Potlatch

Android Capstone Project
(source review)

author: Sergeev Ilya
master.hensh@gmail.com
Russia, Tver
november 2014

Description

https://class.coursera.org/androidcapstone-001/wiki/Potlatch

Sources

Server side:
https://github.com/IlyaSergeev/Potlach_server

Client side for Android:
https://github.com/IlyaSergeev/Potlach_client

Basic Project Requirements

- App supports multiple users via individual user accounts
  User info stored in server database as login and password.
  Auth end point {SERVER_URL}/oauth/token
  All requests must send by authorized user
  Server classes **UserInfo**, **UserInfoRepository**, **CustomUserDetailsService**
  Client classes **UserInfo**, **LoginActivity**

- App contains at least one user facing function available only to authenticated users
  There are 3 roles in system: ADMIN, USER, NOT_USER
  System take role on authorization in **CustomUserDetailsService**
  See **@PreAuthorize** annotation in classes **GiftController**, **TouchController**, **UserController**, **VoteController**

- App comprises at least 1 instance of each of at least 2 of the following 4 fundamental Android components: Activity, BroadcastReceiver, Service, ContentProvider
  Activities: **CreateGiftActivity**, **LoginActivity**, **MainActivity**
  Service: **TasksMakerService**
  BroadcastReceivers can see in classes: **ListOfGiftsFragment**, **MainActivity** and **SearchFragment**

- App interacts with at least one remotely-hosted Java Spring-based service
- App interacts over the network via HTTP
  Android client send request to server via HTTP. All settings are in **ApplicationSettings**

See also class **ServerSvc** in client

- App allows users to navigate between 3 or more user interface screens at runtime
  See all screen in Design doc or in video and layout folder

- 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.
  Client use photo camera capture
  Classe **DialogHelper**

- App supports at least one operation that is performed off the UI Thread in one or more background Threads of Thread pool.
  All requests to server send in background thread using **AsynkTask**
  Service **TasksMakerService** make operations in background thread
  Images download in background thread using **ExecutorService**

Functional Description and App Requirement

- App defines a Gift as a unit of data containing an image, a title, and optional accompanying text.
  See structure of class **Gift**
  Also **GiftRepository** and **GiftController**

- 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.
  See classes **Gift** and **CreateGiftActivity**

- Once the Gift is complete the User can post the Gift to a Gift Chain (which is one or more related Gifts).
  See classes **Gift** and **CreateGiftActivity**

- Gift data is stored to and retrieved from a web-based service accessible in the cloud.
  Classes in server sources **Gift**, **GiftRepository** and **GiftController**

- The post operation used to store Gift data requires an authenticated user account
  Only authorized users can send requests to server
  Classes in server **Application**, **OAuth2SecurityConfiguration**, **ClientAndUserDetailsService**, **CustomUserDetailsService** and **@PreAuthorize** annotation in controllers

- Users can view Gifts that have been posted.
  See classes in client **ListOfGiftsFragment**, **MyGiftsFragment**, **NewGiftFragment**, **SearchFragment** and **UserGiftsFragment**

- Users can do text searches for Gifts performed only on the Gift's title.
  See class in client **SearchFragment**
  See interface in server **GiftRepository**

- Gifts matching the search criterion are returned for user viewing.
  See class in client **SearchFragment**
  See interface in server **GiftRepository**

- Users can indicate that they were touched by a Gift, at most once per Gift (i.e., double touching is not allowed)
  Indicator «NEW» show if user not touch gift. See Design doc.
  See in server source **TouchController**

- 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.
  User can set like or dislike for every gift
  See classes in server **Vote**, **VoteRepository**, **VoteController**
  See class in client **GiftsAdapter**

- Touched counts are displayed with each Gift
  See in Design doc
  Class in client **GiftsAdapter**

- 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 functional is not useful for my opinion. I'm skip it because all information about touches refresh on touch happened.

- App can display information about the top "Gift givers," i.e., those whose Gifts have touched the most people.
  See class in client **TopRateFragment**
  See classes in server **UserInfoRepository**

Packages overview

Server

---

com.ilya.sergeev.potlach

Contains server application class with all settings.
Contains all spring controllers

---

com.ilya.sergeev.potlach.auth

Contain authorization configuration and classes for authorization.

---

com.ilya.sergeev.potlach.client

Contains all entities and API interfaces.

---

com.ilya.sergeev.potlach.repository

Contains repositories for entities

---

com.ilya.sergeev.potlach.test

Tests for server side.

Client

com.ilya.sergeev.potlach

Contains classes for user interface (screens).

com.ilya.sergeev.potlach.client

Entities for server requests

com.ilya.sergeev.potlach.image_loader

Logic for downloading and caching images from server

com.ilya.sergeev.potlach.mock

Generator for tests

com.ilya.sergeev.potlach.unsafe

Containg EasyHttpClient for server connection.