ΜΥΕ042 - Τεχνολογίες Διαδικτύου
Διδάσκων: Στέργιος Αναστασιάδης

Αναφορά 1ης Εργαστηριακής Άσκησης: Εφαρμογή ιστού για κοινοχρησία φωτογραφιών στο Ruby-on-Rails

Ομάδα:
Κονδυλία Βέργου 4325
Παναγιώτης Βουζαλής 2653

Χειμερινό Εξάμηνο 2022

Περιεχόμενα

Σημείωση: Logins Χρηστών για Επίδειξη της Τελικής Εφαρμογής

Για την παρουσίαση της τελικής εφαρμογής μας, μπορείτε να συνδεθείτε στο treegram που υποβάλλαμε στο turnin χρησιμοποιώντας τα παρακάτω credentials:
- username: bob@gmail.com passwd: bob
- username: tim@gmail.com passwd: tim
- username: linus@gmail.com passwd: linus

Το turnin μας μπορείτε να το βρείτε και στο παρακάτω link (click me).

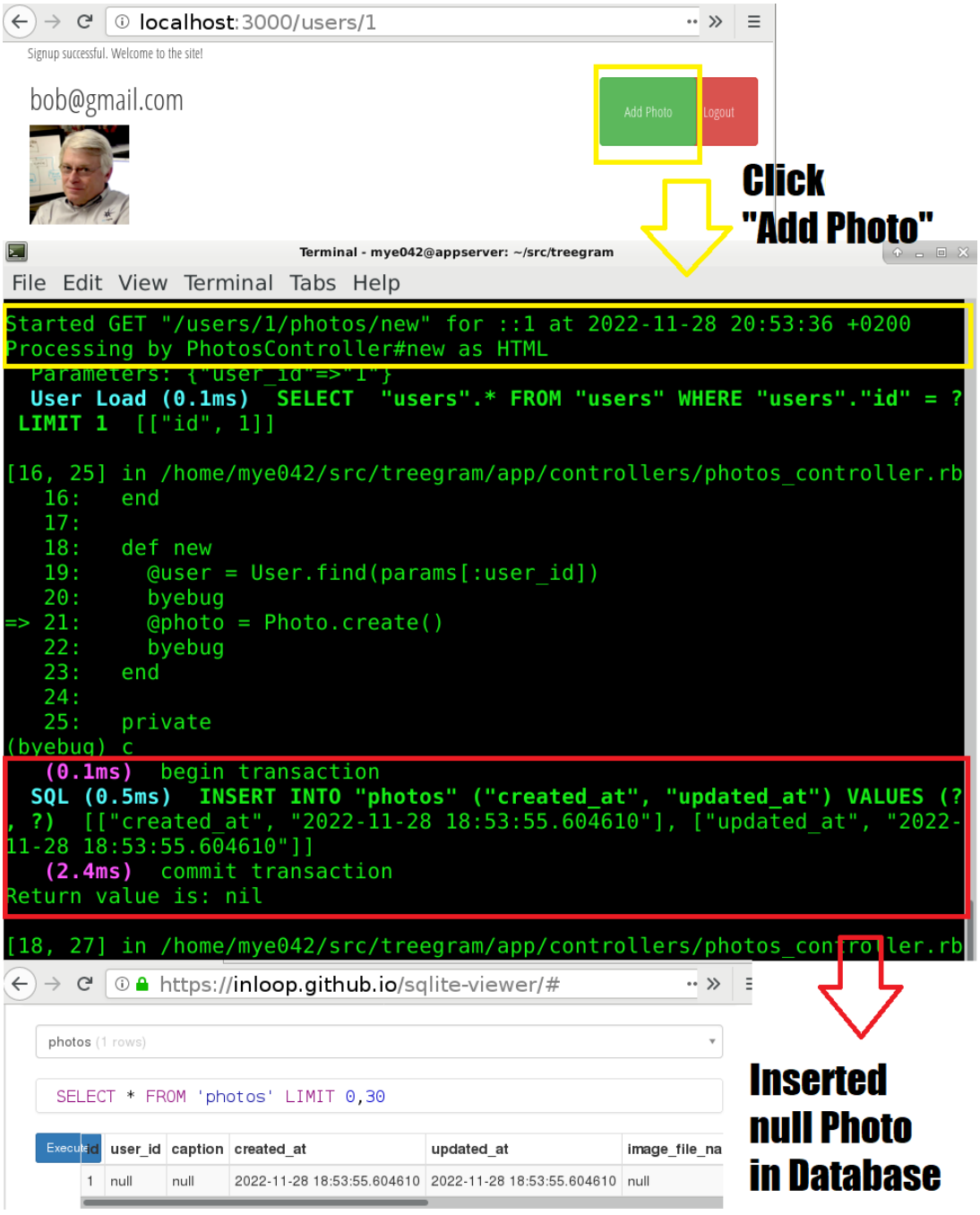
Λίστα Τροποποιημένων Αρχείων

app/models	**app/db/migrate/**
photo.rb	TIMESTAMP_add_title_to_photos.rb
follow.rb	TIMESTAMP_create_follows.rb
comment.rb	TIMESTAMP_create_comments.rb
app/controllers/	**app/views/**
users_controller.rb	users/show.html.haml
photos_controller.rb	photos/new.html.haml
follows_controller.rb	follows/index.html.haml
comments_controller.rb	comment/new.html.haml
app/config/	**app/assets/stylesheets/**
routes.rb	application.sass

1. Προϋπάρχον Error: Δημιουργία Null Photo

Όταν πατούμε το κουμπί Add Photo, δημιουργείται μια null Photo πριν καν φορτωθεί η προβολή new.html.haml. Αυτό γίνεται, διότι η συνάρτηση Ruby που υλοποιεί την ενέργεια new στον PhotoController καλεί την Photo.create().

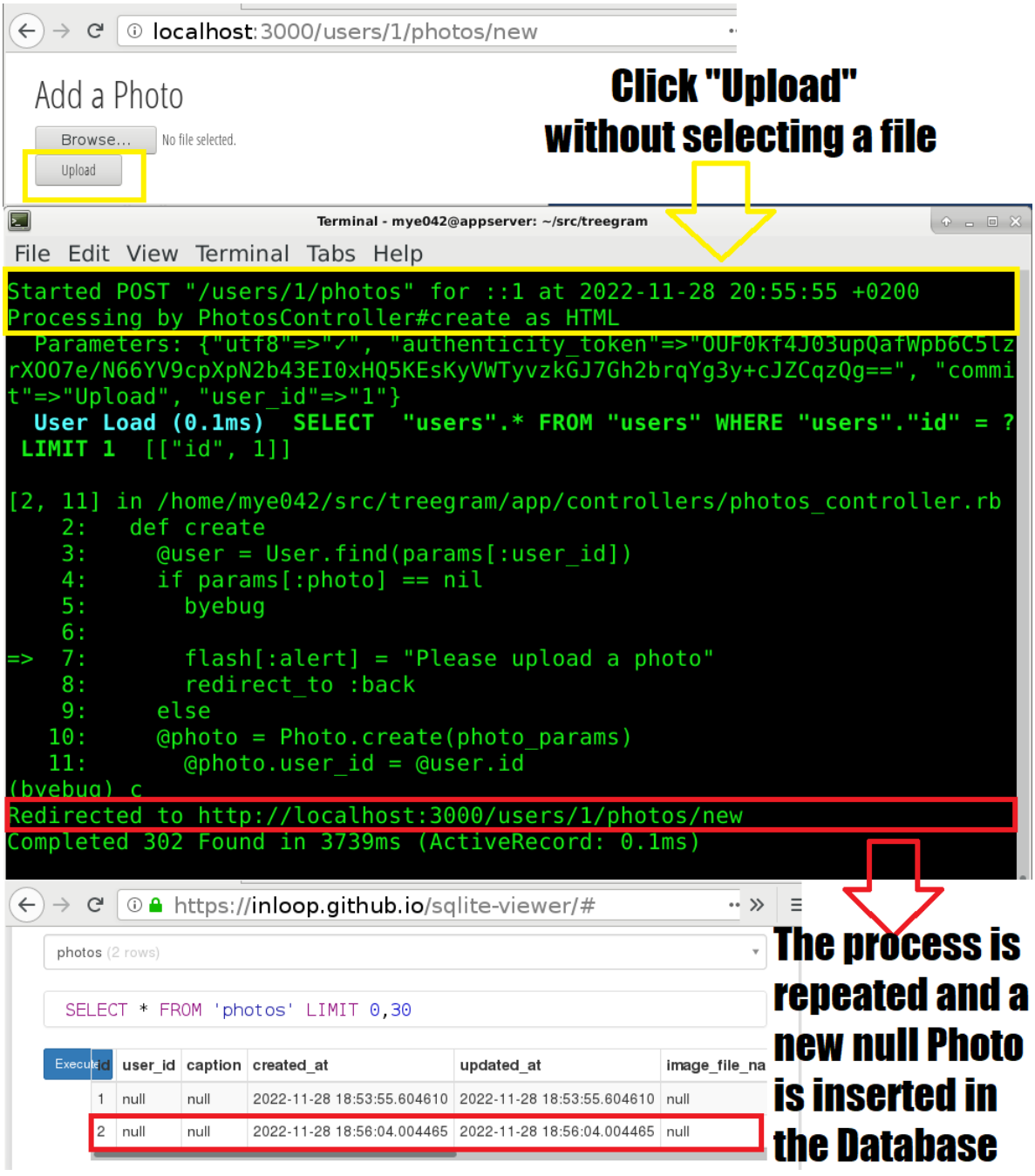
Επιβεβαιώσαμε τα παραπάνω, χρησιμοποιώντας τον διαδραστικό αποσφαλματωτή byebug και τον online SQLite Viewer.



Αφού ολοκληρωθεί η κλήση της new, φορτώνεται η προβολή στον διακομιστή μας.

Σε αυτό το σημείο, μπορούμε να ξαναδημιουργήσουμε μια null Photo, αν πατήσουμε στην φόρμα προσθήκης Photo το Upload χωρίς να έχουμε προσθέσει φωτογραφία.

Αυτό γίνεται καθώς η συνάρτηση Ruby που υλοποιεί την ενέργεια create στον PhotoController, αν δεν έχει επιλεχθεί φωτογραφία, κάνει redirect στην new και επαναλαμβάνονται όσα αναφέραμε παραπάνω.



Για να εξαλείψουμε αυτό το πρόβλημα στην υλοποίησή μας, αφαιρέσαμε τις δυο γραμμές κώδικα της συνάρτησης new του αρχείου “app/controllers/photos_controller.rb”:

```
$ @user = User.find(params[:user_id])
$ @photo = Photo.create()
```

Η πρώτη γραμμή έκανε απλά GET έναν User που δεν τον χρησιμοποιούσε και η δεύτερη γραμμή δημιουργούσε τη null Photo.

2. Προσθήκη Τίτλου σε Φωτογραφία

2.1. Δημιουργία Migration add_title_to_photos

Από το guides.rubyonrails.org διαβάσαμε ότι είναι προτιμότερο να δημιουργήσουμε καινούργιο migration παρά να τροποποιήσουμε ένα ήδη υπάρχον.

5 Changing Existing Migrations

Occasionally you will make a mistake when writing a migration. If you have already run the migration, then you cannot just edit the migration and run the migration again: Rails thinks it has already run the migration and so will do nothing when you run `bin/rails db:migrate`. You must rollback the migration (for example with `bin/rails db:rollback`), edit your migration, and then run `bin/rails db:migrate` to run the corrected version.

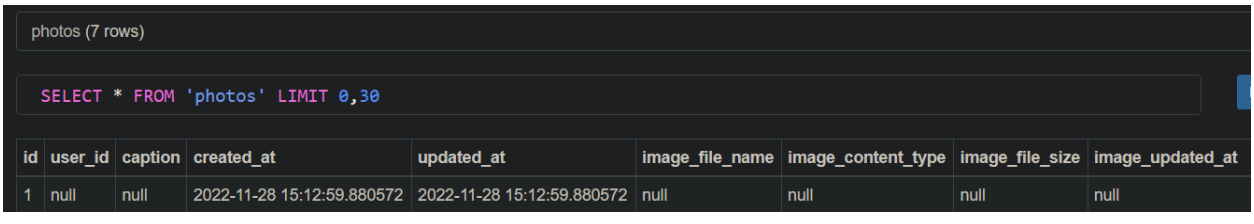
In general, editing existing migrations is not a good idea. You will be creating extra work for yourself and your co-workers and cause major headaches if the existing version of the migration has already been run on production machines. Instead, you should write a new migration that performs the changes you require. Editing a freshly generated migration that has not yet been committed to source control (or, more generally, which has not been propagated beyond your development machine) is relatively harmless.

The `revert` method can be helpful when writing a new migration to undo previous migrations in whole or in part (see Reverting Previous Migrations above).

Για αυτόν τον λόγο θα φτιάξουμε ένα καινούργιο migration έτσι ώστε να προσθέσουμε τον τίτλο στις φωτογραφίες.

Για την προσθήκη του column “title” στον πίνακα “photos” της βάσης μας εκτελούμε τις εξής εντολές στο root directory της εφαρμογής:

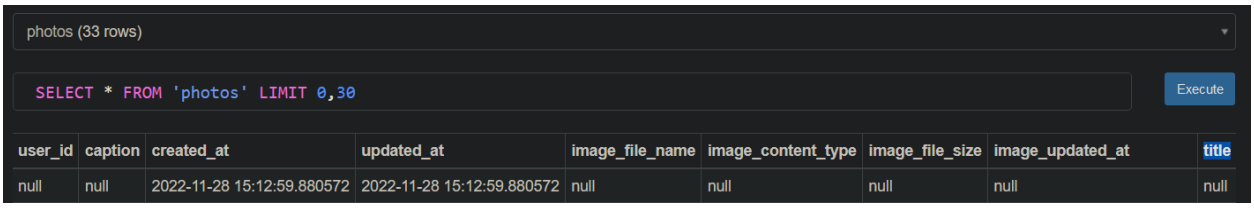
```
$ rails generate migration add_title_to_photos title:string
$ rake db:migrate
```



photos (7 rows)

```
SELECT * FROM 'photos' LIMIT 0,30
```

id	user_id	caption	created_at	updated_at	image_file_name	image_content_type	image_file_size	image_updated_at
1	null	null	2022-11-28 15:12:59.880572	2022-11-28 15:12:59.880572	null	null	null	null



photos (33 rows)

```
SELECT * FROM 'photos' LIMIT 0,30
```

Execute

user_id	caption	created_at	updated_at	image_file_name	image_content_type	image_file_size	image_updated_at	title
null	null	2022-11-28 15:12:59.880572	2022-11-28 15:12:59.880572	null	null	null	null	null

2.2. Επεξεργασία Προβολής photos/new

Στο αρχείο "app/views/photos/new.html.haml" προσθέτουμε τις παρακάτω γραμμές:

```
$ %h2 Add a title: #{form.text_field :title, placeholder:"Look at my dog!"}
$ = link_to 'Back to home view', user_path(current_user)
```

Συνεπώς, το prompt για το upload φωτογραφίας θα φαίνεται έτσι:

2.3. Επεξεργασία Ελεγκτή photos_controller

Στο αρχείο "app/controllers/photos_controller.rb" προσθέτουμε τους εξής ελέγχους, για να αναγκάζεται ο χρήστης να έχει επιλέξει φωτογραφία προς ανέβασμα και να έχει προσθέσει τίτλο, πριν του επιτραπεί να προχωρήσει:

```
$ if params[:photo][:image] == nil
$     flash[:alert] = "Please upload a photo."
$     redirect_to :back
$ elsif params[:photo][:title] == ""
$     flash[:alert] = "You forgot to add a title!"
$     redirect_to :back
```

Επίσης βάζουμε το argument `:title` στη γραμμή

```
$ params.require(:photo).permit(:image, :title)
```

```
class PhotosController < ApplicationController
  def create
    @user = User.find(params[:user_id])
    if params[:photo][:image] == nil
      flash[:alert] = "Please upload a photo."
      redirect_to :back
    elsif params[:photo][:title] == ""
      flash[:alert] = "You forgot to add a title!"
      redirect_to :back
    ..............
  private
  def photo_params
    params.require(:photo).permit(:image, :title)
  end
```

2.4. Επεξεργασία Προβολής users/show

Στο αρχείο “app/views/users/show.html.haml” προσθέτουμε την εξής γραμμή για να υποδείξουμε στον κώδικά μας να εμφανίζει τον τίτλο κάθε φωτογραφίας:

```
$ %h2= photo.title
```

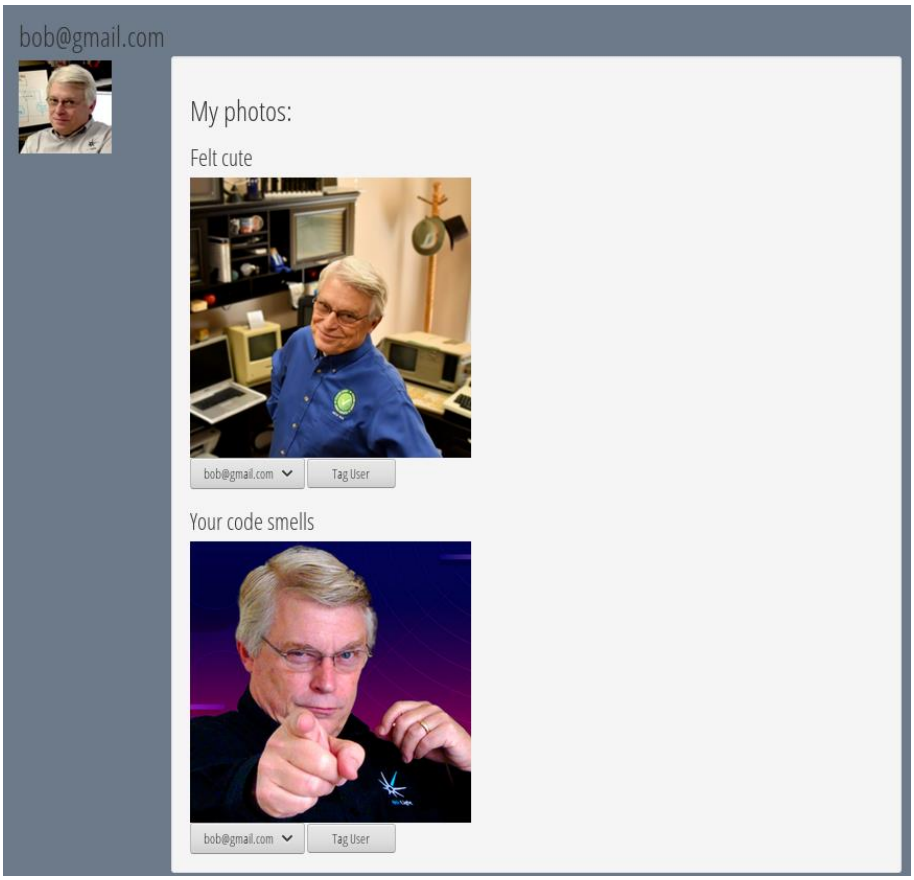
Επίσης προσθέτουμε την εξής γραμμή για να εμφανίζεται το όνομα και το catchphrase της εφαρμογής όσο τη χρησιμοποιούμε:

```
$ %h1.title Treegram - Show off to the world!
```

```
.row.top_row
  %h1.title Treegram - Show off to the world!
  ....................
  - @user.photos.each  do |photo|
    .well.col-sm-4
      %h2= photo.title
```

2.5. Παράδειγμα Χρήσης Title σε Φωτογραφίες

Επομένως, η εφαρμογή μας φαίνεται έτσι, όταν έχουμε κάνει log σε ένα user και έχουμε ανεβάσει φωτογραφίες.



3. Προσθήκη Λειτουργίας follow

3.1. Δημιουργία Μοντέλου follow και Migration create_follows

Δημιουργούμε το μοντέλο follow με την παρακάτω εντολή:

`$ rails generate model follow`

```
mye042@appserver:~/Desktop/project/treegram-notis$ rails generate model follow
Array values in the parameter to `Gem.paths=` are deprecated.
Please use a String or nil.
An Array ({"GEM_PATH"=>["/usr/local/rvm/gems/ruby-2.5.1", "/usr/local/rvm/gems/
Running via Spring preloader in process 6094
      invoke  active_record
      create    db/migrate/20221129113245_create_follows.rb
      create    app/models/follow.rb
      invoke    rspec
      create      spec/models/follow_spec.rb
```

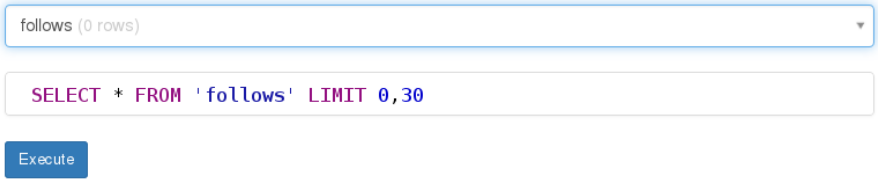
Ταυτόχρονα δημιουργείται το αρχείο “app/db/migrate/TIMESTAMP_create_follows.rb” στο οποίο ορίζουμε τη δομή του πίνακα με τα αναγνωριστικά του follower (αυτός που ακολουθεί τα post κάποιου) και του followee (αυτός του οποίου τα post ακολουθούν άλλοι):

```ruby
class CreateFollows < ActiveRecord::Migration
  def change
    create_table :follows do |t|
      t.integer :follower_id
      t.integer :followee_id
      t.timestamps null: false
    end
    # follows are unique between follower and followee
    add_index :follows, [:follower_id, :followee_id], unique: true
  end
end
```

Εκτελούμε `$ rake db:migrate` για να δημιουργήσουμε τον πίνακα των follow:

```
mye042@appserver:~/Desktop/project/treegram-notis$ rake db:migrate
== 20221129113245 CreateFollows: migrating ===================================
-- create_table(:follows)
   -> 0.0005s
-- add_index(:follows, [:follower_id, :followee_id], {:unique=>true})
   -> 0.0004s
== 20221129113245 CreateFollows: migrated (0.0010s) ==========================
```

Ο οποίος σε αυτό το σημείο είναι άδειος:



3.2. Δημιουργία Ελεγκτή follows_controller

Σε αυτό το σημείο πρέπει να δημιουργήσουμε τη σχέση μεταξύ των πινάκων follow και user. Αυτό θα το κάνουμε με έναν follows controller.

Δημιουργούμε τον follows controller με την εξής εντολή:

```
$ rails generate controller follows
```

```
mye042@appserver:~/Desktop/src_old/treegram (copy 1)$ rails generate controller follows
Array values in the parameter to `Gem.paths=` are deprecated.
Please use a String or nil.
An Array ({"GEM_PATH"=>["/usr/local/rvm/gems/ruby-2.5.1", "/usr/local/rvm/gems/ruby-2.5.
Running via Spring preloader in process 7509
      create  app/controllers/follows_controller.rb
      invoke  erb
      create    app/views/follows
      invoke  rspec
      create    spec/controllers/follows_controller_spec.rb
      invoke  helper
      create    app/helpers/follows_helper.rb
      invoke    rspec
      create      spec/helpers/follows_helper_spec.rb
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/follows.coffee
      invoke    scss
      create      app/assets/stylesheets/follows.scss
```

Τροποποιούμε το αρχείο “app/controllers/follows_controller.rb” προσθέτοντας λειτουργία follow/unfollow:

```ruby
class FollowsController < ApplicationController

    # get the user follows
    def index
        @users = User.all
        @not_current_user = User.where.not(id: current_user)
    end

    # post the user follows
    def create
        # followee means that we follow that user
        @followee = User.find(params[:followee_id])
        # follow functionality
        @follow = Follow.create(follower_id: params[:user_id], followee_id: params[:followee_id])

        flash[:notice] = "You successfully followed #{@followee.email}!"
        redirect_to :back
    end

    def destroy
        # followee means that we follow that user
        @followee = User.find(params[:followee_id])
        # unfollow functionality
        @unfollow = Follow.destroy(follower_id: params[:user_id], followee_id: params[:followee_id])

        flash[:notice] = "You successfully unfollowed #{@followee.email}!"
        redirect_to :back
    end
end
```

3.3. Επεξεργασία Αρχείου Διαδρομών routes

Τροποποιούμε το αρχείο “app/config/routes.rb” προσθέτοντας τη γραμμή

```
$ resources :follows
```

```
Rails.application.routes.draw do
  get '/' => 'home#index'
  resources :users do
    resources :photos
    resources :follows
  end

  resources :tags, only: [:create, :destroy]
  get '/log-in' => "sessions#new"
  post '/log-in' => "sessions#create"
  get '/log-out' => "sessions#destroy", as: :log_out

end
```

3.4. Επεξεργασία Μοντέλου follow

Τροποποιούμε το αρχείο “app/models/follow.rb” προσθέτοντας τη γραμμή

```
$ validates :follower_id, uniqueness: {scope: :followee_id}
```

```
1   class Follow < ActiveRecord::Base
2       validates :follower_id, uniqueness: {scope: :followee_id}
3   end
4
```

3.5. Επεξεργασία Προβολής follow/index

Δημιουργούμε το αρχείο “/app/views/follows/index.html.haml” για προβολή όλων των χρηστών που μπορεί να ακολουθήσει ένας χρήστης:

```
%h1 Would you like to follow some of these users?
%table
  %tbody
    - @not_current_user.each do |user|
      %tr
        %td.bold= user.email
        %td= button_to 'Follow', {:controller => 'follows', :action => '
          create', :followee_id => user.id}, {:method => :post }
%br
= link_to 'Back to home view', user_path(current_user)
```

3.6. Επεξεργασία Ελεγκτή users_controller

Τροποποιούμε το αρχείο "/app/controllers/users_controller.rb".
Προθέτουμε κώδικα για τη δημιουργία ενός array που περιέχει το προσωπικό δίκτυο κάθε χρήστη της εφαρμογής (αποτελείται απο τον ίδιο το χρήστη και απο τα άτομα που ακολουθεί):

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

    # create our user network
    @user_network = [User.find(params[:id])]
    Follow.all.each do |u|
      if u.follower_id == @user.id
        # add to the end of this array
        @user_network << User.find(u.followee_id)
      end
    end
  end
```

3.7. Επεξεργασία Προβολής users/show

Τροποποιούμε το αρχείο "/app/views/users/show.html.haml" για να προσθέσουμε:
- Κουμπί για εμφάνιση εγγεγραμμένων χρηστών
- Εμφάνιση φωτογραφιών του χρήστη και ατόμων που ακολουθεί ο χρήστης (folowees) σε αντίστροφη χρονολογική σειρά.

Σημειώνουμε πως προς το παρόν τα άτομα που έχει ακολουθήσει ένας χρήστης εμφανίζονται σε κανονική χρονολογική σειρά (οι φωτογραφίες τους εμφανίζονται όμως όπως ζητείται σε αντίστροφη χρονολογική σειρά).

```
  .row
    = link_to 'Share a photo!', new_user_photo_path(@user), class: ['btn', 'btn-success', 'add_photo_btn']
    = link_to 'Follow people!', user_follows_path(@user), class: ['btn', 'btn-primary', 'follow_users_btn']
.........................
.row
  - @user_network.each do |user|
    - user_photos = user.photos.sort.reverse
    - if user_photos.any?
      .well.col-sm-8
        - if user == current_user
          %h2.text My photos:
        - else
          %h2.text= user.email
        %img
          -# Display each photo
          - user_photos.each do |photo|
            %h3.text= photo.title
            %img{alt: photo.title, src: photo.image.url(:medium)}
            -# Tag form
            = form_for @tag do |f|
              = f.hidden_field :photo_id, value: photo.id
              = f.collection_select :user_id, @users, :id, :email
              = f.submit "Tag User"
            - photo.tags.each do |tag|
              = tag.user.email
```

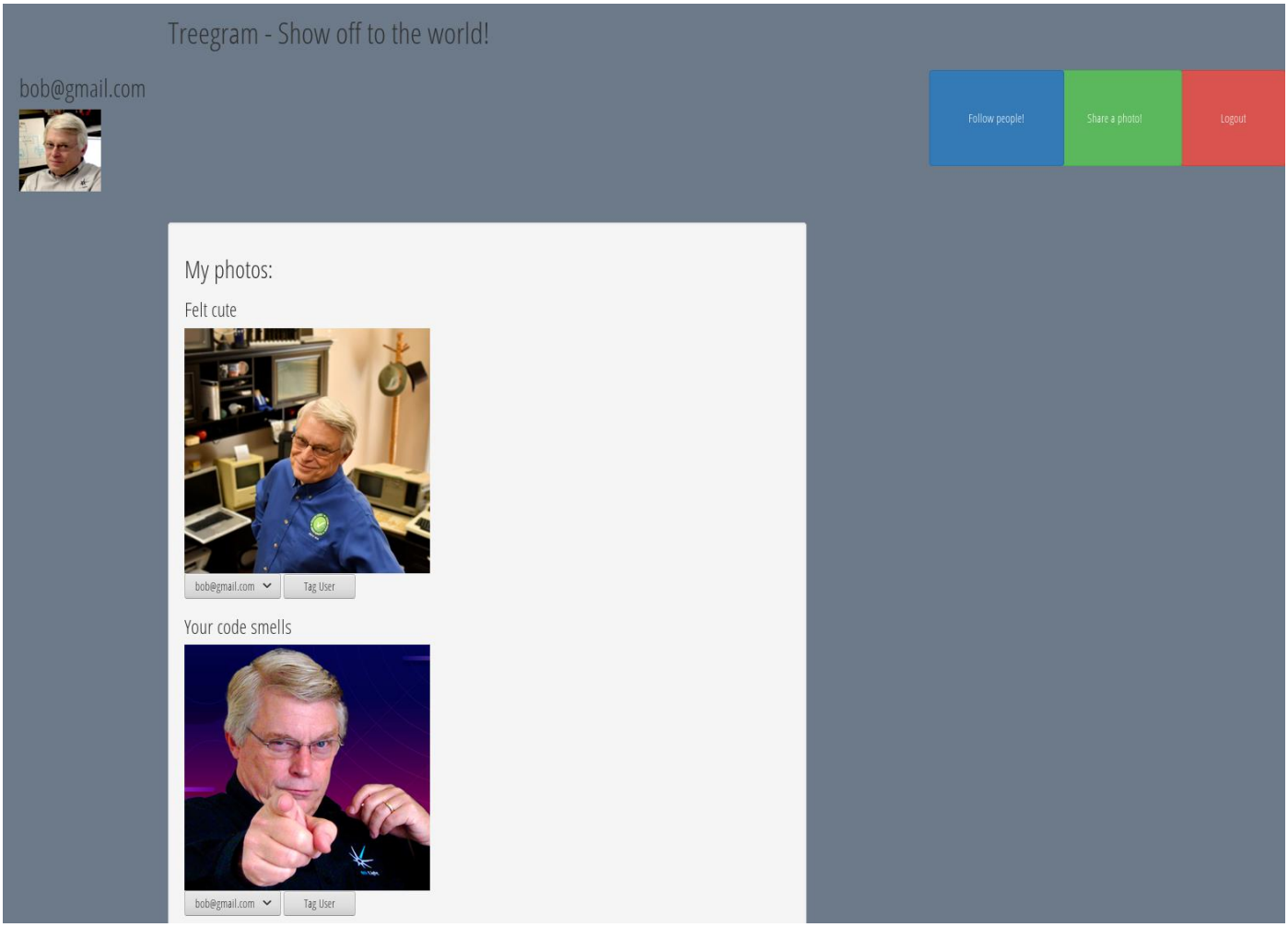
3.8. Τοποθέτηση Κατάλληλου CSS

Τροποποιούμε το αρχείο “app/assets/stylesheets/application.sass” για τη σωστή προβολή των κουμπιών και των text πεδίων username και τίτλου φωτογραφίας:

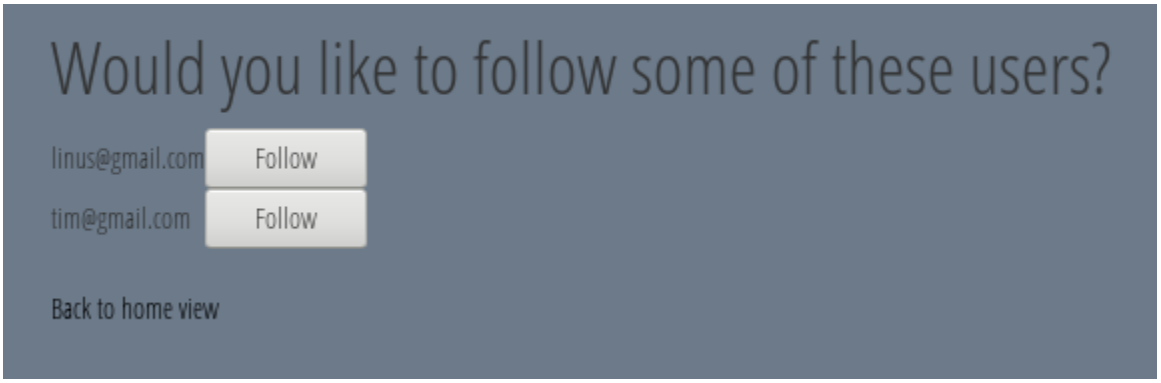
```
body
  font-family: 'Open Sans Condensed', sans-serif
  background-color: hsl(211, 12, 48)

a
  text-decoration: none
  color: black

.logout_btn
  position: fixed
  right: 0
  padding: 3%
  margin: 15px

.top_row
  max-height: fill-avaliable

.add_photo_btn
  position: fixed
  right: 8%
  padding: 3%
  margin: 15px

.follow_users_btn
  position: fixed
  right: 17%
  padding: 3%
  margin: 15px

.email
  font-size: 7em

.user_att
  position: fixed
  left: 0
  margin-left: 15px

.center
  margin: auto

.text
  text-align: left

.bold
  font-weight: bold
```

3.9. Παρουσίαση Εφαρμογής Μέχρι Τώρα

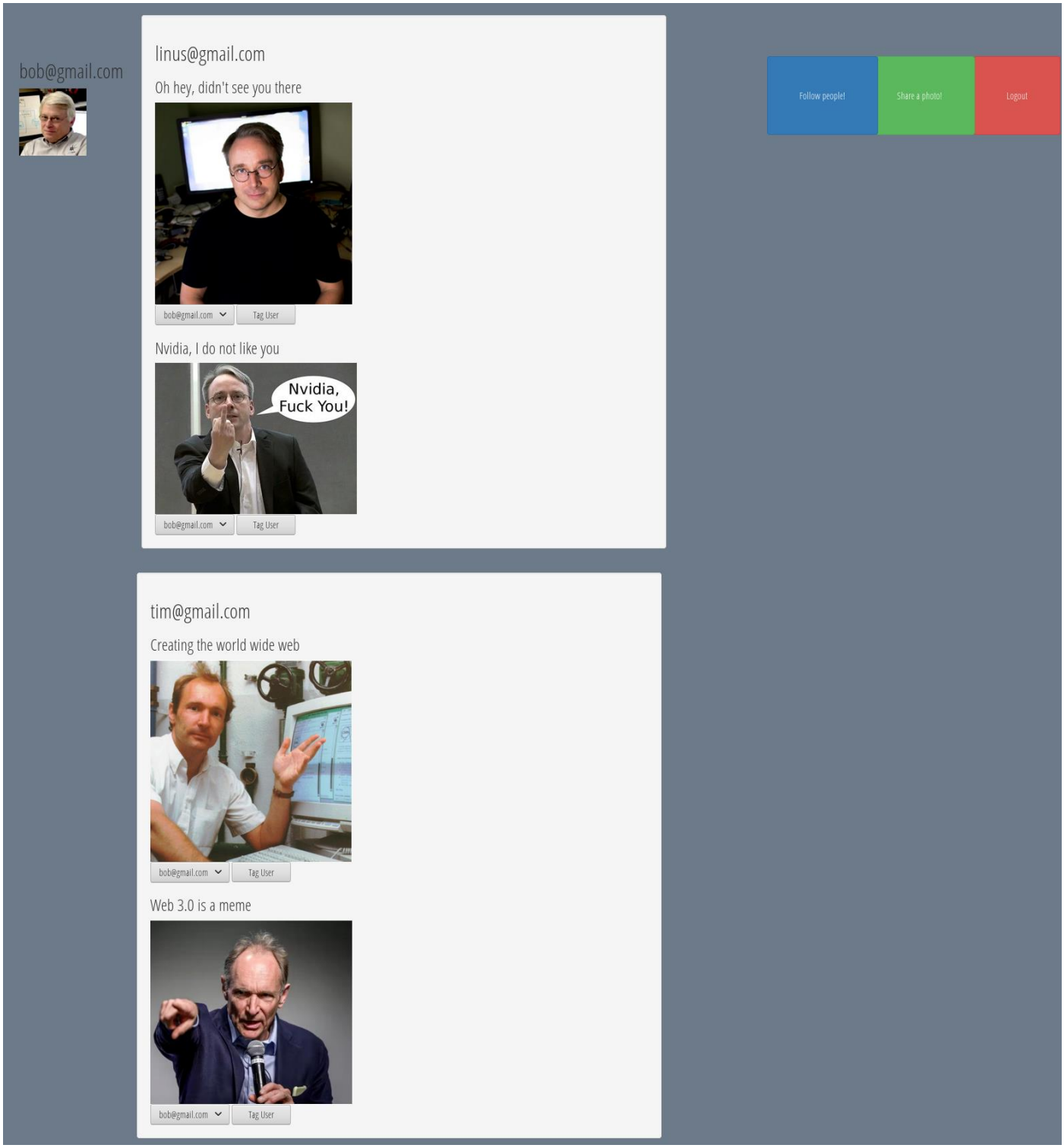
Αρχική σελίδα που βλέπει ο χρήστης:



Prompt που βλέπει ο χρήστης όταν πατήσει το κουμπί follow people:



Σκρολλάροντας προς τα κάτω ο χρήστης μπορεί να δεί όλο το feed του:



4. Προσθήκη Λειτουργίας comments

4.1. Δημιουργία Μοντέλου comment και Migration create_comments

Δημιουργούμε το μοντέλο follow με την παρακάτω εντολή:

```
$ rails generate model comment
```

```
mye042@appserver:~/Desktop/project/treegram-notis/app$ rails generate model comment
Array values in the parameter to `Gem.paths=` are deprecated.
Please use a String or nil.
An Array ({"GEM_PATH"=>["/usr/local/rvm/gems/ruby-2.5.1", "/usr/local/rvm/gems/ruby-2.5.1@global"]}) wa
s passed in from bin/rails:3:in `load'
Running via Spring preloader in process 10708
      invoke  active_record
      create    db/migrate/20221129221936_create_comments.rb
      create    app/models/comment.rb
      invoke    rspec
      create      spec/models/comment_spec.rb
```

Έπειτα στο αρχείο “app/db/migrate/TIMESTAMP_create_comments.rb” ορίζουμε τη δομή του πίνακα με τα αναγνωριστικά του comment:

```
class CreateComments < ActiveRecord::Migration
  def change
    create_table :comments do |t|
      t.references :photo, index: true, foreign_key: true
      t.references :user, index: true, foreign_key: true
      t.text :context

      t.timestamps null: false
    end
  end
end
```

Εκτελούμε `$ rake db:migrate` για να δημιουργήσουμε τον πίνακα των comment:

```
mye042@appserver:~/Desktop/project/treegram-notis/app$ rake db:migrate
(in /home/mye042/Desktop/project/treegram-notis)
== 20221129221936 CreateComments: migrating ===================================
-- create_table(:comments)
   -> 0.0012s
== 20221129221936 CreateComments: migrated (0.0012s) ==========================
```

4.2. Δημιουργία Ελεγκτή comments_controller

Δημιουργούμε τον comments controller με την εξής εντολή:

```
$ rails generate controller comments
```

```
mye042@appserver:~/Desktop/project/treegram-notis/app$ rails generate controller comments
Array values in the parameter to `Gem.paths=` are deprecated.
Please use a String or nil.
An Array ({"GEM_PATH"=>["/usr/local/rvm/gems/ruby-2.5.1", "/usr/local/rvm/gems/ruby-2.5.1@
s passed in from bin/rails:3:in `load'
Running via Spring preloader in process 10761
      create  app/controllers/comments_controller.rb
      invoke  erb
      create    app/views/comments
      invoke  rspec
      create    spec/controllers/comments_controller_spec.rb
      invoke  helper
      create    app/helpers/comments_helper.rb
      invoke    rspec
      create      spec/helpers/comments_helper_spec.rb
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/comments.coffee
      invoke    scss
      create      app/assets/stylesheets/comments.scss
```

Τροποποιούμε το αρχείο “app/controllers/comments_controller.rb” προσθέτοντας λειτουργία δημιουργίας comment:

```
class CommentsController < ApplicationController

  # get our user comments
  def index
    @photo_comments = Comment.where(photo_id: params[:photo_id])
  end

  # post the user comments
  def create
    @comment = Comment.create(photo_id: params[:photo_id],
        user_id: params[:user_id], context: params[:comment][:context])
    flash[:notice] = "You successfully added a comment"
    redirect_to :back
  end
end
```

4.3. Επεξεργασία Μοντέλου comment

Τροποποιούμε το αρχείο “app/models/comment.rb” προσθέτοντας τις γραμμές

```
$     belongs_to :user
$     belongs_to :photo
```

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

4.4. Επεξεργασία Μοντέλου photo

Τροποποιούμε το αρχείο “app/models/photo.rb” προσθέτοντας τη γραμμή

```
$ has_many :comments
```

```ruby
class Photo<ActiveRecord::Base
  belongs_to :user
  has_many :tags
  has_many :comments
```

4.5. Επεξεργασία Αρχείου Διαδρομών routes

Τροποποιούμε το αρχείο “app/config/routes.rb” προσθέτοντας τη γραμμή

```
$ resources :comments
```

```ruby
Rails.application.routes.draw do
  get '/' => 'home#index'
  resources :users do
    resources :photos do
      resources :comments
    resources :follows
  end
```

4.6. Δημιουργία Προβολής comment/new

Δημιουργούμε το αρχείο για τη φόρμα υποβολής comment:

```
%br
%br
.text
  = form_for Comment.new(), :url => user_photo_comments_path do |f|
    = label :context, 'Add a Comment!'
    %br
    = f.text_area :context
    %br
    = f.submit 'Post'
    %br
%br
= link_to 'Back to home view', user_path(current_user)
```

4.7. Επεξεργασία Προβολής users/show

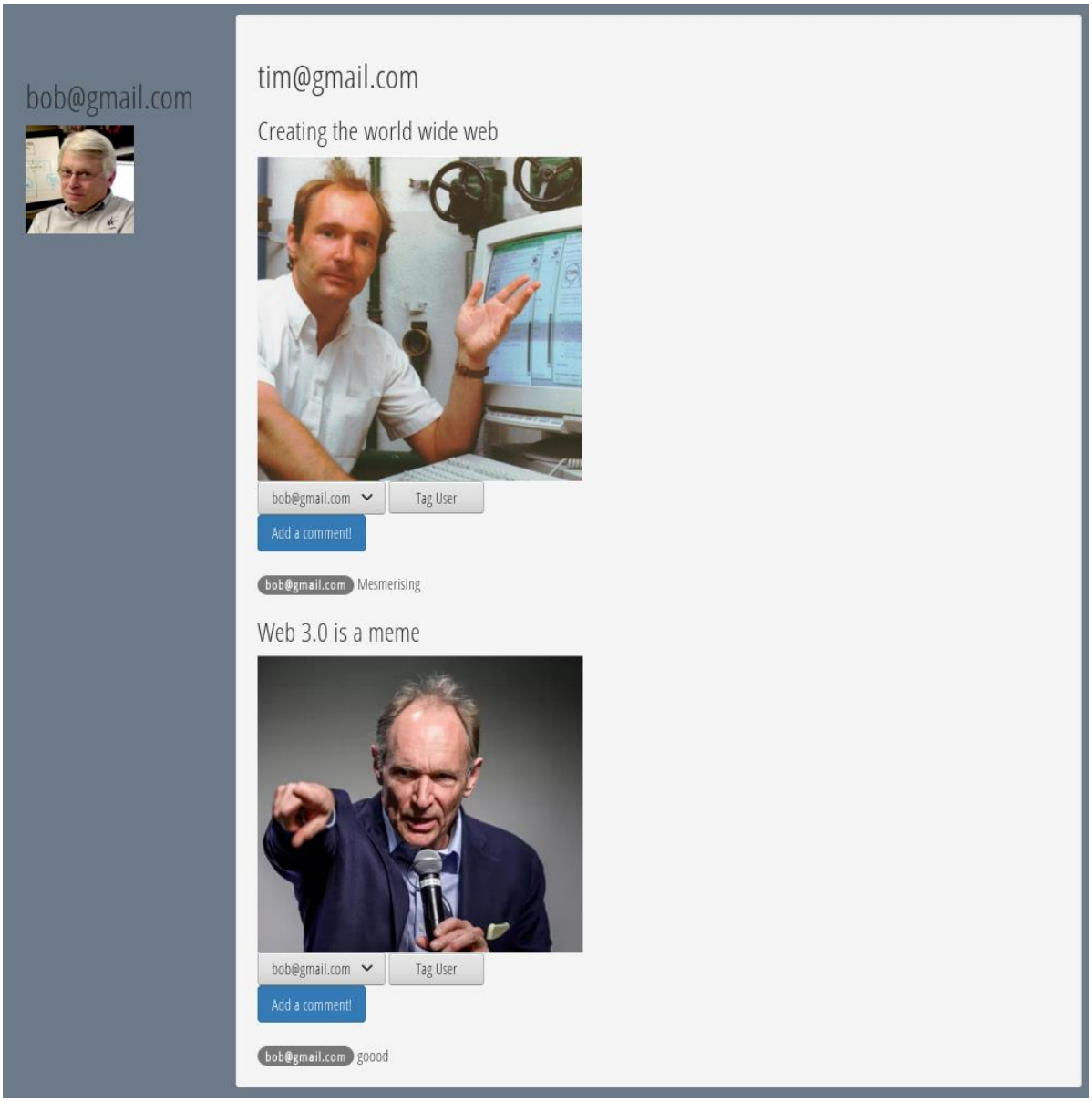
Τροποποιούμε το αρχείο "/app/views/users/show.html.haml" για να προσθέσουμε:

- Κουμπί για προσθήκη νέου comment κάτω από κάθε φωτογραφία
- Προβολή λίστας σχολίων κάτω από κάθε φωτογραφία

```
.row
  - @user_network.each do |user|
    - user_photos = user.photos.sort.reverse
    - if user_photos.any?
      .well.col-sm-8
        - if user == current_user
          %h2.text My photos:
        - else
          %h2.text= user.email
        %img
          -# Display each photo
          - user_photos.each do |photo|
            %h3.text= photo.title
            %img{alt: photo.title, src: photo.image.url(:medium)}
            -# Tag form
            = form_for @tag do |f|
              = f.hidden_field :photo_id, value: photo.id
              = f.collection_select :user_id, @users, :id, :email
              = f.submit "Tag User"
            - photo.tags.each do |tag|
              = tag.user.email
            -# Add a comment
            = button_to "Add a comment!", new_user_photo_comment_path(photo_id: photo.id, user_id: current_user.id), method:
                :get, :class => "btn btn-primary"
            -# Display comments
            - photo.comments.each do |c|
              %br
              %span.badge.badge-secondary
                #{c.user.email}
              = c.context
```

4.8. Παρουσίαση Εφαρμογής Μέχρι Τώρα

Τα comments βρίσκονται κάτω απο κάθε φωτογραφία με κανονική σειρά (όπως και στο αληθινό instagram)



Prompt που βλέπει ο χρήστης όταν πατήσει το κουμπί add a comment:

5. Bonus: Προσθήκη Λειτουργίας Διαγραφής post

5.1. Επεξεργασία Μοντέλου photo

Τροποποιούμε το αρχείο "/app/models/photo.rb" για να προσθέσουμε τις γραμμές:
$ has_many :tags, dependent: :destroy
$ has_many :comments, dependent: :destroy
Έτσι ώστε όταν διαγράφεται μια φωτογραφία να καταστρέφονται και τα tags και τα comments που σχετίζονται με αυτή

```
class Photo<ActiveRecord::Base
  belongs_to :user
  has_many :tags, dependent: :destroy
  has_many :comments, dependent: :destroy
```

5.2. Επεξεργασία Προβολής users/show

Τροποποιούμε το αρχείο "/app/views/users/show.html.haml" για να προσθέσουμε:
- Κουμπί για διαγραφή φωτογραφίας εαν η φωτογραφία ανήκει στο χρήστη
Επιπλέον αλλάξαμε την λειτουργία των tags: Κάθε χρήστης μπορεί πια να κάνει tags μόνο στις δικές του φωτογραφίες.

```
%img
  -# Display each photo
  - user_photos.each do |photo|
    %h3.text= photo.title
    %img{alt: photo.title, src: photo.image.url(:medium)}
    -# Tag form
    - if user.id.equal?(current_user.id)
      = form_for @tag do |f|
        = f.hidden_field :photo_id, value: photo.id
        = f.collection_select :user_id, @users, :id, :email
        = f.submit "Tag User"
      Tagged users:
    - photo.tags.each do |tag|
      - if tag.equal?(photo.tags.last)
        = tag.user.email
      - else
        = tag.user.email + ","
    -# Add a comment
    = button_to "Add a comment!", new_user_photo_comment_path(photo_id: photo.id,
      user_id: current_user.id), method: :get, :class => "btn btn-primary"
    - if user.id.equal?(current_user.id)
      = button_to "Delete post", user_photo_path(photo_id: photo.id, user_id:
        current_user.id), method: :delete, :class => "btn btn-danger"
    -# Display comments
    Comments:
    - photo.comments.each do |c|
      %br
      %span.badge.badge-secondary
        #{c.user.email}
      = c.context
```

5.3. Επεξεργασία Ελεγκτή photos_controller

Τροποποιούμε το αρχείο "/app/controllers/photos_controller.rb" για να προσθέσουμε τη μέθοδο destroy που υλοποιεί τη λειτουργία διαγραφής post.

```
class PhotosController < ApplicationController
  def create
    @user = User.find(params[:user_id])
    if params[:photo][:image] == nil
      flash[:alert] = "Please upload a photo."
      redirect_to :back
    elsif params[:photo][:title] == ""
      flash[:alert] = "You forgot to add a title!"
      redirect_to :back
    else
    @photo = Photo.create(photo_params)
      @photo.user_id = @user.id
      @photo.save
      flash[:notice] = "Successfully uploaded a photo!"
      redirect_to user_path(@user)
    end
  end

  def new
    #@user = User.find(params[:user_id])
    #@photo = Photo.create()
  end

  def destroy
    @photo = Photo.find(params[:photo_id])
    @photo.destroy
    redirect_to user_path(id: params[:user_id])
  end

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

6. Τελική Παρουσίαση Εφαρμογής