Catstagram

Simple Cat version of Instagram in Ruby on Rails - for Webrails class at HEIG-VD

Provisional Report

Context and Authors

This is a Ruby on Rails project for WebRails class at HEIG-VD, Yverdon-les-Bains, Switzerland.

Authors:

- Eleonore d'Agostino
- Valentin Minder

General Description

Catstagram is a simple version of Instagram for photos of cats only. It allows registered users to upload photos of cats, tag them and share them with all their friends, because sharing is caring ;3

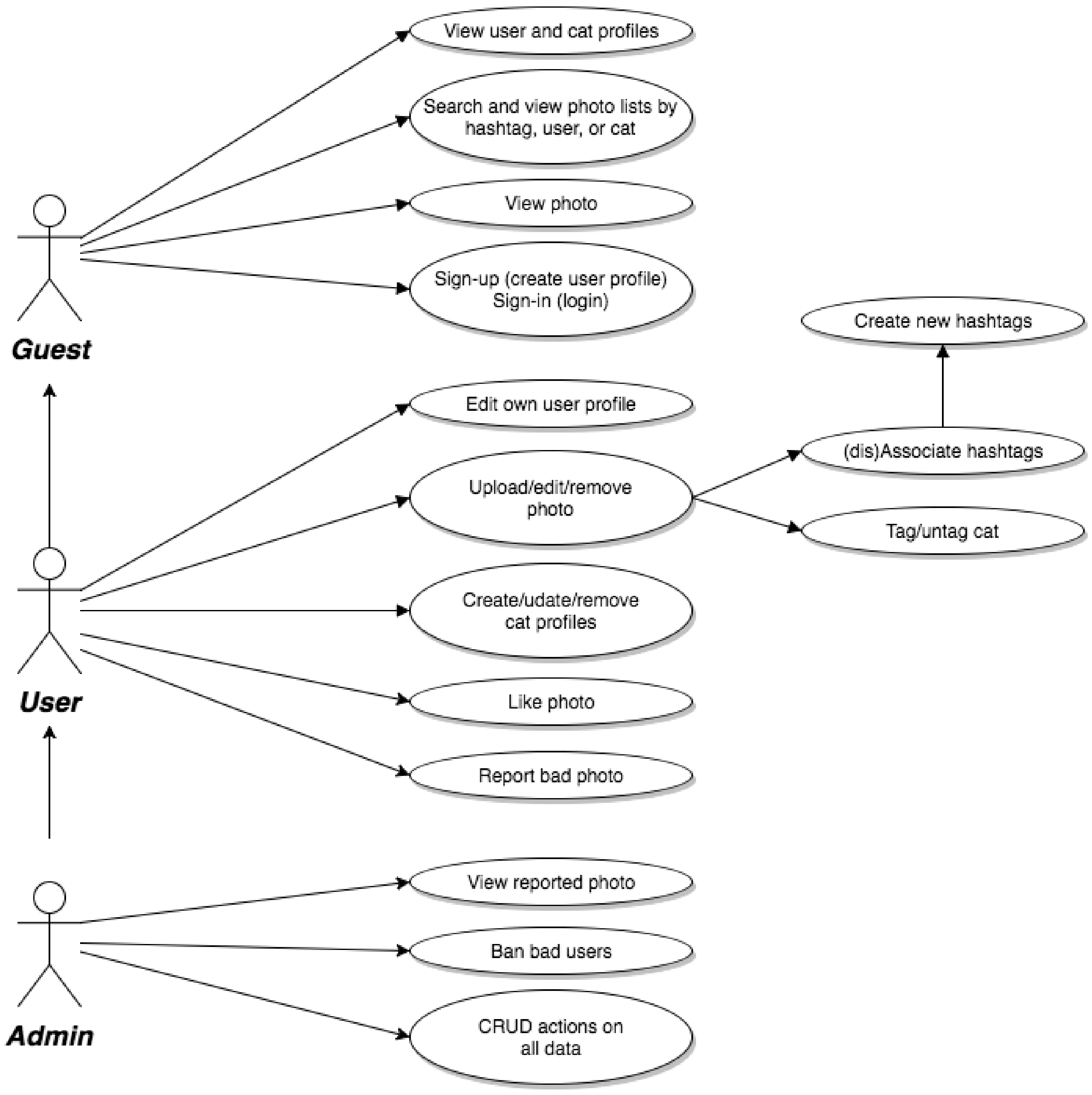
The more you share, the more views you get! :D

Guests can see photos of specific cats or uploaded by specific users! And cats by categories, of course!

Be cautious, if you upload anything other than a cat photo and you get reported too many times, the admins might ban you!

Usecases

See the following image:



A guest is a non-registered or non-logged visitor of the website. They are limited to VIEW capacities on everything, and have no right to EDIT. A guest may:

- view user and cat profile details
- search and view photos by hashtag, user or cat
- view photos (direct link to single photo), increasing the view count of the photo
- create an account with *sign-up*, or login with *sign-in*

A user is a registered and logged visitor of the website who doesn't have admin rights. A user has VIEW capabilites on everything, and EDIT capabilites on their own data. A user may:

- do everything a guest can
- edit their own user profile
- upload or edit photos on their own account. Uploading a photo includes optionally tagging cats and categories (hashtags). This includes creating new hashtags (categories), when those available do not suffice. Of course, they can upload photos of cats that don't belong to them.

- create or update cat profiles (of their own cats only)
- report bad photos, like those with dogs, increasing the report count of the photo
- like photos, increasing the like count of the photo

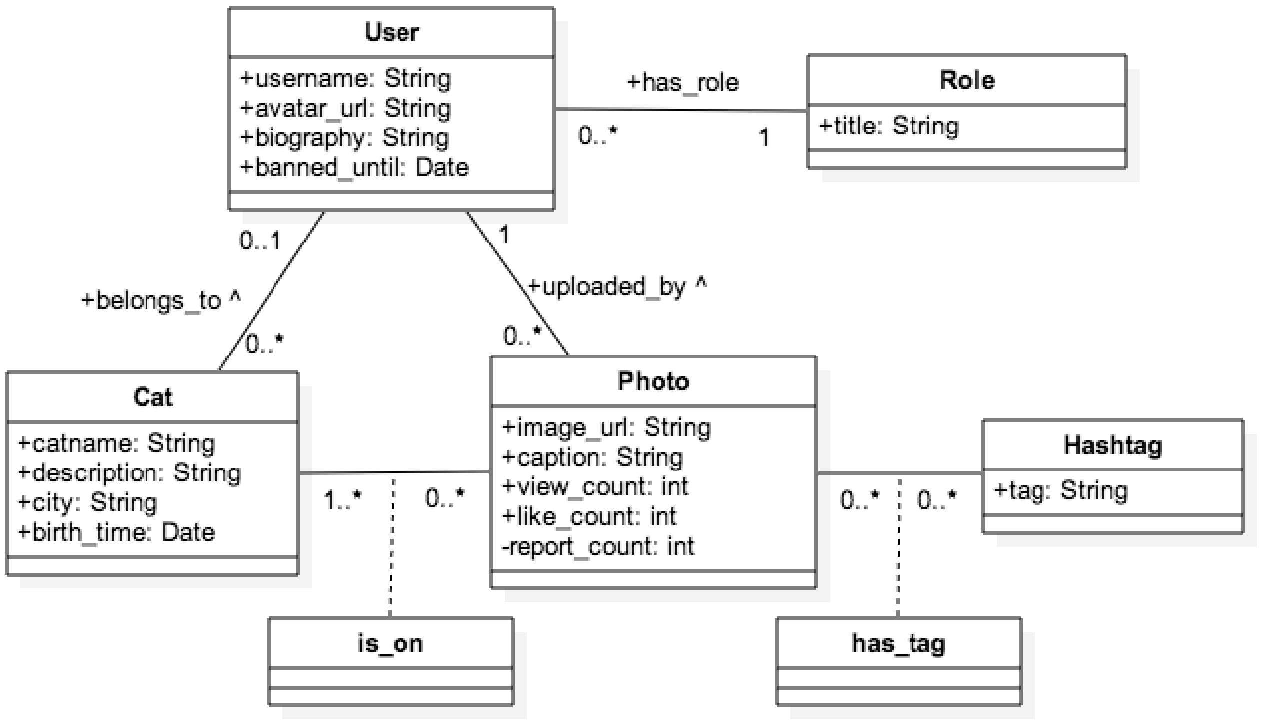
An admin is a registered and logged visitor of the website that has admin rights. An admin has the right to VIEW and EDIT on everything. An admin may:

- do everything a user can
- view the report count of a photo, recieve some kind of alerts of new reports, and reset report counts
- decide what to do with reported photos, including banning users, when they were reported for bad behaviors
- edit everything (CRUD actions), including removing reported photos and associated users, or editing any non-generated field/relation

Note that views, likes and reports counts are not related to users. View counts reflects the number of times a photo was opened in a browser, even if it was done 10x by the same person. The same goes with like and reports, but those are only available for logged users. This system is open to abuse but it allows for a simpler system, otherwise it would require a lot more relations and be more complex.

Database schema

See the following image:



A user is a registered human visitor that may have some role (and admin rights). Fields generated by Devise for account creation and authentication are not shown, as well as fields handled by Ruby by default. Users own cats and upload photos (therefore cats belongs to their user owner, and photos are uploaded by users).

Cats are tagged on photos, and at least one cat must be tagged on every photo. If the cat is unknown, a special entity called the "Chanonyme" (Catnonymous) is tagged instead.

Hashtags are similar to categories, where any photo may belong to any number of hashtags (from 0 to N).

About permissions and rights

The following fields are set once and forever, and cannot be edited, even by an admin:

- id
- created_at
- updated_at

All the counts fields cannot be edited, even by an admin. They are set to 0 by default and incremented automatically when the related actions are performed. However, admins may reset the report count to 0 when handling reports.

Admins may set the banned_until field to ban a user (otherwise this field is left blank). Users cannot connect until the end of the ban period.

In all entities, there is at least one field that is mandatory. All the others are optional (or they have special meaning, explained before)

- username
- title
- catname
- image_url
- tag

Users may add/edit/remove all **other** data:

- in their profile (username, avatar_url, biography)
- in their cats' profile (catname, city, description, birth_time)
- in a photo they uploaded (caption, cats tagged, hashtags tagged). They cannot change the image URL though. If a user wishes to change photos, they have to delete the current one and upload a new one, as likes/hashtags/cats tagged may not be relevant to the new photo.

Users may remove a photo, a profile of a cat they own, or even their whole profile (deleting all data from their photos and cats).

As the admin has CRUD rights on everything, they can do everything a user can on all users' data. Moreover, they have special CRUD rights (in that sense, edit means all CRUD actions)

- They can set the `banned_until` field and reset the `report_count` field
- A cat without an owner is only editable by an admin
- They may assign cats to an owner, or remove ownership

- They may assign roles
- They may edit hashtags

Iterations plan

Week 8-9 (18.04. - 01.05.2016)

- basic structure of whole project (scaffold, db, basic usable layout...)
- no permissions: basic CRUD operations on everything

Week 10-11 (02.05. - 15.05.2016)

- account creation and authentification (with Devise)
- permissions (guest, user, admin) (with CanCanCan)
- basic upload form, with tagging cats and hashtags

Week 12-13-14 (16.05. - 05.06.2016)

- AJAX: autocompletion of hashtags in upload form
- AJAX: update of views/likes count asynchronous
- pretty upload form
- clean and pretty layout (UI)
- report and presentations preparations

Week 15-16 (06.06 - 19.06.2016)

Oral presentations