ΜΥΕ042 - Τεχνολογίες Διαδικτύου

Διδάσκων: Στέργιος Αναστασιάδης

Αναφορά 1ης Εργαστηριακής Άσκησης:

Εφαρμογή ιστού για κοινοχρησία φωτογραφιών στοRuby-on-Rails

Ομάδα:

Κονδυλία Βέργου 4325

Παναγιώτης Βουζαλής 2653

Χειμερινό Εξάμηνο 2022

Περιεχόμενα

[Σημείωση: Logins Χρηστών για Επίδειξη της Τελικής Εφαρμογής 3](#_Toc120714007)

[1. Προϋπάρχoν Error: Δημιουργία Null Photo 4](#_Toc120714008)

[2. Προσθήκη Τίτλου σε Φωτογραφία 7](#_Toc120714009)

[2.1. Δημιουργία Migration add\_title\_to\_photos 7](#_Toc120714010)

[2.2. Επεξεργασία Προβολής photos/new 8](#_Toc120714011)

[2.3. Επεξεργασία Ελεγκτή photos\_controller 8](#_Toc120714012)

[2.4. Επεξεργασία Προβολής users/show 9](#_Toc120714013)

[2.5. Παράδειγμα Χρήσης Title σε Φωτογραφίες 9](#_Toc120714014)

[3. Προσθήκη Λειτουργίας follow 10](#_Toc120714015)

[3.1. Δημιουργία Μοντέλου follow και Migration create\_follows 10](#_Toc120714016)

[3.2. Δημιουργία Ελεγκτή follows\_controller 11](#_Toc120714017)

[3.3. Επεξεργασία Αρχείου Διαδρομών routes 12](#_Toc120714018)

[3.4. Επεξεργασία Μοντέλου follow 12](#_Toc120714019)

[3.5. Επεξεργασία Προβολής follow/index 12](#_Toc120714020)

[3.6. Επεξεργασία Ελεγκτή users\_controller 13](#_Toc120714021)

[3.7. Επεξεργασία Προβολής users/show 13](#_Toc120714022)

[3.8. Τοποθέτηση Κατάλληλου CSS 14](#_Toc120714023)

[3.9. Παρουσίαση Εφαρμογής Μέχρι Τώρα 15](#_Toc120714024)

[4. Προσθήκη Λειτουργίας comments 17](#_Toc120714025)

[4.1. Δημιουργία Μοντέλου comment και Migration create\_comments 17](#_Toc120714026)

[4.2. Δημιουργία Ελεγκτή comments\_controller 18](#_Toc120714027)

[4.3. Επεξεργασία Μοντέλου comment 19](#_Toc120714028)

[4.4. Επεξεργασία Αρχείου Διαδρομών routes 19](#_Toc120714029)

[4.5. Επεξεργασία Αρχείου Διαδρομών routes 19](#_Toc120714030)

[4.6. Δημιουργία Προβολής comment/new 20](#_Toc120714031)

[4.7. Επεξεργασία Προβολής users/show 20](#_Toc120714032)

[4.8. Παρουσίαση Εφαρμογής Μέχρι Τώρα 21](#_Toc120714033)

[5. Bonus: Προσθήκη Λειτουργίας Διαγραφής post 22](#_Toc120714034)

[5.1. Επεξεργασία αρχείου models/photo 22](#_Toc120714035)

[5.2. Επεξεργασία Προβολής users/show 22](#_Toc120714036)

[5.3. Επεξεργασία Ελεγκτή photos\_controller 23](#_Toc120714037)

[6. Τελική Παρουσίαση Εφαρμογής 24](#_Toc120714038)

[7. Προβληματισμοί 25](#_Toc120714039)

# 

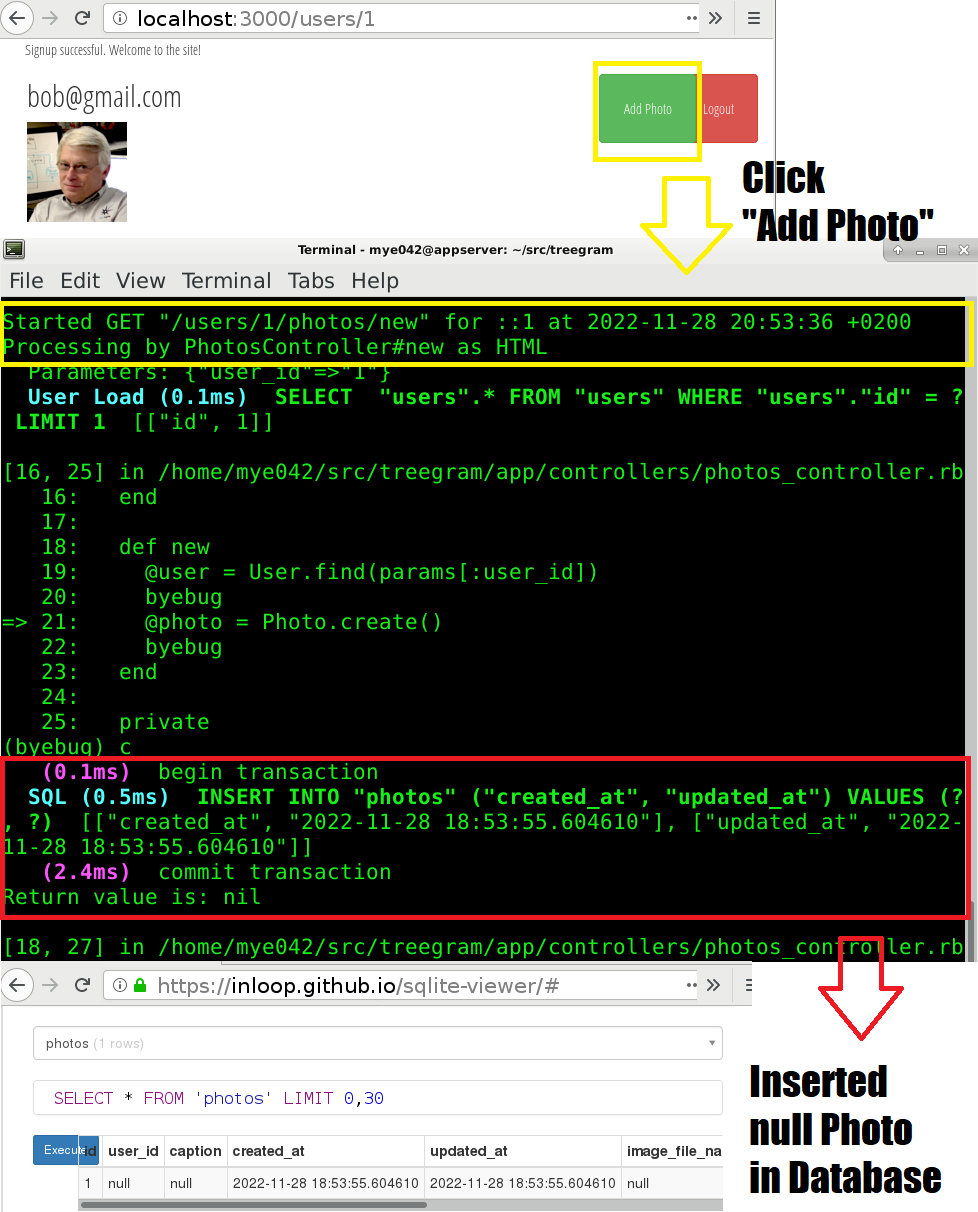
# Σημείωση: Logins Χρηστών για Επίδειξη της Τελικής Εφαρμογής

Για την παρουσίαση της τελικής εφαρμογής μας, μπορείτε να συνδεθείτε στο treegram που υποβάλλαμε στο turnin χρησιμοποιώντας τα παρακάτω credentials:  
- username: bob@gmail.com passwd: bob  
- username: tim@gmail.com passwd: tim  
- username: linus@gmail.com passwd: linus

To turnin μας μπορείτε να το βρείτε και στο παρακάτω link ([click me](https://drive.google.com/drive/folders/1794yepasfdfzMbKoRyB0wd8oIjzFNWQb?usp=sharing)).

1. Προϋπάρχoν Error: Δημιουργία Null Photo   
Όταν πατούμε το κουμπί Add Photo, δημιουργείται μια null Photo πριν καν φορτωθεί η προβολή new.html.haml. Αυτό γίνεται, διότι η συνάρτηση Ruby που υλοποιεί την ενέργεια new στον PhotoController καλεί την Photo.create().

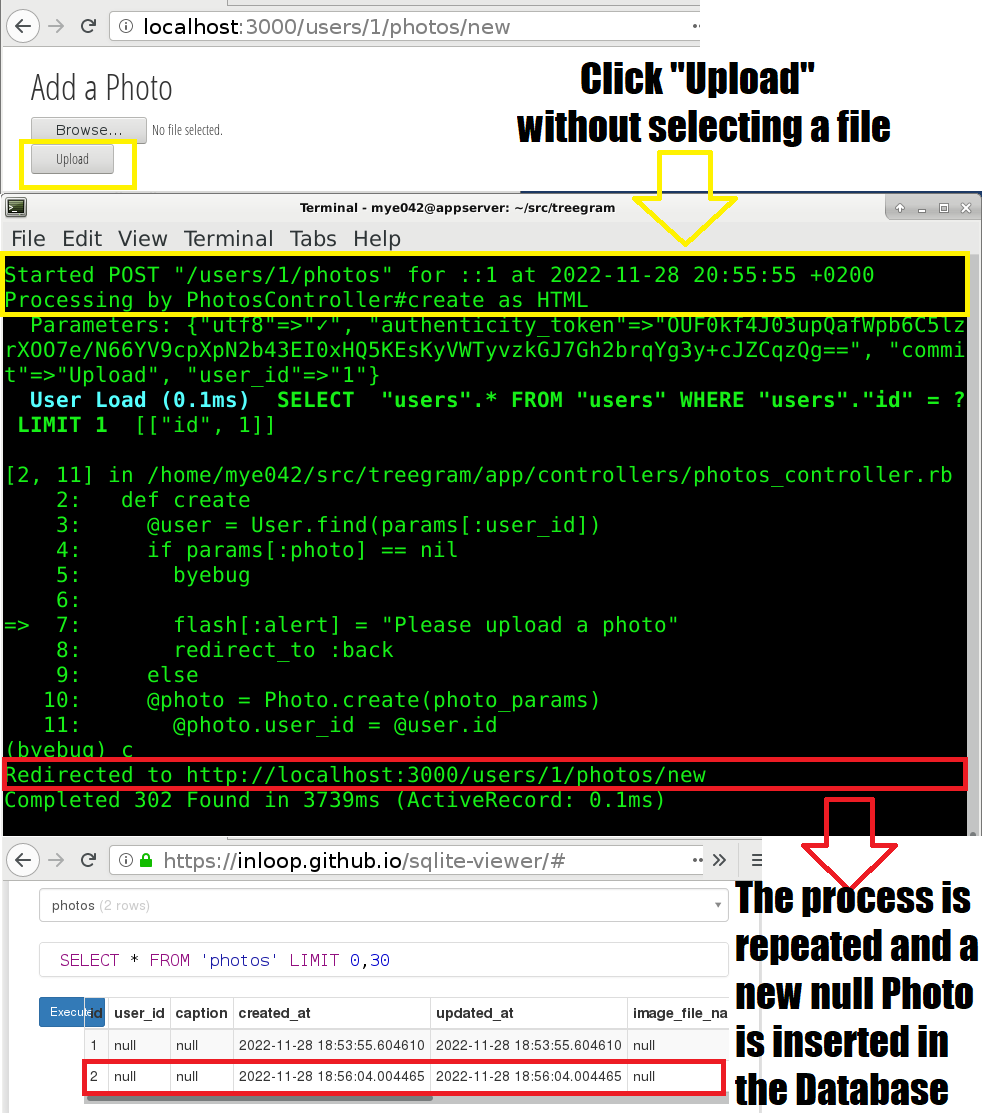
Επιβεβαιώσαμε τα παραπάνω, χρησιμοποιώντας τον διαδραστικό αποσφαλματωτή byebug και τον online [SQLite Viewer](https://inloop.github.io/sqlite-viewer).



Αφού ολοκληρωθεί η κλήση της new, φορτώνεται η προβολή στον διακομιστή μας.

Σε αυτό το σημείο, μπορούμε να ξαναδημιουργήσουμε μια null Photo, αν πατήσουμε στην φόρμα προσθήκης Photo το Upload χωρίς να έχουμε προσθέσει φωτογραφία.

Αυτό γίνεται καθώς η συνάρτηση Ruby που υλοποιεί την ενέργεια create στον PhotoController, αν δεν έχει επιλεχθεί φωτογραφία, κάνει redirect στην new και επαναλαμβάνονται όσα αναφέραμε παραπάνω.



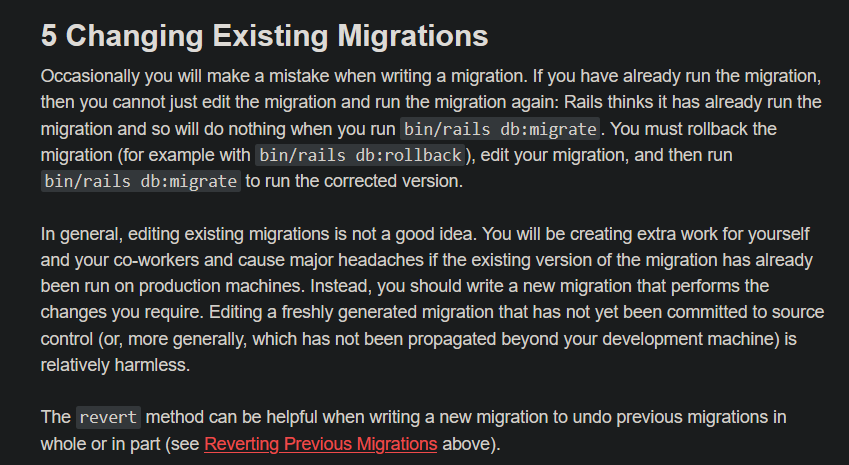
Για να εξαλείψουμε αυτό το πρόβλημα στην υλοποίησή μας, αφαιρέσαμε τις δυο γραμμές κώδικα της συνάρτησης new του αρχείου “app/controllers/photos\_controller.rb”:  
$ @user = User.find(params[:user\_id])  
$ @photo = Photo.create()

Η πρώτη γραμμή έκανε απλά GET έναν User που δεν τον χρησιμοποιούσε και η δεύτερη γραμμή δημιουργούσε τη null Photo.

# 2. Προσθήκη Τίτλου σε Φωτογραφία

## 2.1. Δημιουργία Migration add\_title\_to\_photos

Από το [guides.rubyonrails.org](https://guides.rubyonrails.org/active_record_migrations.html#changing-existing-migrations) διαβάσαμε ότι είναι προτιμότερο να δημιουργήσουμε καινούργιο migration παρά να τροποποιήσουμε ένα ήδη υπάρχον.



Για αυτόν τον λόγο θα φτιάξουμε ένα καινούργιο migration έτσι ώστε να προσθέσουμε τον τίτλο στις φωτογραφίες.

Για την προσθήκη του column “title” στον πίνακα “photos” της βάσης μας εκτελούμε τις εξής εντολές στo root directory της εφαρμογής:  
$ rails generate migration add\_title\_to\_photos title:string  
$ rake db:migrate  
  
Graphical user interface, text, application

Description automatically generated

A screenshot of a computer screen

Description automatically generated

## 2.2. Επεξεργασία Προβολής photos/new

Στο αρχείο “app/views/photos/new.html.haml” προσθέτουμε τις παρακάτω γραμμές:  
$ %h2 Add a title: #{form.text\_field :title, placeholder:"Look at my dog!"}  
$ = link\_to 'Back to home view', user\_path(current\_user)  
Συνεπώς, το prompt για το upload φωτογραφίας θα φαίνεται έτσι:  
Timeline

Description automatically generated with low confidence

## 2.3. Επεξεργασία Ελεγκτή photos\_controller

Στο αρχείο “app/controllers/photos\_controller.rb” προσθέτουμε τους εξής ελέγχους, για να αναγκάζεται ο χρήστης να έχει επιλέξει φωτογραφία προς ανέβασμα και να έχει προσθέσει τίτλο, πριν του επιτραπεί να προχωρήσει:  
$ if params[:photo][:image] == nil  
$ flash[:alert] = "Please upload a photo."  
$ redirect\_to :back  
$ elsif params[:photo][:title] == ""  
$ flash[:alert] = "You forgot to add a title!"  
$ redirect\_to :back  
  
Επίσης βάζουμε το argument :title στη γραμμή   
$ params.require(:photo).permit(:image, :title)  
Text

Description automatically generated

## 2.4. Επεξεργασία Προβολής users/show

Στο αρχείο “app/views/users/show.html.haml” προσθέτουμε την εξής γραμμή για να υποδείξουμε στον κώδικά μας να εμφανίζει τον τίτλο κάθε φωτογραφίας:  
$ %h2= photo.title

Επίσης προσθέτουμε την εξής γραμμή για να εμφανίζεται το όνομα και το catchphrase της εφαρμογής όσο τη χρησιμοποιούμε:  
$ %h1.title Treegram - Show off to the world!  
Timeline

Description automatically generated

## 2.5. Παράδειγμα Χρήσης Title σε Φωτογραφίες

Επομένως, η εφαρμογή μας φαίνεται έτσι, όταν έχουμε κάνει log σε ένα user και έχουμε ανεβάσει φωτογραφίες.  
Graphical user interface, application, website

Description automatically generated

# 3. Προσθήκη Λειτουργίας follow

## 3.1. Δημιουργία Μοντέλου follow και Migration create\_follows

Δημιουργούμε το μοντέλο follow με την παρακάτω εντολή:  
$ rails generate model follow  
Text

Description automatically generated

Έπειτα στο αρχείο “app/db/migrate/TIMESTAMP\_create\_follows.rb” ορίζουμε τη δομή του πίνακα με τα αναγνωριστικά του follower (αυτός που ακολουθεί τα post κάποιου) και του followee (αυτός του οποίου τα post ακολουθούν άλλοι):  
Text

Description automatically generated

Εκτελούμε $ rake db:migrate για να δημιουργήσουμε τον πίνακα των follow:  
Text

Description automatically generated

Ο οποίος σε αυτό το σημείο είναι άδειος:  
Graphical user interface, application

Description automatically generated

## 3.2. Δημιουργία Ελεγκτή follows\_controller

Σε αυτό το σημείο πρέπει να δημιουργήσουμε τη σχέση μεταξύ των πινάκων follow και user. Αυτό θα το κάνουμε με έναν follows controller.

Δημιουργούμε τον follows controller με την εξής εντολή:  
$ rails generate controller follows  
Text

Description automatically generated

Τροποποιούμε το αρχείο “app/controllers/follows\_controller.rb” προσθέτοντας λειτουργία follow/unfollow:  
Text

Description automatically generated

## 3.3. Επεξεργασία Αρχείου Διαδρομών routes

Τροποποιούμε το αρχείο “app/config/routes.rb” προσθέτοντας τη γραμμή  
$ resources :follows  
A screenshot of a computer

Description automatically generated with medium confidence

## 3.4. Επεξεργασία Μοντέλου follow

Τροποποιούμε το αρχείο “app/models/follow.rb” προσθέτοντας τη γραμμή  
$ validates :follower\_id, uniqueness: {scope: :followee\_id}  
Graphical user interface, text

Description automatically generated

## 3.5. Επεξεργασία Προβολής follow/index

Δημιουργούμε το αρχείο “/app/views/follows/index.html.haml” για προβολή όλων των χρηστών που μπορεί να ακολουθήσει ένας χρήστης:  
Text

Description automatically generated

## 3.6. Επεξεργασία Ελεγκτή users\_controller

Τροποποιούμε το αρχείο “/app/controllers/users\_controller.rb”.  
Προθέτουμε κώδικα για τη δημιουργία ενός array που περιέχει το προσωπικό δίκτυο κάθε χρήστη της εφαρμογής (αποτελείται απο τον ίδιο το χρήστη και απο τα άτομα που ακολουθεί):  
Text

Description automatically generated

## 3.7. Επεξεργασία Προβολής users/show

Τροποποιούμε το αρχείο “/app/views/users/show.html.haml” για να προσθέσουμε:  
- Κουμπί για εμφάνιση εγγεγραμμένων χρηστών  
- Εμφάνιση φωτογραφιών του χρήστη και ατόμων που ακολουθεί ο χρήστης (folowees) σε αντίστροφη χρονολογική σειρά.  
Σημειώνουμε πως προς το παρόν τα άτομα που έχει ακολουθήσει ένας χρήστης εμφανίζονται σε κανονική χρονολογική σειρά (οι φωτογραφίες τους εμφανίζονται όμως όπως ζητείται σε αντίστροφη χρονολογική σειρά).Text

Description automatically generated

## 3.8. Τοποθέτηση Κατάλληλου CSS

Τροποποιούμε το αρχείο “app/assets/stylesheets/application.sass” για τη σωστή προβολή των κουμπιών και των text πεδίων username και τίτλου φωτογραφίας:  
Text

Description automatically generated

## 3.9. Παρουσίαση Εφαρμογής Μέχρι Τώρα

Graphical user interface, application, website

Description automatically generatedΑρχική σελίδα που βλέπει ο χρήστης:

Prompt που βλέπει ο χρήστης όταν πατήσει το κουμπί follow people:  
A picture containing logo

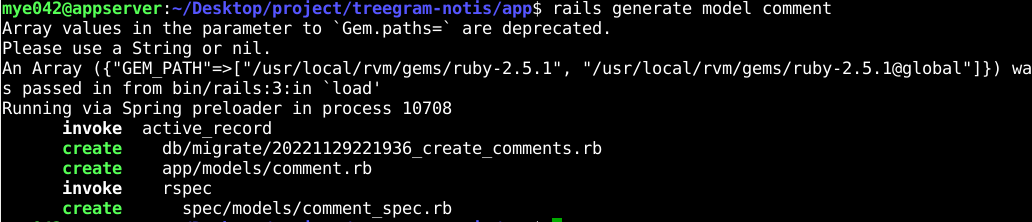
Description automatically generated

Graphical user interface, application, website

Description automatically generatedΣκρολλάροντας προς τα κάτω o χρήστης μπορεί να δεί όλο το feed του:

# 4. Προσθήκη Λειτουργίας comments

## 4.1. Δημιουργία Μοντέλου comment και Migration create\_comments

Δημιουργούμε το μοντέλο follow με την παρακάτω εντολή:  
$ rails generate model comment  


Έπειτα στο αρχείο “app/db/migrate/TIMESTAMP\_create\_comments.rb” ορίζουμε τη δομή του πίνακα με τα αναγνωριστικά του comment:  
Text

Description automatically generated

Εκτελούμε $ rake db:migrate για να δημιουργήσουμε τον πίνακα των comment:  
Text

Description automatically generated

## 4.2. Δημιουργία Ελεγκτή comments\_controller

Δημιουργούμε τον comments controller με την εξής εντολή:  
$ rails generate controller comments  
Text

Description automatically generated

Τροποποιούμε το αρχείο “app/controllers/comments\_controller.rb” προσθέτοντας λειτουργία δημιουργίας comment:  
A screenshot of a computer

Description automatically generated with medium confidence

## 4.3. Επεξεργασία Μοντέλου comment

Τροποποιούμε το αρχείο “app/models/comment.rb” προσθέτοντας τις γραμμές  
$ belongs\_to :user  
$ belongs\_to :photo  
Text

Description automatically generated

## 4.4. Επεξεργασία Αρχείου Διαδρομών routes

Τροποποιούμε το αρχείο “app/models/photo.rb” προσθέτοντας τη γραμμή  
$ has\_many :comments  
Text

Description automatically generated

## 4.5. Επεξεργασία Αρχείου Διαδρομών routes

Τροποποιούμε το αρχείο “app/config/routes.rb” προσθέτοντας τη γραμμή  
$ resources :comments  
Graphical user interface, text

Description automatically generated

## 4.6. Δημιουργία Προβολής comment/new

Δημιουργούμε το αρχείο για τη φόρμα υποβολής comment:Text

Description automatically generated

## 4.7. Επεξεργασία Προβολής users/show

Text

Description automatically generatedΤροποποιούμε το αρχείο “/app/views/users/show.html.haml” για να προσθέσουμε:  
- Κουμπί για προσθήκη νέου comment κάτω από κάθε φωτογραφία  
- Προβολή λίστας σχολίων κάτω από κάθε φωτογραφία

## 4.8. Παρουσίαση Εφαρμογής Μέχρι Τώρα

Τα comments βρίσκονται κάτω απο κάθε φωτογραφία με κανονική σειρά  
(όπως και στο αληθινό instagram)  
Graphical user interface, application

Description automatically generated

Prompt που βλέπει ο χρήστης όταν πατήσει το κουμπί add a comment:  
Graphical user interface, application

Description automatically generated

# 5. Bonus: Προσθήκη Λειτουργίας Διαγραφής post

## 5.1. Επεξεργασία αρχείου models/photo

Τροποποιούμε το αρχείο “/app/models/photo.rb” για να προσθέσουμε τις γραμμές:  
$ has\_many :tags, dependent: :destroy  
$ has\_many :comments, dependent: :destroy  
Έτσι ώστε όταν διαγράφεται μια φωτογραφία να καταστρέφονται και τα tags και τα comments που σχετίζονται με αυτή  
Text

Description automatically generated

## 5.2. Επεξεργασία Προβολής users/show

Τροποποιούμε το αρχείο “/app/views/users/show.html.haml” για να προσθέσουμε:  
- Κουμπί για διαγραφή φωτογραφίας εαν η φωτογραφία ανήκει στο χρήστη  
Επιπλέον αλλάξαμε την λειτουργία των tags: Κάθε χρήστης μπορεί πια να κάνει tags μόνο στις δικές του φωτογραφίες.  
Text

Description automatically generated

## 5.3. Επεξεργασία Ελεγκτή photos\_controller

Τροποποιούμε το αρχείο “/app/controllers/photos\_controller.rb” για να προσθέσουμε τη μέθοδο destroy που υλοποιεί τη λειτουργία διαγραφής post.  
Text

Description automatically generated

# Graphical user interface, text, application Description automatically generated6. Τελική Παρουσίαση Εφαρμογής

# 7. Προβληματισμοί