ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ

Εφαρμογή ιστού για διαμοιρασμό φωτογραφιών στο Ruby-on-Rails.

Team

Γκόβαρης Χρήστος-Γρηγόριος
Σπανού Μαρία

Χρονοδιάγραμμα 1ης Εργαστηριακής Άσκησης

Έκδοση	Ημερομηνία	Progress
1.0	8/11/2024	Ανακοίνωση 1ης εργαστηριακής άσκησης
2.0	16/11 - 18/11	Υλοποίηση του ερωτήματος (1)
2.1	19/11/2024	Υλοποίηση του ερωτήματος (2)
2.2	20/11 - 21/11	Υλοποίηση του ερωτήματος (3)
2.3	22/11 - 23/11	Υλοποίηση του ερωτήματος (4)
2.4	23/11 - 24/11	Υλοποίηση του bonus ερωτήματος (5)
3.0	16/11 – 24/11	Υλοποίηση Report εργαστηριακής άσκησης

Ανάλυση Χαρακτηριστικών

Η εργασία υλοποιήθηκε σε ένα μηχάνημα (Lenovo Ideapad 3), με τα εξής χαρακτηριστικά πυρήνα:

- AMD Ryzen 7 (7730U)
- 8 CPU cores
- 16 Threads
- Boost Clock up to 4.5GHz
- Base Clock 2.0GHz
- CPU Socket FP6
- Intergraded Graphics (Radeon Graphics)

Επιπλέον, η **εργασία** υλοποιήθηκε σε **Debian 64-bit Linux**, χρησιμοποιώντας το **Mousepad**.

Αρχεία που τροποποιήθηκαν

Τροποποιήθηκαν τα παρακάτω αρχεία:

Models	Controllers	Views	Routes	Migrations
app/models/user.rb	app/controllers/users_controller.rb	app/views/users/show.html.haml	config/routes.rb	db/migrate/create_photos.rb
app/models/photo.rb	app/controllers/photos_controller.rb	app/views/users/new.html.haml		db/migrate/create_comments.rb
app/models/comment.rb	app/controllers/comments_controller.rb	app/views/photos/show.html.haml		db/migrate/create_follows.rb
app/models/follow.rb				db/migrate/add_title_to_photos.rb

2.0 Υλοποίηση του ερωτήματος (1)

Για το 1ο ερώτημα, προσθέσαμε τη δυνατότητα κάθε φωτογραφία να έχει τίτλο. Δημιουργήσαμε ένα migration (**add_title_to_photos.rb**) για τη στήλη title στον πίνακα photos και ενημερώσαμε το μοντέλο Photo για να περιλαμβάνει validation για τον τίτλο. Στη φόρμα ανέβασμα φωτογραφίας (**users/show.html.haml**), προσθέσαμε πεδίο για την εισαγωγή τίτλου, ενώ τροποποιήσαμε το **photo_params** στο **PhotosController** ώστε να αποθηκεύεται σωστά. Μετά τη migration, εμφανίσαμε τον τίτλο κάτω από κάθε φωτογραφία στη σελίδα του χρήστη και επαληθεύσαμε τη λειτουργικότητα μέσω δοκιμών. Πιο συγκεκριμένα, οι αλλαγές ήταν οι εξής:

app/models/photo.rb

```
validates :title, presence: true
```

Προστέθηκε για να διασφαλίσουμε ότι κάθε φωτογραφία πρέπει να έχει τίτλο κατά τη δημιουργία της.

db/migrate/add_title_to_photos.rb

```
class AddTitleToPhotos < ActiveRecord::Migration
  def change
    add_column :photos, :title, :string
  end
end
```

Η δημιουργία του **migration** για την προσθήκη της στήλης **title** στον πίνακα **photos** στη βάση δεδομένων.

app/controllers/ photos_controller.rb

```
def photo_params
  params.require(:photo).permit(:image, :title)
end
```

Ενημερώθηκε για να επιτρέπει την παράμετρο **title** κατά τη δημιουργία ή ενημέρωση μιας φωτογραφίας.

app/views/users/show.html.haml

```
%p
  %strong Title:
  = photo.title
```

Προστέθηκε για να εμφανίζει τον **τίτλο** της κάθε φωτογραφίας κάτω από την εικόνα της.

2.1 Υλοποίηση του ερωτήματος (2)

Στο 2ο ερώτημα, υλοποιήσαμε τη δυνατότητα **follow/unfollow** μεταξύ χρηστών, δημιουργώντας το μοντέλο **Follow** για τη διαχείριση σχέσεων **follower** και **followed**. Ενημερώσαμε το **routes.rb** για τις διαδρομές και τον **UsersController** για τη διαχείριση των ενεργειών. Στο **view**, προσθέσαμε κουμπιά **follow/unfollow** και εμφανίσαμε φωτογραφίες ακολουθούμενων χρηστών στη σελίδα χρήστη. Πιο συγκεκριμένα, οι αλλαγές ήταν οι εξής:

app/models/user.rb

```
has_many :follows_as_follower, class_name: 'Follow', foreign_key: 'follower_id'
has_many :followed_users, through: :follows_as_follower, source: :followed

has_many :follows_as_followed, class_name: 'Follow', foreign_key: 'followed_id'
has_many :followers, through: :follows_as_followed, source: :follower
```

Προστέθηκαν οι συσχετίσεις για ακολούθους (**followers**) και ακολουθούμενους (**followed_users**) χρήστες.

app/models/follow.rb

```
class Follow < ActiveRecord::Base
  belongs_to :follower, class_name: 'User', foreign_key: 'follower_id'
  belongs_to :followed, class_name: 'User', foreign_key: 'followed_id'

  validates :follower_id, presence: true
  validates :followed_id, presence: true
  validates :follower_id, uniqueness: { scope: :followed_id }
end
```

Δημιουργήθηκε το μοντέλο **Follow** για να αντιπροσωπεύει τις σχέσεις ακολουθίας μεταξύ χρηστών.

app/controllers/users_controller.rb

```
def follow
  user_to_follow = User.find(params[:id])
  if current_user.followed_users.include?(user_to_follow)
    flash[:alert] = "You are already following this user."
  else
    current_user.followed_users << user_to_follow
    flash[:notice] = "You are now following #{user_to_follow.email}!"
  end
  redirect_to user_path(current_user)
end

def unfollow
  user_to_unfollow = User.find(params[:id])
  if current_user.followed_users.include?(user_to_unfollow)
    current_user.followed_users.delete(user_to_unfollow)
    flash[:notice] = "You unfollowed #{user_to_unfollow.email}."
  else
    flash[:alert] = "You are not following this user."
  end
  redirect_to user_path(current_user)
end
```

Προστέθηκαν οι μέθοδοι **follow** και **unfollow** για να διαχειρίζονται τις λειτουργίες ακολούθησης και απεγγραφής.

app/views/users/show.html.haml

```
- if current_user != user
  - if current_user.followed_users.include?(user)
    = button_to 'Unfollow', unfollow_user_path(user), method: :delete, class: ['btn', 'btn-warning']
  - else
    = button_to 'Follow', follow_user_path(user), method: :post, class: ['btn', 'btn-primary']
```

Προστέθηκε το κουμπί **Follow/Unfollow** δίπλα από κάθε χρήστη.

db/migrate/create_follows.rb

```
class CreateFollows < ActiveRecord::Migration
  def change
    create_table :follows do |t|
      t.integer :follower_id, null: false
      t.integer :followed_id, null: false

      t.timestamps null: false
    end

    add_index :follows, :follower_id
    add_index :follows, :followed_id
    add_index :follows, [:follower_id, :followed_id], unique: true
  end
end
```

Δημιουργήθηκε το **migration** για τον πίνακα **follows**, ο οποίος αποθηκεύει τις σχέσεις ακολουθίας μεταξύ χρηστών.

config/routes.rb

```
resources :users do
  member do
    post :follow
    delete :unfollow
  end
end
```

Προστέθηκαν οι διαδρομές **follow** και **unfollow** για τους χρήστες.

2.2 Υλοποίηση του ερωτήματος (3)

Στο 3ο ερώτημα, υλοποιήσαμε τη δυνατότητα οι χρήστες να προσθέτουν και να διαγράφουν σχόλια σε φωτογραφίες. Δημιουργήσαμε το μοντέλο Comment για να διαχειρίζεται τα σχόλια, με συσχετίσεις με τα μοντέλα **Photo** και **User**. Προσθέσαμε το **CommentsController** για να διαχειρίζεται τις λειτουργίες δημιουργίας και διαγραφής σχολίων, με έλεγχο δικαιωμάτων ώστε ένας χρήστης να μπορεί να διαγράψει μόνο τα δικά του σχόλια ή σχόλια στις δικές του φωτογραφίες. Στη σελίδα προβολής χρηστών (**users/show.html.haml**), προσθέσαμε φόρμα για νέα σχόλια και λίστα για την εμφάνισή τους κάτω από κάθε φωτογραφία, με κουμπί διαγραφής για εξουσιοδοτημένους χρήστες. Τέλος, ενημερώσαμε το **routes.rb** για τις διαδρομές δημιουργίας και διαγραφής σχολίων. Πιο συγκεκριμένα, οι αλλαγές ήταν οι εξής:

app/models/comment.rb

```
class Comment < ActiveRecord::Base
  belongs_to :photo
  belongs_to :user

  validates :text, presence: true
end
```

Δημιουργήθηκε το μοντέλο **Comment** για να διαχειρίζεται τα σχόλια. Περιλαμβάνει τις σχέσεις **belongs_to** με το **Photo** και τον **User** και έναν **validation** κανόνα για το πεδίο **text**.

app/models/photo.rb

```
has_many :comments, dependent: :destroy
```

Προστέθηκε η σχέση **has_many :comments** στο μοντέλο **Photo** για να συνδέονται τα σχόλια με κάθε φωτογραφία. Ορίστηκε η επιλογή **dependent: :destroy** για να διαγράφονται αυτόματα τα σχόλια όταν διαγράφεται η φωτογραφία.

app/controllers/comments_controller.rb

```
def create
  @photo = Photo.find(params[:photo_id])
  @comment = @photo.comments.new(comment_params)
  @comment.user = current_user
  if @comment.save
    flash[:notice] = "Comment added successfully!"
  else
    flash[:alert] = "Failed to add comment. Please try again."
  end
  redirect_to user_path(@photo.user)
end

def destroy
  @comment = Comment.find(params[:id])
  @photo = @comment.photo
  if current_user == @comment.user || current_user == @photo.user
    @comment.destroy
    flash[:notice] = "Comment deleted successfully!"
  else
    flash[:alert] = "You are not authorized to delete this comment."
  end
  redirect_to user_path(@photo.user)
end

private

def comment_params
  params.require(:comment).permit(:text)
end
```

Δημιουργήθηκε ο **controller CommentsController** με τις μεθόδους **create** για την προσθήκη σχολίων και **destroy** για τη διαγραφή σχολίων. Περιλαμβάνει ελέγχους εξουσιοδότησης για διασφάλιση των δικαιωμάτων του χρήστη.

app/controllers/users_controller.rb

```
@comment = Comment.new
```

Προστέθηκε στη μέθοδο **show** για να δημιουργηθεί αντικείμενο **Comment**, το οποίο χρησιμοποιείται στη φόρμα σχολίων.

app/views/users/show.html.haml

```
%h3 Comments:
- if photo.comments.any?
  - photo.comments.each do |comment|
    %p
      %strong= comment.user.email
      = comment.text
      - if current_user == comment.user || current_user == photo.user
        = button_to 'Delete', photo_comment_path(photo, comment), method: :delete, data: { confirm: 'Are you sure?' }, class: ['btn', 'btn-danger', 'btn-sm']
- else
  %p No comments yet.

%h4 Add a Comment:
= form_for [photo, Comment.new] do |f|
  = f.text_area :text, placeholder: "Write your comment here...", rows: 3, class: "form-control"
  %br/
  = f.submit "Add Comment", class: ['btn', 'btn-primary']
```

Προστέθηκε τμήμα **HTML** για την εμφάνιση σχολίων και την προσθήκη νέου σχολίου μέσω φόρμας. Περιλαμβάνεται κουμπί διαγραφής σχολίων, ορατό μόνο στον δημιουργό ή στον ιδιοκτήτη της φωτογραφίας.

db/migrate/create_comments.rb

```
class CreateComments < ActiveRecord::Migration
  def change
    create_table :comments do |t|
      t.integer :photo_id
      t.integer :user_id
      t.text :text

      t.timestamps null: false
    end

    add_index :comments, :photo_id
    add_index :comments, :user_id
  end
end
```

Δημιουργήθηκε ο πίνακας **comments** στη βάση δεδομένων με πεδία για το αναγνωριστικό της φωτογραφίας, του χρήστη και το κείμενο του σχολίου. Προστέθηκαν **indexes** για τα πεδία **photo_id** και **user_id**.

config/routes.rb

```
resources :photos, only: [:destroy] do
  resources :comments, only: [:create, :destroy]
end
```

Προστέθηκε η διαδρομή για σχόλια, επιτρέποντας τη δημιουργία (**create**) και διαγραφή (**destroy**) μέσω των **routes**.

2.4 Υλοποίηση του ερωτήματος (4)

Στο 4ο ερώτημα, υλοποιήσαμε τη δυνατότητα διαγραφής σχολίων με συγκεκριμένα δικαιώματα. Προσθέσαμε τη μέθοδο destroy στον CommentsController, ώστε οι χρήστες να διαγράφουν σχόλια που έχουν γράψει ή σχόλια στις δικές τους φωτογραφίες. Στο view (users/show.html.haml), προστέθηκε κουμπί Delete δίπλα σε κάθε σχόλιο, το οποίο εμφανίζεται μόνο αν ο χρήστης έχει δικαίωμα. Ενημερώσαμε το routes.rb για να υποστηρίξει τη λειτουργία destroy για τα σχόλια. Πιο συγκεκριμένα, οι αλλαγές ήταν οι εξής:

app/controllers/comments_controller.rb

```
def destroy
  @comment = Comment.find(params[:id])
  @photo = @comment.photo

  # Check if the user has the right to delete the comment
  if current_user == @comment.user || current_user == @photo.user
    @comment.destroy
    flash[:notice] = "Comment deleted successfully!"
  else
    flash[:alert] = "You are not authorized to delete this comment."
  end
  redirect_to user_path(@photo.user)
end
```

Προστέθηκε ο κώδικας για να επιτρέπεται η διαγραφή σχολίων. Οι χρήστες μπορούν να διαγράψουν τα δικά τους σχόλια ή σχόλια στις δικές τους φωτογραφίες.

config/routes.rb

```
resources :photos, only: [:destroy] do
  resources :comments, only: [:create, :destroy] # Allow deletion of comments
end
```

Προστέθηκε η μέθοδος **destroy** για τα σχόλια στη διαδρομή **comments**, ώστε να υποστηρίζεται η λειτουργία διαγραφής.

views: users/show.html.haml

```
- if photo.comments.any?
  - photo.comments.each do |comment|
    %p
      %strong= comment.user.email
      = comment.text
      - if current_user == comment.user || current_user == photo.user
        = button_to 'Delete', photo_comment_path(photo, comment), method: :delete, data: { confirm: 'Are you sure?' }, class: ['btn', 'btn-danger', 'btn-sm']
- else
  %p No comments yet.
```

Προστέθηκε κουμπί διαγραφής δίπλα στα σχόλια. Το κουμπί είναι ορατό μόνο εάν ο τρέχων χρήστης είναι ο δημιουργός του σχολίου ή ο ιδιοκτήτης της φωτογραφίας.

2.5 Υλοποίηση του bonus ερωτήματος (5)

Στο 5ο ερώτημα, υλοποιήσαμε τη δυνατότητα διαγραφής φωτογραφιών από τον ιδιοκτήτη τους, μαζί με όλα τα σχόλια και τις ετικέτες που σχετίζονται με αυτές. Προσθέσαμε τη μέθοδο destroy στον PhotosController, με έλεγχο ώστε μόνο ο χρήστης που ανέβασε τη φωτογραφία να μπορεί να τη διαγράψει. Στο view (users/show.html.haml), προσθέσαμε κουμπί διαγραφής δίπλα σε κάθε φωτογραφία που εμφανίζεται μόνο αν ο τρέχων χρήστης είναι ο ιδιοκτήτης. Ενημερώσαμε επίσης το routes.rb για να περιλαμβάνει τη διαδρομή destroy για τις φωτογραφίες. Πιο συγκεκριμένα, οι αλλαγές ήταν οι εξής:

app/controllers/photos_controller.rb

```
def destroy
  @photo = Photo.find(params[:id])
  if current_user == @photo.user
    @photo.destroy
    flash[:notice] = "Photo and all related comments and tags were successfully deleted!"
  else
    flash[:alert] = "You are not authorized to delete this photo."
  end
  redirect_to user_path(current_user)
end
```

Προστέθηκε ο κώδικας για να επιτρέπεται η διαγραφή σχολίων. Οι χρήστες μπορούν να διαγράψουν τα δικά τους σχόλια ή σχόλια στις δικές τους φωτογραφίες.

app/views/users/show.html.haml

```
- if current_user == photo.user
  = button_to 'Delete Photo', photo_path(photo), method: :delete, data: { confirm: 'Are you sure? This will delete the photo and all its comments and tags.' }, class: ['btn', 'btn-danger', 'btn-sm']
```

Προστέθηκε κουμπί διαγραφής δίπλα στα σχόλια. Το κουμπί είναι ορατό μόνο εάν ο τρέχων χρήστης είναι ο δημιουργός του σχολίου ή ο ιδιοκτήτης της φωτογραφίας.

config/routes.rb

```
resources :photos, only: [:destroy] do
  resources :comments, only: [:create, :destroy]
end
```

Προστέθηκε η μέθοδος **destroy** για τα σχόλια στη διαδρομή **comments**, ώστε να υποστηρίζεται η λειτουργία διαγραφής.