ΑΝΑΦΟΡΑ 2ης ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ

Δημήτριος Παγώνης, AM:4985
Δημήτριος Τζαλοκώστας, AM:4994

19 Δεκεμβρίου 2024

Περίληψη

1 Προβολή παρουσίασης φωτογραφιών

1.1 Δημιουργία της photosslideshow.js

1.1.1 Στην παρακάτω συνάρτηση αρχικοποιήσαμε το slideshow μέσα σε ένα συγκεκριμένο στοιχείο container.Το slideIndex είναι ένας δείκτης για την παρακολούθηση της τρέχουσας διαφάνειας, ανακτά και αντιστρέφει τον πίνακα με τα στοιχεία των slides ώστε να εμφανίζεται πρώτα η πιο πρόσφατη διαφάνεια και χρησιμοποιεί μια μεταβλητή slideInterval για την αποθήκευση του interval id για αυτόματες εναλλαγές. Η εσωτερική συνάρτηση showSlide διασφαλίζει ότι μόνο η διαφάνεια στον τρέχοντα δείκτη είναι ορατή, ενώ οι υπόλοιπες κρύβονται. Αρχικά, η πρώτη διαφάνεια εμφανίζεται με τη χρήση της showSlide(slideIndex), και οι επόμενες διαφάνειες εμφανίζονται κάθε 3 δευτερόλεπτα μέσω της setInterval, με το interval id του διαστήματος να αποθηκεύεται στο data attribute του container για μελλοντική χρήση.

```
1
2       function startSlideshow(container) {
3       var slideIndex = 0;
4       var slides = $(container).find('.slide').toArray();
5       var slideInterval;
6
7       function showSlide(index) {
8         slides.forEach((slide, i) => {
9           $(slide).css('display', i === index ? 'block' : 'none');
10        });
11      }
12
13      showSlide(slideIndex);
14
15      slideInterval = setInterval(function() {
16        slideIndex = (slideIndex + 1) % slides.length;
17        showSlide(slideIndex);
18      }, 3000);
19
20
21      $(container).data('slideInterval', slideInterval);
22  }
```

1.1.2 Η συνάρτηση stopSlideshow σταματά το slideshow μέσα στο καθορισμένο container, ανακτώντας το container ID που είναι αποθηκευμένο στο data attribute του container και στη συνέχεια χρησιμοποιεί το clearInterval(slideInterval) για να σταματήσει το slideshow

```
1
2     function stopSlideshow(container) {
3     var slideInterval = $(container).data('slideInterval');
4     clearInterval(slideInterval);
5   }
```

1.1.3 Εδώ ουσιαστικά δημιουργήσαμε έναν handler για να μπορούμε να χειριζόμαστε τις ενέργειες του ποντικιού όταν το τοποθετούμε πάνω σε μια εικόνα.Πιο συγκεκριμένα, ξεκινά την παρουσίαση διαφανειών όταν ο δείκτης του ποντικιού εισέρχεται στο στοιχείο .slideshow-container. Ανακτά το πλησιέστερο στοιχείο .slideshow-container χρησιμοποιώντας τη συνάρτηση:(this).closest('.slideshow-container'), εμφανίζει το αναδυόμενο παράθυρο της φωτογραφίας ορίζοντας την ιδιότητα CSS display σε block και στη συνέχεια ξεκινά την παρουσίαση διαφανειών καλώντας τη συνάρτηση startSlideshow(slideshowContainer).

```
1     $(document).on('mouseenter', '.slideshow-container', function() {
2     var slideshowContainer = $(this).closest('.slideshow-container');
3     var photoPopup = slideshowContainer.find('.photo');
4     photoPopup.css('display', 'block');
5     startSlideshow(slideshowContainer);
6   });
```

1.1.4 Αυτός ο χειριστής σταματά την παρουσίαση διαφανειών όταν ο δείκτης του ποντικιού εξέρχεται από το στοιχείο .slideshow-container. Ανακτά το τρέχον στοιχείο .slideshow-container χρησιμοποιώντας τη συνάρτηση: $(this), κρύβει το αναδυόμενο παράθυρο της φωτογραφίας ορίζοντας την ιδιότητα CSS display σε none και σταματά την παρουσίαση διαφανειών καλώντας τη συνάρτηση stopSlideshow(slideshowContainer).

```
1     $(document).on('mouseleave', '.slideshow-container', function() {
2     var slideshowContainer = $(this);
3     var photoPopup = slideshowContainer.find('.photo');
4     photoPopup.css('display', 'none');
5     stopSlideshow(slideshowContainer);
6   });
```

1.2 Αλλαγές στη users_controller

1.2.1 Προσθέσαμε την @followed_users, η οποία ανακτά τη λίστα των χρηστών που ακολουθεί ο τρέχων χρήστης μαζί με τις σχετικές φωτογραφίες του. Αυτή η προσθήκη συμπληρώνει την υπάρχουσα μεταβλητή @followed_photos, η οποία ανακτά όλες τις φωτογραφίες από χρήστες που ακολουθούν.

```
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_users = @user.followed_users.includes(:photos)
8     @followed_photos = Photo.joins(user: :followers)
9                             .where(follows: { follower_id:
                                 current_user.id })
10                            .order(created_at: :desc)
11                            .distinct
12                            .includes(:comments)
13    end
```

1.3 Αλλαγές στην show.haml.html

Οι αλλαγές που κάναμε στην show.haml.html είναι κυρίως στις κατηγορίες «.slideshow-container» και «.slide». Το ".slideshow-container" συγκρατεί όλες τις διαφάνειες και διαχειρίζεται τη συμπεριφορά της παρουσίασης. Κάθε «.slide» αντιπροσωπεύει μια μεμονωμένη διαφάνεια μέσα στο κοντέινερ, που περιέχει μια εικόνα και μια λεζάντα και είναι αρχικά κρυφή. Η κλάση '.photo' χρησιμοποιείται για το αναδυόμενο παράθυρο φωτογραφιών που εμφανίζεται όταν η παρουσίαση είναι ενεργή. Αυτά τα στοιχεία χρησιμοποιούνται τόσο στην ενότητα φωτογραφιών του τρέχοντος χρήστη όσο και στην ενότητα φωτογραφιών των χρηστών που ακολουθούν για τη δημιουργία και την εμφάνιση των προβολών διαφανειών.

1.3.1 .slideshow-container:

```
1     .slideshow-container{ data: { user_id: @current_user.id }}
```

1.3.2 .photo:

```
1   .photo{ style:"display:none;position:absolute;top:0;left:0;width
       :100%;height:100%;background:white;z-index:10;" }
```

1.3.3 .slide:

```
1   .slide{ style: "display:none" }
2    %img{ src: photo.image.url(:medium), alt: photo.title, style: "
        width:100%", class: "photo-click␣recent-user-photo", data: {
        photo_id: photo.id, url: comments_photo_path(photo), user_email:
         @current_user.email  } }
3   %div.caption{style: "text-align:center;font-size:20px;"}= photo.
       title
```

1.3.4 Photos of Followed Users

```
1   %div{ style: "margin-top:␣50px;" }
2   %h1{ style: "text-align:␣center;␣font-size:␣50px;" } Photos of
       Followed Users
3   .photos-container{ style: "display:␣flex;␣flex-wrap:␣wrap;" }
4     - @followed_users.order(created_at: :desc).each do |followed_user|
5       %div{ style: "margin-right:␣20px;" }
6         %h2{ style: "text-align:␣center;" }= followed_user.email
7         - if followed_user.photos.any?
8           - recent_photo = followed_user.photos.order(created_at: :
                desc).first
9           .slideshow-container{ data: { user_id: followed_user.id } }
```

```
10            %img{ src: recent_photo.image.url(:medium), alt:
                  recent_photo.title, class: "photo-click", data: {
                  photo_id: recent_photo.id, url: comments_photo_path(
                  recent_photo), user_email: followed_user.email } }
11            .photo{ style: "display:␣none;␣position:␣absolute;␣top:␣0;
                  ␣left:␣0;␣width:␣100%;␣height:␣100%;␣background:␣white;
                  ␣z-index:␣10;" }
12              - followed_user.photos.order(created_at: :desc).
                    each_with_index do |photo, index|
13                .slide{ style: "display:none" }
14                  %img{ src: photo.image.url(:medium), alt: photo.
                        title, style: "width:100%", class: "photo-click",
                         data: { photo_id: photo.id, url:
                        comments_photo_path(photo), user_email:
                        followed_user.email } }
15                  %div.caption{style: "text-align:␣center;␣font-size:␣
                        20px;"}= photo.title
16        - else
17          %p No photos to display.
```

2 Αναδυόμενο παράθυρο με σχόλια και πλαίσιο κειμένου

2.1 Αλλαγές στη routes.rb

Η κύρια αλλαγή μεταξύ της παλιάς και της νέας διαδρομής είναι η προσθήκη ενός nested "photos" με μια διαδρομή μέλους για τα "comments". Αυτή η νέα διαδρομή επιτρέπει τη λήψη σχολίων για μια συγκεκριμένη φωτογραφία, ενισχύοντας την ικανότητα της εφαρμογής να χειρίζεται πιο αποτελεσματικά αιτήματα που σχετίζονται με σχόλια φωτογραφιών. Οι υπόλοιπες διαδρομές παραμένουν αμετάβλητες, διατηρώντας την υπάρχουσα δομή για τους χρήστες, τα σχόλια, τους ακόλουθους, τις ετικέτες και τη διαχείριση περιόδων σύνδεσης.

```
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 :photos do
12    member do
13      get 'comments', to: 'photos#comments', as: 'comments'
14    end
15  end
16
17  resources :comments, only: [:create, :destroy]
18  resources :follows, only: [:create, :destroy]
19  resources :tags, only: [:create, :destroy]
20
21  get '/log-in' => "sessions#new"
22  post '/log-in' => "sessions#create"
23  get '/log-out' => "sessions#destroy", as: :log_out
24 end
```

2.2 Δημιουργία της _comments.html.haml

Δημιουργήσαμε την _comments.html.haml για καλύτερο χειρισμό των σχολίων στο αναδυόμενο παράθυρο.Πιο συγκεκριμένα το σχεδιάσαμε ώστε να εμφανίζει μια λίστα σχολίων που σχετίζονται με μια φωτογραφία. Ξεκινά με μια επικεφαλίδα "Σχόλια" και ελέγχει εάν υπάρχουν σχόλια προς εμφάνιση. Για κάθε σχόλιο, εμφανίζει το email του χρήστη που έκανε το σχόλιο και το κείμενο του σχολίου. Επιπλέον, εάν ο τρέχων χρήστης είναι είτε ο συγγραφέας του σχολίου είτε ο κάτοχος της φωτογραφίας, παρέχεται ένα κουμπί "Διαγραφή" δίπλα στο σχόλιο. Αυτό το κουμπί επιτρέπει στο χρήστη να διαγράψει το σχόλιο, με μια προτροπή επιβεβαίωσης για την αποφυγή τυχαίων διαγραφών. Το κουμπί διαγραφής είναι διαμορφωμένο ως ένα μικρό, κόκκινο κουμπί και εμφανίζεται ενσωματωμένα με το κείμενο του σχολίου.Κάτω από τη λίστα των σχολίων, υπάρχει μια φόρμα για την προσθήκη νέου σχολίου, η οποία περιλαμβάνει ένα κρυφό πεδίο για το αναγνωριστικό φωτογραφίας, μια περιοχή κειμένου για το κείμενο σχολίου και ένα κουμπί υποβολής. Η φόρμα έχει ρυθμιστεί να υποβάλλεται μέσω AJAX.

```
   %h3
  %strong
    %u Comments

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


= form_for Comment.new, url: comments_path, remote: true, html: {
   class: 'new_comment', id: 'comment-form' } do |f|
  = f.hidden_field :photo_id, value: photo.id
  .form-group
    = f.label :text, 'Add a comment:'
    = f.text_area :text, class: 'form-control', rows: 1
    = f.submit 'Post Comment', class: 'btn btn-primary'
```

2.3 Αλλαγές στην comments_controller.rb

Οι ενημερώσεις στο CommentsController περιλαμβάνουν την προσθήκη render json οπου ψάχνει τη φωτοφραφία και δημιουργέι το σχόλιο και επιχειρεί να το αποθηκεύσει. Όταν ένα σχόλιο δημιουργείται με επιτυχία, επιστρέφει το σχόλιο ως αντικείμενο JSON. Εάν η δημιουργία σχολίου αποτύχει, επιστρέφει τα μηνύματα σφάλματος ως αντικείμενο JSON, επιτρέποντας τον χειρισμό σφαλμάτων επικύρωσης από την πλευρά του πελάτη και τα σχόλια των χρηστών.Η ενέργεια καταστροφής βρίσκει το σχόλιο και τη σχετική φωτογραφία, ελέγχει εάν ο τρέχων χρήστης έχει ανεβάσει τη φωτογραφία και διαγράψει το σχόλιο απαντώντας με ανακατεύθυνση ή απάντηση JSON που υποδηλώνει επιτυχία.

```
     ef create
    @photo = Photo.find(params[:comment][:photo_id])
    @comment = @photo.comments.build(comment_params)
    @comment.user = current_user

    if @comment.save
      render json: {
```

```
8         html: render_to_string(partial: 'users/comments', locals: {
              photo: @photo, comments: @photo.comments.includes(:user) },
               formats: [:html]),
9         text: @comment.text
10      }, status: :created
11    else
12      render json: { errors: @comment.errors.full_messages }, status:
           :unprocessable_entity
13    end
14  end
15
16
17  def destroy
18    @comment = Comment.find(params[:id])
19    @photo = @comment.photo
20    if @comment.user == current_user || @photo.user == current_user
21      @comment.destroy
22      respond_to do |format|
23        format.html { redirect_to request.referrer, notice: 'Comment␣
              was␣successfully␣deleted.' }
24        format.json { head :no_content }
25      end
26    end
27  end
```

2.4 Αλλαγές στην photos_controller

Οι κύριες αλλαγές περιλαμβάνουν την προσθήκη της δράσης show για την εμφάνιση μιας συγκεκριμένης φωτογραφίας, και την προσθήκη της δράσης comments για την απόκτηση των σχολίων μιας φωτογραφίας περιλαμβάνοντας και τουε χρήστες που κάνουν τα σχόλια. Αυτές οι αλλαγές βελτιώνουν τη λειτουργικότητα και την ευελιξία του PhotosController.

```
1  class PhotosController < ApplicationController
2    def create
3      @user = User.find(params[:user_id])
4      if params[:photo].nil?
5        flash[:alert] = "Please␣upload␣a␣photo"
6        redirect_back(fallback_location: user_path(@user))
7      else
8        @photo = Photo.create(photo_params)
9        @photo.user_id = @user.id
10       @photo.save
11       flash[:notice] = "Successfully␣uploaded␣a␣photo"
12       redirect_to user_path(@user)
13     end
14   end
15
16   def show
17     @photo = Photo.find(params[:id])
18   end
19
20   def new
21     @user = User.find(params[:user_id])
22     @photo = Photo.create()
23   end
24
```

```
25    def destroy
26      @photo = Photo.find(params[:id])
27      if @photo.user == current_user
28        @photo.destroy
29        flash[:notice] = "Photo␣and␣its␣associated␣comments␣and␣tags␣
              deleted␣successfully."
30      else
31        flash[:alert] = "You␣can␣only␣delete␣your␣own␣photos."
32      end
33      redirect_to :back
34    end
35
36
37    def comments
38      @photo = Photo.find(params[:id])
39      @comments = @photo.comments.includes(:user)
40      render partial: 'users/comments', locals: { photo: @photo,
            comments: @comments }
41    end
42
43
44
45    private
46
47    def photo_params
48      params.require(:photo).permit(:image, :title)
49    end
50  end
```

2.5 Αλλαγές στην show.html.haml

Προσθέσαμε την ετικέτα #comments-popup για να αναπαράγονται τα σχόλια της κάθε φωτογραφίας όταν το popup είναι ανοιχτό. Αρχικά, το παράθυρο αυτό είναι κρυμμένο και σχεδιασμένο ώστε να καλύπτει ένα σημαντικό τμήμα της οθόνης, με ημιδιαφανές μαύρο φόντο και λευκό κείμενο, με σκοπό να επικαλύπτει βασικό τμήμα της οθόνης.

Η αναδυόμενη κεφαλίδα περιλαμβάνει ένα κουμπί κλεισίματος με την ετικέτα #close-comments. Ο κώδικας επαναλαμβάνεται πάνω από τη συλλογή των φωτογραφιών κάθε χρήστη. Για κάθε φωτογραφία, εάν η φωτογραφία είναι παρούσα, αποδίδεται #render partial 'users/comments' με τοπικές μεταβλητές τη φωτογραφία και τα σχόλιά της, περιλαμβανομένου του κάθε χρήστη που έγραψε το κάθε σχόλιο, στο αρχείο ../app/views/users/$_c$*omments.html.haml*.

```
1   #comments-popup{ style: "display: none; position: fixed; top: 10%;
        left: 10%; width: 80%; height: 80%; background: rgba(0, 0, 0, 0.8);
        color: white; z-index: 20; overflow: auto;" }
2     .popup-header{ style: "position:␣relative;␣padding:␣10px;" }
3       %h5{ style: "margin:␣0;" } Photo Comments
4       %button#close-comments &times;
5     - @photos.each do |photo|
6       - if photo.present?
7         #comments-container
8           = render partial: 'users/comments', locals: { photo: photo,
                comments: photo.comments.includes(:user) }
9       - else
10        %p No comments to display.
```

2.6 Αλλαγές στην photosslideshow.js

2.6.1 Αλλαγές στα mouse enter, mouseleave

Αρχικά πήγαμε και προσθέσαμε στους παραπάνω χειριστές μια συνθήκη if με την οποία ελέγχουμε αν το αναδυόμενο παράθυρο ειναι ορατό. Αν είναι, το slideshow δεν θα ξεκινήσει ποτέ , ενώ αν δεν είναι ορατό το παράθυρο, τότε θα ξεκινήσει ή θα σταματήσει αντιστοίχως το slideshow με το τρόπο που εξηγήσαμε στα 1.1.3 και 1.1.4 αντίστοιχα.

```
1
2   $(document).on('mouseenter', '.slideshow-container', function() {
3     var slideshowContainer = $(this).closest('.slideshow-container');
4     var photoPopup = slideshowContainer.find('.photo');
5     if (!$('#comments-popup').is(':visible')) {
6       photoPopup.css('display', 'block');
7       startSlideshow(slideshowContainer);
8     }
9   });
10
11  $(document).on('mouseleave', '.slideshow-container', function() {
12    var slideshowContainer = $(this);
13    var photoPopup = slideshowContainer.find('.photo');
14    if (!$('#comments-popup').is(':visible')) {
15      photoPopup.css('display', 'none');
16      stopSlideshow(slideshowContainer);
17    }
18  });
```

2.6.2 Προσθήκη του αντικειμένου PhotoPopUp

Στο κώδικα μας προσθέσαμε το αντικειμένο PhotoPopUp, το οποίο διαχειρίζεται την εμφάνιση και την αλληλεπίδραση του αναδυόμενου παραθύρου σχολίων μιας φωτογραφίας μέσα στη προβολή παρουσίασης των φωτογραφιών.Μέσα στο αντικείμενο, ορίζουμε πεντε συναρτήσεις και δυο μεταβλητές τις newCommentHtml και isSetup.Η συνάρτηση setup ορίζει το αναδυόμενο παράθυρο και setup με στόχο να υποβάλλει νεα σχόλια μεσω AJAX, αλλά και για να διαγράψει σχόλια με τον ίδιο τρόπο.

```
1     var PhotoPopup = {
2
3   newCommentHtml: '',
4   isSetup: false,
5
6   setup: function() {
7
8     if (this.isSetup) {
9       console.log('Setup␣already␣done,␣skipping.');
10      return;
11    }
12    var popupDiv = $('<div␣id="comments-popup"></div>');
13    popupDiv.hide().appendTo($('body'));
14
15    $(document).off('submit', '.new_comment').on('submit', '.
          new_comment', function(event) {
16      event.preventDefault();
17      const $form = $(this);
18      $.ajax({
19        type: 'POST',
20        url: '/comments',
21        data: $form.serialize(),
```

```
22          dataType: 'json',
23          success: function(data) {
24            $('#comments-container').html(data.html);
25
26            $form[0].reset();
27          },
28          error: function(xhrObj, textStatus, exception) {
29            console.error('Error saving comment:', exception);
30            alert('Error saving comment.');
31          }
32        });
33      });
34
35      $(document).off('click', '.btn-danger.delete-comment').on('click',
            '.btn-danger.delete-comment', function(event) {
36        event.preventDefault(); // Prevent default anchor behavior
37        event.stopPropagation(); // Stop the event from propagating
38
39        var buttondangercontainer = $(this).closest('.btn-danger');
40        const commentId = $(buttondangercontainer).data('comment-id');
41        console.log('Deleting comment:', commentId);
42        $.ajax({
43          type: 'DELETE',
44          url: '/comments/' + commentId,
45          dataType: 'json',
46          success: function(data) {
47            if (data && data.errors) {
48              alert("Failed to delete comment: " + data.errors.join(", "
                    ));
49            } else {
50              location.reload();
51              //$('#comments-container').html(data.html);
52              //$('#comments-popup').show();
53            }
54          }
55        });
56      });
57
58      this.isSetup = true;
59    },
```

Στην συνέχεια, ορίζουμε τη συνάρτηση getPhotoInfo, η οποία δέχεται τα δεδομένα μιας φωτογραφίας και τα αναπαριστά στο αναδυόμενο παράθυρο μέσω AJAX.

```
1     getPhotoInfo: function(photoId, photoUrl, commentsUrl,
          photoCaption, userEmail) {
2
3     console.log('Using URL:', commentsUrl); // Log the URL
4     console.log('Caption:', photoCaption); // Log the caption
5     console.log('User Email:', userEmail); // Log the user email
6     $.ajax({
7       type: 'GET',
8       url: commentsUrl,
9       dataType: 'html',
10      timeout: 5000,
11      success: function(data) {
12        PhotoPopup.showPhotoInfo(data, photoUrl, photoCaption, photoId
```

```
            , userEmail);
13        },
14        error: function(xhrObj, textStatus, exception) {
15          alert('Error␣fetching␣comments.');
16        }
17      });
18      return false;
19    },
```

Επιπλέον, ορίζουμε τη συνάρτηση showPhotoInfo, η οποία αναπαριστά το αναδυόμενο παράθυρο με τα δεδομένα και τα σχόλια της φωτογραφίας.

```
1
2     showPhotoInfo: function(data, photoUrl, photoCaption, photoId,
          userEmail) {
3       // Center a floater 1/2 as wide and 1/4 as tall as screen
4       var oneFourth = Math.ceil($(window).width() / 4)
5       $('#comments-popup')
6         .css({
7           'left': oneFourth,
8           'width': 2 * oneFourth,
9           'top': 100,
10          'background-color': 'rgba(0,␣0,␣0,␣0.8)', // Set background
                color to light black (semi-transparent black)
11          'color': 'white', // Set text color to white for better
                readability
12          'position': 'fixed',
13        })
14        .html('<div␣class="popup-header"␣style="position:␣relative;␣
              padding:␣10px;">' +
15          '<h5␣style="margin:␣0;">Photo␣Comments</h5>' +
16          '<button␣id="close-comments"␣type="button"␣style="position:␣
                absolute;␣top:␣10px;␣right:␣10px;␣background:␣none;␣border:
                ␣none;␣color:␣white;␣font-size:␣50px;␣cursor:␣pointer;">&
                times;</button>' +
17          '</div>' +
18          '<img␣src="' + photoUrl + '"␣style="width:50%;␣display:block;␣
                margin:␣20px␣auto␣10px␣auto;">' +
19          '<div␣class="caption"␣style="text-align:␣center;␣font-size:␣20
                px;␣margin-top:␣10px;">' + photoCaption + '</div>' +
20          '<div␣id="comments-container"␣style="margin-top:␣20px;">' +
                data + '</div>'
21        )
22        .show();
23
24
25      $('.slideshow-container').each(function() {
26        var slideshowContainer = $(this);
27        PhotoPopup.stopSlideshow(slideshowContainer);
28      });
29      $('#close-comments').off('click').on('click', PhotoPopup.
            hidePhotoInfo);
30
31      return false;
32    },
```

Τέλος, ορίζουμε τις συναρτήσεις hidePhotoInfo, η οποία κρύβει το αναδυόμενο παράθυρο κι συνεχίζει τη προβολή των φωτογραφιών που έχουμε ορίσει. Και η συνάρτηση stopSlideShow, η οποία ορίζεται με τον ίδιο τρόπο όπως αναφέρεται και στο 1.1.2 και ελέγχει επιπρόσθετα σε ποια φωτογραφία σταμάτησε το slideShow και όποτε κλείσει το αναδυόμενο παράθυρο, καλεί τη μέθοδο startSlideShow, η οποία ξεκινά το slideShow απο το σημείο που σταμάτησε.

```
    hidePhotoInfo: function() {
    $('#comments-popup').hide();
    $('.slideshow-container').each(function() {
      var slideshowContainer = $(this);
      if (slideshowContainer.is(':hover')) {
        startSlideshow(slideshowContainer);
      }
    });
    return false;
  },

  stopSlideshow: function(container) {
    var slideInterval = $(container).data('slideInterval');
    clearInterval(slideInterval);
    var currentSlideIndex = $(container).find('.slide:visible').index
       ();
    $(container).data('slideIndex', currentSlideIndex);
  }

};
```

2.6.3 Προσθήκη χειριστή για κλικ φωτογραφίας από το slideshow.

Στο συγκεκριμένο χειριστή, ελέγχουμε αν ο χρήστης εχει πατήσει κλικ σε φωτογραφία σε slideshow μέσα στην ετικέτα slideshow-container.Αν το έχει κάνει, τότε σταματάει το κατευθείαν το slideshow και λαμβάνει τα δεδομένα της φωτογραφίας μέσω της ../app/views/users/show.html.haml και καλεί τη συνάρτηση

ώστε να ανοίξει το αναδυόμενο παράθυρο.

```
     $(document).on('click', '.slideshow-container', function() {
    var slideshowContainer = $(this).closest('.slideshow-container');
    stopSlideshow(slideshowContainer);
    var currentSlide = slideshowContainer.find('.slide:visible');
    var photoId = currentSlide.find('img.photo-click').data('photo-id'
       );
    var photoUrl = currentSlide.find('img.photo-click').attr('src');
    var commentsUrl = currentSlide.find('img.photo-click').data('url')
       ;
    var photoCaption = currentSlide.find('.caption').text();
    var userEmail = currentSlide.find('img.photo-click').data('user-
       email');
    PhotoPopup.getPhotoInfo(photoId, photoUrl, commentsUrl,
       photoCaption, userEmail);
  });
```

2.6.4 Προσθήκη χειριστή για κλείσιμο αναδυόμενου παραθύρου.

Στο συγκεκριμένο χειριστή, ελέγχουμε αν ο χρήστης πατήσει το εικονίδιο για κλείσιμο του αναδυόμενου παραθύρου. Αυτός ο χειριστής μας διασφαλίζει πως όταν το αναδυόμενο παράθυρο κλείσει, το slideshow θα συνεχίσει από τη φωτογραφία που σταμάτησε και τέλος καλούμε τη setup συνάρτηση του αντικειμένου PhotoPopUp.

```
     $(document).off('click', '#close-comments').on('click', '#close-
        comments', function() {
    $('#comments-popup').hide();
    $('.slideshow-container').each(function() {
      startSlideshow($(this));
    });
  });




});


$(PhotoPopup.setup);
```

2.7 Αλλαγές στην ../app/assets/stylesheets/application.sass

Οι προσθέσεις που κάναμε ειναι:

```
.slideshow-container
  position: relative
  max-width: 100%
  margin: auto


.slideshow-inner
  display: flex


.photo
  display: inline-block
  margin-right: 10px
```

3 Λειτουργία διαγραφής φωτογραφίας με διπλό κλικ.

3.1 Διαχωρισμός διπλού με απλού κλικ.

Για να μπορέσουμε να διαχωρίσουμε το διπλό με το απλό κλικ, χρησιμοποιήσαμε ένα αντικείμενο clickTimer, το οποίο θα δώσει προτεραιότητα στο διπλό κλικ να εκτελεστεί πρώτο, κάνοντας το απλό κλικ και την εμφάνιση του αναδυόμενου παραθύρου να περιμένει.Αυτό επιτυγχάνεται χρησιμοποιώντας setTimeoutγια ενέργειες με ένα κλικ και clearTimeout για να ακυρώσουμε την ενέργεια ενός κλικ εάν εντοπιστεί διπλό κλικ. Τα setTimeout και clearTimeout χρησιμοποιήσαμε στον χειριστή για απλό κλικ στην ετικέτα slideshow-container.

```
1     $(document).on('dblclick', '.slideshow-container', function() {
2    clearTimeout(clickTimer);
3    var slideshowContainer = $(this).closest('.slideshow-container');
4    var currentSlide = slideshowContainer.find('.slide:visible');
5    var photoId = currentSlide.find('img.photo-click').data('photo-id'
       );
6    var userId = slideshowContainer.data('user-id');
7    console.log('User ID:', userId);
8
9    if ($(this).find('img.photo-click').hasClass('recent-user-photo'))
       {
10     if (confirm('Are you sure you want to delete this photo?')) {
11       $.ajax({
12         type: 'DELETE',
13         url: '/users/' + userId + '/photos/' + photoId,
14         dataType: 'json',
15         success: function(data) {
16           if (data.success) {
17             alert('Photo deleted successfully.');
18             location.reload();
19             $('#comments-popup').hide();
20             $('.slideshow-container').each(function() {
21               startSlideshow($(this));
22             });
23           } else {
24             alert('Failed to delete photo: ' + data.error);
25           }
26         }
27       });
28     }
29   } else {
30     alert('You can only delete your own photos.');
31   }
32 });
```

3.2 Προσθήκη χειριστή για διπλό κλικ.

Ο νέος χειριστής ελέγχει αν έχει γίνει διπλό κλικ σε κάποια φωτογραφία όταν γίνεται το slideshow. 'Οταν έχει γίνει διπλό κλικ, σταματάει οποιαδήποτε ενεργό click timer,σταματάει το slideshow και δέχεται τα δεδομένα της φωτογραφίας που είναι ορατή.Άν η φωτογραφία ανήκει στον ενεργό χρήστη, τότε τον ρωτάει αν θέλει να τη διαγράψει μαζι με όλα της δεδομένα. Αν ο χρήστης πατήσει OK, διαγράφει τη φωτογραφία και τα δεδομένα της μέσω AJAX και κανει reload την αρχική σελίδα και ξαναξεκινά το slideshow. Αν όμως η φωτογραφία δεν ανήκει στον χρήστη , του εμφανίζει μήνυμα πως δεν μπορεί να τη διαγράψει.

```
    $(document).on('dblclick', '.slideshow-container', function() {
    clearTimeout(clickTimer);
    var slideshowContainer = $(this).closest('.slideshow-container');
    var currentSlide = slideshowContainer.find('.slide:visible');
    var photoId = currentSlide.find('img.photo-click').data('photo-id'
       );
    var userId = slideshowContainer.data('user-id');
    console.log('User ID:', userId);

    if ($(this).find('img.photo-click').hasClass('recent-user-photo'))
        {
      if (confirm('Are you sure you want to delete this photo?')) {
        $.ajax({
          type: 'DELETE',
          url: '/users/' + userId + '/photos/' + photoId,
          dataType: 'json',
          success: function(data) {
            if (data.success) {
              alert('Photo deleted successfully.');
              location.reload();
              $('#comments-popup').hide();
              $('.slideshow-container').each(function() {
                startSlideshow($(this));
              });
            } else {
              alert('Failed to delete photo: ' + data.error);
            }
          }
        });
      }
    } else {
      alert('You can only delete your own photos.');
    }
  });
```

4 Παρουσίαση ιστοσελίδας

Figure 1: Ανοίγουμε την ιστοσελίδα και βλέπουμε ομαδοποιημένες ολες τις φωτογραφίες του χρήστη με επικεφαλίδα την πιο πρόσφατα ανεβασμένη φωτογραφία.

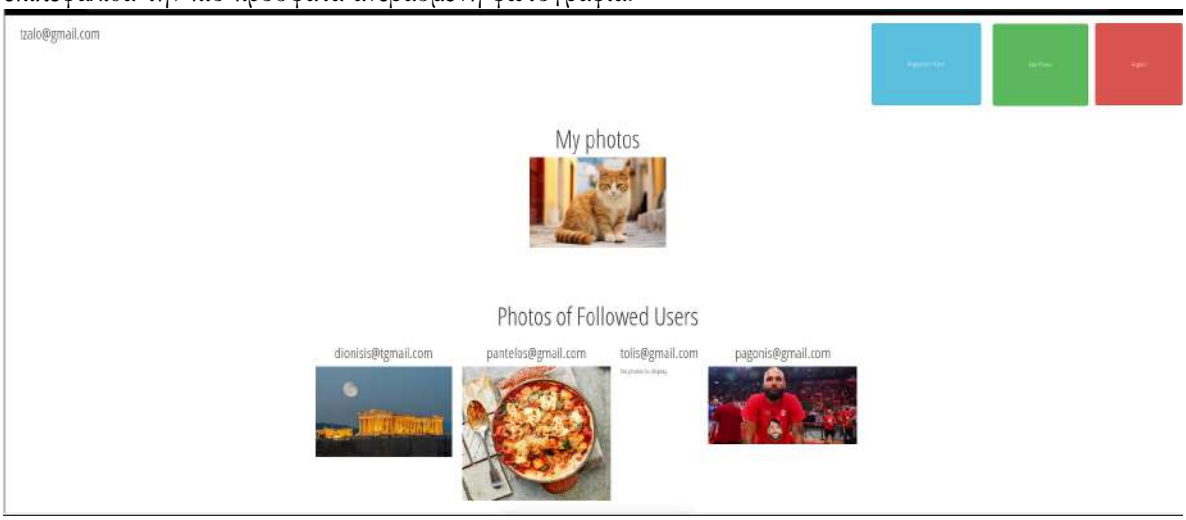


Figure 2: Οταν βάλουμε το ποντίκι πανω στις φωτογραφίες παρατηρούμε πως θα εμφανιστεί το caption της φωτογραφίας αν υπάρχει και θα αρχίσει το slideshow

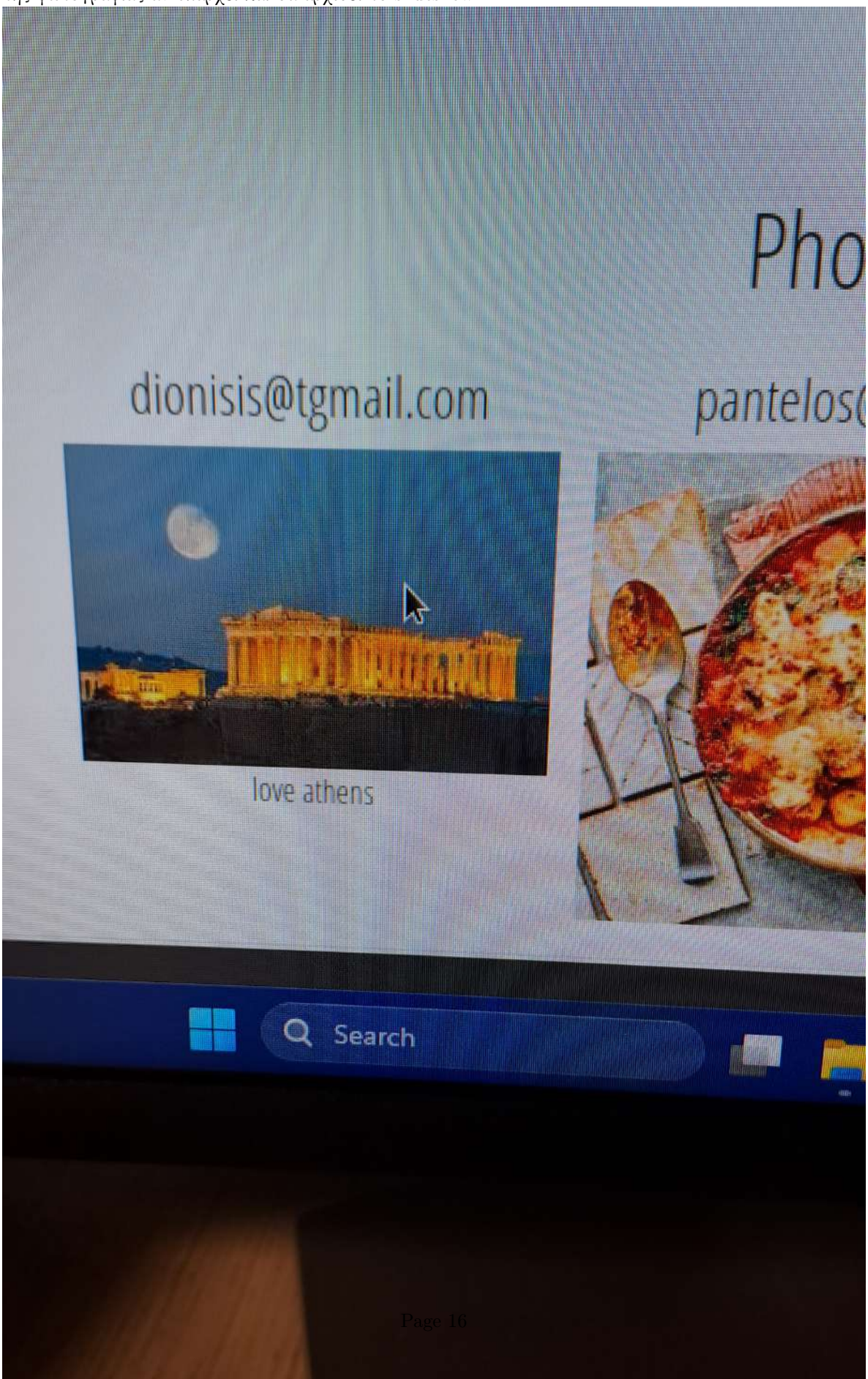


Figure 3: Μετά απο 3 δευτερόλεπτα αλλάζει και η φωτογραφία

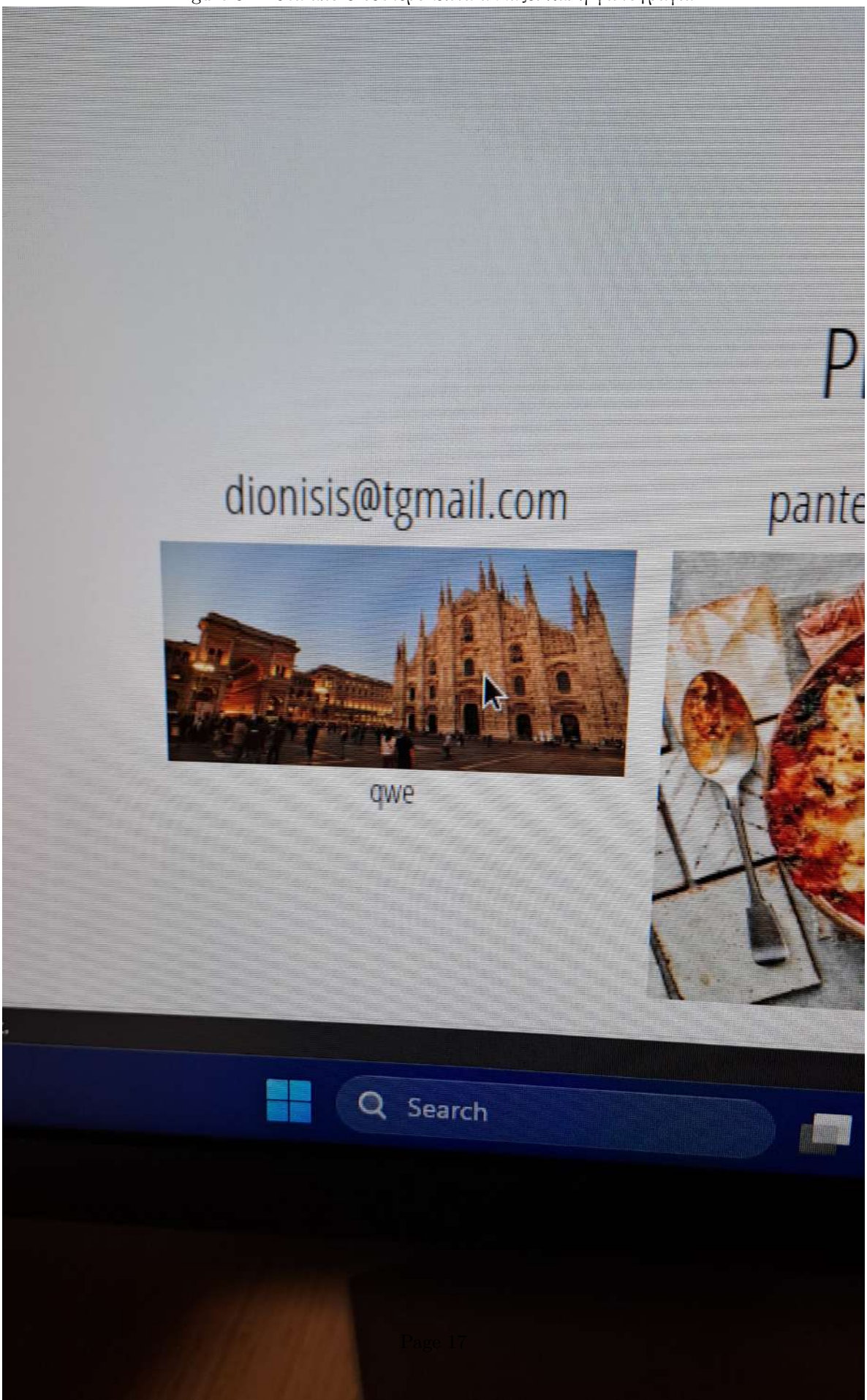


Figure 4: Όταν απομακρύνουμε το ποντίκι, σταματά το slideshow και επιστρέφει στη πιο πρόσφατα ανεβασμένη φωτογραφία.

Figure 5: Όταν πατάμε με μονο κλικ σε κάποια δικιά μας φωτογραφία, εμφανίζεται το αναδυόμενο παράθυρο.

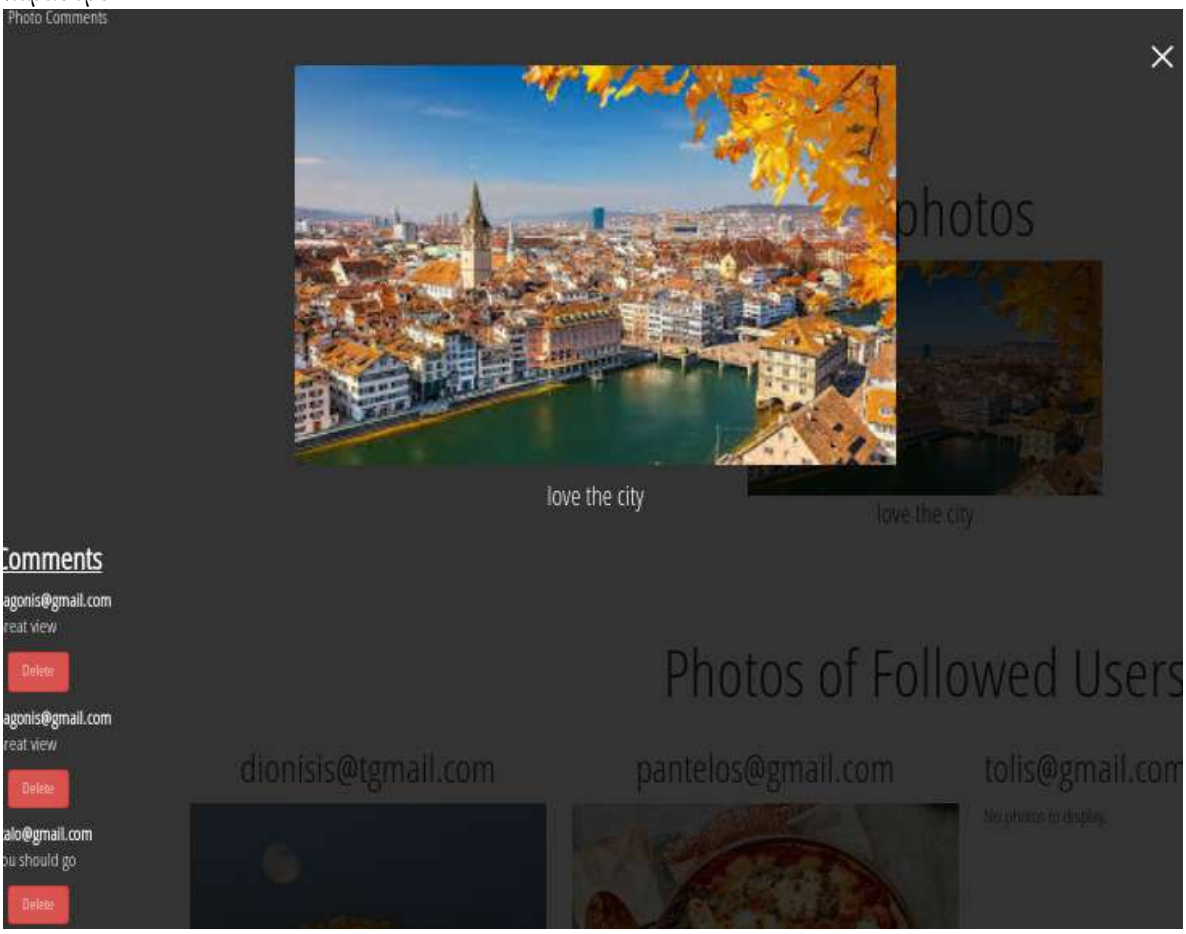


Figure 6: Προσθέτουμε και το σχόλιο

Figure 7: Όταν πατάμε με μονο κλικ σε κάποια φωτογραφία ενος χρήστη που ακολουθούμε, εμφανίζεται το αναδυόμενο παράθυρο και μπορούμε να διαγράψουμε μονο δικά μας σχόλια.

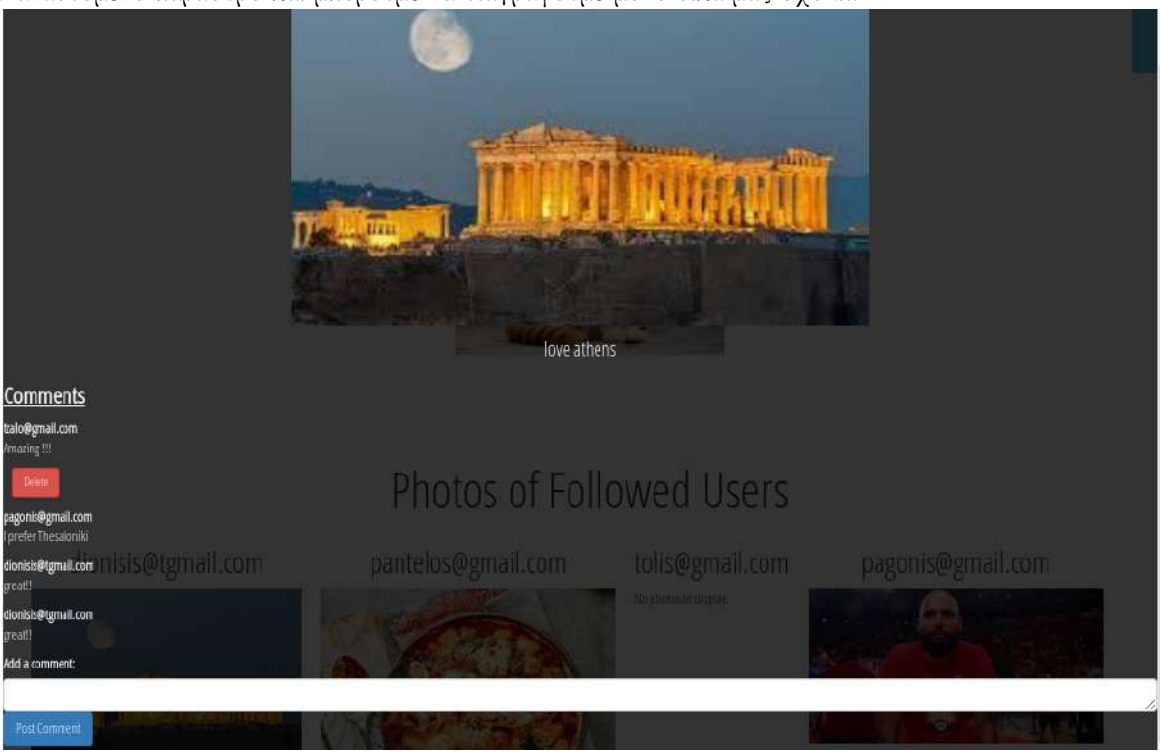


Figure 8: Όταν θέλουμε να διαγράψουμε ενα σχόλιο δικο μας εμφανίζεται το παρακάτω μήνυμα.

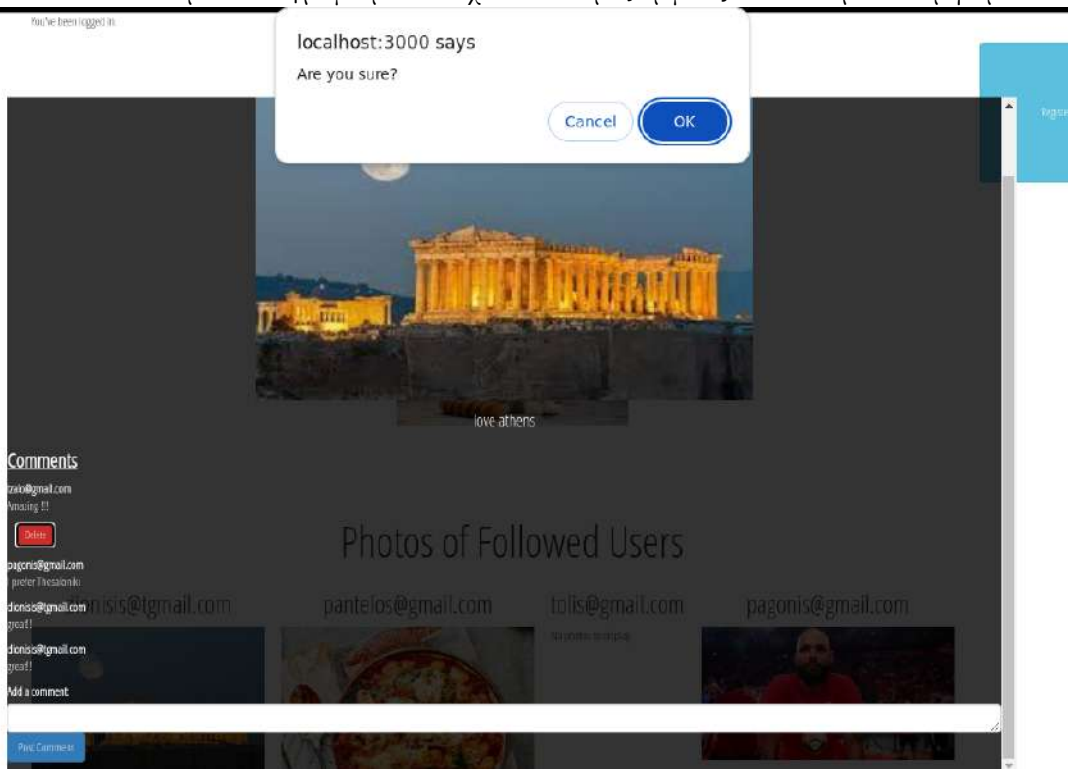


Figure 9: Το σχολιό μας εχει διαγραφθεί.

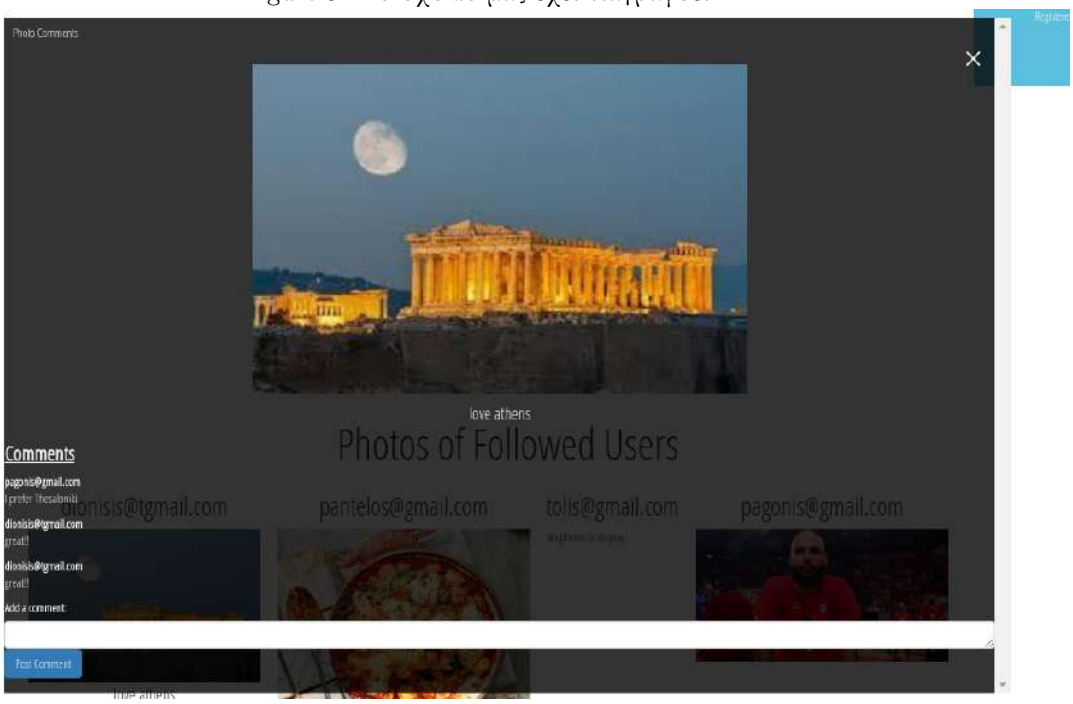


Figure 10: Στις δικές μας φωτογραφίες μπορούμε να διαγράψουμε οτι σχόλιο θέλουμε.

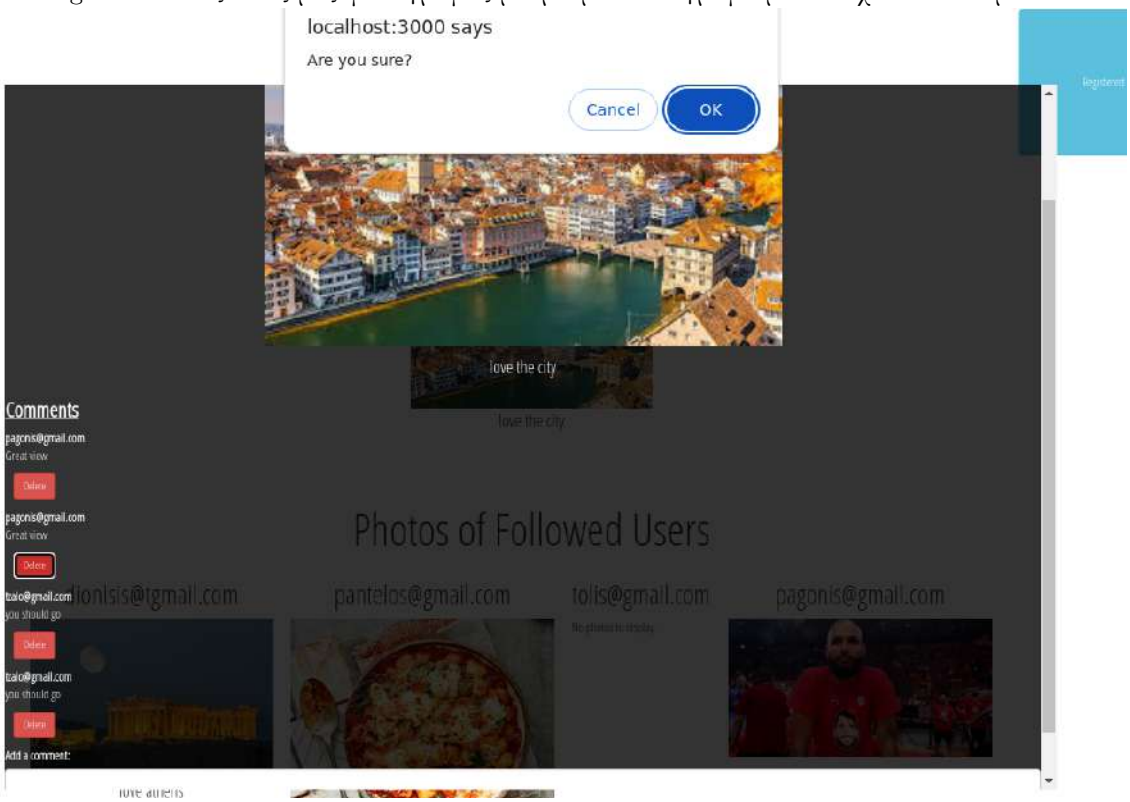


Figure 11: Το σχόλιο θα εχει διαγραφθεί.

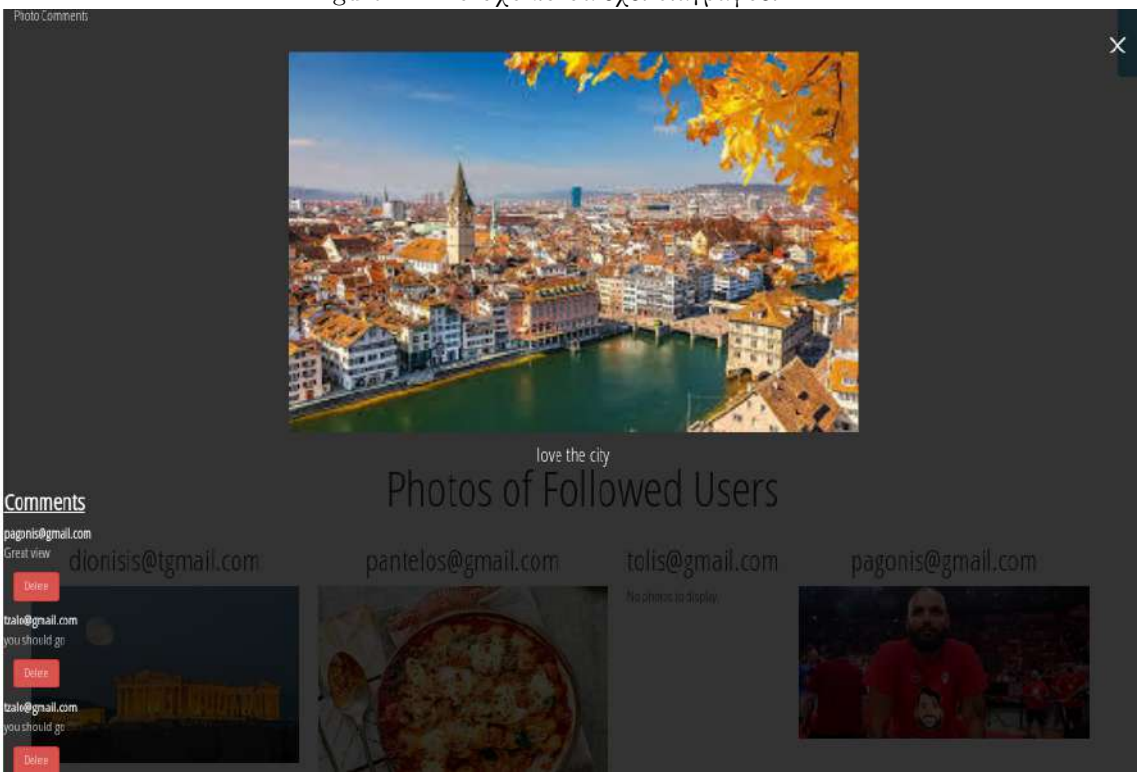


Figure 12: Πατώντας διπλό κλικ σε δικια μας φωτογραφία μας εμφανίζεται το παρακάτω μήνυμα.

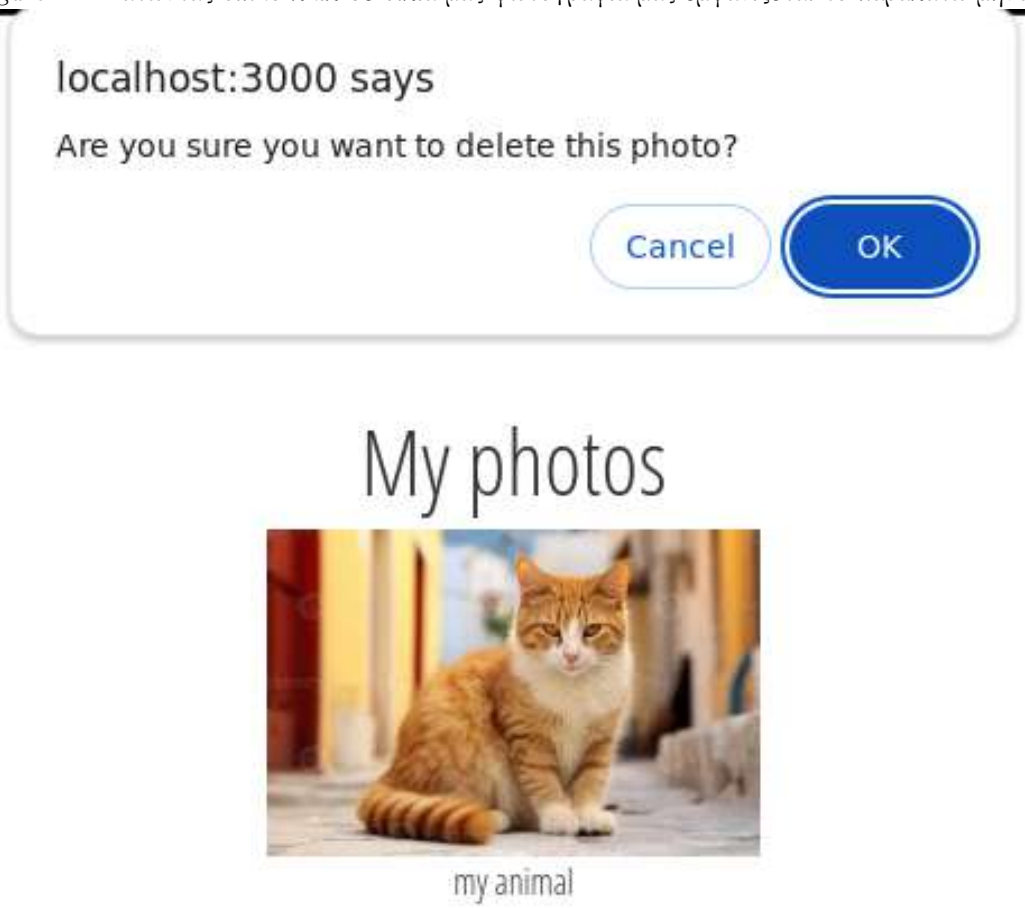


Figure 13: Ανανεώνουμε τη σελίδα και βλέπουμε πως η φωτογραφία εχει διαγραφθέι και η επικεφαλίδα αλλάζει φωτογραφία.

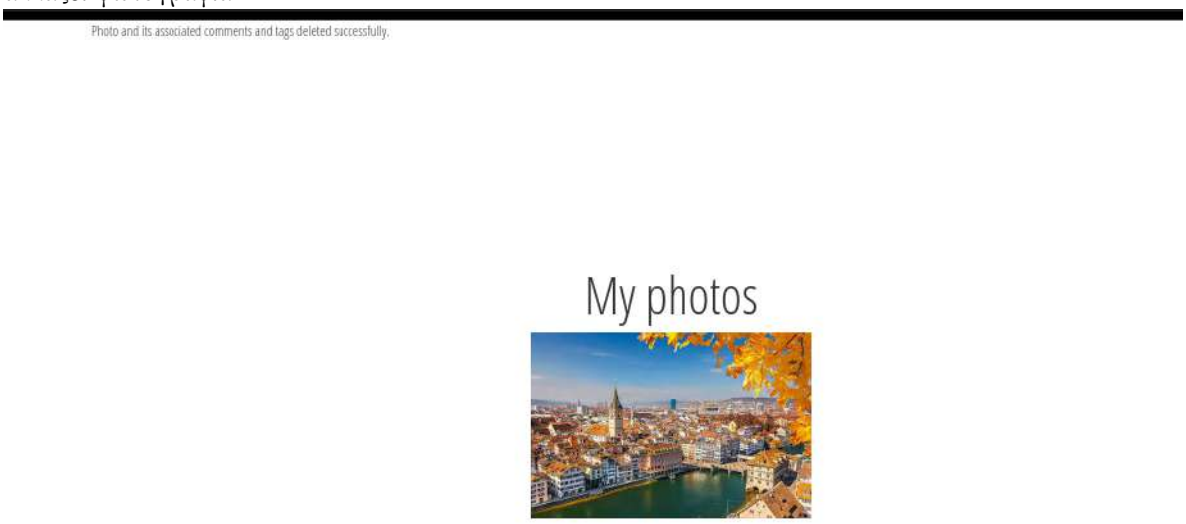
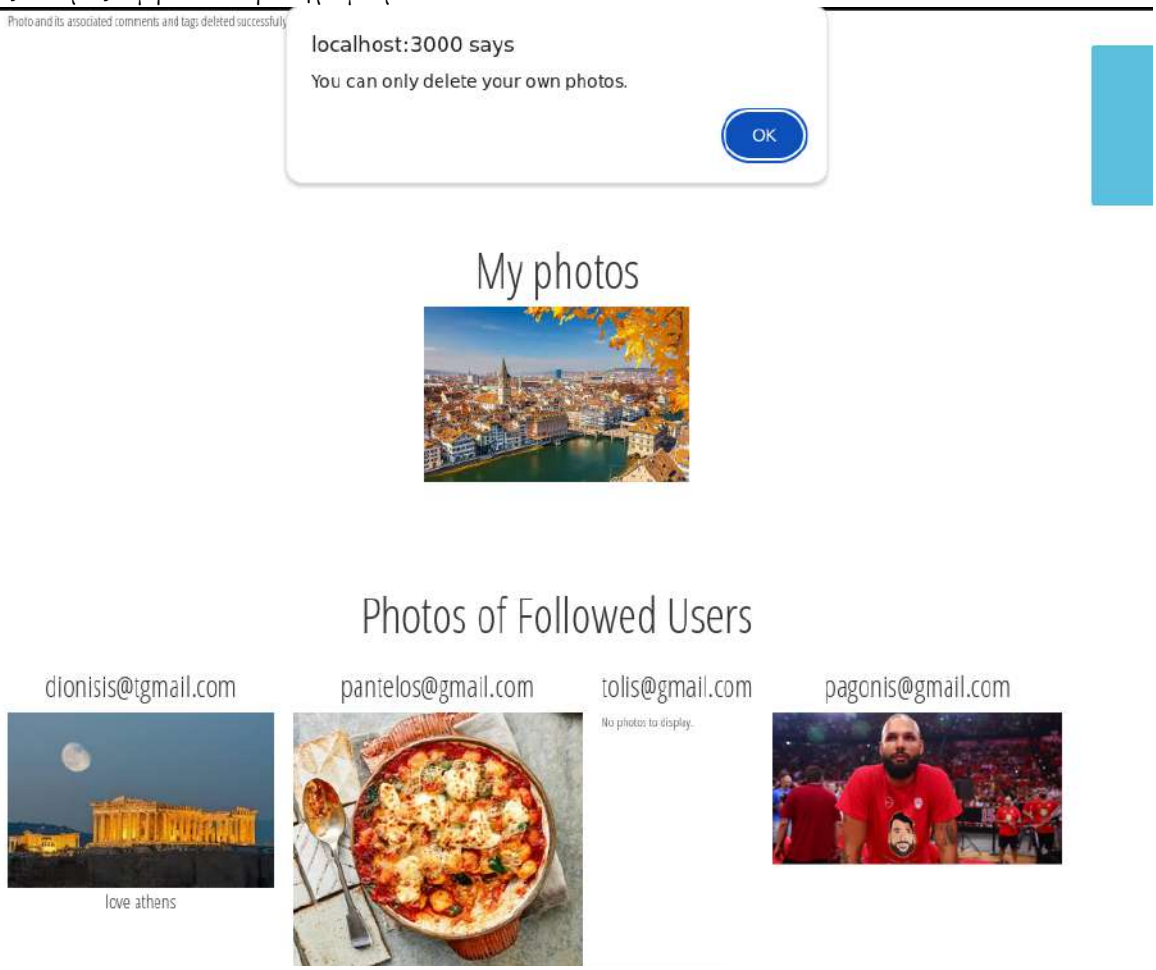


Figure 14: Πατώντας διπλό κλικ σε κάποια φωτογραφία καποιου χρήστη που ακολουθουμε, βλέπουμε πως δεν μας αφήνει να τη διαγράψουμε.



5 Σημέιωση

Οταν δοκιμαζουμε να βάλουμε νεο σχόλιο για καποιο λόγο παρουσιάζεται στο αναδυόμενο παράθυρο 2 φορές.