompanying text.

This is the basic functionality of the CreateGiftActivity if initiated through the “Add” button (on the GiftListActivity or GiftChainListActivity).

3a. Functional Description and App Requirement:

Once the Gift is complete the User can post the Gift to a Gift Chain (which is one or more related Gifts).

When a gift is added from the `GiftChainViewActivity` the gift is added to the current gift chain.

3b. Functional Description and App Requirement:

Gift data is stored to and retrieved from a web-based service accessible in the cloud.

All gift and gift chain data is retrieved from a REST-based Spring-Application and cached locally on the device. All cached gifts are regularly refreshed (their touched-counts) according to the preference Touched-Counts-Update-Interval.
This Application will be hosted on Amazon EC2. The images will be stored on Google Drive.

3c. Functional Description and App Requirement:

The post operation used to store Gift data requires an authenticated user account

Upon opening the CreateGiftActivity it is checked wheather or not the user is authenticated. If not the LoginAcitivity is open for authentication, and only if successful the CreateGiftActivity is opened.
The Delete-button in the GiftViewActivity is only shown if the user is authenticated, the same is true for the “I'm touched”-button.

4. Functional Description and App Requirement:

Users can view Gifts that have been posted.

Even not authenticated users will see gifts and are able to browse through gift chains and gifts. However, to be touched they have to log in first.

5a. Functional Description and App Requirement:

Users can do text searches for Gifts performed only on the Gift's title.

The gift search is implemented by title. While typing in the Textbox in the GiftListActivity or GiftChainListActivity the cached gifts are shown/hidden as the user types. Upon pressing the Search button a search is performed on the server and a cache refresh is initiated.

5b. Functional Description and App Requirement:

Gifts matching the search criterion are returned for user viewing.

Search will open the GiftListView with search results. From there users can open individual gifts.

6a. Functional Description and App Requirement:

Users can indicate that they were touched by a Gift, at most once per Gift (i.e., double touching is not allowed)

To be touched by a gift

- the user needs to be logged in
- the gift has not been touched by this user yet

To remove a being-touched-mark (which decreases the touch count)

- the user needs to be logged in
- the user has marked this gift as touching and has not removed this mark yet.

If these conditions are not met the “I'm Touched”-buttons changes to “I'm not touched” once the gift was marked as touching by the user in the GiftViewActivity, and vice versa. So the user can only toggle its status. This is only possible if the user is logged in, otherwise this button is invisible.
The information what user marked what gifts as touched is stored on the server.

6b. Functional Description and App Requirement:

Users can flag Gifts as being obscene or inappropriate. Users can set a preference that prevents the display of Gifts flagged as obscene or inappropriate.

This preference can be changed in the SettingsActivity.

7a. Functional Description and App Requirement:

Touched counts are displayed with each Gift

This is part of the

- GiftListActivity
- GiftChainListActivity
- GiftViewActivity
- TopGiftListActivity

The touch counts are read-only.

7b. Functional Description and App Requirement:

Touched counts can be periodically updated in accordance with a user-specified preference (e.g., Touched counts are updated every 1, 5 or 60 minutes) or updated via push notifications for continuous updates.

This setting is realized as a ListPreference in the SettingsActivity.

8. Functional Description and App Requirement:

App can display information about the top "Gift givers", i.e., those whose Gifts have touched the most people.

This implemented in the TopGiftListActivity.