



MODUL 151

TEIL 7: POST DETAIL, POST DELETE, LIKE MODEL, LIKE CONTROLLER, POST LIKE

Raloh Maurer

Inhaltsverzeichnis AB151-07

Modul 151: Instagram mit Rails bauen

Teil 7: Post Detail, Post Delete, Like Model, Like Controller, Post Like

Inhaltsverzeichnis AB151-07	1
Modul 151: Instagram mit Rails bauen	1
Post Detail.....	2
Post Delete.....	6
Berechtigung Post Delete.....	8
Model: Posts mit Like ergänzen	8
Likes-Controller	11
Post Like.....	13
Auftrag: Quicknotes AB151-07	22

Post Detail

In diesem Abschnitt erstellen wir die Detailansicht für einen Post. Betrachten wir die Routen unserer Instagram Applikation. Gehen Sie im Terminal ins Applikations-verzeichnis und geben Sie das Kommando ein: `rails routes`

```
vagrant@bmLP1:~/workspace/instagram$ cd workspace/instagram/
vagrant@bmLP1:~/workspace/instagram$ atom .
vagrant@bmLP1:~/workspace/instagram$ rails routes
Prefix Verb URI Pattern
      root GET  /
new_user_session GET  /users/sign_in(.:format)
user_session POST  /users/sign_in(.:format)
destroy_user_session DELETE /users/sign_out(.:format)
new_user_password GET  /users/password/new(.:format)
edit_user_password GET  /users/password/edit(.:format)
user_password PATCH /users/password(.:format)
                  PUT   /users/password(.:format)
                  POST  /users/password(.:format)
cancel_user_registration GET  /users/cancel(.:format)
new_user_registration GET  /users/sign_up(.:format)
edit_user_registration GET  /users/edit(.:format)
user_registration PATCH /users(.:format)
                  PUT   /users(.:format)
                  DELETE /users(.:format)
                  POST  /users(.:format)
users GET  /users(.:format)
user GET  /users/:id(.:format)
post_photos POST  /posts/:post_id/photos(.:format)
posts GET  /posts(.:format)
      POST  /posts(.:format)
      GET  /posts/:id(.:format)
      DELETE /posts/:id(.:format)
rails_service_blob GET  /rails/active_storage/blobs/:signed_id/*filename(.:format)
rails_blob_representation GET  /rails/active_storage/representations/:signed_blob_id/:variation_key/*filename(.:format)
rails_disk_service GET  /rails/active_storage/disk/:encoded_key/*filename(.:format)
update_rails_disk_service PUT  /rails/active_storage/disk/:encoded_token(.:format)
rails_direct_uploads POST  /rails/active_storage/direct_uploads(.:format)

Controller#Action
posts#index
devise/sessions#new
devise/sessions#create
devise/sessions#destroy
devise/passwords#new
devise/passwords#edit
devise/passwords#update
devise/passwords#update
devise/passwords#create
registrations#cancel
registrations#new
registrations#edit
registrations#update
registrations#destroy
registrations#create
users#index
users#show
photos#create
posts#index
posts#create
posts#show
posts#destroy
active_storage/blobs#show
active_storage/representations#show
active_storage/disk#show
active_storage/disk#update
active_storage/direct_uploads#create

vagrant@bmLP1:~/workspace/instagram$
```

Vergewissern Sie sich, dass Sie einen GET Eintrag für `posts#show` haben. Falls nicht müssen Sie die Routen anpassen (`routes.rb`):

```
routes.rb
show.html.erb          _posts_list.html.erb        posts_controller.rb
1 Rails.application.routes.draw do
2   root 'posts#index'
3   devise_for :users, controllers: { registrations: 'registrations' }
4   # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
5   resources :users, only: [:index, :show]
6
7   resources :posts, only: [:index, :show, :create, :destroy] do
8     resources :photos, only: [:create]
9   end
10 end
11
```

Der Eintrag `resources :posts` ist notwendig. Mit `only:` schränken wir die Actions bzw. Methoden ein, dabei muss `:show` vorhanden sein.

Damit eine Detailview für die Posts erstellt werden kann, benötigen wir im Controller `app/controllers/posts/posts_controller.rb` die `show`-Methode, die wir bereits mit einem früheren Arbeitsblatt erstellt haben. Vergewissern Sie sich, dass die `show`-Methode wie folgt aussieht:

```
27   def show
28     @photos = @post.photos
29   end
```

Nun können Sie die dazu passende View erstellen, d.h. die Datei `app/views/posts/show.html.erb`.

Wir erstellen ein Grundgerüst mit zwei Spalten:

```
1  <div class="d-flex flex-column align-items-center mt-3">
2    <div class="row post box col-xl-10 col-lg-11 col-xs-12">
3      <div class="col-lg-8 col-md-7 px-0 d-flex post-show-img">
4        <div class="align-self-center">
5
6          </div>
7        </div>
8        <div class="col-lg-4 col-md-5 mt-sm-4 mt-md-0">
9          <div class="row px-3 d-flex align-items-center">
10
11        </div>
12        </div>|<br/>
13      </div>
14    </div>
```

Sie können aus der Partial-View `app/views/posts/_posts_list.html.erb` folgenden Bereich kopieren und in die 1. Spalte der Detailview (Zeile 5) einsetzen:

```
<% if post.photos.size == 1%>
  
<% else %>
  <div class="carousel slide" data-ride="carousel" id="carousel-post-<%= post.id %>">
    <div class="carousel-inner">
      <% post.photos.each do |photo| %>
        <% if photo == post.photos.first %>
          <div class="carousel-item active">
        <% else %>
          <div class="carousel-item">
        <% end %>
        
      </div>
      <% end %>
    </div>
    <a class="carousel-control-prev" href="#carousel-post-<%= post.id %>" role="button" data-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="sr-only">Previous</span>
    </a>
    <a class="carousel-control-next" href="#carousel-post-<%= post.id %>" role="button" data-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="sr-only">Next</span>
    </a>
  </div>
<% end %>
```

Beachten Sie, dass die Instanzvariable `post` mit `@` als Präfix angepasst werden muss, da diese aus der `show`-Methode des Posts-Controller bereitgestellt wird (an 4 Stellen!).

Als nächstes erstellen wir in der Partial-View `app/views/posts/_posts_list.html.erb` einen Link auf die Detailview.

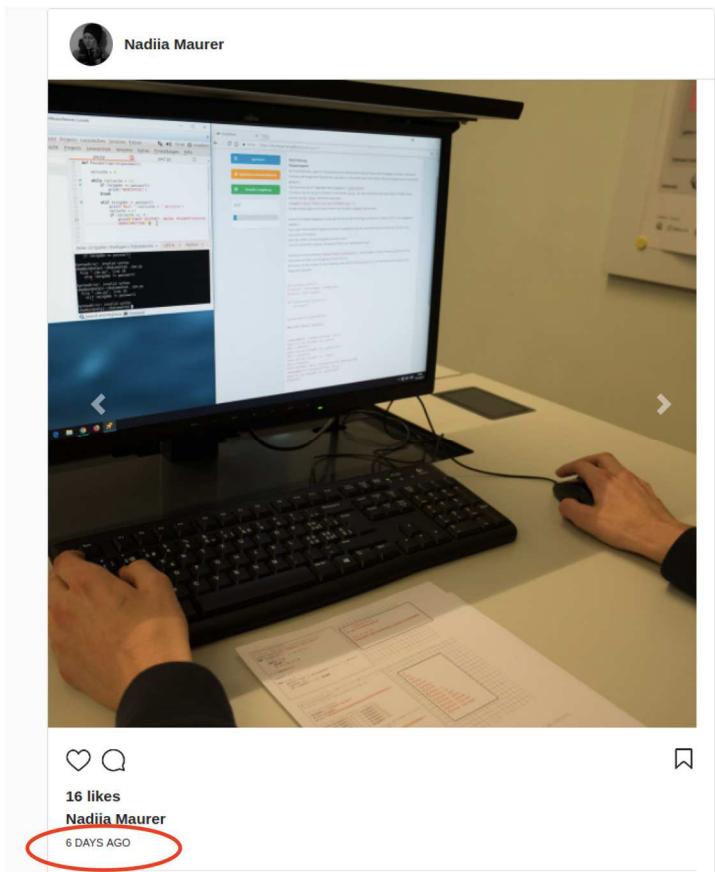
Wir löschen folgende zwei Zeilen unten in der Datei:

```
<div class="light-color post-time">
<%= time_ago_in_words(post.created_at).upcase %> AGO </div>
```

und ersetzen diese mit

```
<%= link_to time_ago_in_words(post.created_at).upcase + " AGO",
post_path(post), class: "light-color post-time no-text-decoration"%>
```

Der Link X DAYS AGO unterhalb der Posts sollte nun auf die Detailview führen:



Jetzt ergänzen wir die 2. Spalte der Detailview `posts#show`, so dass ein Link (Avatar und Benutzername) auf das Profil führt und Kommentare, Icons und weitere Informationen angezeigt werden. Dabei erzeugen wir im Moment 100 Dummy-Kommentare:

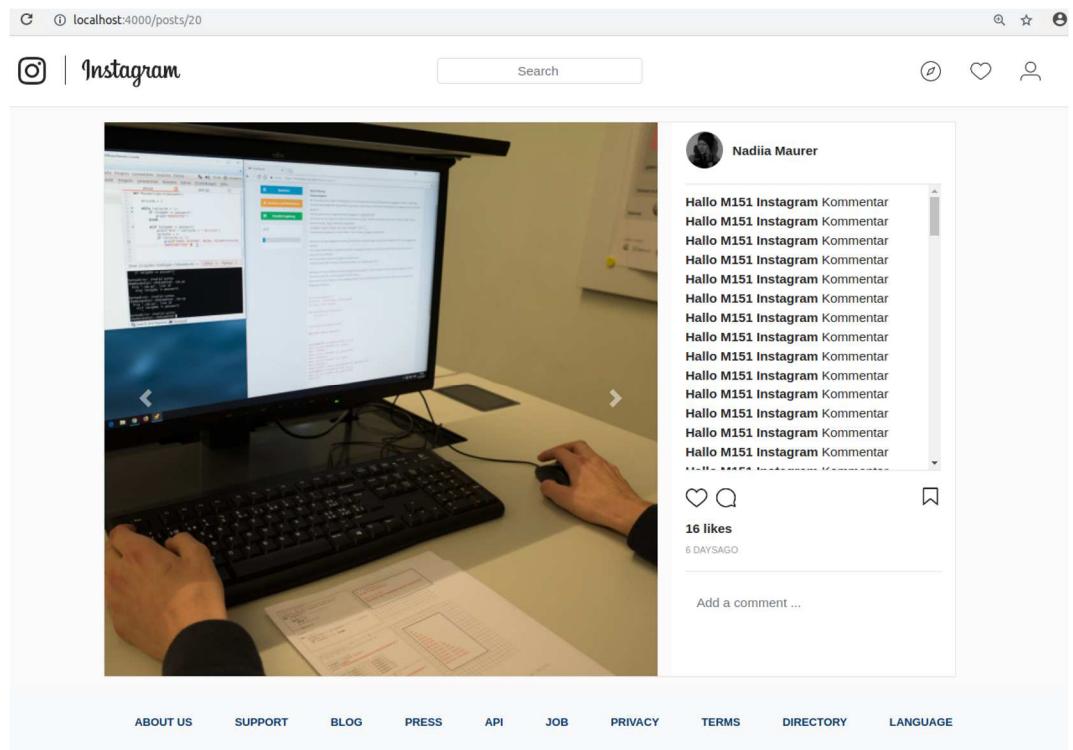
```
33 <div class="col-lg-4 col-md-5 mt-sm-4 mt-md-0">
34   <div class="row px-3 d-flex align-items-center">
35     <%= link_to user_path(@post.user), class: "no-text-decoration" do %>
36       <%= image_tag avatar_url(@post.user), class: "post-author-icon" %>
37     <% end %>
38     <%= link_to user_path(@post.user), class: "normal-color no-text-decoration d-flex align-self-center", title: @post.user.name do %>
39       <strong><%= @post.user.name %></strong>
40     <% end %>
41   </div>
42   <hr class="mb-0">
43   <div class="comment-list py-2">
44     <% if @post.content.present? %>
45       <div>
46         <span><strong><%= @post.user.name %></strong></span>
47         <span><%= @post.content %></span>
48       </div>
49     <% end %>
50     <!-- Load comment-->
51     <%100.times do%>
52       <div>
53         <span><strong>Hallo M151 Instagram </strong></span>
54         <span>Kommentar </span>
55       </div>
56     <% end %>
57   </div>
58   <hr class="mt-0">
59
60   <div class="row actions">
61     <a href="#" class="core-sprite love hide-text"> Love </a>
62     <a href="#" class="core-sprite comment hide-text"> Comment </a>
63     <a href="#" class="core-sprite bookmark hide-text ml-auto"> Bookmark </a>
64   </div>
65   <div><strong><%= pluralize(16, "like") %></strong></div>
66
67   <div class="light-color post-time"><%= time_ago_in_words(@post.created_at).upcase %>AGO</div>
68   <hr>
69   <div class="row actions">
70     <form action="#" class="w-100">
71       <div>
72         <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." rows="1"></textarea>
73       </div>
74     </form>
75   </div>
76
77   </div>
78 </div>
79 </div>
80 </div>
```

Die zwei obersten und die vier untersten Zeilen sind bereits vorhanden, fügen Sie einfach den restlichen Code ein.

Wir ergänzen app/assets/stylesheets/application.scss mit den zwei SCSS-Klassen comment-list und post-show-img:

```
.comment-list {  
    height: 310px;  
    overflow: auto;  
}  
  
.post-show-img {  
    background-color: #fafafa;  
    margin: -10px 0;  
}
```

Kontrollieren Sie die Detailview posts#show, indem Sie in der View mit allen Posts auf einen Link X DAYS AGO klicken:



Als Alternative für die drei Icons können wir wiederum die Icons von *Font Awesome* verwenden, was folgendermassen aussehen könnte:

```
<a href="#" class="far fa-heart fa-2x" style="color:#888888"></a>  
<a href="#" class="far fa-comment fa-2x" style="color:#888888"></a>  
<a href="#" class="far fa-bookmark fa-2x" style="color:#888888"></a>
```

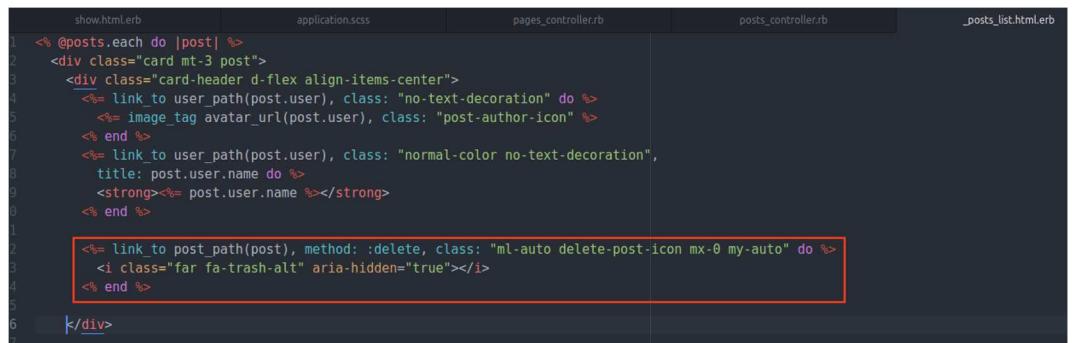
Post Delete

Um einen Post löschen zu können, müssen wir im Posts-Controller eine `destroy`-Methode programmieren.

Wir überprüfen zuerst, ob ein Benutzer berechtigt ist, einen Post zu löschen. Kann der Post gelöscht werden, geben wir eine Erfolgsmeldung (`flash[:notice]`) aus; ansonsten eine Fehlermeldung (`flash[:alert]`). Nach dem Vorgang führen wir den Benutzer auf die Startseite.

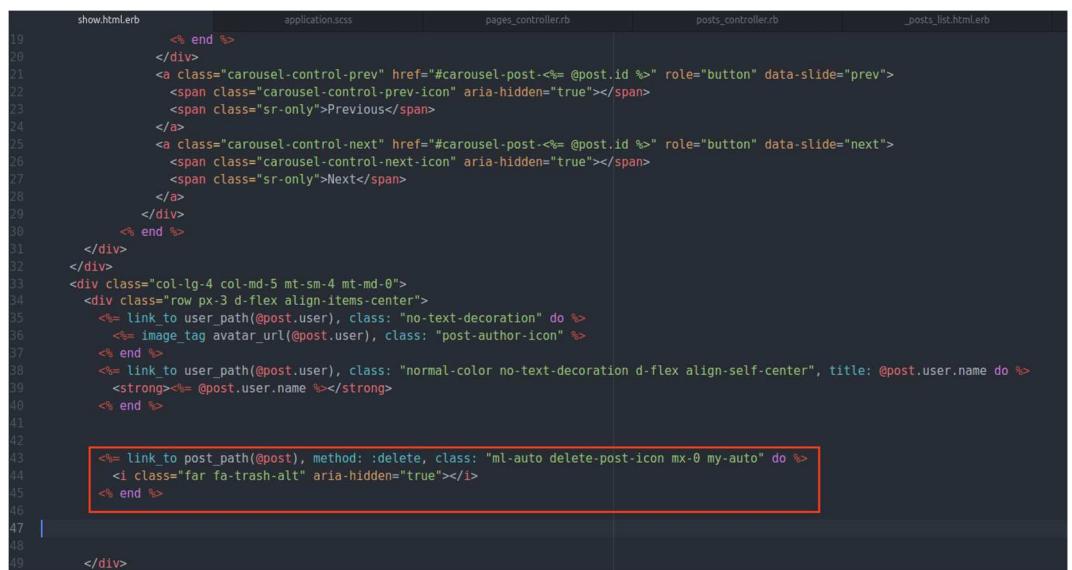
```
def destroy
  if @post.user == current_user
    if @post.destroy
      flash[:notice] = "Post deleted!"
    else
      flash[:alert] = "Something went wrong..."
    end
  else
    flash[:alert] = "You don't have the permission to delete this post!"
  end
  redirect_to root_path
end
```

In `app/views/posts/_posts_list.html` bauen wir einen Link ein:



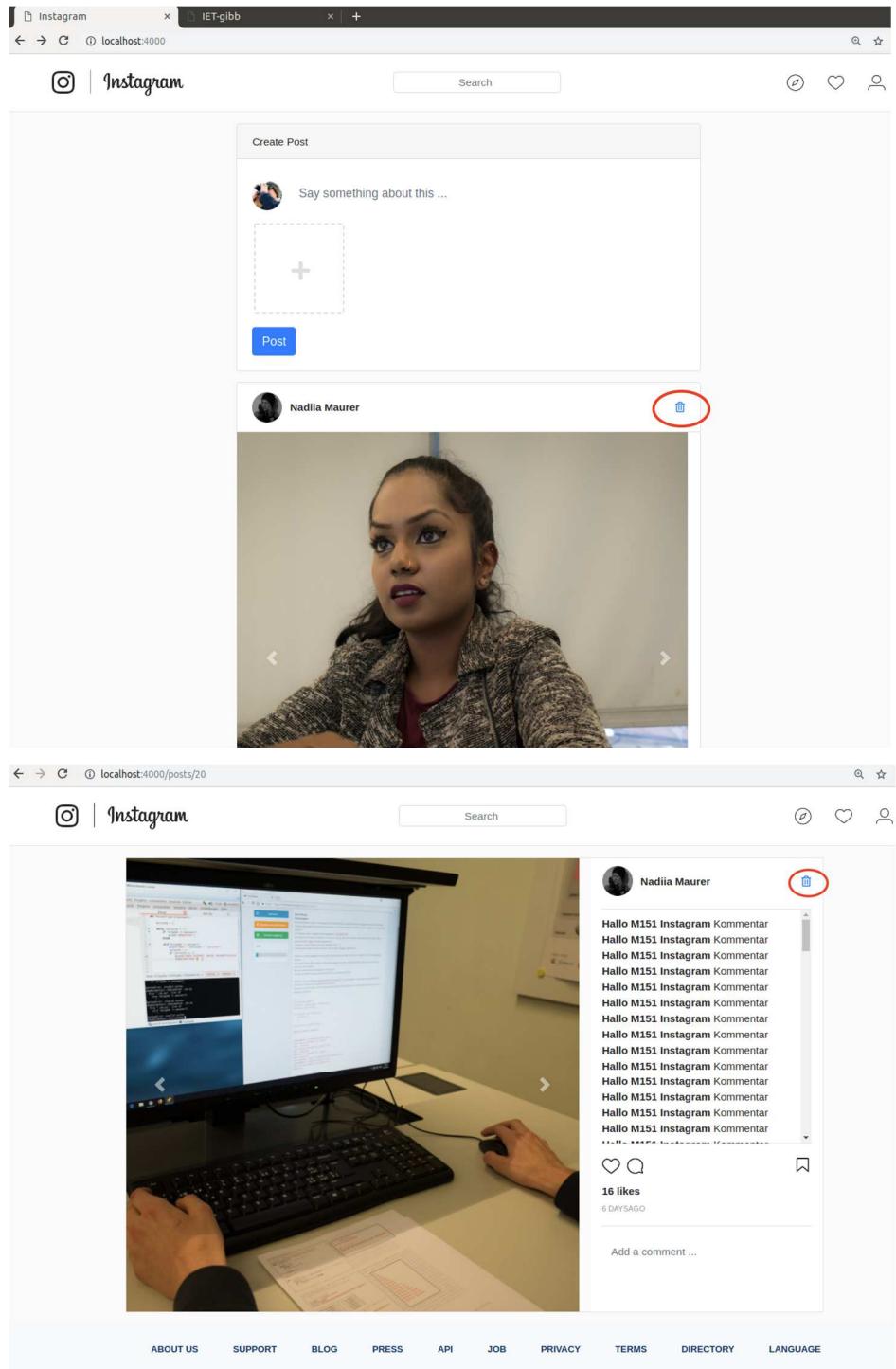
```
show.html.erb           application.scss          pages_controller.rb        posts_controller.rb      _posts_list.html.erb
1  <% @posts.each do |post| %>
2    <div class="card mt-3 post">
3      <div class="card-header d-flex align-items-center">
4        <%= link_to user_path(post.user), class: "no-text-decoration" do %>
5          <%= image_tag avatar_url(post.user), class: "post-author-icon" %>
6        <% end %>
7        <%= link_to user_path(post.user), class: "normal-color no-text-decoration",
8          title: post.user.name do %>
9          <strong><%= post.user.name %></strong>
10         <% end %>
11
12         <%= link_to post_path(post), method: :delete, class: "ml-auto delete-post-icon mx-0 my-auto" do %>
13           <i class="far fa-trash-alt" aria-hidden="true"></i>
14         <% end %>
15
16     </div>
```

Und den gleichen Link - aber mit einer Instanzvariable - schreiben wir auch in `app/views/posts/show.html.erb`:



```
show.html.erb           application.scss          pages_controller.rb        posts_controller.rb      _posts_list.html.erb
19   <% end %>
20   </div>
21   <a class="carousel-control-prev" href="#carousel-post-<%= @post.id %>" role="button" data-slide="prev">
22     <span class="carousel-control-prev-icon" aria-hidden="true"></span>
23     <span class="sr-only">Previous</span>
24   </a>
25   <a class="carousel-control-next" href="#carousel-post-<%= @post.id %>" role="button" data-slide="next">
26     <span class="carousel-control-next-icon" aria-hidden="true"></span>
27     <span class="sr-only">Next</span>
28   </a>
29   </div>
30   <% end %>
31 </div>
32 </div>
33 <div class="col-lg-4 col-md-5 mt-sm-4 mt-md-0">
34   <div class="row px-3 d-flex align-items-center">
35     <%= link_to user_path(@post.user), class: "no-text-decoration" do %>
36       <%= image_tag avatar_url(@post.user), class: "post-author-icon" %>
37     <% end %>
38     <%= link_to user_path(@post.user), class: "normal-color no-text-decoration d-flex align-self-center", title: @post.user.name do %>
39       <strong><%= @post.user.name %></strong>
40     <% end %>
41
42     <%= link_to post_path(@post), method: :delete, class: "ml-auto delete-post-icon mx-0 my-auto" do %>
43       <i class="far fa-trash-alt" aria-hidden="true"></i>
44     <% end %>
45   </div>
46
47   </div>
48 </div>
```

In den Views posts#index und posts#show haben wir nun blaue Mülleimer:



The image consists of two vertically stacked screenshots of a web-based Instagram clone. The top screenshot shows the 'Create Post' interface, which includes a placeholder for an image (a dashed square with a plus sign) and a 'Post' button. The bottom screenshot shows a user's profile page for 'Nadia Maurer'. It features a large photo of a woman, her name, and a delete icon (a blue trash can) in the top right corner of the header. Below the photo is a list of comments, each preceded by a delete icon. At the bottom of the post area, there are like, comment, and share buttons, followed by a timestamp ('6 DAYS AGO') and a placeholder for adding a comment.

Ergänzen Sie das Stylesheet app/assets/stylesheets/application.scss:

```
.delete-post-icon {  
  color: #262626;  
  font-size: 20px;  
}  
  
.delete-comment {  
  width: 11px;  
  height: 11px;  
  background-position: -216px -112px;  
  float: right;  
  margin: 5px 0 0 10px;  
}
```

Testen Sie die Delete-Funktion in beiden Views.

Berechtigung Post Delete

In der `destroy`-Methode verhindern wir bereits, dass ein fremder Post gelöscht werden kann. Wir bauen noch eine zweite Sicherheitsmaßnahme ein: Jeder Post wird ja via Fremdschlüssel einem Benutzer zugewiesen. Wir können zudem noch definieren, dass der Post genau dem Benutzer gehört, der ihn erstellt hat. Hierzu programmieren wir die Methode `belongs_to?` in `app/models/post.rb`:

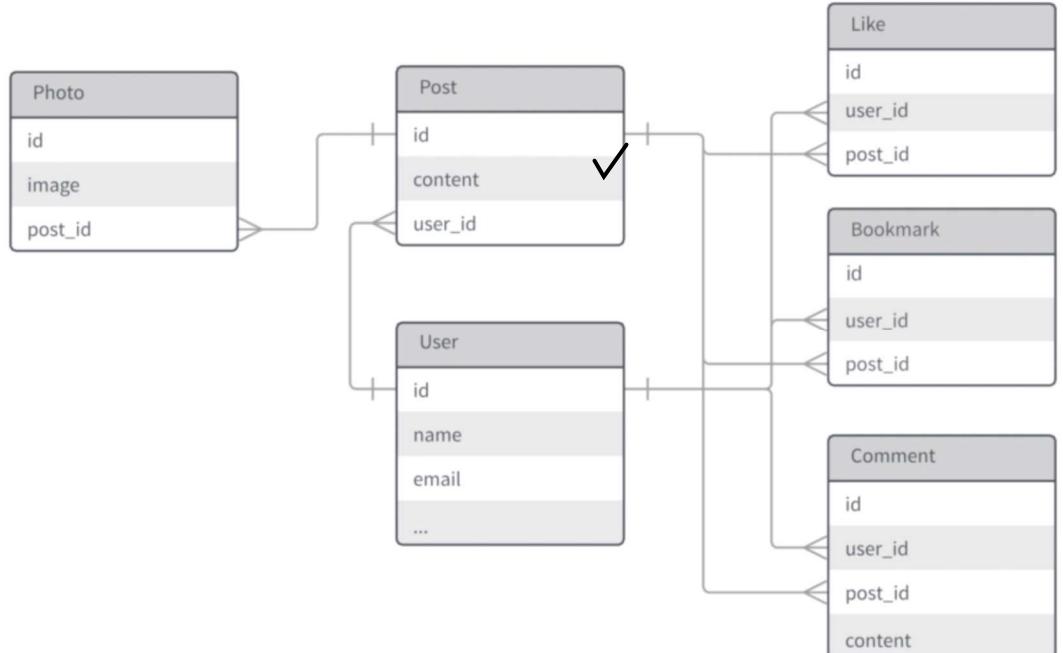
```
class Post < ApplicationRecord
  belongs_to :user
  has_many :photos, dependent: :destroy
  def belongs_to? user
    Post.find_by(user_id: user.id, id: id)
  end
end
```

Erklären Sie die Zeile: `Post.find_by(user_id: user.id, id: id)`:

Man sucht einen Post, bei welchen die `user_id` der id des param User entspricht und die Post id der Id des Posts.

Model: Posts mit Like ergänzen

Ein Post kann mehrere Likes von unterschiedlichen Usern haben. Ein Like gehört zu einem Post und wurde von einem User erfasst:



Ergänzen wir die Routen in `config/routes.rb` für die Likes indem `create` und `destroy` zugelassen werden:

```
routes.rb
1 Rails.application.routes.draw do
2   root 'posts#index'
3   devise_for :users, controllers: { registrations: 'registrations' }
4   # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
5   resources :users, only: [:index, :show]
6
7   resources :posts, only: [:index, :show, :create, :destroy] do
8     resources :photos, only: [:create]
9     resources :likes, only: [:create, :destroy]
10    end
11  end
```

Als nächstes müssen wir das Like-Model mit den entsprechenden Referenzen auf `User` und `Post` erstellen. Wie lautet die korrekte Generierung?

```
rails g model Like user:belongs_to post:belongs_to
```

Die Migration `db/migrate>NNNNN_create_likes.rb` sollte so aussehen:

```
routes.rb          20181217212228_create_likes.rb
1 class CreateLikes < ActiveRecord::Migration[5.2]
2   def change
3     create_table :likes do |t|
4       t.references :post, foreign_key: true
5       t.references :user, foreign_key: true
6
7       t.timestamps
8     end
9   end
10  end
11
```

Schliessen Sie die Migration mit `rails db:migrate` ab.

Um sicherzustellen, dass ein User einen Post nur genau einmal «liken» kann, müssen wir eine entsprechende Validierung auf dem Model `app/models/like.rb` definieren:

```
routes.rb          like.rb
1 class Like < ApplicationRecord
2   belongs_to :post
3   belongs_to :user
4
5   validates :user_id, uniqueness: { scope: :post_id }
6 end
7 |
```

In der Klasse des Models app/models/post.rb geben wir an, dass ein Post viele Likes haben kann und sortieren diese nach Aktualität. Zudem hinterlegen wir eine Methode, die prüft, ob ein Benutzer einen Post bereits mit einem Like gekennzeichnet hat:

```
routes.rb          like.rb          post.rb
1 class Post < ApplicationRecord
2   belongs_to :user
3   has_many :photos, dependent: :destroy
4   has_many :likes, -> {order(:created_at => :desc)}
5
6   def belongs_to? user
7     Post.find_by(user_id: user.id, id: id)
8   end
9
10  def is_liked user
11    Like.find_by(user_id: user.id, post_id: id)
12  end
13 end
14
```

In der Klasse des Models app/models/user.rb geben wir an, dass ein User viele Likes erfassen kann:

```
routes.rb          like.rb          post.rb
1 class User < ApplicationRecord
2   has_many :posts, dependent: :destroy
3   has_many :likes|
4   # Include default devise modules. Others available are:
5   # :confirmable, :lockable, :timeoutable, :trackable and :omniauthable
6   devise :database_authenticatable, :registerable,
7         :recoverable, :rememberable, :validatable
8
9   validates :name, presence: true, length: { maximum: 50 }
10 end
11
```



Rails verwendet eine JS-Bibliothek namens Turbolinks, die das Navigieren innerhalb einer App schneller machen soll. Leider funktioniert diese JS-Bibliothek nicht mit unserer Like-Funktionalität und muss aus assets/javascripts/application.js entfernt werden. Löschen Sie die Zeile // = require turbolinks.

Likes-Controller

Erstellen Sie den neuen Controller Likes-Controller und erfassen Sie folgenden Code:

```
1  class LikesController < ApplicationController
2    before_action :authenticate_user!
3
4    def create
5      @like = current_user.likes.build(like_params)
6      @post = @like.post
7      if @like.save
8        respond_to :js
9      else
10        flash[:alert] = "Something went wrong ..."
11      end
12    end
13
14    def destroy
15      @like = Like.find(params[:id])
16      @post = @like.post
17      if @like.destroy
18        respond_to :js
19      else
20        flash[:alert] = "Something went wrong ..."
21      end
22    end
23
24    private
25    def like_params
26      params.permit :post_id
27    end
28  end
29
```

Bedeutung:

- Zeile 2: Der Controller bzw. die Methoden dürfen nur von authentifizierten Benutzern ausgeführt werden.
- Zeile 4: Create-Methode für Likes, wird ausgeführt beim Klick auf das entsprechende Icon.
- Zeile 5: Der Benutzer macht einen Like und speichert diesen in eine Instanzvariable.
- Zeile 6: Wir speichern den Like.
- Zeilen 7-9: Bei Speicherung geben wir an ein asynchrones Javascript kurz AJAX weiter, um nicht die ganze Seite neu laden zu müssen.
- Zeile 14: Delete Methode für like
- Zeile 15: Speichere zu löschen like in Instanzvariable

In app/controllers/posts_controller.rb müssen wir die Methoden index und show ergänzen:

```
1 class PostsController < ApplicationController
2   before_action :authenticate_user!
3   before_action :find_post, only: [:show, :destroy]
4
5   def index
6     @posts = Post.all.limit(10).includes(:photos, :user, :likes).order('created_at desc')
7     @post = Post.new
8   end
9
10  def create
11    @post = current_user.posts.build(post_params)
12    if @post.save
13      if params[:images]
14        params[:images].each do |img|
15          @post.photos.create(image: img[1])
16        end
17      end
18
19      redirect_to posts_path
20      flash[:notice] = "Saved ..."
21    else
22      flash[:alert] = "Something went wrong ..."
23      redirect_to posts_path
24    end
25  end
26
27  def show
28    @photos = @post.photos
29    @likes = @post.likes.includes(:user)
30    @is_liked = @post.is_liked(current_user)
31  end
32
33  def destroy
34    if @post.user == current_user
35      if @post.destroy
36        flash[:notice] = "Post deleted!"
37      else
38        flash[:alert] = "Something went wrong ..."
39      end
40    else
41      flash[:notice] = "You don't have permission to delete this post!"
42    end
43    redirect_to root_path
44  end
```

Als nächstes passen wir die Datei app/config/routes.rb an:

```
routes.rb           likes_controller.rb
1 Rails.application.routes.draw do
2   root 'posts#index'
3   devise_for :users, controllers: { registrations: 'registrations' }
4   # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
5   resources :users, only: [:index, :show]
6
7   resources :posts, only: [:index, :show, :create, :destroy] do
8     resources :photos, only: [:create]
9     resources :likes, only: [:create, :destroy], shallow: true
10  end
11 end
```

Notieren Sie, was mit dem Schlüsselwort shallow: true genau gemeint ist (Infos unter <https://guides.rubyonrails.org/routing.html>):

bietet keine Indexressource. Man wird nicht auf /likes zugreifen können

Post Like

Damit wir einen Post liken können, muss die View posts#index (Datei app/views/_post_list.html.erb) angepasst werden:

The image shows a screenshot of a social media post. At the top left is a circular profile picture of a woman with dark hair. To its right, the name "Nadiia Maurer" is written in a dark font. Below the profile picture is a photograph of a man with dark hair and a light beard, wearing a white long-sleeved shirt, sitting at a desk and looking down at a computer keyboard. In the background, there are other people and computer monitors in what appears to be a shared workspace or office environment. On the far left of the post, there are three small icons: a red heart, a white speech bubble with a question mark, and a white ribbon-like bookmark icon. Below these icons, the text "Ralph Maurer like this" is followed by "Nadiia Maurer" and "28 DAYS AGO". At the bottom of the post, there is a placeholder text "Add a comment ...".

Hierzu verändern wir die Sektion, wo der Like-Link steht, wie folgt:

```
routes.rb          _posts_list.html.erb      likes_controller.rb
...
35               <span class="carousel-control-prev-icon" aria-hidden="true"></span>
36               <span class="sr-only">Previous</span>
37             </a>
38             <a class="carousel-control-next" href="#carousel-post-<%= post.id %>" role="button" data-slide="next">
39               <span class="carousel-control-next-icon" aria-hidden="true"></span>
40               <span class="sr-only">Next</span>
41             </a>
42           </div>
43         <% end %>
44
45       <div class="card-body">
46         <div class="row actions">
47           <% if post.is_liked(current_user).present? %>
48             <%= link_to "Love", like_path(post.is_liked(current_user)), method: :delete, remote: true,
49             class: "core-sprite loved hide-text" %>
50           <% else %>
51             <%= link_to "Loved", post_likes_path(post), method: :post, remote: true,
52             class: "core-sprite love hide-text" %>
53           <% end %>
54           <a href="#" class="core-sprite comment hide-text"> Comment </a>
55           <a href="#" class="core-sprite bookmark hide-text ml-auto"> Bookmark </a>
56         </div>
57       </div>
58       <strong>
59         <% post.likes.each.with_index do |like, index| %>
60           <% if post.likes.size == 1 %>
61             <%= like.user.name %> <strong> like this
62           <% elsif like == post.likes.last %>
63             </strong> and <strong>
64             <%= + like.user.name %> </strong> like this
65           <% elsif index > 1 %>
66             </strong> <%= "and " + (post.likes.size-index).to_s + " others " %> like this
67           <% break %>
68           <% elsif index==likes.size-2 || index==1%>
69             <%= like.user.name %>
70           <% else %>
71             <%= like.user.name + ", " %>
72           <% end %>
73         <% end %>
74       </strong>
75
76     </div>
77     <div>
78       <span><strong><%= post.user.name %></strong></span>
79       <span><%= post.content %></span>
80     </div>
81     <%= link_to time_ago_in_words(post.created_at).upcase + " AGO", post_path(post), class: "light-color post-time"
82   <hr>
83   <div class="row actions">
84     <form action="#" class="w-100">
85       <div>
86         <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." rows="1"></textarea>
87       </div>
88     </form>
89   </div>
90 </div>
91 <%end%>
```

Die Funktionsweise ist relativ einfach:

- Es wird geprüft, ob der Post bereits vom aktuellen Benutzer geliked wurde. Je nachdem wird das rote oder weisse Like-Herz angezeigt.
- Danach prüfen wir, wie viele Male der Post bereits geliked wurde und geben den letzten Benutzer, der einen Like gemacht hat, mit Namen aus.

Testen Sie den Like-Button. Nach dem Liken müssen Sie manuell einen Refresh im Browser durchführen.

Das Codechaos soll nun strukturiert werden. Erstellen Sie die neue Datei:
app/views/posts/_like_icon.html.erb.

Schneiden Sie den folgenden Ausschnitt der Datei app/views/_post_list.html.erb aus:

```
routes.rb          _posts_list.html.erb  ●  like_icon.html.erb      application.scss      likes_controller.rb
29      <% end %>
30      <%= image_tag photo.image.url(:standard), class: "card_img_top" %>
31      </div>
32      <% end %>
33      </div>
34      <a class="carousel-control-prev" href="#carousel-post-<%= post.id %>" role="button" data-slide="prev">
35          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
36          <span class="sr-only">Previous</span>
37      </a>
38      <a class="carousel-control-next" href="#carousel-post-<%= post.id %>" role="button" data-slide="next">
39          <span class="carousel-control-next-icon" aria-hidden="true"></span>
40          <span class="sr-only">Next</span>
41      </a>
42      </div>
43  <% end %>
44
45  <div class="card-body">
46      <div class="row actions">
47
48      <% if post.is_liked(current_user).present? %>
49          <%= link_to "Love", like_path(post.is_liked(current_user)), method: :delete, remote: true,
50          class: "core-sprite loved hide-text" %>
51      <% else %>
52          <%= link_to "Loved", post_likes_path(post), method: :post, remote: true,
53          class: "core-sprite love hide-text" %>
54      <% end %>
55
56      <a href="#" class="core-sprite comment hide-text"> Comment </a>
57      <a href="#" class="core-sprite bookmark hide-text ml-auto"> Bookmark </a>
58  </div>
59  <div>
60      <strong>
61          <% post.likes.each.with_index do |like, index| %>
62              <% if post.likes.size == 1 %>
63                  <%= like.user.name %> </strong> like this
64              <% elsif like == post.likes.last %>
65                  </strong>and<strong>
66                  <%= + like.user.name %></strong> like this
67              <% elsif index > 1 %>
68                  </strong><=%= "and " + (post.likes.size-index).to_s + " others " %> like this
69                  <% break %>
70              <% elsif index==likes.size-2 || index==1%>
71                  <%= like.user.name %>
72              <% else %>
73                  <%= like.user.name + ", " %>
74              <% end %>
75          <% end %>
76      </strong>
77
78  </div>
79  <div>
80      <span><strong><%= post.user.name %></strong></span>
81      <span><%= post.content %></span>
82  </div>
83      <%= link_to time_ago_in_words(post.created_at).upcase + " AGO", post_path(post), class: "light-color" %>
84      <hr>
85  <div class="row actions">
86      <form action="#" class="w-100">
```

Fügen Sie den Code in die Datei app/views/posts/_like_icon.html.erb ein und ergänzen Sie _post_list.html.erb mit dem Rendering der neuen Partial-View:

```
<div id="like-icon-post-<%= post.id.to_s %>">
    <%= render "like_icon", {is_liked: post.is_liked(current_user), post: post} %>
</div>
```

Der Code in der Datei app/views/posts/_like_icon.html.erb muss noch angepasst werden, da wir uns im Like-Bereich der Instagram-App befinden. Am Ende sollte die Datei wie folgt aussehen:

```
1  <% if is_liked.present? %>
2    <%= link_to "Love", like_path(is_liked), method: :delete, remote: true,
3      class: "core-sprite loved hide-text" %>
4  <% else %>
5    <%= link_to "Loved", post_likes_path(post), method: :post, remote: true,
6      class: "core-sprite love hide-text" %>
7  <% end %>
```

Auch den Like-Text wollen wir in eine Partial-View auslagern. Erstellen Sie die Datei: app/views/posts/_like_text.html.erb. Schneiden Sie den folgenden Bereich aus der Datei app/views/posts/_post_list.html.erb aus und fügen Sie diesen in die Datei _like_text.html.erb ein:

```
</div>
<a class="carousel-control-prev" href="#carousel-post-<%= post.id %>" role="button" data-sli
  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
  <span class="sr-only">Previous</span>
</a>
<a class="carousel-control-next" href="#carousel-post-<%= post.id %>" role="button" data-sli
  <span class="carousel-control-next-icon" aria-hidden="true"></span>
  <span class="sr-only">Next</span>
</a>
</div>
<% end %>

<div class="card-body">
  <div class="row actions">

    <div id="like-icon-post-<%= post.id.to_s %>">
      <%= render "like_icon", {is_liked: post.is_liked(current_user), post: post} %>
    </div>

    <a href="#" class="core-sprite comment hide-text"> Comment </a>
    <a href="#" class="core-sprite bookmark hide-text ml-auto"> Bookmark </a>
  </div>
<div>
  <strong>
    <% post.likes.each.with_index do |like, index| %>
      <% if post.likes.size == 1 %>
        <%= like.user.name %> </strong> like this
      <% elsif like == post.likes.last %>
        </strong>and<strong>
      <%= + like.user.name %></strong> like this
      <% elsif index > 1 %>
        </strong>&lt;= "and " + (post.likes.size-index).to_s + " others " %> like this
        <% break %>
      <% elsif index==likes.size-2 || index==1%>
        <%= like.user.name %>
      <% else %>
        <%= like.user.name + ", " %>
      <% end %>
    <% end %>
  </strong>
</div>
<div>
  <span><strong><%= post.user.name %></strong></span>
  <span><%= post.content %></span>
</div>
  <%= link_to time_ago_in_words(post.created_at).upcase + " AGO", post_path(post), class: "light-col
<hr>
<div class="row actions">
  <form action="#" class="w-100">
    <div>
      <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." rows="1"><
    </div>
  </form>
</div>
</div>
<%end%>
```

Fügen Sie den Code in die Datei app/views/posts/_like_text.html.erb ein und ergänzen Sie _post_list.html.erb mit dem Rendering der neuen Partial-View:

```
<div id="like-text-post-<%= post.id.to_s %>">
  <%= render "like_text", {likes: post.likes} %>
</div>
```

Auch in der Datei _like_text.html.erb nehmen wir ein paar Anpassungen vor. Achten Sie auf die neue Verschachtelung des Elementes **:**

```
1  <% likes.each.with_index do |like, index| %>
2    <strong>
3      <% if likes.size == 1 %>
4        <%= like.user.name %> </strong> like this
5      <% elsif like == likes.last %>
6        </strong>and<strong>
7        <%= + like.user.name %></strong> like this
8      <% elsif index > 1 %>
9        </strong><%= "and " + (likes.size-index).to_s + " others " %> like this
10       <% break %>
11     <% elsif index==likes.size-2 || index==1%>
12       <%= like.user.name %>
13     <% else %>
14       <%= like.user.name + ", " %>
15     <% end %>
16   <% end %>
17 </strong>
```

Damit wir nach einem Like nicht immer die Seite im Browser aktualisieren müssen, benötigen wir zwei AJAX Dateien: `create.js.erb` und `destroy.js.erb`.

Diese können Sie unter `sh-modules/iet-151/03_Arbeitsblaetter/instagram_files/js` beziehen. Kopieren Sie die beiden Dateien ins Verzeichnis `app/views/likes/`. Mit den AJAX Dateien wird nun beim Setzen eines Like die Seite neu geladen.

Das Like-Icon bzw. die Like-Funktion ist sowohl bei der Posts-Übersicht als auch beim Detail eines Posts identisch. Wir haben bereits eine Partial-View dafür erstellt und können nun diese in `app/views/posts/show.html.erb` wieder verwenden. Wir ersetzen den folgenden Link:

```
64      <hr class="mt-0">
65
66      <div class="row actions">
67        <a href="#" class="core-sprite love hide-text"> Love </a>
68        <a href="#" class="core-sprite comment hide-text"> Comment </a>
69        <a href="#" class="core-sprite bookmark hide-text ml-auto"> Bookmark </a>
70      </div>
71      <div><strong><%= pluralize(16, "like")%></strong></div>
72
73      <div class="light-color post-time"><%= time_ago_in_words(@post.created_at).upcase %>AGO</div>
74      <hr>
75      <div class="row actions">
76        <form action="#" class="w-100">
77          <div>
78            <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." rows="1" cols="30" type="text" value=""></div>
79          </form>
80        </div>
81      </div>
82
83      </div>
84    </div>
85  </div>
86 </div>
```

mit:

```
<div id="like-icon-post-<%= @post.id.to_s %>">
  <%= render "like_icon", {is_liked: @is_liked, post: @post} %>
</div>
```

Auch die Partial-View für den Like-Text soll in app/views/posts/show.html.erb wiederverwendet werden. Ersetzen Sie die folgende Zeile:

```
65          <div class="row actions">
66              <div id="like-icon-post-<%= post.id.to_s %>">
67                  <%= render "like_icon", {is_liked: @post.is_liked(current_user), post: @post} %>
68              </div>
69              <a href="#" class="core-sprite comment hide-text"> Comment </a>
70              <a href="#" class="core-sprite bookmark hide-text ml-auto"> Bookmark </a>
71          </div>
72          <div><strong><%= pluralize(16, "like")%></strong></div>
73
74      <div class="light-color post-time"><%= time_ago_in_words(@post.created_at).upcase %>AGO</div>
75      <hr>
76      <div class="row actions">
77          <form action="#" class="w-100">
78              <div>
79                  <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." ro
80              </div>
81          </form>
82      </div>
83
84      </div>
85  </div>
86</div>
87</div>
```

mit:

```
<div id="like-text-post-<%= @post.id.to_s %>">
    <%= render "like_text", {likes: @likes} %>
</div>
```

Nun erstellen wir eine weitere Partial-View für die Anzeige eines Posts. Schneiden Sie folgenden Code aus app/views/posts/show.html.erb aus:

```
show.html.erb      _posts_list.html.erb      _like_text.html.erb      _like_icon.html.erb      posts_controller.rb      likes_controller.rb
1  <div class="d-flex flex-column align-items-center mt-3">
2      <div class="row post box col-xl-10 col-lg-11 col-xs-12">
3          <div class="col-lg-8 col-md-7 px-0 d-flex post-show-img">
4              <div class="align-self-center">
5
6                  <% if @post.photos.size == 1%>
7                      <%= image_tag @post.photos.first.image.url(:standard), class: "card_img_top" %>
8                  <% else %>
9                      <div class="carousel slide" data-ride="carousel" id="carousel-post-<%= @post.id %>">
10                         <div class="carousel-inner">
11                             <% @post.photos.each do |photo| %>
12                                 <% if photo == @post.photos.first %>
13                                     <div class="carousel-item active">
14                                         <% else %>
15                                             <div class="carousel-item">
16                                                 <%= image_tag photo.image.url(:standard), class: "card_img_top" %>
17                                             </div>
18                                         <% end %>
19                                     </div>
20                                 <% end %>
21                             </div>
22                             <a class="carousel-control-prev" href="#carousel-post-<%= @post.id %>" role="button" dat
23                                 <span class="carousel-control-prev-icon" aria-hidden="true"></span>
24                                 <span class="sr-only">Previous</span>
25                             </a>
26                             <a class="carousel-control-next" href="#carousel-post-<%= @post.id %>" role="button" dat
27                                 <span class="carousel-control-next-icon" aria-hidden="true"></span>
28                                 <span class="sr-only">Next</span>
29                             </a>
30                         <% end %>
31                     </div>
32                 </div>
33                 <div class="col-lg-4 col-md-5 mt-sm-4 mt-md-0">
34                     <div class="row px-3 d-flex align-items-center">
35                         <%= link_to_user_path(@post.user), class: "no-text-decoration" do %>
36                             <%= image_tag avatar_url(@post.user), class: "post-author-icon" %>
37                         <% end %>
38                         <%= link_to_user_path(@post.user), class: "normal-color no-text-decoration d-flex align-self-cen
39                             <strong><%= @post.user.name %></strong>
40                         <% end %>
41
42                         <% if @post.belongs_to? current_user %>
43                             <%= link_to_post_path(@post), method: :delete, class: "ml-auto delete-post-icon mx-0 my-auto"
44                             <i class="far fa-trash-alt" aria-hidden="true"></i>
45                         <% end %>
46                     <% end %>
47                 </div>
48             </div>
49         </div>
50     </div>
51     <div class="col-lg-4 col-md-5 mt-sm-4 mt-md-0">
52         <div class="row px-3 d-flex align-items-center">
53             <%= link_to_user_path(@post.user), class: "no-text-decoration" do %>
54                 <%= image_tag avatar_url(@post.user), class: "post-author-icon" %>
55             <% end %>
56             <%= link_to_user_path(@post.user), class: "normal-color no-text-decoration d-flex align-self-cen
57                 <strong><%= @post.user.name %></strong>
58             <% end %>
59
60             <% if @post.belongs_to? current_user %>
61                 <%= link_to_post_path(@post), method: :delete, class: "ml-auto delete-post-icon mx-0 my-auto"
62                 <i class="far fa-trash-alt" aria-hidden="true"></i>
63             <% end %>
64         <% end %>
65     </div>
66 
```

Erstellen Sie die Partial `app/views/posts/_photos.html.erb` und fügen Sie den Code ein. Ergänzen Sie die erste und letzte Zeile und entfernen Sie bei der Instanzvariablen `post` das `@`-Zeichen an drei Stellen. Schlussendlich sollte die Datei wie folgt aussehen:

```
1  <%= link_to post_path(post) do %>
2    <% if post.photos.size == 1 %>
3      <%= image_tag post.photos.first.image.url(:standard), class: "card-img-top" %>
4    <% else %>
5      <div class="carousel slide" data-ride="carousel" id="carousel-post-<%= @post.id %>">
6        <div class="carousel-inner">
7          <% post.photos.each do |photo| %>
8            <% if photo == post.photos.first %>
9              <div class="carousel-item active">
10                <% else %>
11                  <div class="carousel-item">
12                    <% end %>
13                    <%= image_tag photo.image.url(:standard), class: "card-img-top" %>
14                  </div>
15                <% end %>
16              </div>
17              <a class="carousel-control-prev" href="#carousel-post-<%= @post.id %>" role="button" data-slide="prev">
18                <span class="carousel-control-prev-icon" aria-hidden="true"></span>
19                <span class="sr-only">Previous</span>
20              </a>
21              <a class="carousel-control-next" href="#carousel-post-<%= @post.id %>" role="button" data-slide="next">
22                <span class="carousel-control-next-icon" aria-hidden="true"></span>
23                <span class="sr-only">Next</span>
24              </a>
25            </div>
26          <% end %>
27        <% end %>
```

Bauen wir das Rendering in `app/views/posts/show.html.erb` ein:

```
<div class="align-self-center">
  <%= render "photos", {post: @post} %>
</div>
```

Wir sollten generell auf Instanzvariablen in Partial-Views verzichten. Mit der Methode `render` übergeben wir im zweiten Parameter die Instanzvariable `@post` der Variablen `post`. `post` kann in der Partial-View als lokale Variable verwendet werden.

Als nächstes passen wir die Menü-Punkte in `app/views/shared/_navbar.html.erb` an:

```
<nav class="navbar navbar-expand-lg navbar-light">
  <div class="container">
    <%= link_to "Icon", root_path, class: "navbar-brand core-sprite hide-text" %>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <%= form_tag users_path, method: :get, class: "form-inline my-2 my-lg-0 ml-md-auto", id: "users-search",
        remote: true do %>
        <%= text_field_tag :term, params[:term], class: "form-control mr-sm-2 text-center common-btn",
          placeholder: "Search", autocomplete: "off" %>
        <div id="users-result"></div>
      <% end %>
      <ul class="navbar-nav ml-md-auto">
        <li class="nav-item">
          <a class="nav-link core-sprite explore-icon hide-text" href="#">Explore</a>
        </li>
        <li class="nav-item">
          <a class="nav-link core-sprite notification-icon hide-text" href="#">Notification</a>
        </li>
        <li class="nav-item">
          <%= link_to "Profile", user_path(current_user), class: "nav-link core-sprite profile-icon hide-text" %>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Als letztes können wir aus app/views/posts/_posts_list.html.erb folgenden Code löschen:

```

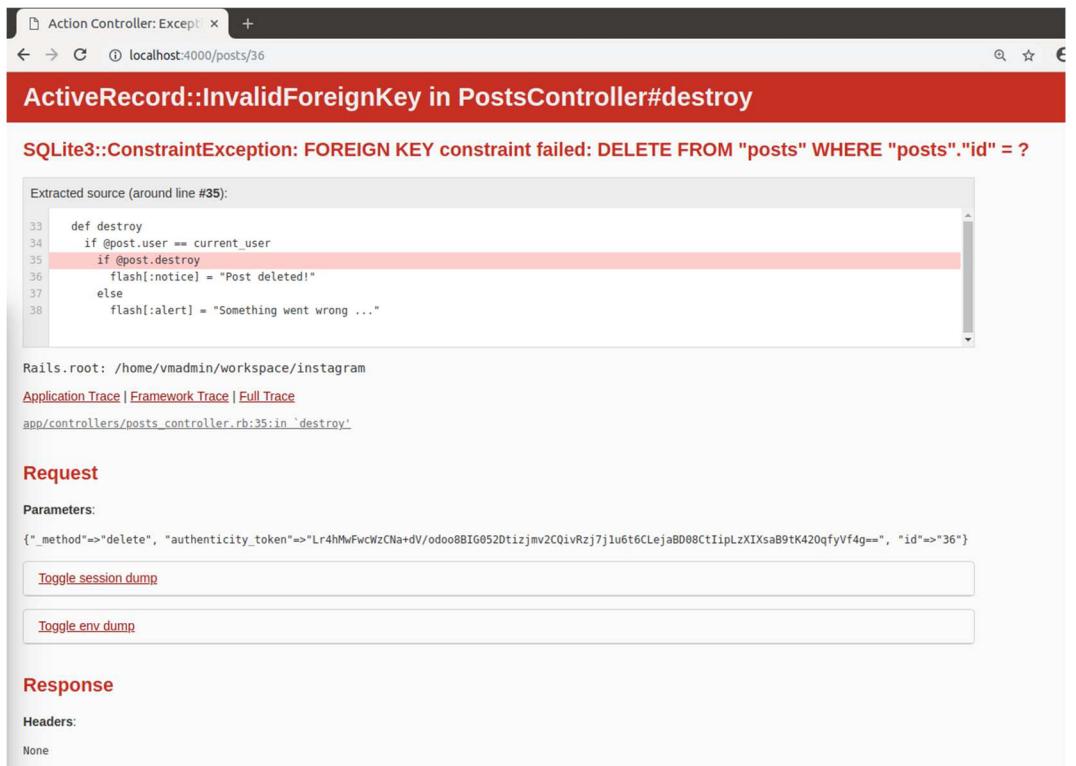
8         title: post.user.name do %>
9             <strong><%= post.user.name %></strong>
10            <% end %>
11
12            <% if post.belongs_to? current_user %>
13                <%= link_to post_path(post), method: :delete, class: "ml-auto delete-post-icon mx-0 my-auto" do %>
14                    <i class="far fa-trash-alt" aria-hidden="true"></i>
15                <% end %>
16            <% end %>
17        </div>
18
19        <% if post.photos.size == 1%>
20            <%= image_tag post.photos.first.image.url(:standard), class: "card_img_top" %>
21        <% else %>
22            <div class="carousel slide" data-ride="carousel" id="carousel-post-<%= post.id %>">
23                <div class="carousel-inner">
24                    <% post.photos.each do |photo| %>
25                        <% if photo == post.photos.first %>
26                            <div class="carousel-item active">
27                        <% else %>
28                            <div class="carousel-item">
29                        <% end %>
30                            <%= image_tag photo.image.url(:standard), class: "card_img_top" %>
31                        </div>
32                    <% end %>
33                </div>
34                <a class="carousel-control-prev" href="#carousel-post-<%= post.id %>" role="button" data-slide="prev">
35                    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
36                    <span class="sr-only">Previous</span>
37                </a>
38                <a class="carousel-control-next" href="#carousel-post-<%= post.id %>" role="button" data-slide="next">
39                    <span class="carousel-control-next-icon" aria-hidden="true"></span>
40                    <span class="sr-only">Next</span>
41                </a>
42            </div>
43        <% end %>
44
45        <div class="card-body">
46            <div class="row actions">
47
48                <div id="like-icon-post-<%= post.id.to_s %>">
49                    <%= render "like_icon", {is_liked: post.is_liked(current_user), post: post} %>
50                </div>
51
52                <a href="#" class="core-sprite comment hide-text"> Comment </a>
53                <a href="#" class="core-sprite bookmark hide-text ml-auto" data-aria="Bookmark" > Bookmark </a>
54            </div>
55        </div>
56
57        <div id="like-text-post-<%= post.id.to_s %>">
58            <%= render "like_text", {likes: post.likes} %>
59        </div>
60
61    </div>
62    <div>
63        <span><strong><%= post.user.name %></strong></span>
64        <span><%= post.content %></span>
65    </div>

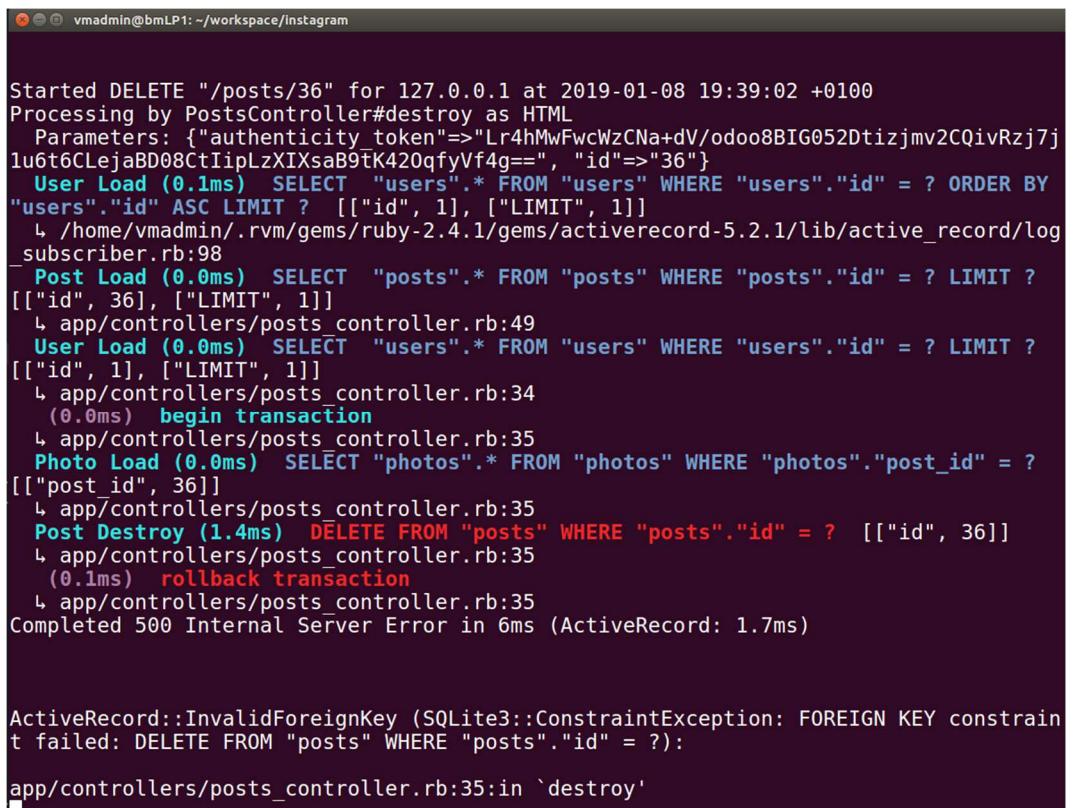
```

und ersetzen mit:

```
<%= render "photos", {post: post} %>
```

Sobald Sie nun einen bereits gelikten Post löschen, erhalten Sie einen DB-Fehler aus dem ActiveRecord:



```
Extracted source (around line #35):  
33     def destroy  
34       if @post.user == current_user  
35         if @post.destroy  
36           flash[:notice] = "Post deleted!"  
37         else  
38           flash[:alert] = "Something went wrong..."  
  
Rails.root: /home/vmadmin/workspace/instagram  
Application Trace | Framework Trace | Full Trace  
app/controllers/posts_controller.rb:35:in `destroy'  
  
Request  
Parameters:  
{  
  "_method"=>"delete",  
  "authenticity_token"=>"Lr4hMwFwcWzCNa+dV/odoo8BIG052Dtizjm...v2CQivRzj7j1u6t6CLEjaBD08CtIipLzXIXsaB9tK420qfyVf4g==",  
  "id"=>"36"  
}  
  
Toggle session dump  
  
Toggle env dump  
  
Response  
Headers:  
None  
  


```
Started DELETE "/posts/36" for 127.0.0.1 at 2019-01-08 19:39:02 +0100
Processing by PostsController#destroy as HTML
 Parameters: {"authenticity_token"=>"Lr4hMwFwcWzCNa+dV/odoo8BIG052Dtizjm...v2CQivRzj7j1u6t6CLEjaBD08CtIipLzXIXsaB9tK420qfyVf4g==", "id"=>"36"}
 User Load (0.1ms) SELECT "users".* FROM "users" WHERE "users"."id" = ? ORDER BY "users"."id" ASC LIMIT ? [[{"id": 1}, {"LIMIT": 1}]]
 ↳ /home/vmadmin/.rvm/gems/ruby-2.4.1/gems/activerecord-5.2.1/lib/active_record/log_subscriber.rb:98
 Post Load (0.0ms) SELECT "posts".* FROM "posts" WHERE "posts"."id" = ? LIMIT ? [[{"id": 36}, {"LIMIT": 1}]]
 ↳ app/controllers/posts_controller.rb:49
 User Load (0.0ms) SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ? [[{"id": 1}, {"LIMIT": 1}]]
 ↳ app/controllers/posts_controller.rb:34
 (0.0ms) begin transaction
 ↳ app/controllers/posts_controller.rb:35
 Photo Load (0.0ms) SELECT "photos".* FROM "photos" WHERE "photos"."post_id" = ? [[{"post_id": 36}]]
 ↳ app/controllers/posts_controller.rb:35
 Post Destroy (1.4ms) DELETE FROM "posts" WHERE "posts"."id" = ? [[{"id": 36}]]
 ↳ app/controllers/posts_controller.rb:35
 (0.1ms) rollback transaction
 ↳ app/controllers/posts_controller.rb:35
Completed 500 Internal Server Error in 6ms (ActiveRecord: 1.7ms)

ActiveRecord::InvalidForeignKey (SQLlite3::ConstraintException: FOREIGN KEY constraint failed: DELETE FROM "posts" WHERE "posts"."id" = ?):
app/controllers/posts_controller.rb:35:in `destroy'
```


```

Lösen Sie das Problem:

in post.rb:

has_many :likes, -> {order(:created_at => :desc)}, dependent: :destroy

Auftrag: Quicknotes AB151-07

Alle Aufträge des Typen Quicknote sind Bestandteil der Bewertung. Erstellen Sie Quicknotes mit 2-4 A4-Seiten Text. Dazu kommen Screenshots und Bilder, so dass schlussendlich mehr als 4 Seiten resultieren können.

Die Dokumentation muss zusammen mit der Rails-Applikation via Git abgegeben werden (z.B. GitLab oder GitHub). Das Format ist entweder *md* (Markdown) oder *pdf*. Eine Anleitung zu GitLab der Abteilung IET finden Sie auf dem Modulshare. Vergessen Sie nicht, das Projekt für die Lehrperson freizugeben!

Zusammenfassung und Anwendungszweck

Die Quicknotes sollen eine kurze, prägnante Zusammenfassung des Dokuments AB151-07 und einen Überblick über **alle** vorgestellten Techniken, Methoden und Konzepte beinhalten. Fassen Sie nicht die Arbeitsschritte des Auftrages zusammen, sondern beschreiben Sie nur kurz, was sie gemacht haben. Z.B. «Installation des Gem devise mit einem Eintrag im Gemfile und dem Ausführen des Befehls 'bundle install'».

Führen Sie ausführlicher auf, was die Techniken, Methoden und Konzepte bedeuten. Z.B. «Mit Gem devise wird eine Authentifizierung in Rails erstellt. Dazu gehören das Model (Tabelle 'users') und die Seiten/Views Sign-up (users/sign_up) und Sign-in (users/sign_in). Nach erfolgreichem Sign-in wird eine Session erstellt...usw...».

Achten Sie darauf, dass alle wesentliche Themen inkl. Anwendungszweck beschrieben werden und wenn möglich in Zusammenhang gebracht sind.

Die Fragen zum Anwendungszweck werden direkt mit der Zusammenfassung beantwortet: Wie und wo können die vorgestellten Techniken, Methoden und Konzepte in einer Rails-Applikation (z.B. Instagram) angewendet werden?

Zu folgenden Themen/Kapiteln im AB151-07 sollen die Vor- und Nachteile aufgeführt werden:

- Anpassungen des Models (inkl. Erweiterung des Models mit *Like*).
- Partial-Views: Es handelt sich um eine Wiederholung des Themas, führen Sie nochmals die Vor- und Nachteile auf und wie der Einsatz Ihrer Meinung nach Sinn macht.
- Anpassungen des Codes, so wie wir es in diesem Arbeitsblatt durchgeführt haben.

Vergessen Sie nicht, auch die übrigen Themen/Techniken in der Zusammenfassung zu beschreiben!

Selbstreflexion

Folgende Fragen müssen bei der Selbstreflexion beantwortet werden:

- Was habe ich gelernt? Was wusste ich bereits?
- Wie bin ich vorgegangen beim Lernen bzw. Ausführen des Auftrages?
- Was waren die Schwierigkeiten, wie konnte ich diese lösen?
- Was habe ich nicht verstanden bzw. was konnte ich nicht lösen?
- Was kann ich nächstes Mal besser machen?

Es sollen alle wesentlichen Themen des Arbeitsblattes bei der Reflexion berücksichtigt werden. Wenn Sie etwas bereits kannten/wussten und Sie deshalb bei diesem Aspekt nichts lernten, dann muss das bei der Reflexion erwähnt werden! Mit anderen Worten: Reflektieren Sie über alle Themen und Arbeitsschritte und überlegen Sie sich dabei, was Sie bereits gekannt haben, was Sie gelernt haben und wie Sie dabei vorgegangen sind.

Formulieren Sie vollständige Sätze und seien Sie bei der Selbstreflexion ausführlich! Die Selbstreflexion sollte mind. eine halbe Seite lang sein.

Lösungen der Aufgaben

Das Dokument muss die Lösungen der Aufgaben beinhalten. Die Lösungen können - anstatt in einem separaten Kapitel - auch in der Zusammenfassung enthalten sein. In diesem Fall sollten sie speziell markiert werden, damit die Lehrperson diese schnell findet!

Dieses Arbeitsblatt hat nur 4 kleinere Aufgaben: S. 8, 9, 12 und 21.

Abschliessende Reflexion über das Gelernte:

Schreiben Sie eine abschliessende, zusammenfassende Reflexion:

- Eine Zusammenfassung über das Gelernte (das ist eine Wiederholung der Selbstreflexion in komprimierter Form).
- Der Lernfortschritt, den Sie mit diesem Arbeitsblatt erzielt haben, in Bezug auf das Rails-Framework allgemein und/oder in Bezug auf die Instagram-Applikation im Speziellen.

Instagram Applikation

Ziel der Arbeitsblätter ist in erster Linie das Entwickeln der Instagram Applikation, nicht das Erstellen der Dokumentation. In den Quicknotes sollen deshalb Screenshots den Projektfortschritt dokumentieren.

Folgende Screenshots müssen in dieser Dokumentation enthalten sein:

- Liste mit allen Posts, mind. 2 Posts sichtbar
- Detail-Ansicht eines Posts (Post mit mehreren Bildern)

Sie bauen die Screenshots am besten in die Zusammenfassung ein.

In diesem Arbeitsblatt gab es viel zu codieren und nur wenige Aufgaben. Das Endresultat ist umso wichtiger und fliesst in diesem Fall mit mehr Punkten in die Bewertung ein!

Abgabe der Quicknotes und der Applikation

Termin für die Abgabe: Gemäss Angaben der Lehrperson.

Checken Sie Ihr Projekt am Ende des Arbeitsblattes auf dem Git Server ein. Vergessen Sie nicht, dass auch die Dokumentation enthalten sein muss. Versehen Sie das Projekt anschliessend mit einem Release-Tag und pushen auch dieses auf den Server.

Eine Anleitung zu Git finden Sie auf dem Modulshare (Dokument "AB151_GitAnleitung.pdf").