Τεχνολογίες Διαδικτύου

Σωτηρλής Γιώργος, ΑΜ 2827

Μιμης Πάττας, ΑΜ 2331

2018 – 2019

Η εργασία έχει ως σκοπό τη δημιουργία μιας Διαδυκτιακής πλατφόρμας στην οποία δίνεται η δυνατότητα στους Χρήστες της να μοιράζονται περιεχόμενο, να σχολιάζουν φωτογραφίες και να ακολουθούν άλλους χρήστες.

Η αρχιτεκτονική της εφαρμογής στηρίζεται στο μοντέλο MVC, και αποτελείται απο τα εξής τμήματα:

- Server → Ruby (Rails)
- Client → HTML (Haml)
- Database → SQLite

Παρακάτω παρουσιάζονται οι αλλαγές που έχουν γίνει, σε κάθε τμήμα της εφαρμογής ξεχωριστά.

Database

Η βάση δεδομένων αποτελείται απο τους Πίνακες που συλλέγουν τα δεδομένα της εφαρμογής. Για τις ανάγκες της άσκησης, έχουν δημιουργηθεί 2 επιπλέον πίνακες, **Comments** και **Follows** , οι οποίοι περιέχουν τις πληροφορίες σχετικά με τα σχολία των χρηστών και τις σχέσεις των ακολουθούμενων χρηστών.

Το σχήμα του πίνακα **Comments** περιέχει τα πεδία **user_id , image_id , comment** , τα οποία συνδέουν το id του χρήστη με το id της φωτογραφίας , και το αντίστοιχο σχόλιο comment.

```
create_table "comments", force: :cascade do |t|
  t.integer "user_id"
  t.integer "image_id"
  t.string  "comment"
end
```

Comments Table

Το σχήμα του πίνακα **Follows** περιέχει τα πεδία **follower_id, following_id**, τα οποία συνδέουν το id του χρήστη follower_id με τον χρήστη που επιθυμεί να ακολουθήσει, με id following_id.

```
create_table "follows", force: :cascade do |t|
  t.integer "follower_id"
  t.integer "following_id"
end
```

Follows Table

Server Side

Ο κώδικας του server αποτελείται απο τους ελεγκτές (controllers) οι οποίοι είναι υπεύθυνοι για την εξυπηρέτηση των αιτημάτων που καταφθάνουν απο τον Client.

Έχουν δημιουργηθεί 2 αρχεία “**comments_controller.rb**” και “**follow_controller.rb**” , τα οποία περιέχουν τις συναρτήσεις **createComment** και **createFollow,** οι οποίες εκτελούν τις διαδικασίες δημιουργίας σχόλιου σε μια φωτογραφία και δημιουργία σχέσης ακολουθίας μεταξύ 2 χρηστών.

```
class CommentsController < ApplicationController
  def createComment
    @comment = Comment.create("image_id" => params[:image_id], "user_id" => params[:id], "comment" => params[:comment][:comment])
    redirect_to user_path(User.find(params[:id]))
    flash[:notice]= "You have successfully added a Comment."
  end
end
```

Συνάρτηση **createComment**

```
class FollowController < ApplicationController
  def createFollow
    @follow = Follow.create("follower_id" => params[:id], "following_id" => params[:following_id])
    if @follow.save
      redirect_to user_path(User.find(params[:id]))
      flash[:notice]= "You have successfully added a follower."
    else
      flash[:alert] = "There was a problem adding a follower.. Please try again."
    end
  end
end
```

Συνάρτηση **createFollow**

Επιπλέον, εχουν προστεθεί 3 διαδρομές, οι οποίες ενεργοποιούν τις συναρτήσεις **createComment, createFollow** kai **delete**.

```
post '/users/:id/all_users/:following_id' => 'follow#createFollow', as: 'follow'
post '/users/:id/posts/:image_id' => 'comments#createComment', as: 'comment'
get '/users/:id/posts/:image_id/delete' => 'photos#delete' , as: 'delete_photo'
```

Routes.db file

```
def delete
  @user = User.find(params[:id])
  @photo = Photo.find(params[:image_id])
  if Integer(params[:id]) == Integer(@photo.user_id)
    @tags = Tag.where(photo_id:params[:image_id])
    @comment = Comment.where(image_id:params[:image_id])
    @comment.each do |c|
      c.destroy
    end
    @tags.each do |t|
      t.destroy
    end

    @photo.destroy

    flash[:notice] = "photo '#{@photo.title}' deleted."
  end
  redirect_to user_path(@user)
end
```

Συνάρτηση **delete , photos_controller.rb** file

Τέλος, εχουν γίνει τροποποιήσεις στο αρχείο **users_controller.rb** , και οι αλλαγές αφορούν τις συναρτήσεις **posts , all_users** και **show.**

```
def show
  @users = User.all
  @user = User.find(params[:id])
  @comments = Comment.all
  @tag = Tag.new
end

def all_users
  @users = User.all
  @user = User.find(params[:id])
  begin
    @followers = Follow.where(following_id:params[:id])
  rescue ActiveRecord::RecordNotFound => e1
    @followers = []
  end
  begin
    @following = Follow.where(follower_id:params[:id])
  rescue ActiveRecord::RecordNotFound => e2
    @following = []
  end
end

def posts
  @users = User.all
  @user = User.find(params[:id])
  @photos= Photo.all
  @comments = Comment.all
  @tag = Tag.new
end
```

Client Side

Ο κώδικας του Client αποτελείται απο αρχεία τύπου Haml, τα οποία βοηθούν στη δημιουργία δυναμικού περιεχομένου Html, το οποίο τελικά προβάλεται στον χρήστη. Για την ολοκλήρωση της άσκησης, εχουν δημιουργηθεί τα αρχεία **all_users.html.haml , posts.html.haml** και έχει τροποποιηθεί το αρχείο **show.html.haml** .

Το αρχείο **all_users.html.haml** εμφανίζει όλους τους χρήστες της εφαρμογής και δίνει τη δυνατότητα στο συνδεδεμένο χρήστη να ακολουθήσει άλλους χρήστες.

Το αρχείο **posts.html.haml** εμφανίζει όλες τις φωτογραφίες των χρηστών της εφαρμογής, στοιχισμένες σε αντίστροφη χρονολογική σειρά σε σχέση με την εγγραφή τους στο σύστημα. Σε αυτή τη σελίδα, δίνεται η δυνατότητα στον χρήστη να προσθέσει Tags, σχόλια ή να διαγράψει μια φωτογραφία.

Το αρχείο **show.html.haml** εμφανίζει τη βασική σελίδα του χρήστη, στην οποία έχει προστεθεί το κομμάτι **My Posts,** το οποίο εμφανίζει όλες τις φωτογραφίες του.

Για το καλύτερο σχεδιασμό του γραφικού περιβάλλοντος, έχει χρησιμοποιηθεί η βιβλιοθήκη Bootstrap v4 , και έχουν δημιουργηθεί τα αρχεία μορφής **.css posts.css , home.css** και **all_users.css.**

Τέλος Παρουσίασης.

Σωτηρλής Γιώργος 2827

Μιμής Πάττας 2331