

ΑΝΑΦΟΡΑ 1ης ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ

Δημήτριος Παγώνης, ΑΜ:4985
Δημήτριος Τζαλοκώστας, ΑΜ:4994

21 Νοεμβρίου 2024

Περίληψη

1 Προσθέστε τίτλο σε μια φωτογραφία όταν την ανεβάζετε στον διακομιστή

1.1 Προσθήκη τίτλου στον πίνακα "photos"

Επεκτείνουμε το μοντέλο photo και τον αντίστοιχο πίνακα photos της βάσης δεδομένων με μια ιδιότητα title τύπου string στο db/schema.rb.

```
1 create_table "photos", force: :cascade do |t|
2   t.integer "user_id"
3   t.string "caption"
4   t.datetime "created_at"
5   t.datetime "updated_at"
6   t.string "image_file_name"
7   t.string "image_content_type"
8   t.integer "image_file_size"
9   t.datetime "image_updated_at"
10  t.string "title"
11 end
```

1.2 Επικύρωση τίτλου στην ../models/photos.rb

Προσθέσαμε την εντολή validates :title, presence: true ώστε να προσθέτει μια επικύρωση για να διασφαλίσει ότι υπάρχει το χαρακτηριστικό τίτλος. Εάν λείπει ο τίτλος, η φωτογραφία δεν θα αποθηκευτεί στη βάση δεδομένων.

```
1 class Photo < ActiveRecord::Base
2   belongs_to :user
3   has_many :tags, dependent: :destroy
4
5
6   has_attached_file :image, :styles => { :medium => "300x300>", :thumb
7     => "100x100>" }, :default_url => "/images/:style/missing.png"
8   validates_attachment_content_type :image, :content_type => /\Aimage
9     \.\/.*\Z/
10   validates :title, presence: true
11 end
```

1.3 Προσθήκη παραμέτρου στην photos_controller.rb

Ουσιαστικά καθορίζουμε μία ακόμα παράμετρο προκειμένου να χρησιμοποιήσουμε τον τίτλο και να αποτρέψουμε χρήση άλλης παραμέτρου.

```
1 def photo_params
2   params.require(:photo).permit(:title)
3 end
```

1.4 Επέκταση Φόρμας στο app/views/photos/new.html.haml

Στο σημείο αυτό επεκτείνουμε την φόρμα που χρησιμοποιούμε για το ανέβασμα συμπεριλαμβάνοντας τον τίτλο

```
1 %h1 Add a Photo
2 = form_for Photo.new(), :url => user_photos_path, :html => {:multipart
  => true} do |form|
3   = form.file_field :image
4   = form.label :title, 'Give a title:'
5   = form.text_field :title
6   = form.submit 'Upload'
```

1.5 Εμφάνιση περιγραφής στην φωτογραφία

Προκειμένου να φαίνεται ο τίτλος στον χρήστη στο κάτω μέρος της φωτογραφίας προσθέσαμε στο views/users/show/html.haml την εξής εντολή:

```
1 %p
2   %strong= photo.title
```

2 Επιτρέψτε στους χρήστες να καθορίσουν χρήστες που ακολουθούν

2.1 Εισαγωγή μοντέλου Follow

Η εντολή που χρησιμοποιήσαμε για να εισάγουμε το μοντέλο Follow είναι:

```
1 rails generate migration Follow follower_id:integer followed_id:
  integer
2 rake db:migrate
```

Ο πίνακας περιλαμβάνει τέσσερις στήλες: `follower_id` (ακέραιος αριθμός) για αποθήκευση του αναγνωριστικού του χρήστη που ακολουθεί, `followed_id` (ακέραιος αριθμός) για αποθήκευση του αναγνωριστικού του χρήστη που ακολουθείται, καθώς και `created_at` και `updated_at` (και τα δύο τύπου `datetime`) για αποθήκευση των χρονικών στιγμών για το πότε δημιουργήθηκε και ενημερώθηκε τελευταία η σχέση, αντίστοιχα.

Το μοντέλο `Follow` ορίζει μια σχέση μεταξύ των χρηστών, όπου ένας χρήστης μπορεί να ακολουθήσει έναν άλλο. Χρησιμοποιεί συσχετίσεις `belongs_to` για να συνδέσει το μοντέλο `Follow` με το μοντέλο `User` δύο φορές: μία φορά ως `follower` και μία ως `followed`. Καθορίζοντας το `User`, υποδηλώνει ότι τόσο το `follower` όσο και το `followed` είναι παρουσίες του μοντέλου `User`. Αυτή η ρύθμιση επιτρέπει στην εφαρμογή να παρακολουθεί ποιοι χρήστες ακολουθούν ποιους άλλους χρήστες, ενεργοποιώντας λειτουργίες όπως λίστες παρακολούθησης και ακολουθών.

```
1 class Follow < ActiveRecord::Base
2   belongs_to :follower, class_name: 'User'
3   belongs_to :followed, class_name: 'User'
4 end
```

Listing 1: follow.rb Code

2.2 follows_controller.rb

Το FollowsController διαχειρίζεται τη δημιουργία και τη διαγραφή των ακολουθήσεων μεταξύ των χρηστών. Στην ενέργεια create, βρίσκει τον χρήστη που θα ακολουθήσει χρησιμοποιώντας την παράμετρο followed_id, δημιουργεί μια νέα εγγραφή "Follow" με τον τρέχοντα χρήστη ως ακόλουθο, ορίζει ένα μήνυμα flash[:notice] για να επιβεβαιώσει την δημιουργία του follow και ανακατευθύνει τον χρήστη πίσω στην προηγούμενη σελίδα. Στην ενέργεια destroy, βρίσκει την εγγραφή Follow μέσω της εντολής Follow.find(params[:id]) με βάση το :id του, τη διαγράφει, ορίζει ένα μήνυμα flash[:notice] για να επιβεβαιώσει την διαγραφή του follow και ανακατευθύνει τον χρήστη πίσω στην προηγούμενη σελίδα.

```

1  class FollowsController < ApplicationController
2    def create
3      user = User.find(params[:followed_id])
4      Follow.create(follower_id: current_user.id, followed_id: user.id)
5      flash[:notice] = "You are now following #{user.email}"
6      redirect_to :back
7    end
8
9    def destroy
10     follow = Follow.find(params[:id])
11     follow.destroy
12     flash[:notice] = "You have unfollowed #{follow.followed.email}"
13     redirect_to :back
14   end
15 end

```

Listing 2: follows_controller.rb Code

2.3 Τροποποίηση της ../views/users/show.html.haml

Δίπλα στα κουμπιά Add Photo και Logout προσθέσαμε άλλο ένα κουμπί το Registered Users, το οποίο όταν το πατήσει ο χρήστης, τον ανακατευθύνει στη διαδρομή ../views/users/emails_users_path όπου δημιουργήσαμε, περνώντας το email του τρέχοντος χρήστη ως παράμετρο current_user_email: @user.email.

```

1  .row.top_row
2    .col-sm-6.user_att
3      %h2= @user.email
4      - if @user.avatar_file_name
5        = image_tag @user.avatar.url(:thumb)
6    .col-sm-4
7      = link_to 'Registered Users', emails_users_path(current_user_email
8        : @user.email), class: ['btn', 'btn-info', 'users_btn']
9    .col-sm-6
10     = link_to 'Logout', log_out_path, class: ['btn', 'btn-danger', '
11       logout_btn']
12 .row
13 = link_to 'Add Photo', new_user_photo_path(@user), class: ['btn', '
14   btn-success', 'add_photo_btn']

```

Listing 3: ../views/users/show.html.haml Code

2.4 Δημιουργία της ../views/users/emails.html.haml

Στον παρακάτω κώδικα δημιουργείται μια λίστα όλων των εγγεγραμμένων χρηστών, εμφανίζοντας το email κάθε χρήστη και παρέχοντας κουμπί για να τους ακολουθήσει ή όχι. Εάν το email που εμφανίζεται ταιριάζει με το email του τρέχοντος χρήστη, εμφανίζεται με πράσινο χρώμα η ένδειξη Active User. Διαφορετικά, εμφανίζει ένα κουμπί Following εάν ο τρέχων χρήστης ακολουθεί ήδη έναν άλλον χρήστη ή ένα κουμπί Follow εάν όχι. Επιπλέον, στο κάτω μέρος, υπάρχει ένα κουμπί Back to Main Page με κόκκινο χρώμα, το οποίο ανακατευθύνει τον χρήστη στη σελίδα του προφίλ του. Για τη δημιουργία των κουμπιών και των συνδέσμων, χρησιμοποιήσαμε τα link_to και button_to.

```

1 %h1{ style: "text-align: center; font-size: 50px;" } Registered Users
2
3 %ul{ style: "text-align: left; font-size: 25px;" }
4   - @users.each do |user|
5     %ul{ style: "display: flex; align-items: center; justify-content:
6       center; margin-bottom: 10px;" }
7       %span{ style: "flex: 1;" }= user.email
8       - if user.email == @current_user_email
9         %span{ style: "font-size: 25px; color: green; margin-left: 20
10          px;" } Active user
11       - else
12         - if Follow.exists?(follower_id: current_user.id, followed_id:
13           user.id)
14           = button_to 'Following', follow_path(Follow.find_by(
15             follower_id: current_user.id, followed_id: user.id)),
16             method: :delete, class: ['btn', 'btn-secondary'], style:
17             'margin-left: 20px;'
18         - else
19           = button_to 'Follow', follows_path(followed_id: user.id),
20             method: :post, class: ['btn', 'btn-primary'], style: '
21             margin-left: 20px;'
22
23 %br/
24 %br/
25 .row
26   .col-sm-12
27     = link_to 'Back to Main Page', user_path(current_user), class: ['
28       btn', 'btn-danger'], style: 'margin: 50 auto; text-align:
29       center;'

```

Listing 4: ../views/users/emails.html.haml Code

2.5 Συνέχεια τροποποίησης της {../views/users/show.html.haml

Στην σελίδα του προφίλ του χρήστη θα εμφανίζεται μια επικεφαλίδα My photos, και από κάτω οι φωτογραφίες του ταξινομημένες σε αντίστροφη χρονολογική σειρά. Έπειτα θα εμφανίζεται μια επικεφαλίδα Photos of Followd Users, και από κάτω οι φωτογραφίες όλων των χρηστών που ακολουθεί ο ενεργός χρήστης ταξινομημένες σε αντίστροφη χρονολογική σειρά. Για να επιτευχθεί αυτό η ταξινόμηση των φωτογραφιών σε αντίστροφη χρονολογική σειρά τροποποιήσαμε την UsersController, έτσι ώστε να ταξινομούνται σωστά οι φωτογραφίες και του ίδιου του ενεργού χρήστη αλλά και των άλλων χρηστών που ακολουθεί.

```

1 class UsersController < ApplicationController
2   def create
3     @user = User.new(user_params)
4     @user.valid?
5     if !@user.is_email?
6       flash[:alert] = "Input a properly formatted email."

```

```

7       redirect_to :back
8     elsif @user.errors.messages[:email] != nil
9       flash[:notice]= "That email " + @user.errors.messages[:email].
        first
10      redirect_to :back
11    elsif @user.save
12      flash[:notice]= "Signup successful. Welcome to the site!"
13      session[:user_id] = @user.id
14      redirect_to user_path(@user)
15    else
16      flash[:alert] = "There was a problem creating your account.
        Please try again."
17      redirect_to :back
18    end
19  end
20
21  def new
22  end
23
24  def show
25    @users = User.all
26    @user = User.find(params[:id])
27    @tag = Tag.new
28    @photos = @user.photos.order(created_at: :desc)
29    @followed_photos = Photo.joins(user: :followers)
30      .where(follows: { follower_id: current_user.id })
31      .order(created_at: :desc)
32      .distinct
33      .includes(:comments)
34  end
35
36  def emails
37    @users = User.all
38    @current_user_email = params[:current_user_email]
39  end
40
41
42
43  private
44
45    def user_params
46      params.require(:user).permit(:email, :password, :
        password_confirmation, :avatar)
47    end
48
49
50  end

```

Listing 5: UsersController Code

Οπότε η `../views/users/show.html.haml` (view) δέχεται τις φωτογραφίες ταξινομημένες από την UsersController και τις εμφανίζει μέσω της εντολής `@photos.each do |photo|` για τις φωτογραφίες του ενεργού χρήστη.

```

1 %h1{ style: "text-align: center; font-size: 50px;" } My photos
2 .row
3 %ul{ style: "text-align: left; font-size: 25px;" }
4   - @photos.each do |photo|
5     .well.col-sm-4
6       = image_tag photo.image.url(:medium)
7       %p
8       %strong= photo.title
9       %p
10      At:
11      = photo.created_at.strftime("%B %d, %Y %H:%M")
12      = form_for @tag do |f|
13        = f.hidden_field :photo_id, value: photo.id
14        = f.collection_select :user_id, @users, :id, :email
15        = f.submit "Tag User"
16      - photo.tags.each do |tag|
17        %p= tag.user.email

```

Listing 6: MyPhotos at ../views/users/show.html.haml Code

Όμως, για τις `followed_photos` ελέγχει πρώτα αν υπάρχουν φωτογραφίες και αν ναι, τις εμφανίζει μέσω της εντολής `@followed_photos.each do |photo|`, αλλιώς εμφανίζει ένα μήνυμα `No photos to display`.

```

1 %h1{ style: "text-align: center; font-size: 50px;" } Photos of
  Followed Users
2
3 %ul{ style: "text-align: left; font-size: 25px;" }
4   - if @followed_photos.any?
5     - @followed_photos.each do |photo|
6       .well.col-sm-4
7         = image_tag photo.image.url(:medium)
8         %p
9         %strong= photo.title
10        %p
11        By:
12        %strong= photo.user.email
13        %p
14        At:
15        = photo.created_at.strftime("%B %d, %Y %H:%M")
16      - photo.tags.each do |tag|
17        %p= tag.user.email
18    - else
19      %p No photos to display.

```

Listing 7: Photos of Followed Users at ../views/users/show.html.haml Code

3 Επιτρέψτε τους χρήστες να προσθέσουν σχόλια σε φωτογραφίες.

3.1 Εισαγωγή μοντέλου Comment

Η εντολή που χρησιμοποιήσαμε για να εισάγουμε το μοντέλο `Comment` είναι:

```

1 rails generate migration Comment photo_id:integer user_id:integer text
  :text
2 rake db:migrate

```

Ο πίνακας `comments` περιέχει τέσσερις στήλες οι οποίες είναι για συσχέτιση του σχολίου με μια συγκεκριμένη φωτογραφία (`photo_id :integer`), για συσχέτιση του σχολίου με τον χρήστη που το έκανε (`user_id :integer`), για αποθήκευση του περιεχομένου του σχολίου (`text :text`), και για αποθήκευση χρονικών στιγμών για το πότε δημιουργήθηκε (`created_at :datetime`) και ενημερώθηκε (`updated_at :datetime`) το σχόλιο για τελευταία φορά.

Το μοντέλο `Comment` καθορίζει ότι κάθε σχόλιο ανήκει σε μια φωτογραφία και χρήστη, καθιερώνοντας συσχετίσεις με τα μοντέλα `Photo` και `User` με το `belongs_to`. Επιπλέον, διασφαλίζει ότι πρέπει να υπάρχει το χαρακτηριστικό `text`, που σημαίνει ότι ένα σχόλιο δεν μπορεί να αποθηκευτεί στη βάση δεδομένων χωρίς περιεχόμενο.

```

1 class Comment < ActiveRecord::Base
2   belongs_to :photo
3   belongs_to :user
4
5   validates :text, presence: true
6 end

```

Listing 8: `comment.rb` Code

3.2 Τροποποίηση της `comments_controller.rb`

Αρχικά προσθέσαμε στην μέθοδο `show` από την `users_controller.rb` την εντολή `@comment = Comment.new` με σκοπό να ορίζω ένα νέο `comment`, το οποίο θα χρησιμοποιηθεί ώστε να διευκολύνει το χρήστη να γράφει και να προσθέτει σχόλια στις φωτογραφίες κατευθείαν από τη φόρμα που υπάρχει κάτω από κάθε φωτογραφία. Έπειτα ορίσαμε τη `CommentsController` η οποία ελέγχει τη δημιουργία και τη διαγραφή ενός σχολίου από μια φωτογραφία.

```

1   def show
2     @users = User.all
3     @user = User.find(params[:id])
4     @tag = Tag.new
5     @comment = Comment.new
6     @photos = @user.photos.order(created_at: :desc)
7     @followed_photos = Photo.joins(user: :followers)
8                           .where(follows: { follower_id: current_user.id })
9                           .order(created_at: :desc)
10                      .distinct
11                      .includes(:comments)
12   end

```

Listing 9: `show` method at `users_controller.rb` Code

Στην μέθοδο `create`, ο `controller` βρίσκει την κατάλληλη φωτογραφία, δημιουργεί ένα νέο σχόλιο που συνδέεται με αυτήν τη φωτογραφία και τον τρέχοντα χρήστη και επιχειρεί να την αποθηκεύσει, ορίζοντας ένα μήνυμα `flash` με βάση το αποτέλεσμα.

Στην μέθοδο `destroy`, ο `controller` βρίσκει το σχόλιο και τη σχετική φωτογραφία και διαγράφει το σχόλιο εάν ο τρέχων χρήστης είναι είτε ο συγγραφέας του σχολίου είτε ο κάτοχος της φωτογραφίας, ορίζοντας ένα κατάλληλο μήνυμα `flash`.

Στην μέθοδο `comment_params`, ο `controller` διασφαλίζει πως χρησιμοποιούνται μόνο οι επιτρεπόμενες παράμετροι (`text` και `photo_id`) για τη δημιουργία σχολίων.

```

1 class CommentsController < ApplicationController
2
3   def create
4     @photo = Photo.find(params[:comment][:photo_id])

```

```

5      @comment = @photo.comments.build(comment_params)
6      @comment.user = current_user
7
8      if @comment.save
9          flash[:notice] = "Comment added successfully."
10     else
11         flash[:alert] = "Failed to add comment."
12     end
13
14     redirect_to :back
15 end
16
17 def new
18 end
19
20 private
21
22 def comment_params
23     params.require(:comment).permit(:text, :photo_id)
24 end
25 end

```

Listing 10: comments_controller.rb Code

3.3 Τροποποίηση της ../views/users/show.html.haml

Για να μπορέσουμε να εμφανίσουμε τα σχόλια και να μπορέσει ο χρήστης να γράψει κάτω από κάθε φωτογραφία, δημιουργήσαμε αρχικά μια φόρμα που ζητάει από τον χρήστη είτε βρισκείται στις δικές του φωτογραφίες ή σε φωτογραφίες άλλων χρηστών που ακολουθεί μέσω της εντολής `form_for Comment.new, url: comments_path do |f|`, η οποία προετοιμάζει μια φόρμα για ένα νέο σχόλιο, με την υποβολή της φόρμας να κατευθύνεται στη διεύθυνση `comments_path`. Μέσα στη φόρμα, δημιουργείται ένα κρυφό πεδίο για την αποθήκευση του `photo_id`, συνδέοντας το σχόλιο με τη συγκεκριμένη φωτογραφία. Τέλος, παρέχεται ένα κουμπί υποβολής με τον τίτλο `Post Comment` για την υποβολή του σχολίου.

```

1 %h1{ style: "text-align: center; font-size: 50px;" } My photos
2 .row
3   %ul{ style: "text-align: left; font-size: 25px;" }
4     - @photos.each do |photo|
5       .well.col-sm-4
6         = image_tag photo.image.url(:medium)
7         %p
8         %strong= photo.title
9         %p
10        At:
11        = photo.created_at.strftime("%B %d, %Y %H:%M")
12        = form_for @tag do |f|
13          = f.hidden_field :photo_id, value: photo.id
14          = f.collection_select :user_id, @users, :id, :email
15          = f.submit "Tag User"
16        - photo.tags.each do |tag|
17          %p= tag.user.email
18        %h3
19        %strong
20          %u Comments
21        - photo.comments.each do |comment|
22          %p

```



```

23         %strong= comment.user.email
24         %br/
25         = comment.text
26     = form_for Comment.new, url: comments_path do |f|
27     = f.hidden_field :photo_id, value: photo.id
28     .form-group
29     = f.label :text, "Add a comment:"
30     = f.text_area :text, class: "form-control", rows: 1
31     = f.submit "Post Comment", class: "btn btn-primary"

```

Listing 11: My photos at ../views/users/show.html.haml Code

```

1  %h1{ style: "text-align: center; font-size: 50px;" } Photos of
   Followed Users
2
3  %ul{ style: "text-align: left; font-size: 25px;" }
4    - if @followed_photos.any?
5      - @followed_photos.each do |photo|
6        .well.col-sm-4
7          = image_tag photo.image.url(:medium)
8          %p
9            %strong= photo.title
10         %p
11         By:
12         %strong= photo.user.email
13         %p
14         At:
15         = photo.created_at.strftime("%B %d, %Y %H:%M")
16         - photo.tags.each do |tag|
17           %p= tag.user.email
18         %h3
19         %strong
20         %u Comments
21         - photo.comments.each do |comment|
22           %p
23             %strong= comment.user.email
24             %br/
25             = comment.text
26         = form_for Comment.new, url: comments_path do |f|
27         = f.hidden_field :photo_id, value: photo.id
28         .form-group
29         = f.label :text, "Add a comment:"
30         = f.text_area :text, class: "form-control", rows: 1
31         = f.submit "Post Comment", class: "btn btn-primary"
32     - else
33       %p No photos to display.

```

Listing 12: Photos of Followed Users at ../views/users/show.html.haml Code

4 Επιτρέψτε στους χρήστες να διαγράψουν σχόλια

4.1 Δυνατότητα διαγραφής στην ../controllers/comments_controller.rb

Με τη χρήση της `destroy` διαγράφουμε ένα συγκεκριμένο σχόλιο. Αρχικά βρίσκει το σχόλιο χρησιμοποιώντας την παράμετρο `id` και αναζητά τη σχετική φωτογραφία του σχολίου. Στη συνέχεια, η ενέργεια ελέγχει εάν ο τρέχων χρήστης είναι είτε αυτός που έγραψε το σχόλιο είτε ο κάτοχος της φωτογραφίας. Εάν

ο χρήστης έχει τα απαραίτητα δικαιώματα, διαγράφει το σχόλιο και ορίζει ένα μήνυμα επιτυχίας `flash`. Εάν ο χρήστης δεν έχει τα απαραίτητα δικαιώματα, ορίζει ένα μήνυμα φλας ειδοποίησης. Τέλος, η ενέργεια ανακατευθύνει τον χρήστη πίσω στην προηγούμενη σελίδα. Αυτό διασφαλίζει ότι τα σχόλια διαγράφονται μόνο από εξουσιοδοτημένους χρήστες και παρέχει τα κατάλληλα σχόλια στον χρήστη.

```

1  def destroy
2    @comment = Comment.find(params[:id])
3    @photo = @comment.photo
4    if @comment.user == current_user || @photo.user == current_user
5      @comment.destroy
6      flash[:notice] = "Comment deleted successfully."
7    else
8      flash[:alert] = "You can only delete your own comments or
9      comments on your photos."
10   end
11   redirect_to :back
12 end

```

Listing 13: Destroy method at `comments_controller.rb` Code

4.2 Χειρισμός διαγραφής σχολίων στην `views/users/show.html.haml`

Αυτό το τμήμα του κώδικα χειρίζεται τη διαγραφή των σχολίων. Συγκεκριμένα, είναι το `link_to` helper που δημιουργεί έναν σύνδεσμο διαγραφής για κάθε σχόλιο.

```

1  - photo.comments.each do |comment|
2    %p
3    %strong= comment.user.email
4    %br/
5    = comment.text
6    - if comment.user == current_user || photo.user == current_user
7      = link_to 'Delete', comment_path(comment), method: :delete, data
        : { confirm: 'Are you sure?' }, class: 'btn btn-danger btn-sm
        ', style: 'display: inline; margin-left: 10px;'

```

Listing 14: Delete comments at `../views/users/show.html.haml` Code

Αυτή η γραμμή ελέγχει εάν ο τρέχων χρήστης είναι είτε αυτός που έγραψε το σχόλιο είτε ο κάτοχος της φωτογραφίας.

```

1  - if comment.user == current_user || photo.user == current_user

```

Listing 15: Delete comments at `../views/users/show.html.haml` Code

Αυτή η γραμμή δημιουργεί έναν σύνδεσμο διαγραφής για κάθε σχόλιο. Το `comment_path(comment)` καθορίζει τη διαδρομή προς το σχόλιο που θα διαγραφεί. Το `confirm: 'Are you sure?'` προσθέτει ένα παράθυρο διαλόγου επιβεβαίωσης που ζητά από τον χρήστη να επιβεβαιώσει τη διαγραφή, συμβάλλοντας στην αποφυγή τυχαίων διαγραφών.

```

1  = link_to 'Delete', comment_path(comment), method: :delete, data: {
    confirm: 'Are you sure?' }, class: 'btn btn-danger btn-sm', style:
    'display: inline; margin-left: 10px;'

```

Listing 16: Delete comments at `../views/users/show.html.haml` Code

5 Μπόνους ερώτημα: Διαγραφή φωτογραφιών, σχολίων και ετικετών

5.1 Τροποποίηση μοντέλου Photo

Προσθέσαμε στις συμβολοσειρές `:tags` και `:comments` την εντολή `dependent: :destroy` διότι μας διασφαλίζει πως αν μια φωτογραφία θα διαγραφθεί, τότε και τα συσχετιζόμενα σχόλια και ετικέτες θα διαγραφθούν από τη βάση δεδομένων.

```

1 class Photo < ActiveRecord::Base
2   belongs_to :user
3   has_many :tags, dependent: :destroy
4
5   has_attached_file :image, :styles => { :medium => "300x300>", :thumb
     => "100x100>" }, :default_url => "/images/:style/missing.png"
6   validates_attachment_content_type :image, :content_type => /\Aimage
     \/.*\Z/
7   validates :title, presence: true
8   has_many :comments, dependent: :destroy
9 end

```

Listing 17: `dependent: :destroy` at `photo.rb` Code

5.2 Τροποποίηση του `photos_controller.rb`

Στην `photos_controller.rb` προσθέσαμε την μέθοδο `destroy` η οποία θα διαγράφει τις φωτογραφίες. Αρχικά, ο `controller` βρίσκει πρώτα τη φωτογραφία με το αναγνωριστικό του από τις παραμέτρους. Εάν η φωτογραφία ανήκει στον τρέχοντα χρήστη διαγράφει τη φωτογραφία μαζί με τα σχετικά σχόλια και ετικέτες χάρη στο `dependent: :destroy` που αναφέρθηκε παραπάνω. Τέλος, ανακατευθύνει τον χρήστη πίσω στην προηγούμενη σελίδα.

```

1   def destroy
2     @photo = Photo.find(params[:id])
3     if @photo.user == current_user
4       @photo.destroy
5       flash[:notice] = "Photo and its associated comments and tags
        deleted successfully."
6     else
7       flash[:alert] = "You can only delete your own photos."
8     end
9     redirect_to :back
10  end
11
12  private
13
14  def photo_params
15    params.require(:photo).permit(:image, :title)
16  end

```

Listing 18: `destroy` method at `photos_controller.rb`

5.3 Τροποποίηση της ../views/users/show.html.haml

Ο χρήστης θα πρέπει να μπορεί να διαγράφει μόνο τις δίκες του φωτογραφίες, όποτε στο μέρος του κώδικα που απεικονίζει τις φωτογραφίες του χρήστη (My Photos) προσθέσαμε μία συνθήκη `if` που ελέγχει αν ο ενεργός χρήστης είναι ο κάτοχος της φωτογραφίας. Αν είναι, τότε εμφανίζεται ένα κουμπί με όνομα `Delete Photo`, το οποίο αν το πατήσει ο χρήστης, του εμφανίζεται ένα μήνυμα προειδοποίησης αν θέλει να τη διαγράψει μαζί με τις ετικέτες και τα σχόλια και αν πατήσει ναι, η φωτογραφία μαζί με τις ετικέτες και τα σχόλια που έχει θα διαγραφθούν από τη βάση δεδομένων και θα ξαναφορτώσει στο χρήστη η ίδια σελίδα.

```
1 - if photo.user == current_user
2   = link_to 'Delete Photo', user_photo_path(@user, photo), method: :
    delete, data: { confirm: 'Are you sure you want to delete this
    photo and all its comments and tags?' }, class: 'btn btn-danger
    btn-sm', style: 'margin-top: 10px;'
```

Listing 19: Delete photos at ../views/users/show.html.haml

```
1 %h1{ style: "text-align: center; font-size: 50px;" } My photos
2 .row
3   %ul{ style: "text-align: left; font-size: 25px;" }
4     - @photos.each do |photo|
5       .well.col-sm-4
6         = image_tag photo.image.url(:medium)
7         %p
8           %strong= photo.title
9         %p
10          At:
11          = photo.created_at.strftime("%B %d, %Y %H:%M")
12        = form_for @tag do |f|
13          = f.hidden_field :photo_id, value: photo.id
14          = f.collection_select :user_id, @users, :id, :email
15          = f.submit "Tag User"
16        - photo.tags.each do |tag|
17          %p= tag.user.email
18        %h3
19        %strong
20        %u Comments
21        - photo.comments.each do |comment|
22          %p
23            %strong= comment.user.email
24            %br/
25            = comment.text
26          - if comment.user == current_user || photo.user ==
              current_user
27            = link_to 'Delete', comment_path(comment), method: :
                delete, data: { confirm: 'Are you sure?' }, class: '
                btn btn-danger btn-sm', style: 'display: inline;
                margin-left: 10px;';
28        = form_for Comment.new, url: comments_path do |f|
29          = f.hidden_field :photo_id, value: photo.id
30          .form-group
31            = f.label :text, "Add a comment:"
32            = f.text_area :text, class: "form-control", rows: 1
33            = f.submit "Post Comment", class: "btn btn-primary"
34          - if photo.user == current_user
35            = link_to 'Delete Photo', user_photo_path(@user, photo),
                method: :delete, data: { confirm: 'Are you sure you want
```

```
to delete this photo and all its comments and tags?' },  
class: 'btn btn-danger btn-sm', style: 'margin-top: 10px;  
,
```

Listing 20: Full version of My Photos at ../views/users/show.html.haml

6 Δοκιμή της ιστοσελίδας

Figure 1: Αρχικά συνδεόμαστε στην ιστοσελίδα με email:jim@gmail.com και κωδικό: 12345.

Log In

Email Password

Figure 2: Εφόσον συνδεθούμε στη πλατφόρμα, εμφανίζεται η αρχική οθόνη του χρήστη με τις φωτογραφίες του.

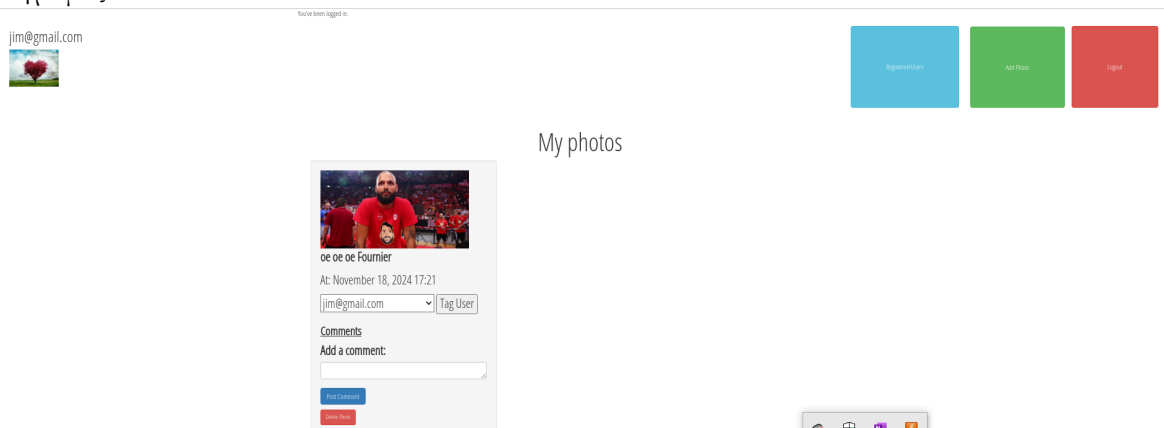


Figure 3: Όταν ο χρήστης πατήσει το κουμπί Add Photo, θα ανακατευθυνθεί στη σελίδα όπου εμφανίζεται η διαδικασία δημοσίευσης νέας φωτογραφίας και η φόρμα υποβολής τίτλου στη φωτογραφία.

Add a Photo

iborra.jpg
Give a title:

Figure 4: Όταν ο χρήστης πατήσει το κουμπί Upload, θα ανακατευθυνθεί στη αρχική του σελίδα όπου εμφανίζονται η ανεβασμένη φωτογραφία δίπλα στις υπόλοιπες δικές του φωτογραφίες με τον τίτλο από κάτω και ταξινομημένες σε αντίστροφη χρονολογική σειρά.

My photos



	
Leader!	oe oe oe Fournier
At: November 26, 2024 15:54	At: November 18, 2024 17:21
<input type="text" value="jim@gmail.com"/> <input type="button" value="Tag User"/>	<input type="text" value="jim@gmail.com"/> <input type="button" value="Tag User"/>
<u>Comments</u>	<u>Comments</u>
Add a comment:	Add a comment:
<input type="text" value="forza"/>	<input type="text"/>
<input type="button" value="Post Comment"/>	<input type="button" value="Post Comment"/>
<input type="button" value="Delete Photo"/>	<input type="button" value="Delete Photo"/>

Figure 5: Όταν ο χρήστης πατήσει το κουμπί **Registered Users**, θα ανακατευθυνθεί στη σελίδα όπου εμφανίζονται όλοι οι εγγεγραμμένοι χρήστες με το κουμπί **Follow** δίπλα στον κάθε χρήστη.

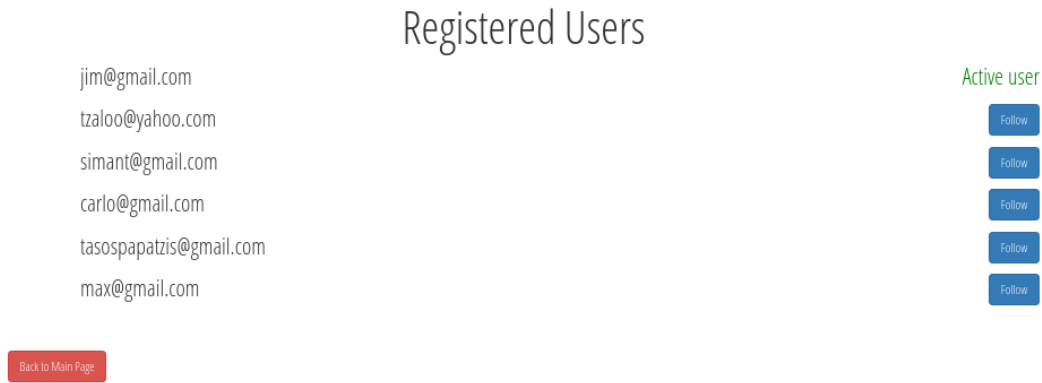


Figure 6: Όταν ο χρήστης θελήσει να ακολουθήσει κάποιον άλλον χρήστη πατάει το κουμπί **Follow** και όταν το πατήσει το κουμπί γίνεται άσπρο και θα γράφει **Following** και εμφανίζεται το μήνυμα επιβεβαίωσης πάνω αριστερά.

You are now following tasospapatzis@gmail.com

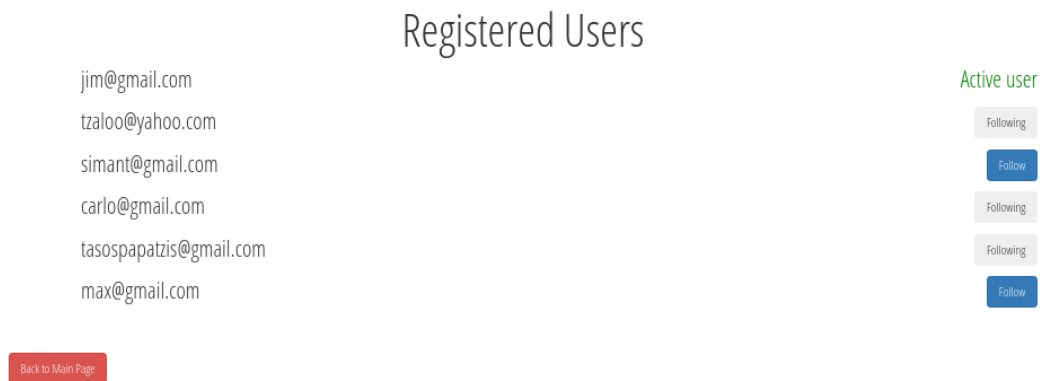



Figure 7: Όταν ο χρήστης θέλει να βάλει ένα σχόλιο σε μια δικιά του φωτογραφία, συμπληρώνει τη φόρμα που βρίσκεται κάτω από το Add a comment.

My photos



Leader!

At: November 26, 2024 15:54


▼ Tag User

Comments

Add a comment:

Post Comment

Delete Photo



oe oe oe Fournier

At: November 18, 2024 17:21

▼ Tag User

Comments

Add a comment:

Post Comment

Delete Photo

Figure 8: Όταν ο χρήστης θέλει να δημοσιεύσει ένα σχόλιο σε μια δικιά του φωτογραφία, πατάει το κουμπί Post Comment, το σχόλιο θα εμφανίζεται κάτω από τον τίτλο Comments και έχει τη δυνατότητα να διαγράψει το σχόλιο του με το κουμπί Delete.

Comment added successfully.

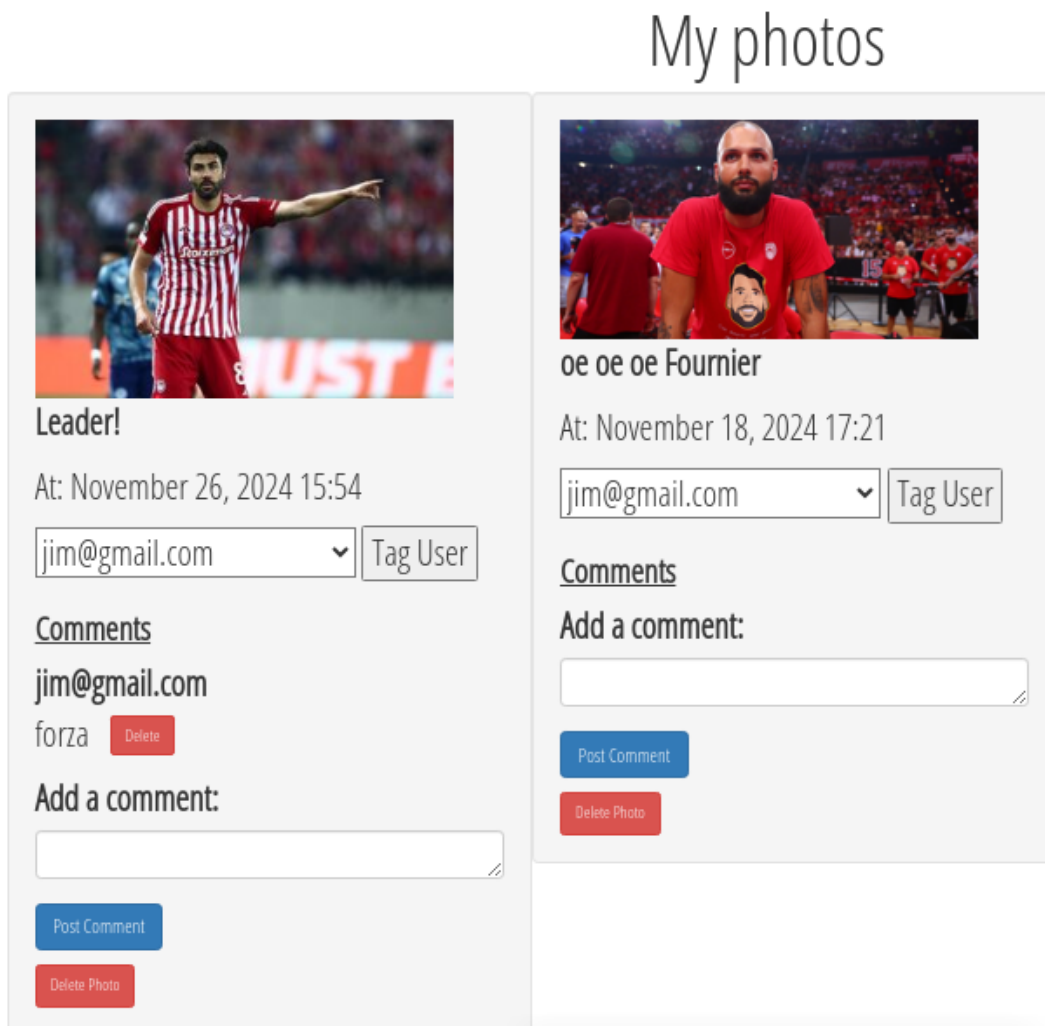



Figure 9: Παρατηρούμε πως λάβαμε το μήνυμα ότι το σχόλιό μας προστέθηκε επιτυχώς.

Comment added successfully.

Figure 10: Προσθέτουμε σχόλιο σε έναν χρήστη που ακολουθούμε

Photos of Followed Users




my love
By: tasospatzis@gmail.com
At: November 19, 2024 19:09

Comments
max@gmail.com
awww so cute

Add a comment:

Post Comment




2 mac parakalw
By: tzaloo@yahoo.com
At: November 19, 2024 18:35

Comments

Add a comment:

Post Comment




My favorite footballer
By: carlo@gmail.com
At: November 19, 2024 18:07

Comments
tzaloo@yahoo.com
you should sign him at Brescia

Add a comment:

Post Comment

Figure 11: Το σχόλιο δημοσιεύτηκε.



my love

By: tasospapatzis@gmail.com

At: November 19, 2024 19:09

Comments

max@gmail.com
awww so cute

jim@gmail.com
I prefer campari Delete

Add a comment:

Post Comment

Figure 12: Εάν πατήσουμε το κουμπί διαγραφής σχολίου μας εμφανίζεται το παρακάτω μήνυμα.

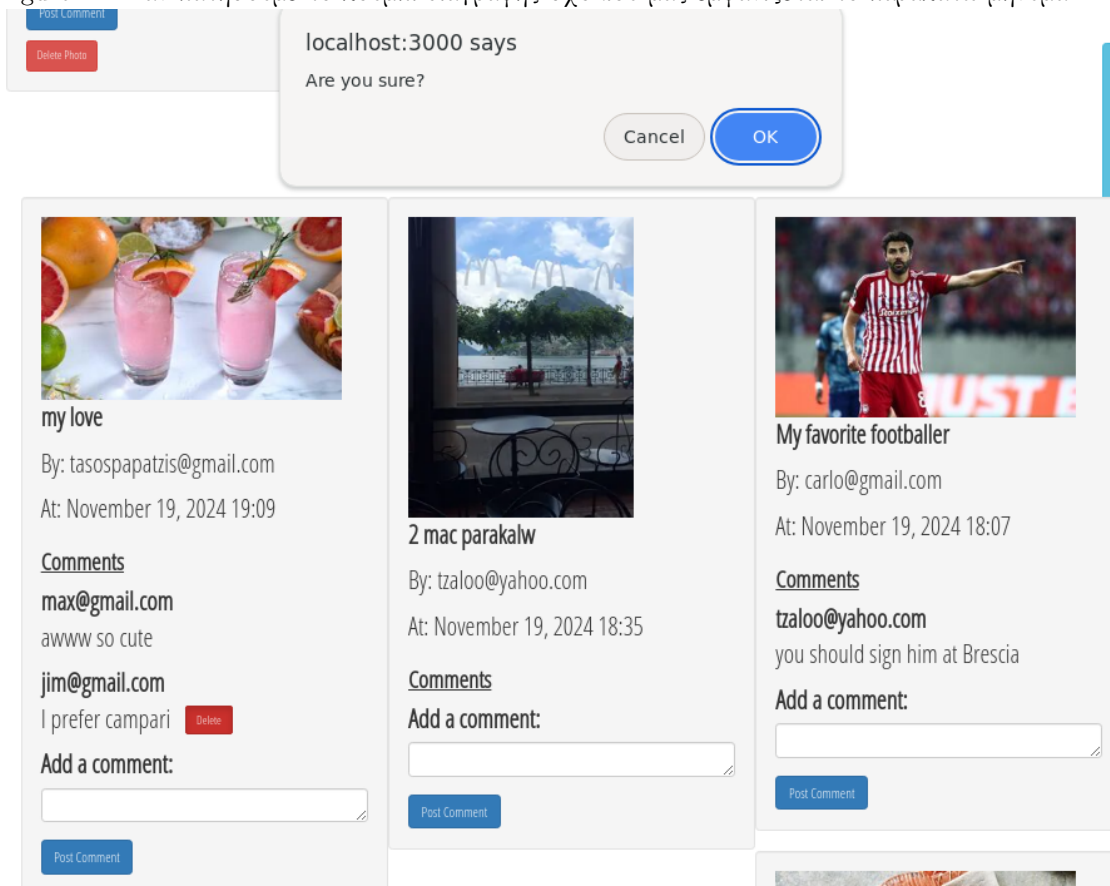


Figure 13: Το μήνυμα επιβεβαίωσης διαγραφής σχολίου.

Comment deleted successfully.

Figure 14: Έχουμε τη δυνατότητα να διαγράψουμε όλα τα σχόλια δικά μας και μη στις φωτογραφίες μας.

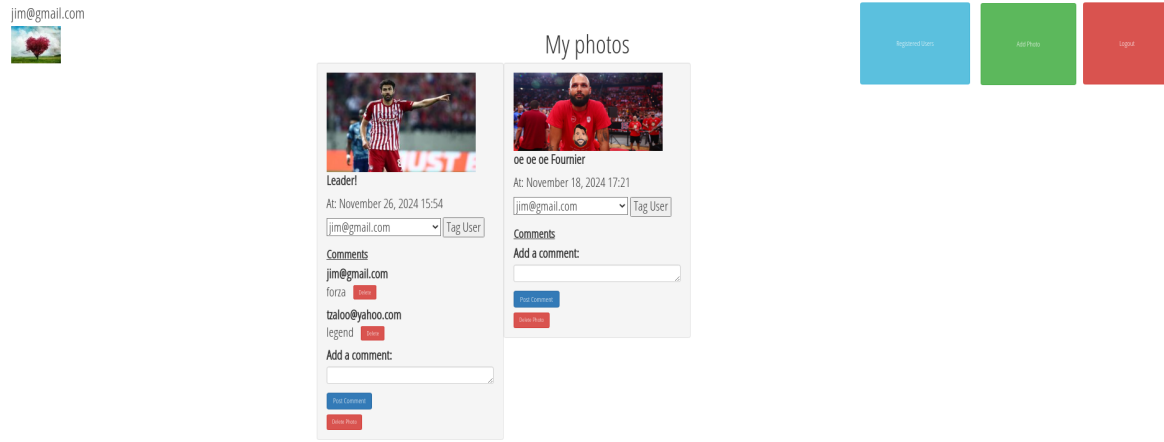


Figure 15: Μήνυμα που εμφανίζεται όταν επιλέγουμε να διαγράψουμε μια φωτογραφία μας.

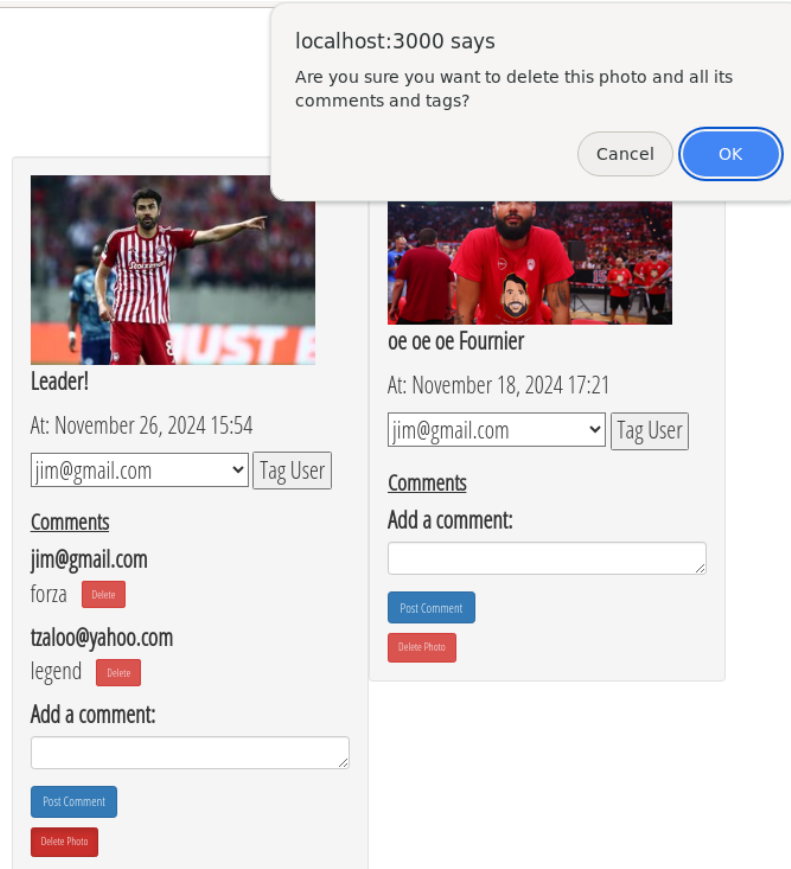


Figure 16: Όταν τελικά την διαγράφουμε.

Photo and its associated comments and tags deleted successfully.

My photos



7 Κάποια άλλα αρχεία που επεξεργαστήκαμε

1. Επεξεργασία του ../config/routes.rb

```
1 Rails.application.routes.draw do
2   get '/' => 'home#index'
3
4   resources :users do
5     resources :photos
6     collection do
7       get '/emails', to: 'users#emails', as: 'emails'
8     end
9   end
10
11  resources :comments, only: [:create, :destroy]
12
13  resources :follows, only: [:create, :destroy]
14  resources :tags, only: [:create, :destroy]
15
16  get '/log-in' => "sessions#new"
```

```
17   post '/log-in' => "sessions#create"  
18   get  '/log-out' => "sessions#destroy", as: :log_out  
19 end
```

Listing 21: ../config/routes.rb

2. Επεξεργασία του ../db/schema.db