Τμήμα Μηχ. Η/Υ & Πληροφορικής,
Πανεπιστήμιο Ιωαννίνων, Ιωάννινα

Ζώης Νικόλαος (2103)
Τσιόλκας Μιχαήλ (2152)

ΜΥΕ042 Τεχνολογίες Διαδικτύου

Εργαστηριακή Άσκηση 1

Εφαρμογή ιστού για κοινοχρησία φωτογραφιών στο Ruby-on-Rails

Εισαγωγή

Σκοπός αυτής της εργασίας ήταν να δημιουργήσουμε μια *Web* εφαρμογή χρησιμοποιώντας το framework *Ruby-on-Rails* το οποίο υλοποιεί το *MVC* pattern. Η εφαρμογή επιτρέπει την εγγραφή διάφορων χρηστών που μπορούν να ακολουθούν άλλους χρήστες, το ανέβασμα φωτογραφιών, το σχολιασμό των φωτογραφιών και την προσθήκη tag στις φωτογραφίες.

Βασιζόμενοι στην αρχική έκδοση της εφαρμογής προσθέσαμε επιπλέον λειτουργίες, τροποποιώντας τα ήδη υπάρχοντα στοιχεία είτε προσθέτοντας καινούργια.

Υλοποίηση

1.Προσθήκη τίτλου στις φωτογραφίες

Ζητούμενο:

Κάθε φωτογραφία θα πρέπει να περιέχει ένα τίτλο. Ο χρήστης όταν ανεβάζει μια νέα φωτογραφία προσθέτει και τον τίτλο, διαφορετικά εμφανίζεται μήνυμα ώστε να ξαναπροσπαθήσει. Όταν οι φωτογραφίες εμφανίζονται θα έχουν στο πάνω μέρος και τον τίτλο της.

Αρχικά για την υλοποίηση αυτής της λειτουργίας χρειάζεται να προσθέσουμε στη φόρμα όπου γίνεται η προσθήκη της νέας φωτογραφίας ένα νέο πεδίο “title” τύπου text_field ώστε να εισαχθεί και ο τίτλος μαζί με τις άλλες παραμέτρους της φωτογραφίας. Αυτό γίνεται στον ‘προβολέα’ *new.html.haml* και πλέον ο ‘ελεγκτής’ έχει πρόσβαση στις παραμέτρους params[: ...] ώστε να δημιουργήσει μια νέα φωτογραφία.

Στην εμφάνιση των φωτογραφιών του χρήστη για να προβάλλεται και ο τίτλος μπορούμε απλά να πάρουμε το πεδίο .title της κάθε φωτογραφίας στο αρχείο show.html.haml του προβολέα user.

Επίσης στην private μέθοδο photo_params του ‘ελεγκτή’ προσθέσαμε την παράμετρο :title για μεγαλύτερο έλεγχο των παραμέτρων από τον “Action Controller” ώστε να μην μπορούν χρήστες να ενημέρωσουν πεδία που δεν ορίζονται ρητά.

Τέλος στο μοντέλο photos έγινε δημιουργία ενός νέου ‘migration’ - “[...]_add_title_to_photo.rb” ώστε να αποθηκεύεται και ο τίτλος της φωτογραφίας στη βάση δεδομένων. Χρησιμοποιώντας έτσι το ActiveRecord του Rails μπορούμε να αλλάξουμε το μοντέλο με το εργαλείο rake, δίνοντας την εντολή db:migrate.

2.Λειτουργία “follow”

Ζητούμενο:

Ο χρήστης έχει τη δυνατότητα να ακολουθεί χρήστες και να βλέπει τις φωτογραφίες που αυτοί έχουν ανεβάσει. Οι φωτογραφίες θα πρέπει να είναι ταξινομημένες σε χρονολογική σειρά.

Σε πρώτο στάδιο προστέθηκε ένα νέο κουμπί (“Socialize”) στην αρχική σελίδα του χρήστη με το οποίο εμφανίζονται όλοι οι εγγεγραμμένοι χρήστες στην εφαρμογή. Για την εμφάνιση όλων των χρηστών προσθέτουμε μια νέα μέθοδο index στον user controller η οποία αντιστοιχεί στο path /users. Το path αυτό θα προστεθεί ως link_to στο κουμπί “Socialize” στον αντίστοιχο προβολέα

Χρησιμοποιώντας το εργαλείο rake με την εντολή routes μπορούμε να δούμε σε πιο path αντιστοιχουν οι μέθοδοι του ελεγκτή όπως φαίνεται στην παρακάτω εικόνα.

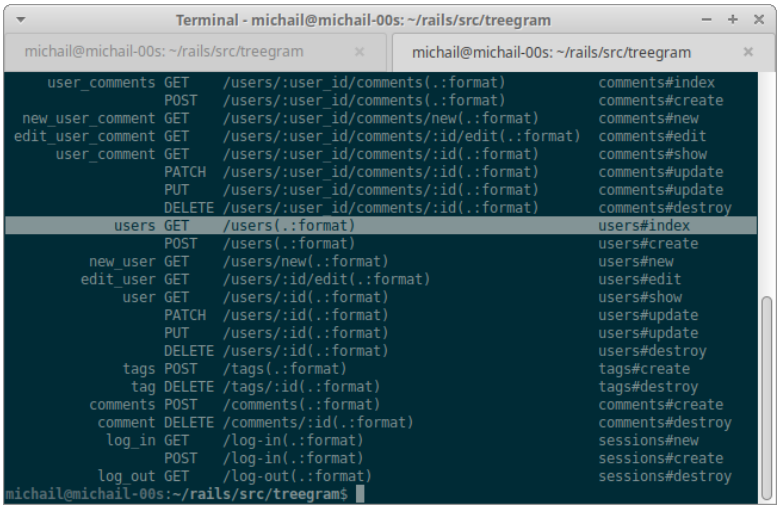

Illustration 1: Εμφάνιση routes με το rake

Δίπλα στο προφίλ κάθε χρήστη εμφανίζεται ένα κουμπί “Follow” ώστε να μπορεί ο συνδεδεμένος χρήστης να ακολουθήσει κάποιον από τους υπόλοιπους. Το κουμπί “Follow” συνδέεται με μία νεα διαδρομή (“new_relation_path”) στον αντίστοιχο ελεγκτή (“relations_controller”) ο οποίος

δημιουργεί μια νέα "σχέση" μεταξύ των δυο χρηστών following – follower. Ο follower είναι ο τρέχων χρήστης και following ο χρήστης που επέλεξε να ακολουθήσει. Αφού προσθέσουμε τη μέθοδο "create" στον relations_controller και καθορίσουμε τα χαρακτηριστικά των χρηστών, μπορούμε να την αποθηκεύσουμε στη βάση δεδομένων στον πίνακα relations χρησιμοποιώντας το μοντέλο του χρήστη. Στο μοντέλο μπορεί να οριστεί ως has_many :relations και το ActiveRecord να εφαρμόσει αυτή τη σχέση.

Εμφάνιση φωτογραφιών των ακολουθούμενων χρηστών

Στο σημείο αυτό τροποποιήσαμε το show.html.haml και τον user controller έτσι ώστε εφόσον έχει following users ο χρήστης, για κάθε έναν που ακολουθεί *παίρνουμε τις εικόνες του* και τις τοποθετούμε όλες μαζί σε ένα κοινό πίνακα. Έπειτα τις κάνουμε ταξινόμηση με βάση το πεδίο created_at και τις δείχνουμε στο χρήστη.

- Επειδή οι εικόνες είναι ένα υποσύνολο του πίνακα user, κάνοντας χρήση του flatmap operator, επιστρέφουμε πάλι ένα πίνακα, απλά μόνο με τις φωτογραφίες κάθε χρήστη.
  [[φ1,φ2,φ3], [φ5,φ6,φ7]]
- Για να μην έχουμε ένα πίνακα της μορφής [[χ],[ζ]] (δηλαδή κάθε στοιχείο του πίνακα να είναι πίνακας) κάναμε χρήση της μεθόδου flatten η οποία εφαρμόζεται σε έναν πίνακα και τα στοιχεία που είναι και αυτά πίνακα, τα αντικαταστεί με το στοιχείο που θέλουμε.

  Π.χ [[1],[2],[3]] flatten = [1,2,3]

3. Σχολιασμός φωτογραφιών

Ζητούμενο:

Στο κάτω μέρος κάθε φωτογραφίας υπάρχει δυνατότητα να προστεθεί σχόλιο από κάποιον χρήστη.

Για την υλοποίηση της λειτουργίας αυτής προστέθηκε ένας νέος ελεγκτής, "comments_controller", το μοντέλο "comment" και τροποποιήθηκε ο προβολέας του χρήστη ώστε να εμφανίζεται η φόρμα για το σχόλιο και τα σχόλια που έχουν ήδη προστεθεί. Επίσης τροποποιήσαμε το μοντέλο photo για να καθορίσουμε τη σχέση μεταξύ μια φωτογραφίας και σχολίων (μια φωτογραφία μπορεί να έχει πολλά σχόλια). Δηλαδή στον κώδικα η προσθήκη

```
class Photo < ActiveRecord::Base
        has_many :comments
```

καθορίζει αυτή τη σχέση στο μοντέλο.

Στον ελεγκτή comments_controller δημιουργούμε ένα νέο σχόλιο με τη μέθοδο create, με τις παραμέτρους που έχουμε πρόσβαση από τον ActionController μέσω του hash "params".

Όταν εμφανίζονται οι φωτογραφίες απο του χρήστες (και των ακολούθων) εμφανίζονται και τα σχόλια. Αυτό υλοποιείται στον προβολέα του χρήστη χρησιμοποιώντας τα block της γλώσσας ruby για να πάρουμε τα αντίστοιχα σχόλια για κάθε φωτογραφία.

Τμήμα Μηχ. Η/Υ & Πληροφορικής,
Πανεπιστήμιο Ιωαννίνων, Ιωάννινα

Ζώης Νικόλαος (2103)
Τσιόλκας Μιχαήλ (2152)