



MODUL 151

TEIL 6: POST VIEW, CREATE FORM VIEW

Ralph Maurer

Inhaltsverzeichnis AB151-06

Modul 133: Instagram mit Rails bauen

Teil 6: Instagram Post view, creat form view

Inhaltsverzeichnis AB151-06.....	1
Modul 133: Instagram mit Rails bauen.....	1
Post view.....	2
DropzoneJS-rails.....	2
Icons von fontawesome.....	3
Post-View HTML.....	4
Post view SCSS	5
+ Button mit DropzoneJS gestalten	6
Posts ohne Photos unterbinden.....	7
Posts mit Photos Test	8
Bildspeichergrößen ändern.....	8
Design für Postliste	9
Partial-View für Posts list	16
Post View Button designen.....	17
Bildschärfe in Posts.....	19
Posts als Startseite.....	19
Auftrag: Quicknote AB151-06.....	20

Post view

DropzoneJS-rails

Zur Gestaltung des Uploads nutzen wir DropzoneJS. Ein Gem, dass uns eine super Anzeige für den Bilderupload ermöglicht. Alle Informationen zu DropzoneJS finden Sie unter <http://www.dropzonejs.com>



DropzoneJS is an open source library that provides drag'n'drop file uploads with image previews.

It's lightweight, doesn't depend on any other library (like jQuery) and is [highly customizable](#).



Erfassen Sie im gemfile den Eintrag:
`gem 'dropzonejs-rails'`

In application.js muss dropzone referenziert werden (Achtung: die Referenz muss nach `bootstrap-sprockets` und vor `tree .` formuliert werden!):

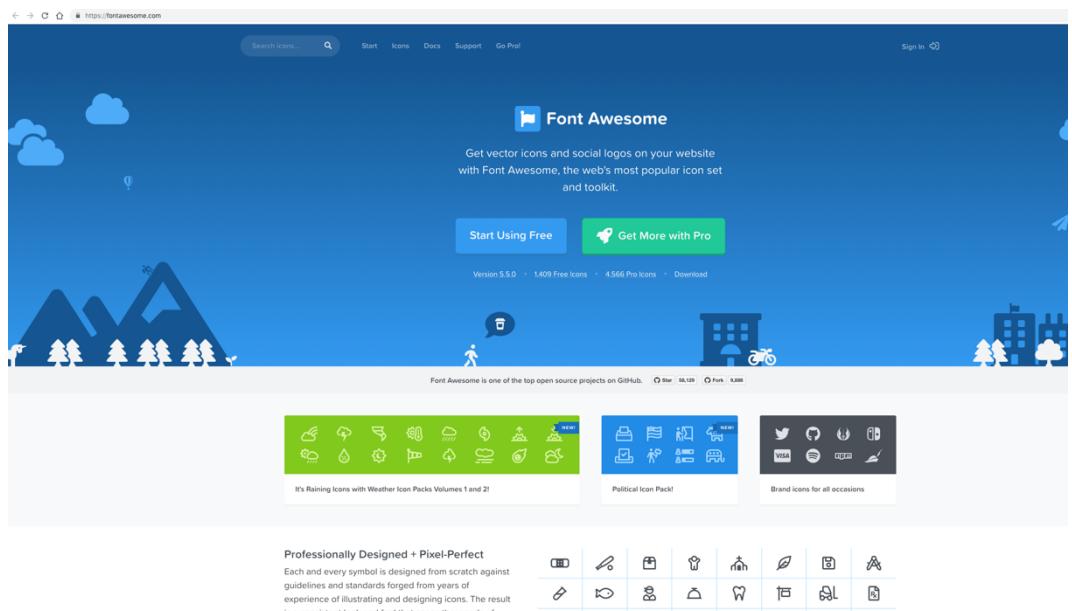
```
application.scss           application.js          upload_post_images.js      index.html.erb
1 // This is a manifest file that'll be compiled into application.js, which will include all the files
2 // listed below.
3 //
4 // Any JavaScript/Coffee file within this directory, lib/assets/javascripts, or any plugin's
5 // vendor/assets/javascripts directory can be referenced here using a relative path.
6 //
7 // It's not advisable to add code directly here, but if you do, it'll appear at the bottom of the
8 // compiled file. JavaScript code in this file should be added after the last require_* statement.
9 //
10 // Read Sprockets README (https://github.com/rails/sprockets#sprockets-directives) for details
11 // about supported directives.
12 //
13 //=- require rails-ujs
14 //=- require activestorage
15 //=- require turbolinks
16 //=- require jquery3
17 //=- require popper
18 //=- require bootstrap-sprockets
19 //=- require dropzone ←
20 //=- require tree .
21 //=- require toastr
22
```

Und im application.scss wird dropzone importiert:

```
@import "dropzone/dropzone";
```

Icons von fontawesome

Die Icons übernehmen wir von <http://fontawesome.com>



Gehen Sie auf «Start Using Free», kopieren Sie den Link:

This screenshot shows the "Start a New Project" section of the Font Awesome website. It features a large input field containing a CDN link for version 5.5.0. Below the input field, there are three numbered steps: 1. Set up on Your Site, 2. Add Icons to Your UI, and 3. Style Your Icons. Each step has a brief description and a "Read more" link.

Und fügen Sie diesen in views/layouts/application.html.erb ein:

```
application_controller.rb registrations_controller.rb Gemfile application.js application.html.erb application.scss routes.rb
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Instagram</title>
5     <%= csrf_meta_tags %>
6     <%= csp_meta_tag %>
7
8     <%= stylesheet_link_tag 'application', media: 'all', 'data-turbolinks-track': 'reload' %>
9     <%= javascript_include_tag 'application', 'data-turbolinks-track': 'reload' %>
10    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/all.css" integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzICo0wt...">
11  </head>
12  <body>
13    <%= render 'shared/navbar' if current_user %>
14    <div class="container">
15      <%= yield %>
16    </div>
17    <%= render 'shared/footer' %>
18    <% if flash.any? %>
19      <script type="text/javascript">
20        <% flash.each do |key, value| %>
21          <% type = key.to_s.gsub('alert', 'error').gsub('notice', 'success') %>
22          toastr.<%= type %>('<%= value %>')
23        <% end %>
24      </script>
25    <% end %>
26  </body>
27 </html>
```

Post-View HTML

Jetzt müssen wir die Post-View `app/views/posts/index.html.erb` anpassen. Dies verlangt ein genaues Arbeiten:

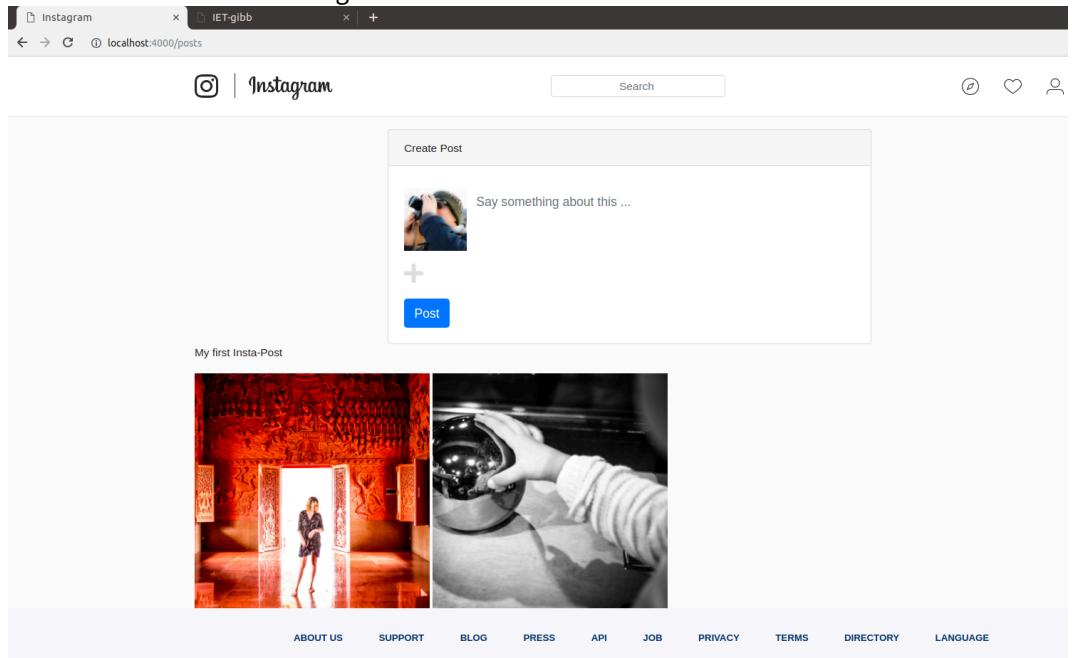
```
application_controller.rb | registrations_controller.rb | Gemfile | application.js | application.html.erb | index.html.erb | application.scss
1  <div class="d-flex flex-column align-items-center mt-3">
2    <div class="col-xl-7 col-lg-8 col-md-10 col-sm-11 post-card">
3      <div class="card">
4        <div class="card-header">
5          Create Post
6        </div>
7        <div class="card-body">
8          <%= form_for @post, :html => {:multipart => true, :class => "dropzone upload-images p-0 border-0"} do |f| %>
9            <div class="form-group row mt-2">
10              <div class="col-auto pr-0">
11                <%= image_tag avatar_url(current_user), class: "post-author-icon" %>
12              </div>
13              <div class="col pl-0">
14                <%= f.text_field :content, class: "form-control border-0", placeholder: "Say something about this ..." %>
15              </div>
16            </div>
17            <div class="fallback">
18              <%= file_field_tag "images[]", type: :file, multiple: true %>
19            </div>
20            <div class="dz-message m-0"></div>
21            <div class="dropzone-previews mb-3">
22              <div class="upload-photos-icon">
23                <i class="fa fa-plus fa-2x" aria-hidden="true" style="color:#dddfef"></i>
24              </div>
25            </div>
26            <div>
27              <%= f.submit "Post", class: "btn btn-primary" %>
28            <% end %>
29          </div>
30        </div>
31      </div>
32    </div>
33  | 
34  <%@posts.each do |post|%>
35    <p><%post.content%></p>
36    <%post.photos.each do |photo|%>
37      <%=image_tag photo.image.url(:standard)%>
38    <%end%>
39  <%end%>
40
```

Installieren Sie nun DropzoneJS mit

`bundle install`

und starten Sie den Server neu.

Ihre Ansicht sollte nun wie folgt aussehen:



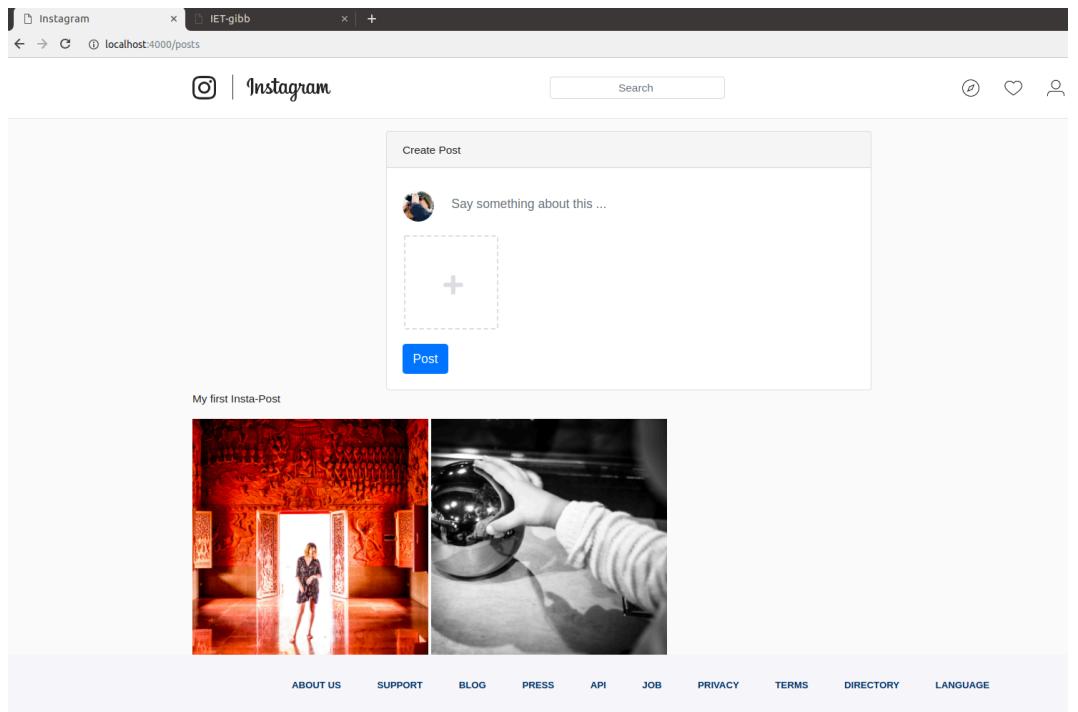
Post view SCSS

Den ganzen Abschnitt «Create post» müssen wir nun mit dem Instagram Layout ausbauen. Insbesondere der + Button soll besser werden. Hierzu schreiben wir reichlich SCSS-Code und nutzen dabei DropzoneJS.

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

```
application.scss index.html.erb
244     font-size: 16px;
245   }
246 }
247 }
248 // Posts
249 .post-author-icon {
250   height: 40px;
251   width: 40px;
252   border-radius: 50%;
253   margin-right: 10px;
254 }
255
256 // Customize upload photos form
257 .upload-photos-icon:hover {
258   cursor: pointer;
259 }
260
261 .dz-preview.dz-image-preview {
262   margin: 2px;
263 }
264
265 .dz-image {
266   border-radius: 0px !important;
267
268   img {
269     width: 100%;
270   }
271 }
272
273 .dz-remove {
274   @extend .core-sprite;
275   @extend .hide-text;
276   position: absolute;
277   top: 9px;
278   right: 5px;
279   z-index: 20;
280   width: 18px;
281   height: 18px;
282   background-position: -386px -25px;
283   opacity: 0;
284   -webkit-transition: opacity 0.2s liner;
285   transition: opacity 0.2s liner;
286 }
287
288 .dropzone .dz-preview:hover .dz-remove {
289   opacity: 1;
290 }
291
292 .upload-photos-icon {
293   height: 120px;
294   width: 120px;
295   border: 2px dashed #ddfe2;
296   border-radius: 2px;
297   padding: 48px;
298   display: inline-block;
299   margin: 2px;
300 }
301
```

Wenn wir das Ganze nun betrachten, stellen wir fest, dass der + Button noch gar nichts macht.



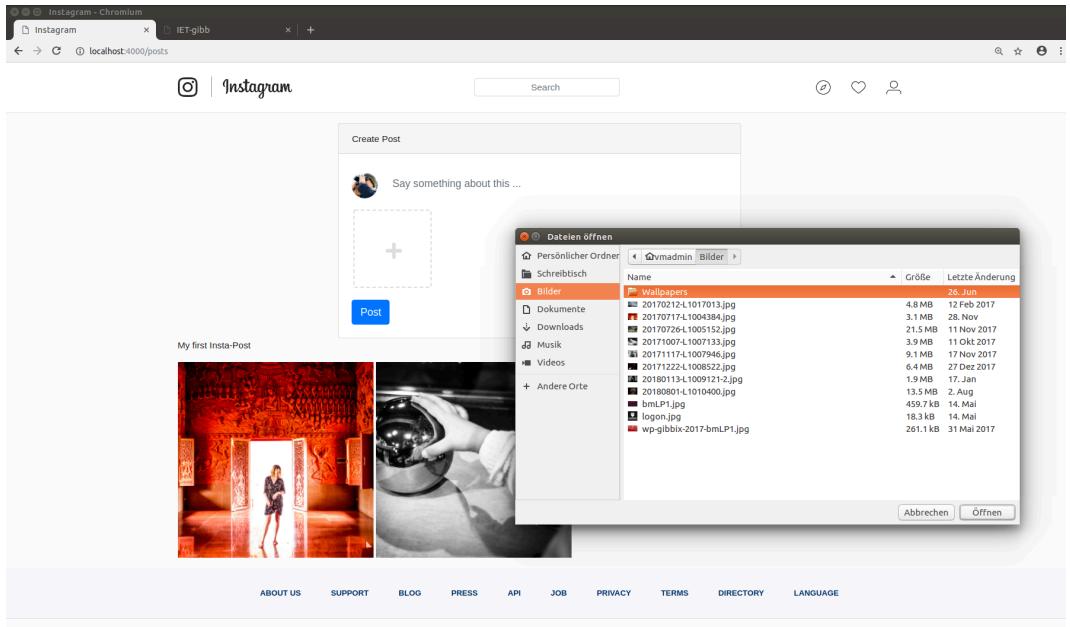
+ Button mit DropzoneJS gestalten

Der + Button soll nun seine Funktionalität erhalten mit DropzoneJS in Javascript. Erstellen Sie eine neue Datei app/assets/javascripts/upload_post_images.js und erfassen Sie folgenden Code:

```
application.scss | upload_post_images.js | index.html.erb
```

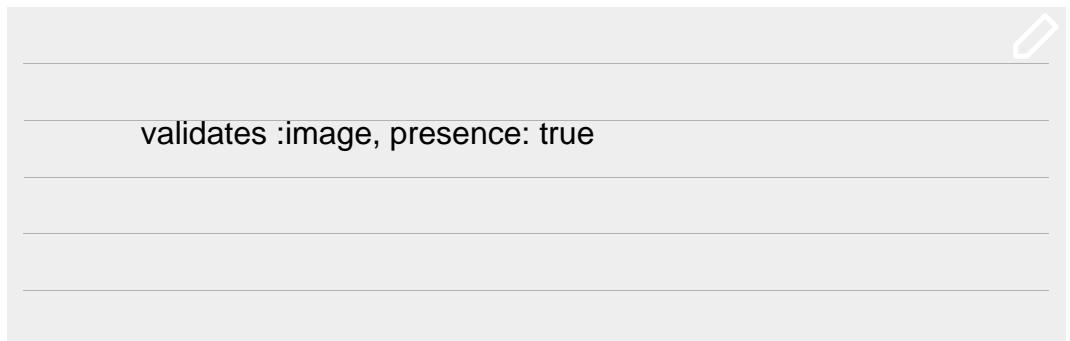
```
1 Dropzone.autoDiscover = false;
2
3 $(document).ready(function(){
4   $(".upload-images").dropzone({
5     addRemoveLinks: true,
6     maxFilesize: 1,
7     autoProcessQueue: false,
8     uploadMultiple: true,
9     parallelUploads: 100,
10    maxFiles: 100,
11    paramName: "images",
12    previewsContainer: ".dropzone-previews",
13    clickable: ".upload-photos-icon",
14    thumbnailWidth: 100,
15    thumbnailHeight: 100,
16
17    init: function(){
18      var myDropzone = this;
19
20      this.element.querySelector("input[type=submit]").addEventListener("click", function(e){
21        e.preventDefault();
22        e.stopPropagation();
23        myDropzone.processQueue();
24      });
25
26      this.on("successmultiple", function(files, response){
27        window.location.reload();
28      });
29
30      this.on("errormultiple", function(files, response){
31        toastr.error(response);
32      });
33    }
34  });
35});
```

Nun sollte der + Button funktionieren:



Posts ohne Photos unterbinden

Damit Benutzer nicht einfach Posts ohne Photos erstellen können, müssen wir eine Validierung auf der Datenbank einbauen; diese besagt, dass nur Posts mit Photos erstellt werden können. Formulieren Sie die korrekte Validierung im Model Photo.

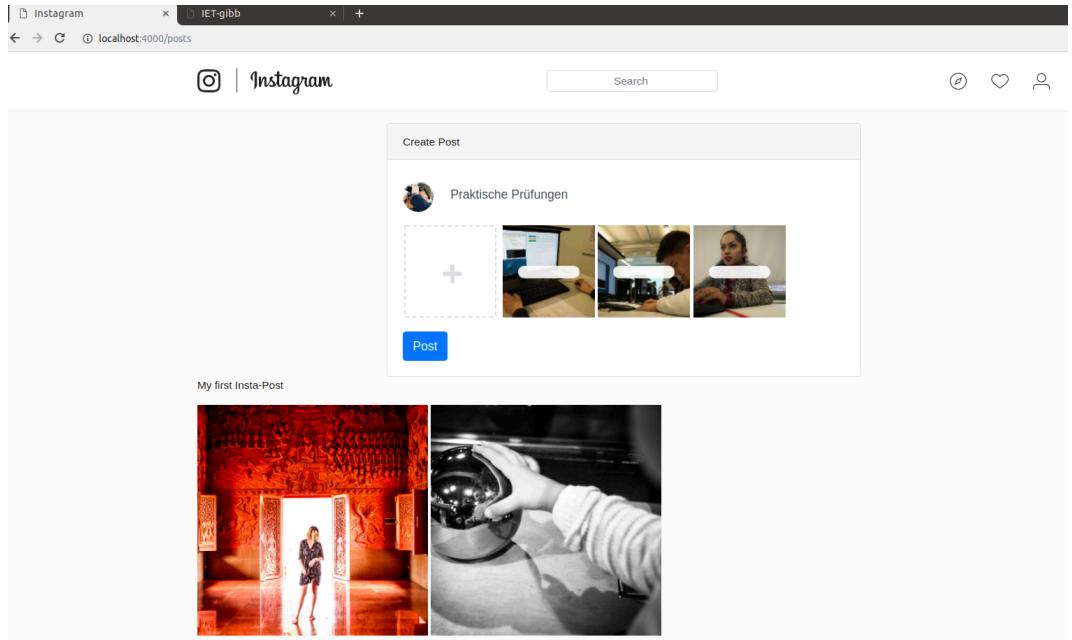


In app/controllers/posts_controller.rb müssen wir noch die Bildübergabe als Parameter erfassen:

```
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
```

Posts mit Photos Test

Erstellen Sie nun einen Post mit drei Bildern und testen Sie das Ganze:



Bildspeichergrößen ändern

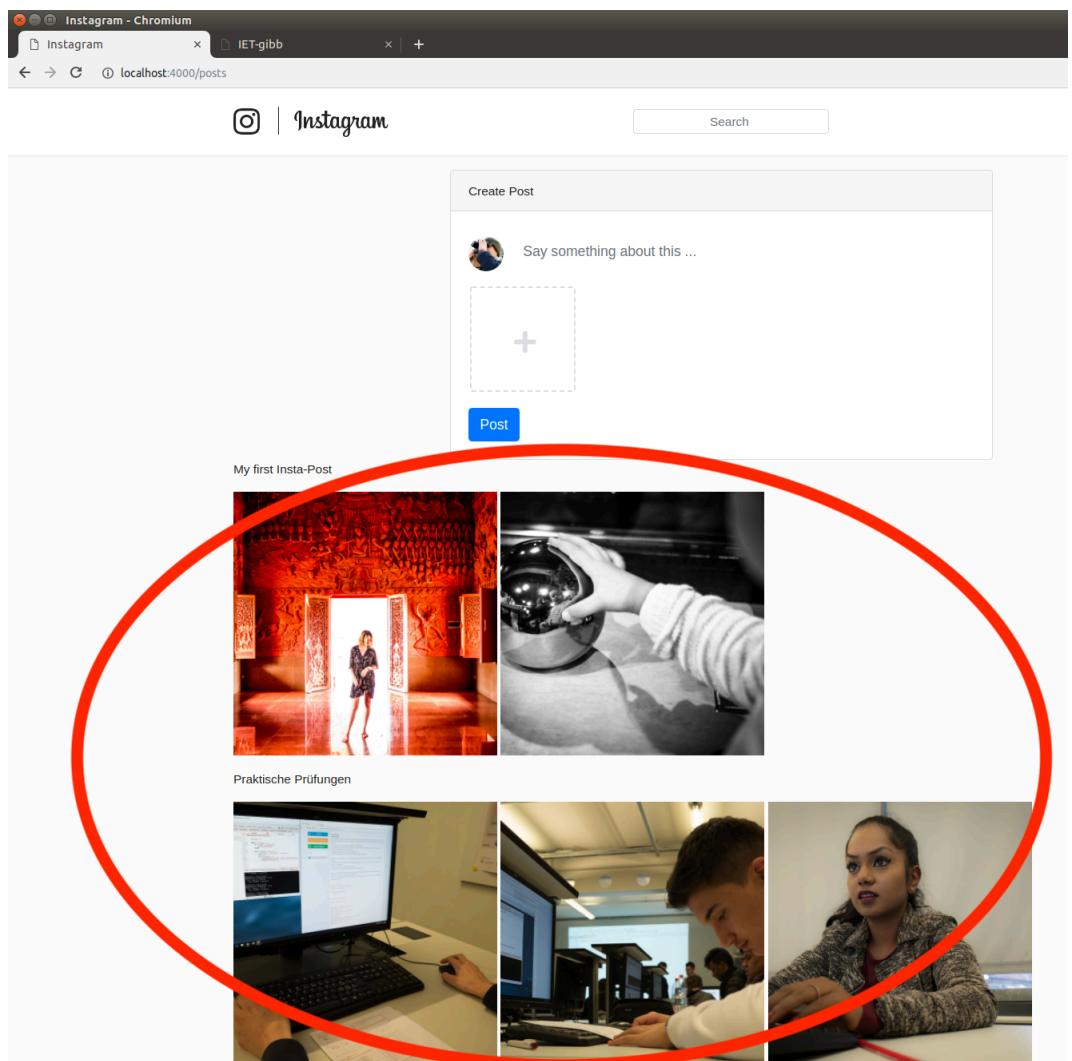
Sie können nur Bilder mit einer Grösse bis zu 1 MB hochladen. Finden Sie den entsprechenden Parameter und verändern Sie diesen, so dass 3 MB grosse Bilder angewandt werden können. Welche Datei und welcher Eintrag sind betroffen?



Design für Postliste

Als nächstes müssen wir die Postliste gestalten. Folgende Anforderungen sind zu erfüllen:

1. Username und Avatar sollen angezeigt werden.
2. Sortierung von neu nach alt (letzter Post soll als erster erscheinen).



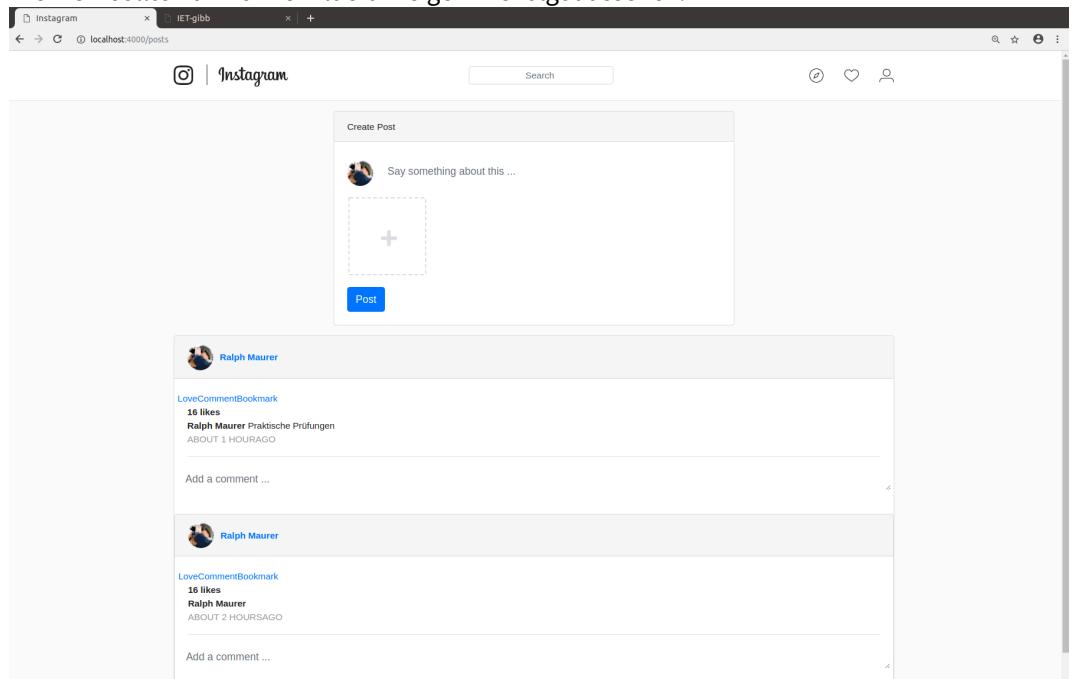
Diese Anforderungen können wir schnell im app/controllers/posts_controller.rb umsetzen:

```
5   def index
6     @posts = Post.all.limit(10).includes(:photos, :user).order('created_at desc')
7     @post = Post.new
8   end
9 |
```

Der Aufwand für die Darstellung in HTML (app/views/posts/index.html.erb) ist wesentlich grösser. Viel Spass beim schreiben:

```
index.html.erb
1 <div class="d-flex flex-column align-items-center mt-3">
2   <div class="col-xl-7 col-lg-8 col-md-10 col-sm-11 post-card">
3     <div class="card">
4       <div class="card-header"> Create Post </div>
5       <div class="card-body">
6         <%= form_for @post, :html => { :multipart => true, :class => "dropzone upload-images p-0 border-0" } do |f| %>
7           <div class="form-group row mt-2">
8             <div class="col-auto pr-0">
9               <%= image_tag avatar_url(current_user), class: "post-author-icon" %>
10            </div>
11            <div class="col pl-0">
12              <%= f.text_field :content, class: "form-control border-0",
13                placeholder: "Say something about this ..." %>
14            </div>
15          </div>
16          <div class="fallback"> <%= file_field_tag "images[]", type: :file, multiple: true %> </div>
17          <div class="dz-message m-0"></div>
18          <div class="dropzone-previews mb-3">
19            <div class="upload-photos-icon">
20              <i class="fa fa-plus fa-2x" aria-hidden="true" style="color:#ddafe2"></i>
21            </div>
22            <div>
23              <%= f.submit "Post", class: "btn btn-primary" %>
24            <% end %>
25          </div>
26        </div>
27      </div>
28    </div>
29  <% @posts.each do |post| %>
30    <div class="card mt-3 post">
31      <div class="card-header d-flex align-items-center">
32        <%= link_to user_path(post.user), class: "no-text-decoration" do %>
33          <%= image_tag avatar_url(post.user), class: "post-author-icon" %>
34        <% end %>
35        <%= link_to user_path(post.user), class: "normal-color no-text-decoration",
36          title: post.user.name do %>
37          <strong><%= post.user.name %></strong>
38        <% end %>
39      </div>
40      <div class="card-body">
41        <div class="row actions">
42          <a href="core-sprite love hide-text"> Love </a>
43          <a href="core-sprite comment hide-text"> Comment </a>
44          <a href="core-sprite bookmark hide-text"> Bookmark </a>
45        </div>
46        <div><strong><%= pluralize(16, "like")%></strong></div>
47        <div>
48          <span><strong><%= post.user.name %></strong></span>
49          <span><%= post.content %></span>
50        </div>
51        <div class="light-color post-time"><%= time_ago_in_words(post.created_at).upcase %>AGO</div>
52        <hr>
53        <div class="row actions">
54          <form action="#" class="w-100">
55            <div>
56              <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." rows="1"></textarea>
57            </div>
58          </form>
59        </div>
60      </div>
61    <%end%>
```

Die View sollte nun - ohne Bilderanzeige - wie folgt aussehen:



Für die Darstellung der Bilder nutzen wir wiedermal Bootstrap Carousel. Gehen Sie auf <http://www.bootstrap.com> und suchen Sie in der Dokumentation nach Carousel. Wir kopieren dann den Code für einen Slider mit Controls (with controls).

```
<div id="carouselExampleControls" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleControls" 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="#carouselExampleControls" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

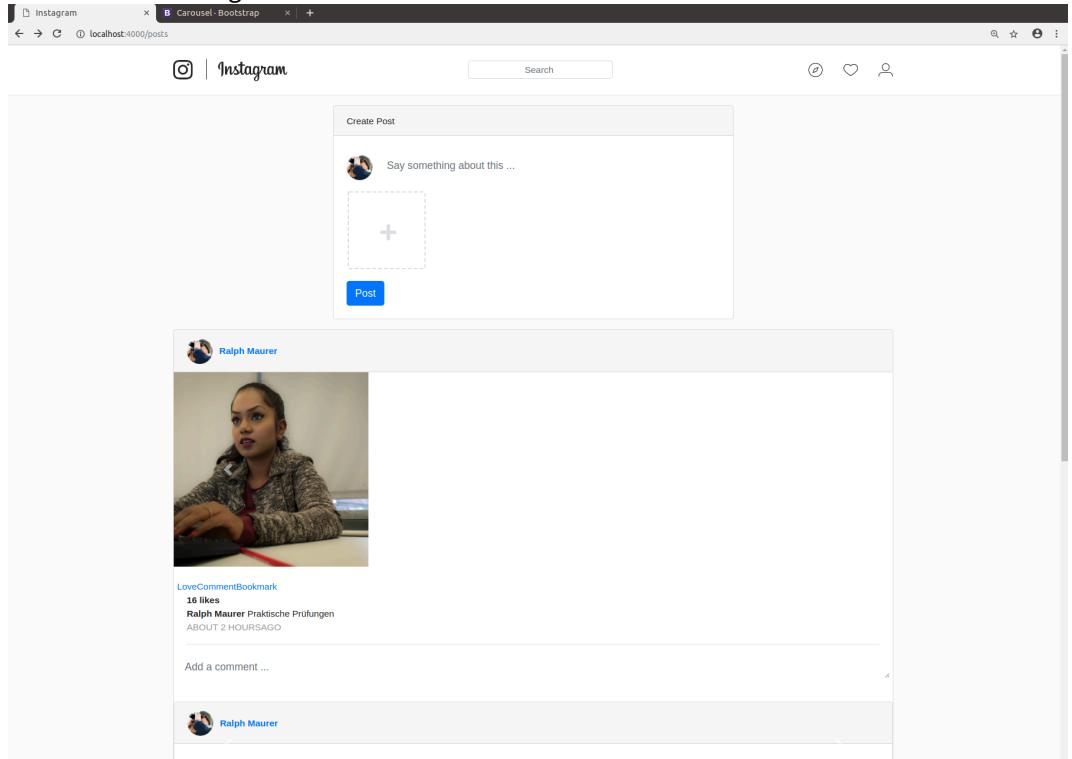
Kopieren Sie den Quelltext in der Datei app/views/posts/index.html.erb an die richtige Stelle:

```
index.html.erb
1<div class="card mt-3 post">
2  <div class="card-header d-flex align-items-center">
3    <a href="#"><img class="w-100" alt="User profile picture" /></a>
4    <div class="flex-grow-1 p-2">
5      <strong><%= post.user.name %></strong>
6      <p><%= post.content %></p>
7      <small><%= time_ago_in_words(post.created_at) %></small>
8    </div>
9  </div>
10 <div class="card-body">
11   <div class="row actions">
12     <a href="#">Love</a>
13     <a href="#">Comment</a>
14     <a href="#">Bookmark</a>
15   </div>
16   <div><strong><%= pluralize(16, "like") %></strong></div>
17   <div>
18     <span><strong><%= post.user.name %></strong></span>
19     <span><%= post.content %></span>
20   </div>
21   <div class="light-color post-time"><%= time_ago_in_words(post.created_at).upcase %>AGO</div>
22   <hr>
23   <div class="row actions">
24     <form action="#" class="w-100">
25       <div>
26         <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." rows="1"></textarea>
27       </div>
28     </form>
29   </div>
30 </div>
31 <% end %>
```

Wir können Teile davon übernehmen, müssen aber den Slider erheblich umbauen:

```
27      <% @posts.each do |post| %>
28          <div class="card mt-3 post">
29              <div class="card-header d-flex align-items-center">
30                  <%= link_to user_path(post.user), class: "no-text-decoration" do %>
31                      <%= image_tag avatar_url(post.user), class: "post-author-icon" %>
32                  <% end %>
33                  <%= link_to user_path(post.user), class: "normal-color no-text-decoration",
34                      title: post.user.name do %>
35                      <strong><%= post.user.name %></strong>
36                  <% end %>
37              </div>
38
39          <% if post.photos.size == 1%>
40              <%= image_tag post.photos.first.image.url(:standard), class: "card_img_top" %>
41          <% else %>
42              <div class="carousel slide" data-ride="carousel" id="carousel-post-<%= post.id %>">
43                  <div class="carousel-inner">
44                      <% post.photos.each do |photo| %>
45                          <% if photo == post.photos.first %>
46                              <div class="carousel-item active">
47                          <% else %>
48                              <div class="carousel-item">
49                          <% end %>
50                          <%= image_tag photo.image.url(:standard), class: "card_img_top" %>
51                      </div>
52                  <% end %>
53              </div>
54              <a class="carousel-control-prev" href="#carousel-post-<%= post.id %>" role="button" data-slide="prev">
55                  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
56                  <span class="sr-only">Previous</span>
57              </a>
58              <a class="carousel-control-next" href="#carousel-post-<%= post.id %>" role="button" data-slide="next">
59                  <span class="carousel-control-next-icon" aria-hidden="true"></span>
60                  <span class="sr-only">Next</span>
61              </a>
62          </div>
63      <% end %>
64
65      <div class="card-body">
66          <div class="row actions">
67              <a href="core-sprite love hide-text"> Love </a>
68              <a href="core-sprite comment hide-text"> Comment </a>
69              <a href="core-sprite bookmark hide-text"> Bookmark </a>
70          </div>
71          <div><strong><%= pluralize(16, "like")%></strong></div>
72          <div>
73              <span><strong><%= post.user.name %></strong></span>
74              <span><%= post.content %></span>
75          </div>
76          <div class="light-color post-time"><%= time_ago_in_words(post.created_at).upcase %>AGO</div>
77          <hr>
78          <div class="row actions">
79              <form action="#" class="w-100">
80                  <div>
81                      <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." rows="1"></textarea>
82                  </div>
83              </form>
84          </div>
85      </div>
86  <%end%>
87  </div>
```

Dies führt uns zu folgender Ansicht:



Damit alle Posts gleich breit sind wie die Box Create Post, müssen wir noch <div>-Elemente korrekt strukturieren. Kopieren Sie folgenden Codeausschnitt

```
29   <% @posts.each do |post| %>
30     <div class="card mt-3 post">
31       <div class="card-header d-flex align-items-center">
32         <%= link_to user_path(post.user), class: "no-text-decoration" do %>
33           <%= image_tag avatar_url(post.user), class: "post-author-icon" %>
34         <% end %>
35         <%= link_to user_path(post.user), class: "normal-color no-text-decoration", title: post.user.name do %>
36           <strong><%= post.user.name %></strong>
37         <% end %>
38       </div>
39     </div>
40
41     <% if post.photos.size == 1%>
42       <%= image_tag post.photos.first.image.url(:standard), class: "card_img_top" %>
43     <% else %>
44       <div class="carousel slide" data-ride="carousel" id="carousel-post-<%= post.id %>">
45         <div class="carousel-inner">
46           <% post.photos.each do |photo| %>
47             <% if photo == post.photos.first %>
48               <div class="carousel-item active">
49             <% else %>
50               <div class="carousel-item">
51             <% end %>
52               <%= image_tag photo.image.url(:standard), class: "card_img_top" %>
53             </div>
54           <% end %>
55         </div>
56         <a class="carousel-control-prev" href="carousel-post-<%= post.id %>" role="button" data-slide="prev">
57           <span class="carousel-control-prev-icon" aria-hidden="true"></span>
58           <span class="sr-only">Previous</span>
59         </a>
60         <a class="carousel-control-next" href="carousel-post-<%= post.id %>" role="button" data-slide="next">
61           <span class="carousel-control-next-icon" aria-hidden="true"></span>
62           <span class="sr-only">Next</span>
63         </a>
64       </div>
65     <% end %>
66
67     <div class="card-body">
68       <div class="row actions">
69         <a href="#" class="core-sprite love hide-text"> Love </a>
70         <a href="#" class="core-sprite comment hide-text"> Comment </a>
71         <a href="#" class="core-sprite bookmark hide-text"> Bookmark </a>
72       </div>
73       <div><strong><%= pluralize(16, "like") %></strong></div>
74       <div>
75         <span><strong><%= post.user.name %></strong></span>
76         <span><%= post.content %></span>
77       </div>
78       <div class="light-color post-time"><%= time_ago_in_words(post.created_at).upcase %>AGO</div>
79       <hr>
80       <div class="row actions">
81         <form action="#" class="w-100">
82           <div>
83             <textarea class="form-control comment-input border-0" placeholder="Add a comment ..." rows="1"></textarea>
84           </div>
85         </form>
86       </div>
87     </div>
88   </end%>
89 
```

und fügen Sie diesen im <div class="col-xl-7 col-lg-8 col-md-10 col-sm-11 post-card"> Element (Zweites <div>) ein:

```
index.html.erb
1  <div class="d-flex flex-column align-items-center mt-3">
2    <div class="col-xl-7 col-lg-8 col-md-10 col-sm-11 post-card">
3      <div class="card">
4        <div class="card-header"> Create Post </div>
5        <div class="card-body">
6          <%= form_for @post, :html => {:multipart => true, :class => "dropzone upload-images p-0 border-0"} do |f| %>
7            <div class="form-group row mt-2">
8              <div class="col-auto pr-0">
9                <%= image_tag avatar_url(current_user), class: "post-author-icon" %>
10              </div>
11              <div class="col pl-0">
12                <%= f.text_field :content, class: "form-control border-0",
13                  placeholder: "Say something about this ..." %>
14              </div>
15            </div>
16            <div class="fallback"> <%= file_field_tag "images[]", type: :file, multiple: true %> </div>
17            <div class="dz-message m-0"></div>
18            <div class="dropzone-previews mb-3">
19              <div class="upload-photos-icon">
20                <i class="fa fa-plus fa-2x" aria-hidden="true" style="color:#ddfe2"></i>
21              </div>
22            </div>
23            <%= f.submit "Post", class: "btn btn-primary" %>
24          <% end %>
25        </div>
26      </div>
27      <% @posts.each do |post| %>= .....
28      <%end%>
29    </div>
30  </div>
```

Betrachten wir das Resultat, dass sich sehen lässt:

The screenshot shows a web browser window titled "Instagram - Chromium" with the URL "localhost:4000/posts". The page displays a "Create Post" form and a post by user "Ralph Maurer".

Create Post Form:

- Header: "Create Post"
- User profile picture: A small circular profile picture.
- Text input field: "Say something about this ..."
- Image upload area: A dashed square placeholder with a plus sign (+) in the center.
- Post button: A blue rectangular button labeled "Post".

User Post:

- User profile picture: A small circular profile picture.
- User name: "Ralph Maurer" (in blue link text).
- Image: A photograph of a person with dark hair, wearing a white shirt, sitting at a desk and looking down at a computer screen. Other people and equipment are visible in the background.
- Interaction buttons: "Love", "Comment", "Bookmark" (in blue link text).
- Like count: "16 likes".
- Post caption: "Praktische Prüfungen".
- Timestamp: "ABOUT 2 HOURSAGO".
- Comment input field: "Add a comment ...".

Partial-View für Posts list

Die Post-View ist mit soviel Code unbersichtlich. Deshalb erstellen Sie für den Bereich Posts list eine Partial-View app/views/posts/_posts_list.html.erb und kopieren die Schleife mit Posts in die Partial-View:

```
<% @posts.each do |post| %>
  ...
<% end %>
```

In app/views/posts/index.html.erb packen wir das Rendering in einen <div id="post">-Container:

```
index.html.erb          application.scss          _posts_list.html.erb          upload_post_images.js
1 <div class="d-flex flex-column align-items-center mt-3">
2   <div class="col-11 col-lg-8 col-md-10 col-sm-11 post-card">
3     <div class="card">
4       <div class="card-header"> Create Post </div>
5       <div class="card-body">
6         <%= form_for @post, :html => { multipart: true, class: "dropzone upload-images p-0 border-0" } do |f| %>
7           <div class="form-group row mt-2">
8             <div class="col-auto pr-0">
9               <%= image_tag avatar_url(current_user), class: "post-author-icon" %>
10            </div>
11            <div class="col pl-0">
12              <%= f.text_field :content, class: "form-control border-0", placeholder: "Say something about this ..." %>
13            </div>
14          </div>
15        <div class="fallback"> <%= file_field_tag "images[]", type: :file, multiple: true %> </div>
16        <div class="dz-message m-0"></div>
17        <div class="dropzone-previews mb-3">
18          <div class="upload-photos-icon">
19            <i class="fa fa-plus fa-2x" aria-hidden="true" style="color:#ddafe2"></i>
20          </div>
21        </div>
22        <%= f.submit "Post", class: "btn btn-primary" %>
23      <% end %>
24    </div>
25  </div>
26  <div id="post">
27    <%= render 'posts_list'%>
28  </div>
29 </div>
30 </div>
31 </div>
32
```

Kontrollieren Sie das Resultat.

Post View Button designen

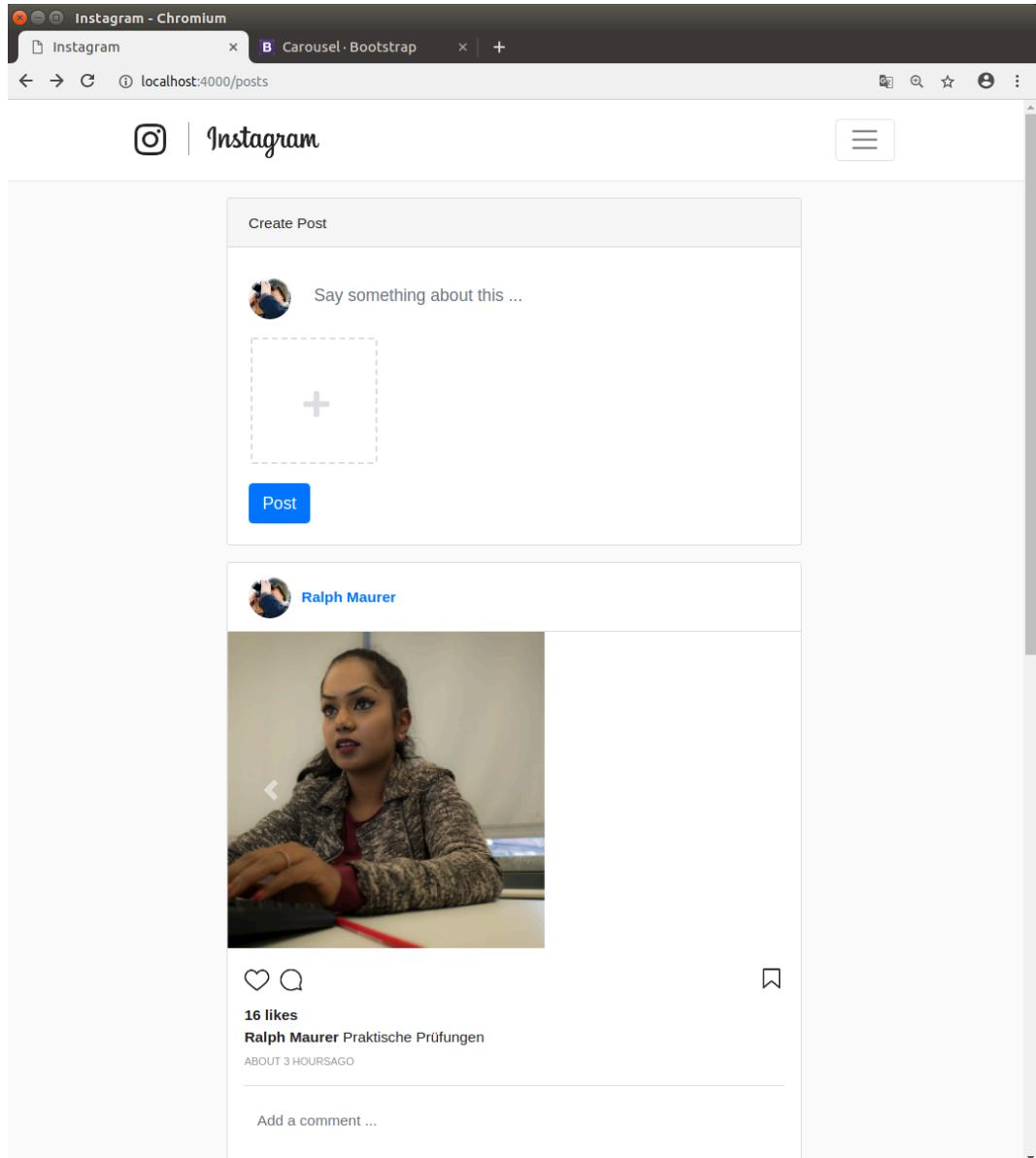
Als nächstes müssen wir die Button Love, Comment und Bookmark anpassen
(app/assets/stylesheets/application.scss):

```
index.html.erb application.scss _posts_list.html.erb
301
302 // Customize post views
303 .post {
304   .card-header {
305     background-color: white;
306   }
307   .card-body {
308     padding: 5px 16px 5px;
309   }
310   .actions {
311     margin: 12px 0;
312   }
313   .post-time {
314     margin-top: 5px;
315     font-size: 10px;
316   }
317   .comment-input {
318     font-size: 14px;
319     resize: none;
320   }
321   .previous-btn {
322     background-position: -281px -280px;
323     height: 30px;
324     opacity: 1;
325     width: 30px;
326   }
327   .next-btn {
328     background-position: -353px 0px;
329     height: 30px;
330     opacity: 1;
331     width: 30px;
332   }
333 }
334
335 .love {
336   background-position: -325px -329px;
337   height: 24px;
338   width: 24px;
339 }
340 .loved {
341   @extend .love;
342   background-position: -200px -330px;
343 }
344 .comment {
345   @extend .love;
346   background-position: -258px -175px;
347   margin-left: 8px;
348 }
349 .bookmark {
350   @extend .love;
351   background-position: -150px -355px;
352 }
353 .bookmarked {
354   @extend .love;
355   background-position: -353px -273px;
356 }
357 }
```

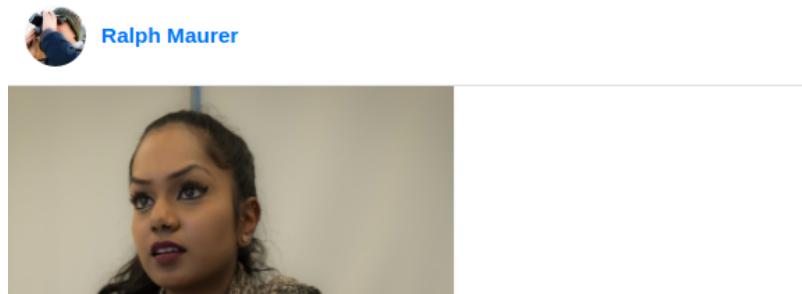
Damit das SCSS greift, müssen noch in der Partial-View app/views/posts/_posts_list.html.erb folgende Links für Love, Comment und Bookmark angepasst werden:

```
<div class="card-body">
  <div class="row actions">
    <a href="#" class="core-sprite love hide-text"> Love </a>
    <a href="#" class="core-sprite comment hide-text"> Comment </a>
    <a href="#" class="core-sprite bookmark hide-text ml-auto"> Bookmark </a>
  </div>
```

Die Button lassen sich nun sehen:



Störend wirkt die blaue Default-Schrift für den Benutzernamen-Link neben dem Avatar.



Erfassen Sie die zwei SCSS-Klassen `.no-text-decoration:hover` und `.normal-text`. Da die Klassen nicht in Post-View gebraucht werden, können Sie diese oberhalb der Post-View Formatierung schreiben.

```
index.html.erb application.scss _posts_list.html.erb upload_post_images.js
302 .no-text-decoration:hover {
303   text-decoration: none;
304   color: #262626;
305 }
306
307 .normal-color {
308   color: #262626;
309 }
310
311 // Customize post views
312 .post {
313   .card-header {
314     background-color: white;
315 }
```

Bildschärfe in Posts

Damit die Bilder schärfen in den Posts erscheinen, passen wir in `app/uploaders/photo_uploader.rb` folgende Parameter an:

```
index.html.erb application.scss photo_uploader.rb routes.rb
1 class PhotoUploader < CarrierWave::Uploader::Base
2   # Include RMagick or MiniMagick support:
3   # include CarrierWave::RMagick
4   # include CarrierWave::MiniMagick
5
6   # Choose what kind of storage to use for this uploader:
7   # storage :file
8   # storage :fog
9
10  # Override the directory where uploaded files will be stored.
11  # This is a sensible default for uploaders that are meant to be mounted:
12  def store_dir
13    "uploads/#{model.class.to_s.underscore}/#{mounted_as}/#{model.id}"
14  end
15
16  include Cloudinary::CarrierWave
17
18  process :convert => 'png'
19  process :tags => ['post_picture']
20
21  version :standard do
22    process :resize_to_fill => [1080, 1080, :center]
23  end
24
25  version :thumbnail do
26    resize_to_fit(100, 100)
27  end
28
```

Die Bilder kommen nun in einer besseren Auflösung daher und sind zentriert und nicht oben links positioniert (north). Cloudinary nimmt nämlich auch eine Kompression vor, die hier parametrisiert werden kann.

Posts als Startseite

Die Posts-View soll als letztes noch als Root-Page definiert werden:

```
root 'posts#index'
```

Auftrag: Quicknote AB151-06

AB151-06 gab viel zu tun. Deshalb wird die übliche Quicknote gekürzt. Sie geben

- › Instagram APP als Archiv ab.

Benennen Sie das Archiv: Klasse _Name_Vorname_AB151-06.zip

In einer kurzen Quicknote schreiben Sie eine persönliche Schlussfolgerung über die vielen Code-Arbeiten in AB151-06:

- › Was haben Sie nicht verstanden?
- › Was müssen Sie in Drittquellen nachschlagen? (Quellenangabe)
- › Was könnten Sie besser machen?

Dem Autor ist bewusst, dass AB151-06 eine Routine (Abschreib)-Arbeit war. Routinearbeiten sollen dazu führen, dass wir bald ein funktionierendes Instagram haben und die Zeit entsprechend effizient nützen.