

# ΜΥΕ042 Τεχνολογίες Διαδικτύου

Εργαστηριακή Άσκηση 2:

Προβολή παρουσίασης και αναδυόμενο παράθυρο  
στην εφαρμογή φωτογραφιών

Όμηρος Χατζηιορδάνης AM: 5388

## **Περιεχόμενα**

1. Προβολή παρουσίασης φωτογραφιών.....	3
2. Pop Up παράθυρο .....	8
3. Διαγραφή φωτογραφιών.....	12
4. Παραδοτέα αρχεία .....	15

# 1. Προβολή παρουσίασης φωτογραφιών

Χρησιμοποιώντας την ήδη υπάρχουσα υλοποίηση της προηγούμενης εργαστηριακής άσκησης, αρχικά έγιναν αλλαγές στη προβολή των φωτογραφιών. Στην μέθοδο show του user controller πλέον αντί για ένα σετ με όλες τις φωτογραφίες που μπορούν να προβληθούν στον τωρινό χρήστη, δημιουργήθηκαν σετ που στη μία θέση βρίσκεται ο χρήστης ή οι ακόλουθοι του και στην άλλη οι φωτογραφίες εκείνου του χρήστη, ταξινομημένες κατά φθίνουσα σειρά ως προς την ημερομηνία δημιουργίας τους. Επίσης για την εμφάνιση πρώτα των φωτογραφιών του χρήστη που είναι συνδεμένος εκείνη τη στιγμή, στην ανάθεση των σετ γίνεται πρώτα το σετ του τωρινού χρήστη και μετά των ακόλουθων του.

```
22 def show
23   @users = User.all
24   @user = User.find(params[:id])
25
26   users_with_photos = [{ user: current_user, photos: current_user.photos.order(created_at: :desc) }]
27   follows = current_user.followed_users
28   follows.each do |followed_user|
29     users_with_photos << { user: followed_user, photos: followed_user.photos.order(created_at: :desc) }
30   end
31   @group_photos = users_with_photos
32 end
```

Έτσι στο αρχείο του haml πλέον διατρέχουμε το κάθε σετ φωτογραφιών και δημιουργούμε στοιχεία slideshow-container για το κάθε ένα, τα οποία θα χρησιμοποιήσει η javascript για να κάνει παρουσίαση τις φωτογραφίες του κάθε χρήστη. Μέσα στο κάθε slideshow-container επίσης δημιουργούνται στοιχεία slides που αντιστοιχούν σε κάθε φωτογραφία που περιέχεται σε αυτό το σετ. Για ευκολία σε επόμενα ερωτήματα, μαζί με κάθε φωτογραφία δίνονται και κάποια δεδομένα σε ένα dataset, περισσότερη ανάλυση στην υλοποίηση του泡泡 up. Επίσης για κάθε φωτογραφία δημιουργείται και ένα στοιχείο caption που περιέχει τον τίτλο της, το οποίο θα προβάλλεται όταν θα προβάλλεται και η φωτογραφία. Τέλος για να μην εμφανίζεται ένα κενό πλαίσιο εάν ένας χρήστης δεν έχει ανεβάσει καμία φωτογραφία, γίνεται έλεγχος άμα υπάρχουν φωτογραφίες σε κάθε σετ ώστε να δημιουργηθεί ένα container και σε κάθε σετ αναγράφεται το email του αντίστοιχου χρήστη.

```
23 .row
24 - @group_photos.each do |group|
25   - if group[:photos].present?
26     %h2= group[:user].email
27
28   %div.slideshow-container
29     - group[:photos].each do |photo|
30       %div.slide{ data:(photo_id: photo.id, user_id: current_user.id, comment_path: get_comments_photo_path(photo.id)) }
31         = image_tag photo.image.url(:medium), class: 'slide-image'
32         %div.caption= photo.title
33
```

To script που δημιουργήθηκε ονομάζεται slideshow.js, στο ερώτημα αυτό μας απασχολούν κυρίως οι συναρτήσεις showSlides, startSlideshow, η setup και τα event listeners του ποντικιού.

Στη setup μαζεύονται όλα τα slideshow-containers στη μεταβλητή slideshowContainers μέσω της document.getElementsByClassName η οποία εντοπίζει τα στοιχεία που είχαν δηλωθεί στο αρχείο haml, δηλαδή στην html της εφαρμογής. Για κάθε ένα από αυτά τα containers καλείται η συνάρτηση showSlides. Τέλος η setup εκτελείται αμέσως μόλις φορτώσει το script.

```
207     setup: function() {
208       var slideshowContainers = document.getElementsByClassName('slideshow-container');
209       for (var i = 0; i < slideshowContainers.length; i++) {
210         SlideshowManager.showSlides(slideshowContainers[i]);
211       }
212     }
213   };
214   $(SlideshowManager.setup);
```

Η showSlides έχει την ευθύνη να μαζέψει όλα τα στοιχεία slides, δηλαδή τις φωτογραφίες, και να τις αποκρύψει από τη σελίδα μέσω του style.display = 'none' και να εμφανίσει τη πρώτη εικόνα, η οποία λόγω της υλοποίησης στο backend κομμάτι, θα είναι η νεότερη. Επίσης λόγω του στοιχείου caption που δημιουργήθηκε στη haml, θα εμφανίζεται και αυτόματα και ο τίτλος μαζί με την φωτογραφία. Τέλος μέσα στην showSlides στεγάζονται όλες οι υπόλοιπες συναρτήσεις του script.

```
1  var SlideshowManager = {
2    popupOpen: false, // Global variable to track the popup state
3    showFirst: true, // Global variable for continuing or not the Slideshow after the
4    // popup window
5    clickTimeout: null, // Timeout for click events
6
7    showSlides: function(container) {
8      var slideIndex = 0;
9      var slides = container.getElementsByClassName('slide');
10     var slideshowInterval;
11
12     // Hide all slides initially and show the first one
13     for (var i = 0; i < slides.length; i++) {
14       slides[i].style.display = 'none';
15     }
16     slides[0].style.display = 'block';
```

Τα δύο event listeners έχουν να κάνουν με την εισαγωγή και εξαγωγή του δείκτη του ποντικιού.

Αν ο χρήστης φέρει το ποντίκι του επάνω στο πλαίσιο, ξεκινάει η διαδικασία για την αλλαγή των φωτογραφιών. Η αλλαγή γίνεται μέσω ενός interval στον οποίο έχουν τεθεί τα 3 δευτερόλεπτα ως τα διαστήματα χρόνου ανάμεσα σε κάθε εκτέλεση της startSlideshow.

```

29 // The slides start to change when mouse enters
30 container.addEventListener('mouseenter', function() {
31   showFirst = false;
32   if (!SlideshowManager.popupOpen) {
33     slideshowInterval = setInterval(startSlideshow, 3000);
34   }
35 });

```

Στην απομάκρυνση του ποντικιού από το πλαίσιο του slideshow, επαναφέρεται η πρώτη φωτογραφία στη προβολή μέσω των style.display, την θέτιση τιμής του index στο 0. Επίσης το interval που έφερνε τις συνεχόμενες εκτελέσεις της startSlideshow σταματάει τη λειτουργία του μέσω της clearInterval.

```

37 // Slideshow resets when the mouse leaves
38 container.addEventListener('mouseleave', function() {
39   showFirst = true;
40   if (!SlideshowManager.popupOpen) {
41     clearInterval(slideshowInterval); // This stops the interval that runs the startSlideshow
42     slides[slideIndex].style.display = 'none';
43     slides[0].style.display = 'block';
44     slideIndex = 0;
45   }
46 });

```

Στην startSlideshow εξαφανίζεται οι προηγούμενη φωτογραφία, αυξάνεται κατά ένα ο δείκτης τους ελέγχοντας πάντα αν ξεπερνάει το πλήθος των φωτογραφιών, που σε τέτοια περίπτωση τίθεται η τιμή 0 που αντιπροσωπεύει τη πρώτη φωτογραφία, και εμφανίζεται ή νέα φωτογραφία.

```

17 // Function that changes the slides
18 function startSlideshow() {
19   slides[slideIndex].style.display = 'none';
20   slideIndex++;
21
22   if (slideIndex == slides.length || showFirst) {
23     slideIndex = 0;
24   }
25
26   slides[slideIndex].style.display = 'block';
27 }

```

Εδώ δίνεται το αρχείο slideshow.sass στο οποίο περιέχονται ορισμοί της css για τη μορφοποίηση των slideshow οι οποίοι προήλθαν από διάφορα παραδείγματα στο διαδίκτυο. Επιτυγχάνουν μια καλύτερη οργάνωση και προβολή του κάθε σετ με ξεκάθαρα περιβλήματα για τον χρήστη, αλλαγή του κέρσορα για καλύτερη ένδειξη λειτουργίας του της παρουσίασης και κατάλληλη τοποθέτηση των εικόνων μέσα στο πλαίσιο.

```

1 .slideshow-container
2   width: 50%
3   height: 300px
4   overflow: hidden
5   cursor: pointer // Pointer cursor to indicate it's interactive
6   border: 2px solid black
7   padding: 0 0 30px 0 // Add some space below the image for the caption
8
9 .slide
10  display: none
11  width: 100%
12  height: 100%
13  position: relative
14  display: flex // Use flex to center the image and caption vertically and horizontally
15  justify-content: center
16  align-items: center
17
18 .slide-image
19  width: 100%
20  height: auto
21  max-height: calc(100% - 10px) // Ensure the image height doesn't exceed the container height minus space for the caption
22  object-fit: contain // Maintain aspect ratio
23
24 .caption
25  position: relative
26  background-color: rgba(0, 0, 0, 0.6) // Semi-transparent background for readability
27  color: white
28  font-size: 16px
29  padding: 10px
30  text-align: center

```

Για δοκιμή της σωστής λειτουργίας δημιουργήθηκαν τρεις φωτογραφίες, δύο από τον χρήστη 1 και μία από τον χρήστη 2. Για επιβεβαίωση της σωστής σειράς παρουσίασης των φωτογραφιών δίνεται το σύνολο των φωτογραφιών από τη κονσόλα του rails.

```

Photo Load (0.1ms)  SELECT "photos".* FROM "photos"
=> #<ActiveRecord::Relation [#<Photo id: 1, user_id: 1, caption: nil, created_at: "2024-12-21 02:37:44", updated_at: "2024-12-21 02:37:44", image_file_name: "51rCXXKaVGTL.AC_UF894_1000_QL80_.jpg", image_content_type: "image/jpeg", image_file_size: 42326, image_updated_at: "2024-12-21 02:37:44", title: "ball">, #<Photo id: 5, user_id: 1, caption: nil, created_at: "2024-12-21 03:01:10", updated_at: "2024-12-21 03:01:10", image_file_name: "istockphoto-1446199740-612x612.jpg", image_content_type: "image/jpeg", image_file_size: 84861, image_updated_at: "2024-12-21 03:01:10", title: "Tree">, #<Photo id: 6, user_id: 2, caption: nil, created_at: "2024-12-21 03:27:01", updated_at: "2024-12-21 03:27:02", image_file_name: "sheep-closeup-eating-grass.jpg", image_content_type: "image/jpeg", image_file_size: 36911, image_updated_at: "2024-12-21 03:27:01", title: "Sheep">]

```

Οι φωτογραφίες όταν ο χρήστης 1 δεν έχει το ποντίκι πάνω σε κάποιο πλαίσιο.

user1@email.com



user1@email.com



Tree

user2@email.com



Sheep

Οι φωτογραφίες όταν ο χρήστης 1 έχει τον κέρσορα του πάνω στο πρώτο πλαίσιο και έχει αλλάξει η φωτογραφία. Δυστυχώς δεν ήταν δυνατή η απεικόνιση του κέρσορα στο screenshot.

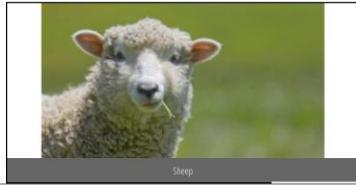
user1@email.com



user1@email.com



user2@email.com



Τέλος για τη σίγουρη λειτουργία του script, έχει δηλωθεί το compilation του slideshow.js στο αρχείο assets.rb του directory config/initializers, όπως ζητείται από τα μηνύματα σφαλμάτων της ruby αν δεν έχει δηλωθεί ήδη.

```
1 # Be sure to restart your server when you modify this file.
2
3 # Version of your assets, change this if you want to expire all your assets.
4 Rails.application.config.assets.version = '1.0'
5
6 # Add additional assets to the asset load path
7 # Rails.application.config.assets.paths << Emoji.images_path
8
9 # Precompile additional assets.
10 # application.js, application.css, and all non-JS/CSS in app/assets folder are already added.
11 # Rails.application.config.assets.precompile += %w( search.js )
12 Rails.application.config.assets.precompile += %w( slideshow.js )
13
```

Επίσης στο application.sass έχει γίνει require το sass για το slideshow.

```
1 /*
2  * This is a manifest file that'll be compiled into application.css, which will include all the files
3  * listed below.
4  *
5  * Any CSS and SCSS file within this directory, lib/assets/stylesheets, vendor/assets/stylesheets,
6  * or any plugin's vendor/assets/stylesheets directory can be referenced here using a relative path.
7  *
8  * You're free to add application-wide styles to this file and they'll appear at the bottom of the
9  * compiled file so the styles you add here take precedence over styles defined in any styles
10 * defined in the other CSS/SCSS files in this directory. It is generally better to create a new
11 * file per style scope.
12 *
13 *= require_tree .
14 *= require_self
15 *= require slideshow
16 */
17
18
19 body
20   font-family: 'Open Sans Condensed', sans-serif
21
```

## 2. Pop Up παράθυρο

Για το pop up δημιουργήθηκε event listener με μονό κλικ επάνω σε κάθε πλαίσιο. Με το κλικ περνάνε πληροφορίες χρήσιμες στα αιτήματα αյαχ, και καλείται η συνάρτηση showPopup. Το timeout εξηγείται στη διαγραφή.

```
48     // Single click event for pop up
49     for (var i = 0; i < slides.length; i++) {
50         slides[i].addEventListener('click', function() {
51             clearTimeout(SlideshowManager.clickTimeout); // Clear the previous click timeout
52             var photo_id = this.dataset.photoId;
53             var user_id = this.dataset.userId;
54             var comment_path = this.dataset.commentPath;
55
56             SlideshowManager.clickTimeout = setTimeout(function() {
57                 showPopup(photo_id, user_id, comment_path);
58             }, 200); // Delay for single click detection
59         });
60     }
61 }
```

Με τη ShowPopup αρχικά απερνεγοποιείται ο interval για το slideshow και η μεταβλητή popupOpen παίρνει τιμή true ώστε να μην επιτρέπει στα υπόλοιπα event listeners του ποντικιού να επηρεάζουν πλέον το slideshow όταν έχουμε το αναδυόμενο παράθυρο ανοικτό, κρατώντας έτσι την ίδια εικόνα. Στη συνέχεια δημιουργείται ένα νέο στοιχείο html το οποίο προστίθεται στη σελίδα και περιέχει ένα container για τα σχόλια της εκάστοτε φωτογραφίας, ένα πλαίσιο κειμένου ώστε ο χρήστης να γράφει νέα σχόλια στη φωτογραφία, ένα κουμπί προσθήκης του σχολίου και ένα κουμπί για να κλείσει το pop up παράθυρο.

```
73     function showPopup(photo_id, user_id, comment_path) {
74         var photoId = photo_id;
75         var userId = user_id;
76         var commentPath = comment_path;
77
78         clearInterval(slideshowInterval);
79         SlideshowManager.popupOpen = true;
80
81         // Create and show popup
82         var existingPopup = document.querySelector('.popup');
83         if (existingPopup) existingPopup.remove();
84
85         var popup = document.createElement('div');
86         popup.classList.add('popup');
87         popup.innerHTML =
88             `<div class="popup-content">
89                 <h2>Comments for Photo #${photoId}</h2>
90                 <div id="comments-container"></div>
91                 <div class="add-comment">
92                     <textarea id="new-comment" placeholder="Write your comment..." rows="4" cols="40"></textarea>
93                     <button id="submit-comment">Add Comment</button>
94                     <button id="close-popup">Close Popup</button>
95                 </div>
96             </div>
97         `;
98         document.body.appendChild(popup);
99     }
```

Τα σχόλια της φωτογραφίας γίνονται διαθέσιμα μέσω ενός αιτήματος αյαχ. Στο αյαχ δίνεται το url της μεθόδου για την ανάκτηση των σχολίων, το οποίο παρέχει το αρχείο haml για κάθε ξεχωριστή φωτογραφία, ο τύπος της μεθόδου που είναι get

διότι γίνεται ανάγνωση δεδομένων και ο τύπος αυτών των δεδομένων που είναι json για εύκολη διαχώριση του κάθε σχολίου που θα έρθει μετά από το αίτημα. Εφόσον εκτελεστεί με επιτυχία το αίτημα, το container με τα σχόλια τροποποιείται με κάθε ένα νέο στοιχείο, αλλιώς εμφανίζεται μήνυμα σφάλματος στην κονσόλα.

```
100    // Fetch comments through AJAX
101    $.ajax({
102      url: commentPath,
103      method: 'GET',
104      dataType: 'json',
105      success: function(data) {
106        var commentsContainer = popup.querySelector('#comments-container');
107        commentsContainer.innerHTML = ''; // Clear existing comments
108        data.forEach(function(comment) {
109          var commentElement = document.createElement('p');
110          commentElement.innerText = `${comment.user_id}: ${comment.comment_text}`;
111          commentsContainer.appendChild(commentElement);
112        });
113      },
114      error: function() {
115        console.error('Error fetching comments');
116      }
117    });
```

Για το αίτημα ajax δημιουργήθηκε μια νέα συνάρτηση στον controller των φωτογραφιών οι οποία βρίσκει όλα τα σχόλια της συγκεκριμένης φωτογραφίας, τα ταξινομεί σύμφωνα με το id κατά φθίνουσα σειρά και δίνει τα τρία πρώτα ώστε να μετατραπούν σε δεδομένα json. Στην εκφώνηση ζητούνται τα πιο πρόσφατα σχόλια χωρίς κάποια παραπάνω διευκρίνηση, οπότε σε αυτήν την υλοποίηση εμφανίζονται συγκεκριμένα τα τρία πιο πρόσφατα. Η ταξινόμηση μέσω του id των σχολίων εξασφαλίζει την εμφάνιση πρώτα των πιο προσφάτων διότι τα id στα σχόλια αυξάνονται αυτόματα με τη κάθε νέα δημιουργία τους.

```
49  def get_comments
50    @photo = Photo.find(params[:id])
51    @comments = @photo.comments.order(id: :desc).limit(3)
52    render json: @comments
53  end
```

Για το κλείσιμο του παραθύρου έχει ανατεθεί ένα μονό κλικ event listener στο κουμπί που είχε δημιουργηθεί στην html του pop up. Όταν πατηθεί το κουμπί το παράθυρο θα αφαιρεθεί από τη σελίδα. Για την συνέχιση ή όχι του slideshow ανάλογα με τη θέση του κέρσορα του χρήστη χρησιμοποιείται η global μεταβλητή showFirst, η οποία όπως φαίνεται σε παραπάνω εικόνες, παίρνει τιμές true ή false ανάλογα άμα ο κέρσορας του χρήστη έχει τοποθετηθεί μέσα ή έξω από το πλαίσιο των φωτογραφιών, ακόμα και όταν είναι ανοιχτό το αναδυόμενο παράθυρο. Άμα έχει τιμή true, μόλις αφαιρεθεί το παράθυρο, θα επανέλθει η πρώτη φωτογραφία στο ανάλογο πλαίσιο μέσω της συνθήκης if που υπάρχει στην startSlideshow η οποία καλείται.

```

119     // Button click for closing the pop up
120     var closeBtn = popup.querySelector('#close-popup');
121     closeBtn.addEventListener('click', function() {
122         SlideshowManager.popupOpen = false;
123         popup.remove();
124         console.log(showFirst)
125         if(showFirst){
126             startSlideshow();
127         }
128     });
129 
```

Για τη δημιουργία σχολίων χρησιμοποιείται πάλι αίτηση ajax, χρησιμοποιώντας ήδη την υπάρχουσα post συνάρτηση για νέα σχόλια που είχε δημιουργηθεί στην προηγούμενη εργασία. Για τις παραμέτρους δημιουργείται νέο σύνολο δεδομένων με τα id της φωτογραφίας και του συνδεμένου χρήστη και με το περιεχόμενο στο πλαίσιο κειμένου, στο οποίο αφαιρούνται τυχόν κενά στην αρχή και στο τέλος του. Άμα η δημιουργία είναι επιτυχής, τότε ξαναγίνεται η ίδια αίτηση για την εμφάνιση των σχολίων.

```

130     // Button click for submitting comment
131     var submitButton = popup.querySelector('#submit-comment');
132     submitButton.addEventListener('click', function() {
133         var newCommentText = popup.querySelector('#new-comment').value.trim();
134         if (newCommentText) {
135             $.ajax({
136                 url: '/comments', // default from routes
137                 method: 'POST',
138                 data: {
139                     comment: {
140                         photo_id: photoId,
141                         user_id: userId,
142                         comment_text: newCommentText
143                     }
144                 },
145                 success: function() {
146                     // Clear the textarea after submission
147                     popup.querySelector('#new-comment').value = '';
148                     // Re-fetch updated comments
149                     $.ajax({
150                         url: commentPath,
151                         method: 'GET',
152                         success: function(comments) {
153                             var commentsContainer = popup.querySelector('#comments-container');
154                             commentsContainer.innerHTML = ''; // Clear existing comments
155                             comments.forEach(function(comment) {
156                                 var commentElement = document.createElement('p');
157                                 commentElement.innerText = `${comment.user_id}: ${comment.comment_text}`;
158                                 commentsContainer.appendChild(commentElement);
159                             });
160                         },
161                         error: function() {
162                             console.error('Error fetching updated comments');
163                         }
164                     });
165                 },
166                 error: function() {
167                     console.error('Error submitting comment');
168                 }
169             });
170         }
171     });
172     popup.style.display = 'block';
173 
```

Στον controller για τα σχόλια, έγινε τροποποίηση ώστε άμα δημιουργηθεί νέο σχόλιο ή κάτι έχει αποτύχει, να σταλθεί στο αίτημα ajax ένα json το οποίο το ενημερώνει για την επιτυχία ή όχι της διαδικασίας. Άμα η διαδικασία δημιουργίας

νέου σχόλιου είναι επιτυχής, τότε γίνεται ξανά η προβολή των σχόλιων αλλιώς βγαίνει μήνυμα στο console του φυλλομετρητή.

```
5     def create
6       @new_comment = Comment.create(
7         photo_id: params[:comment][:photo_id],
8         user_id: params[:comment][:user_id],
9         comment_text: params[:comment][:comment_text]
10        )
11
12       if @new_comment.save
13         render json: @new_comment, status: :created
14       else
15         render json: { errors: @new_comment.errors.full_messages }, status: :unprocessable_entity
16       end
17     end
```

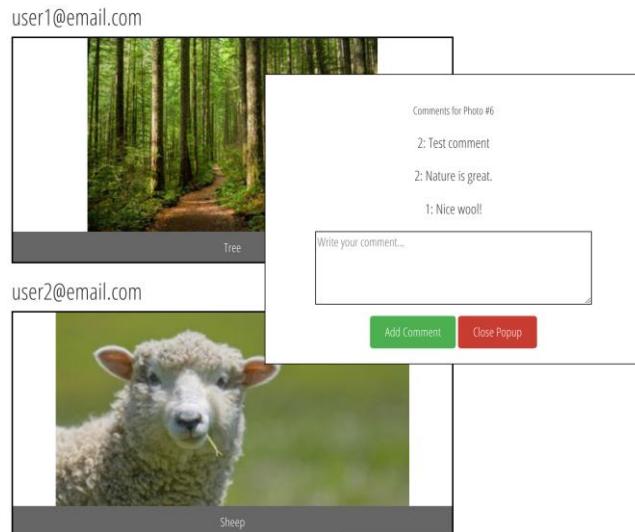
Εδώ παρουσιάζονται τα χαρακτηριστικά css για το αναδυόμενο παράθυρο στην τοποθέτηση, την εμφάνιση και αυτού αλλά και των στοιχείων του. Στην συγκεκριμένη υλοποίηση η τοποθέτηση του παραθύρου δεν επιτρέπει την πλήρη κάλυψη της ορθότητας για την συνέχιση ή όχι του slideshow ανάλογα την τοποθέτηση του κέρσορα, αφού ο χρήστης δεν μπορεί να μετακινήσει το παράθυρο.

```
45 .popup h2
46   font-size: 14px
47   margin-bottom: 20px
48
49 .popup p
50   font-size: 18px
51   margin: 15px
52   line-height: 1.6
53
54 #new-comment
55   width: 80%
56   font-size: 16px
57   border: 1px solid black
58
59 #submit-comment
60   margin-top: 10px
61   padding: 10px 20px
62   font-size: 16px
63   background-color: #4CAF50
64   color: white
65   border: none
66   border-radius: 4px
67   cursor: pointer
68
69 #submit-comment:hover
70   background-color: #45a049
71
72 #close-popup
73   margin-top: 10px
74   padding: 10px 20px
75   font-size: 16px
76   background-color: #c93c32
77   color: white
78   border: none
79   border-radius: 4px
80   cursor: pointer
81
82 #close-popup:hover
83   background-color: #9c251c
84
```

Το σύνολο των σχολίων για την εικόνα με το πρόβατο:

```
Comment Load (0.1ms)  SELECT "comments".* FROM "comments"
=> #<ActiveRecord::Relation [<Comment id: 1, photo_id: 5, user_id: nil, comment_text: "What a nice forest!>, <Comment id: 2, photo_id: 5, user_id: 1, comment_text: "Nature is beautiful">, <Comment id: 3, photo_id: 6, user_id: 1, comment_text: "Bee">, <Comment id: 4, photo_id: 6, user_id: 1, comment_text: "Nice wool!">, <Comment id: 5, photo_id: 6, user_id: 2, comment_text: "Nature is great.">, <Comment id: 6, photo_id: 6, user_id: 2, comment_text: "Test comment">]>
```

Το αναδυόμενο παράθυρο με τα τρία πιο πρόσφατα σχόλια:



Η γραφή και η τοποθέτηση ενός νέου σχόλιου:



### 3. Διαγραφή φωτογραφιών

Για τη διαγραφή των φωτογραφιών δημιουργήθηκε ένα νέο event listener με διπλό κλικ, όμως για να μην λειτουργήσει ταυτόχρονα και ο event listener χρησιμοποιήθηκε ένα time out, δηλαδή ένας χρονοδιακόπτης, στο μονό κλικ. Αν ο χρήστης μέσα σε ένα διάστημα 200ms δεν πατήσει δεύτερη φορά το ποντίκι του, τότε το time out θα τελειώσει και θα εμφανιστεί το αναδυόμενο παράθυρο. Αν πατήσει και δεύτερη φορά, τότε απενεργοποιεί το time out και θα εκτελεστεί η μέθοδος διαγραφής.

```
62 // Double click for photo deletion
63 for (var i = 0; i < slides.length; i++) {
64   slides[i].addEventListener('dblclick', function() {
65     clearTimeout(SlideshowManager.clickTimeout); // Prevent single click event
66     console.log('Double-clicked on slide with photoId:', this.dataset.photoId);
67     var photo_id = this.dataset.photoId;
68
69     deletePhoto(photo_id);
70   });
71 }
```

Η συνάρτηση deletePhoto παίρνει το id της φωτογραφίας και κάνει ένα αίτημα ajax το οποίο οδηγεί στην συνάρτηση διαγραφής στη ruby που είχε δημιουργηθεί στην προηγούμενη άσκηση, βάζοντας το id της φωτογραφίας στο url ώστε να γίνει η σωστή διαγραφή. Ανάλογα με το json που επιστρέφεται, εμφανίζεται και το κατάλληλο μήνυμα στο console του φυλλομετρητή. Επίσης για τη σίγουρη καλή λειτουργία της σελίδας μετά τη διαγραφή, το script τη φορτώνει ξανά μέσω του window.location.reload, ώστε τα σετ να μην περιέχουν οποιοδήποτε ίχνος της διαγραμμένης φωτογραφίας και δημιουργήσει προβλήματα στο slideshow.

```

175     function deletePhoto(photo_id) {
176       var photoId = photo_id;
177       $.ajax({
178         url: `/photos/${photoId}`,
179         method: 'DELETE',
180         success: function(data) {
181           if (data.success) {
182             console.log('Photo deleted successfully');
183             window.location.reload(); // Reload the page after successful deletion
184           } else {
185             console.log('Error deleting photo:', data.message);
186           }
187         },
188         error: function() {
189           console.error('Error deleting photo');
190         }
191       });
192     }
193   },

```

Στη συνάρτηση destroy πλέον γίνεται έλεγχος αν ο χρήστης της φωτογραφίας είναι ο ίδιος με τον συνδεδεμένο χρήστη. Αν ταυτίζονται, η φωτογραφία διαγράφεται και στέλνετε πίσω στο ajax ένα json επιτυχίας, αλλιώς ένα αντίστοιχο json αποτυχίας.

```

26   def destroy
27     @photo_to_delete = Photo.find(params[:id])
28
29     if @photo_to_delete.user_id == current_user.id
30       @photo_to_delete.destroy
31       render json: { success: true, message: "Photo deleted successfully." }
32     else
33       render json: { success: false, message: "You are not authorized to delete this photo." }
34     end
35   end

```

Για παράδειγμα σωστής λειτουργίας της υλοποίησης, θα γίνει διαγραφή της φωτογραφίας Tree με id 5 από τον δημιουργό της, τον χρήστη 1.

Πριν τη διαγραφή:

```

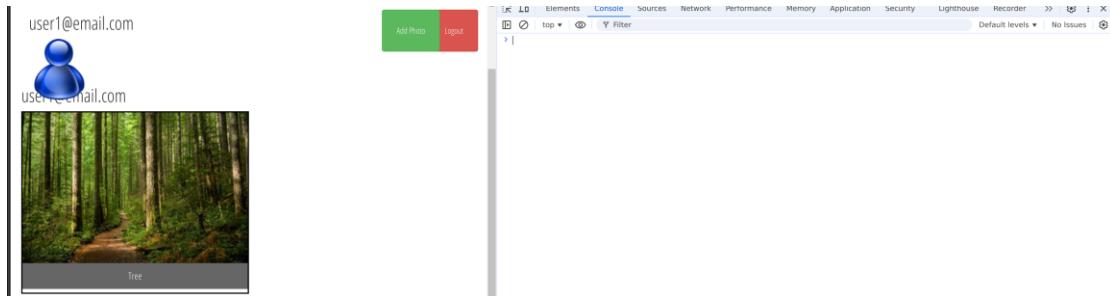
Photo Load (0.7ms)  SELECT "photos".* FROM "photos"
=> #< ActiveRecord::Relation [#<Photo id: 1, user_id: 1, caption: nil, created_at: "2024-12-21 02:37:44", updated_at: "2024-12-21 02:37:44", image_file_name: "51rCXKaVGTL.AC_UF894_1000_QL80_.jpg", image_content_type: "image/jpeg", image_file_size: 42326, image_updated_at: "2024-12-21 02:37:44", title: "ball">, #<Photo id: 5, user_id: 1, caption: nil, created_at: "2024-12-21 03:01:10", updated_at: "2024-12-21 03:01:10", image_file_name: "istockphoto-1446199740-612x612.jpg", image_content_type: "image/jpeg", image_file_size: 84861, image_updated_at: "2024-12-21 03:01:10", title: "Tree">, #<Photo id: 6, user_id: 2, caption: nil, created_at: "2024-12-21 03:27:01", updated_at: "2024-12-21 03:27:02", image_file_name: "sheep-closeup-eating-grass.jpg", image_content_type: "image/jpeg", image_file_size: 36911, image_updated_at: "2024-12-21 03:27:01", title: "Sheep">, #<Photo id: 7, user_id: 2, caption: nil, created_at: "2024-12-21 09:53:46", updated_at: "2024-12-21 09:53:46", image_file_name: "360_F_293712283_EGPqlm1bxpH0ZnrngyjRBol9GnJm2ST7.j...", image_content_type: "image/jpeg", image_file_size: 42603, image_updated_at: "2024-12-21 09:53:46", title: "rock">] >

```

```

Comment Load (0.1ms)  SELECT "comments".* FROM "comments"
=> #< ActiveRecord::Relation [#<Comment id: 1, photo_id: 5, user_id: nil, comment_text: "What a nice forest!">, #<Comment id: 2, photo_id: 5, user_id: 1, comment_text: "Nature is beautiful">, #<Comment id: 3, photo_id: 6, user_id: 1, comment_text: "Bee">, #<Comment id: 4, photo_id: 6, user_id: 1, comment_text: "Nice wool!">, #<Comment id: 5, photo_id: 6, user_id: 2, comment_text: "Nature is great.">, #<Comment id: 6, photo_id: 6, user_id: 2, comment_text: "Test comment">, #<Comment id: 7, photo_id: 6, user_id: 1, comment_text: "New comment">, #<Comment id: 8, photo_id: 5, user_id: 1, comment_text: "Another test comment">, #<Comment id: 9, photo_id: 5, user_id: 1, comment_text: "test">, #<Comment id: 10, photo_id: 5, user_id: 1, comment_text: "test 2">] >

```



Μετά τη διαγραφή:

```

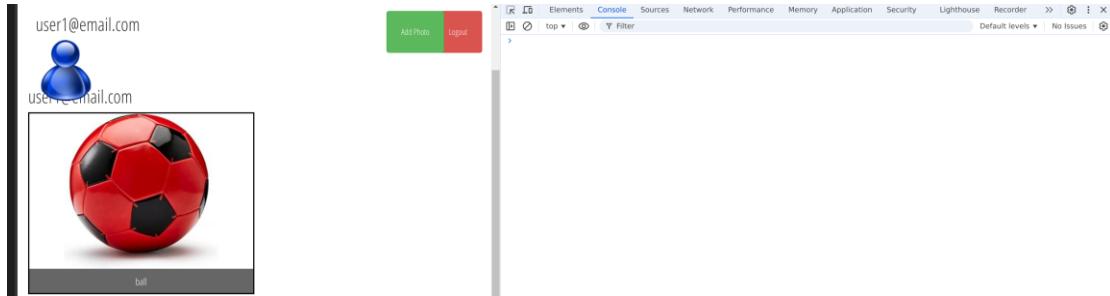
Photo Load (0.2ms)  SELECT "photos".* FROM "photos"
=> #< ActiveRecord::Relation [#<Photo id: 1, user_id: 1, caption: nil, created_at: "2024-12-21 02:37:44", updated_at: "2024-12-21 02:37:44", image_file_name: "51rCXKaVGTL.AC_UF894_1000_QL80_.jpg", image_content_type: "image/jpeg", image_file_size: 42326, image_updated_at: "2024-12-21 02:37:44", title: "ball">, #<Photo id: 6, user_id: 2, caption: nil, created_at: "2024-12-21 03:27:01", updated_at: "2024-12-21 03:27:02", image_file_name: "sheep-closeup-eating-grass.jpg", image_content_type: "image/jpeg", image_file_size: 36911, image_updated_at: "2024-12-21 03:27:01", title: "Sheep">, #<Photo id: 7, user_id: 2, caption: nil, created_at: "2024-12-21 09:53:46", updated_at: "2024-12-21 09:53:46", image_file_name: "360_F_293712283_EGPqlm1bxpH0ZnrngyjRBol9GnJm2ST7.j...", image_content_type: "image/jpeg", image_file_size: 42603, image_updated_at: "2024-12-21 09:53:46", title: "rock">] >

```

```

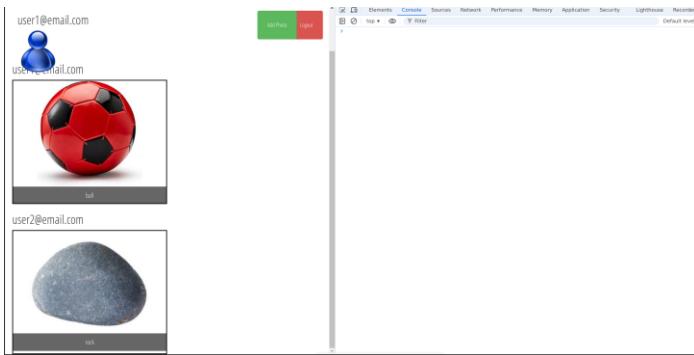
Comment Load (0.2ms)  SELECT "comments".* FROM "comments"
=> #< ActiveRecord::Relation [#<Comment id: 3, photo_id: 6, user_id: 1, comment_text: "Bee">, #<Comment id: 4, photo_id: 6, user_id: 1, comment_text: "Nice wool!">, #<Comment id: 5, photo_id: 6, user_id: 2, comment_text: "Nature is great.">, #<Comment id: 6, photo_id: 6, user_id: 2, comment_text: "Test comment">, #<Comment id: 7, photo_id: 6, user_id: 1, comment_text: "New comment">] >

```

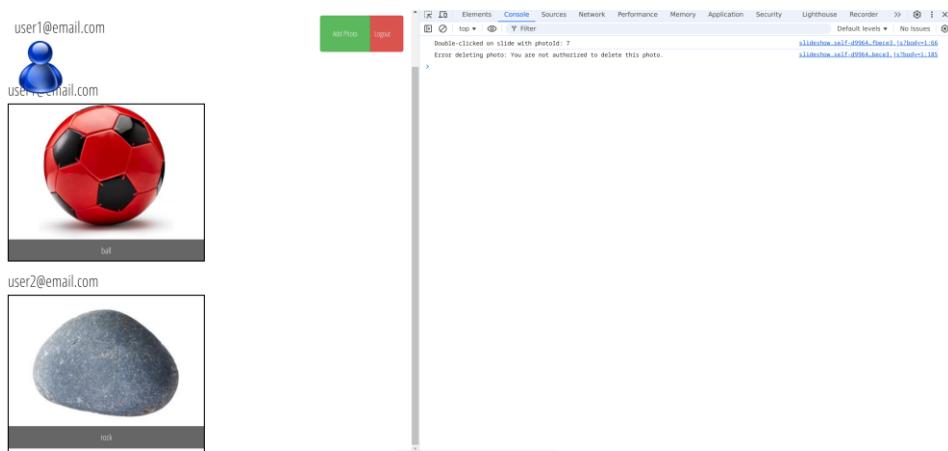


Δοκιμή διαγραφής μιας φωτογραφίας του χρήστη 2 από τον χρήστη 1:

Πριν τη δοκιμή διαγραφής της φωτογραφίας rock:



Μετά τη δοκιμή διαγραφής:



## 4. Παραδοτέα αρχεία

Στη παράδοση έχει δοθεί ένα directory treegram το οποίο περιέχει όλα τα καινούργια και τροποποιημένα αρχεία σε κατάλληλη μορφή φακέλων όπως βρίσκονται και στην εικονική μηχανή. Στον φάκελο της βάσης δεδομένων δίνονται τα migrate, το schema και το sql script με το οποίο δημιουργήθηκαν τα παραδείγματα στις εικόνες, και άλλες δοκιμές. Έχει παρατηρηθεί ότι αν τοποθετηθεί το sql script, η εφαρμογή μπορεί να τρέξει απλά με την εντολή server, αν δεν τοποθετηθεί θα είναι απαραίτητη η εκτέλεση του rake db:migrate σκέτο ή με το πρόθεμα bundle exec. Σε δοκιμή στην εικονική μηχανή της προηγούμενης άσκησης παρατηρήθηκε πρόβλημα με τη σωστή φόρτωση των εικόνων για φωτογραφίες που είχαν ήδη δημιουργηθεί στο παράδειγμα της βάσης που δίνεται, αλλά η εφαρμογή δούλευε κανονικά. Στην εικονική μηχανή που έχει δοθεί για αυτή την άσκηση δεν έχει παρατηρηθεί αυτό το θέμα. Τέλος έχουν δοθεί και οι φωτογραφίες που χρησιμοποιήθηκαν, και αν επιθυμείτε να τρέξετε τη βάση που έχει χρησιμοποιηθεί ως παράδειγμα, οι χρήστες έχουν email της μορφής

userX@email.com, όπου X ο αριθμός (υπάρχουν μόνο δύο και η αρίθμηση ξεκινάει από το 1), με κοινό κωδικό user.