smartlearn
virtual learning. real success.

MODUL 151
TEIL 6: POST VIEW, CREATE FORM VIEW

Ralph Maurer

Inhaltsverzeichnis AB151-06

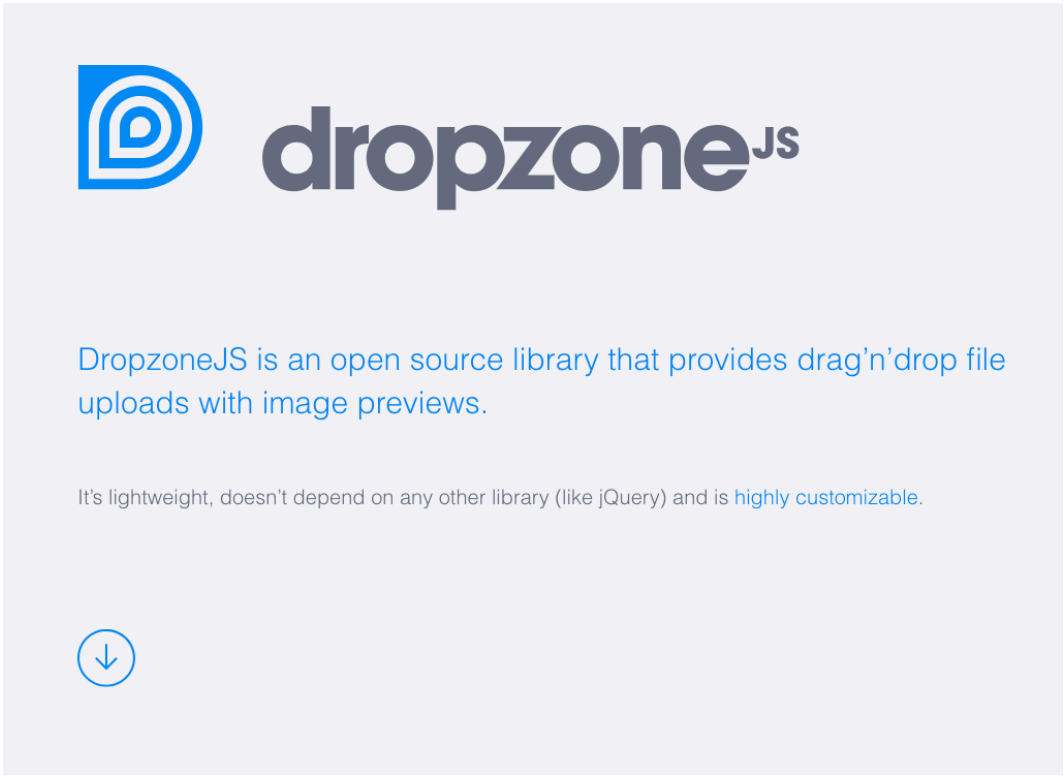
Modul 133: Instagram mit Rails bauen

Teil 6: Instagram Post view, creat form view

Post view

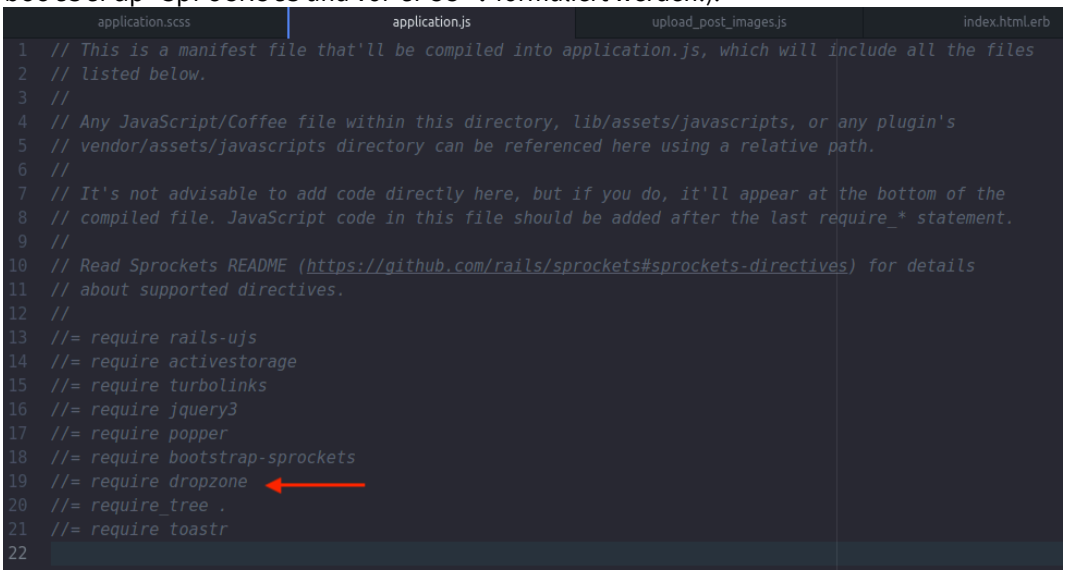
DropzoneJS-rails

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



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

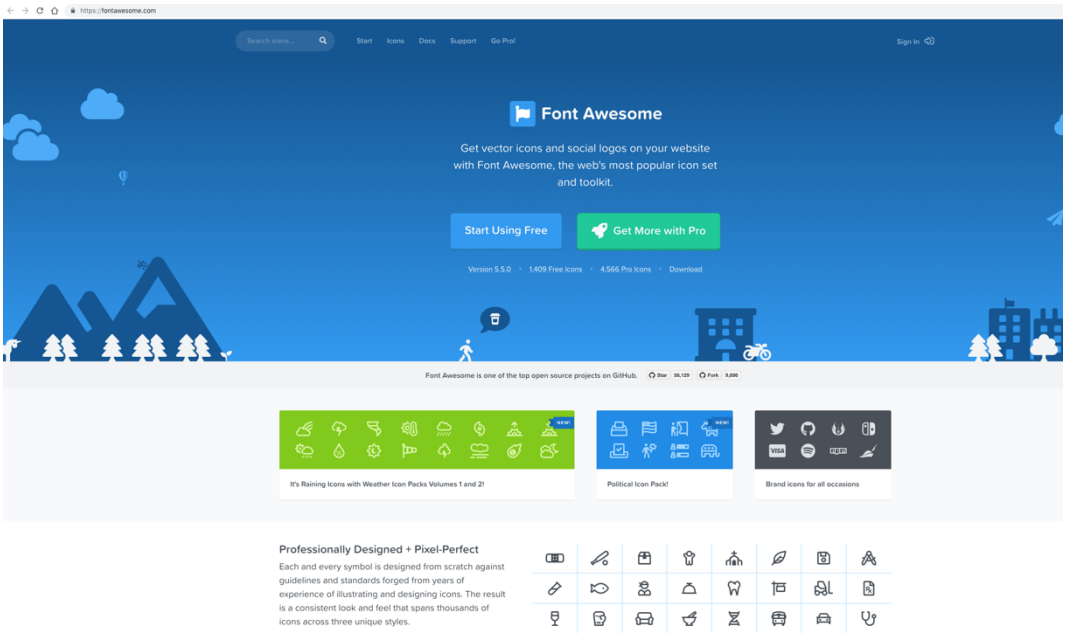
In application.js muss dropzone referenziert werden (Achtung: die Referenz muss nach `bootstrap-sprockets` und vor `tree .` formuliert werden!):



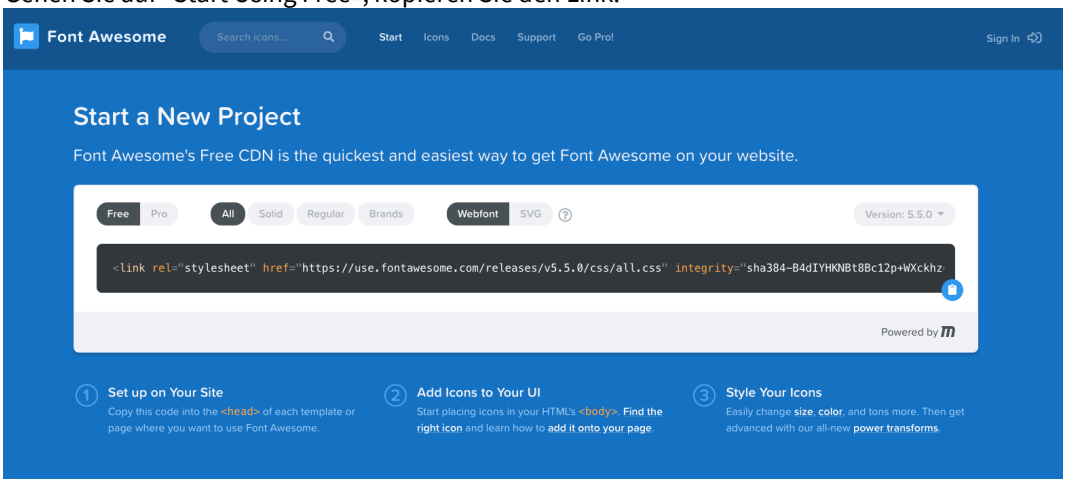
Und im application.scss wird dropzone importiert:
`@import "dropzone/dropzone";`

Icons von fontawesome

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



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



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+WXckhzcICo0wtJ
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>
28
```

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

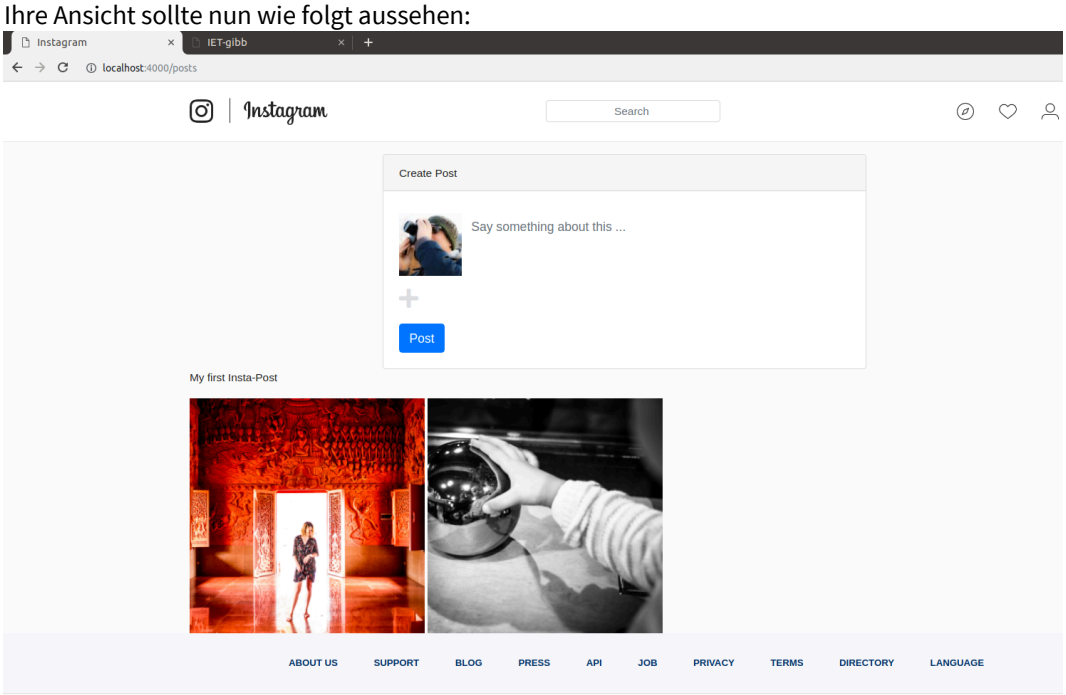
```
<div class="d-flex flex-column align-items-center mt-3">
  <div class="col-xl-7 col-lg-8 col-md-10 col-sm-11 post-card">
    <div class="card">
      <div class="card-header">
        Create Post
      </div>
      <div class="card-body">
        <%= form_for @post, :html => {:multipart => true, :class => "dropzone upload-images p-0 border-0"} do |f| %>
          <div class="form-group row mt-2">
            <div class="col-auto pr-0">
              <%= image_tag avatar_url(current_user), class: "post-author-icon" %>
            </div>
            <div class="col pl-0">
              <%= f.text_field :content, class: "form-control border-0",
                placeholder: "Say something about this ..." %>
            </div>
          </div>
          <div class="fallback">
            <%= file_field_tag "images[]", type: :file, multiple: true %>
          </div>
          <div class="dz-message m-0"></div>
          <div class="dropzone-previews mb-3">
            <div class="upload-photos-icon">
              <i class="fa fa-plus fa-2x" aria-hidden="true" style="color:#dddfe2"></i>
            </div>
          </div>
          <%= f.submit "Post", class: "btn btn-primary" %>
        <% end %>
      </div>
    </div>
  </div>
</div>

<%@posts.each do |post|%>
  <p><%=post.content%></p>
  <%post.photos.each do |photo|%>
    <%=image_tag photo.image.url(:standard)%>
  <%end%>
<%end%>
```

Installieren Sie nun DropzoneJS mit

`bundle install`

und starten Sie den Server neu.

Ihre Ansicht sollte nun wie folgt aussehen:



Post view SCSS

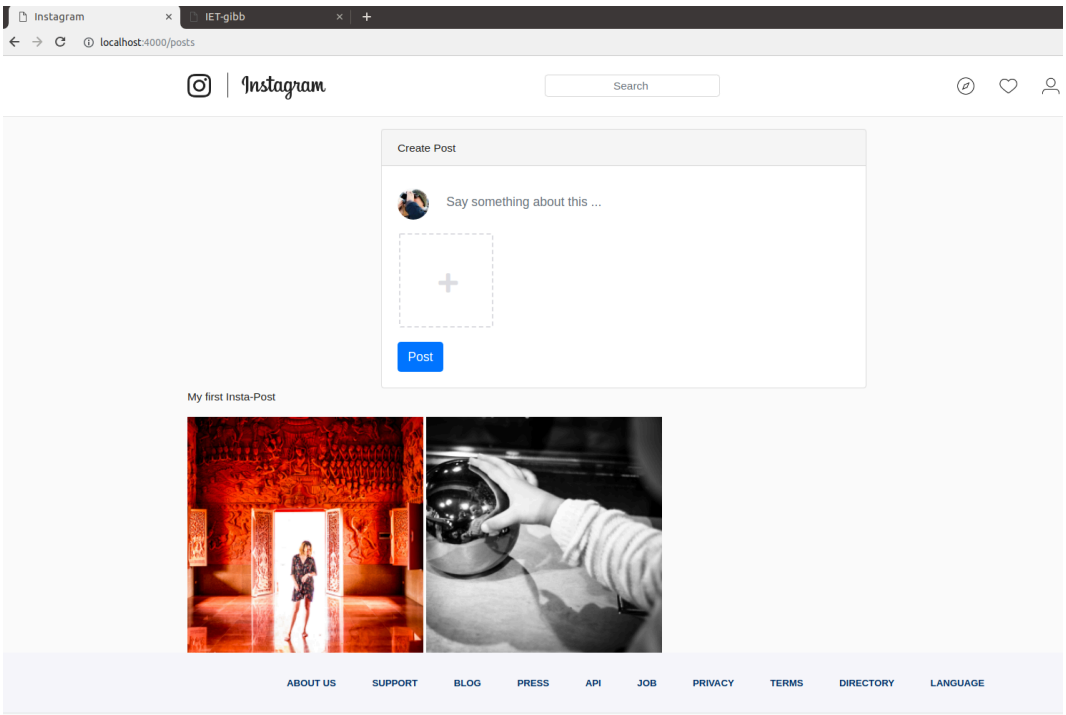
Den ganzen Abschnitt «Create post» müssen wir nun mit dem Instagram Layout ausbauen. Insbesondere der + Button soll besser werden. Hierzu schreiben wir reichlich SCSS-Code und nutzen dabei DropzoneJS.

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

application.scss index.html.erb

```
      font-size: 16px;
    }
  }
}
// Posts
.post-author-icon {
  height: 40px;
  width: 40px;
  border-radius: 50%;
  margin-right: 10px;
}

// Customize upload photos form
.upload-photos-icon:hover {
  cursor: pointer;
}

.dz-preview.dz-image-preview {
  margin: 2px;
}

.dz-image {
  border-radius: 0px !important;

  img {
    width: 100%;
  }
}

.dz-remove {
  @extend .core-sprite;
  @extend .hide-text;
  position: absolute;
  top: 9px;
  right: 5px;
  z-index: 20;
  width: 18px;
  height: 18px;
  background-position: -386px -25px;
  opacity: 0;
  -webkit-transition: opacity 0.2s liner;
  transition: opacity 0.2s liner;
}

.dropzone .dz-preview:hover .dz-remove {
  opacity: 1;
}

.upload-photos-icon {
  height: 120px;
  width: 120px;
  border: 2px dashed #dddfe2;
  border-radius: 2px;
  padding: 48px;
  display: inline-block;
  margin: 2px;
}
```

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



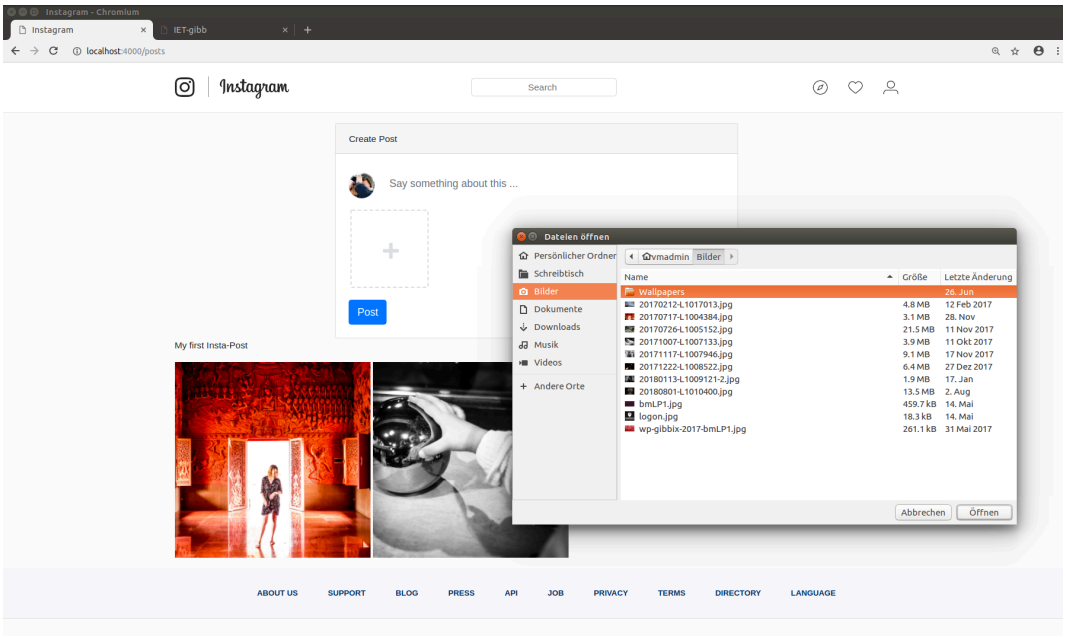
+ Button mit DropzoneJS gestalten

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

```
Dropzone.autoDiscover = false;

$(document).ready(function(){
  $(".upload-images").dropzone({
    addRemoveLinks: true,
    maxFilesize: 1,
    autoProcessQueue: false,
    uploadMultiple: true,
    parallelUploads: 100,
    maxFiles: 100,
    paramName: "images",
    previewsContainer: ".dropzone-previews",
    clickable: ".upload-photos-icon",
    thumbnailWidth: 100,
    thumbnailHeight: 100,

    init: function(){
      var myDropzone = this;

      this.element.querySelector("input[type=submit]").addEventListener("click", function(e){
        e.preventDefault();
        e.stopPropagation();
        myDropzone.processQueue();
      });

      this.on("successmultiple", function(files, response){
        window.location.reload();
      });

      this.on("errormultiple", function(files, response){
        toastr.error(response);
      });
    }
  })
});

```

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.

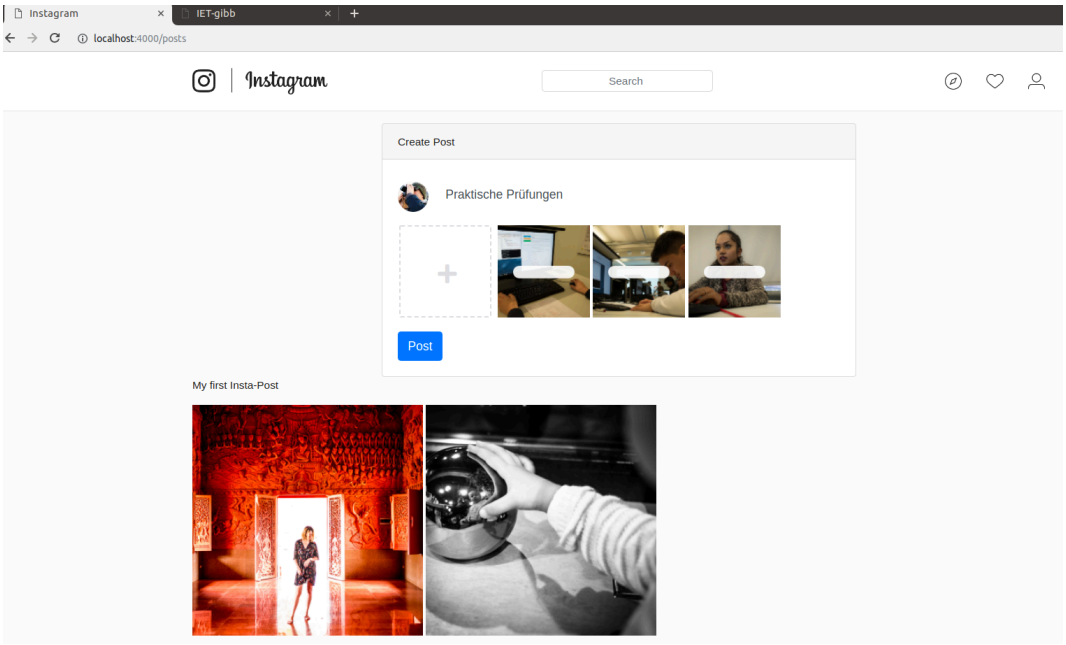
```
validates :image, presence: true
```

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

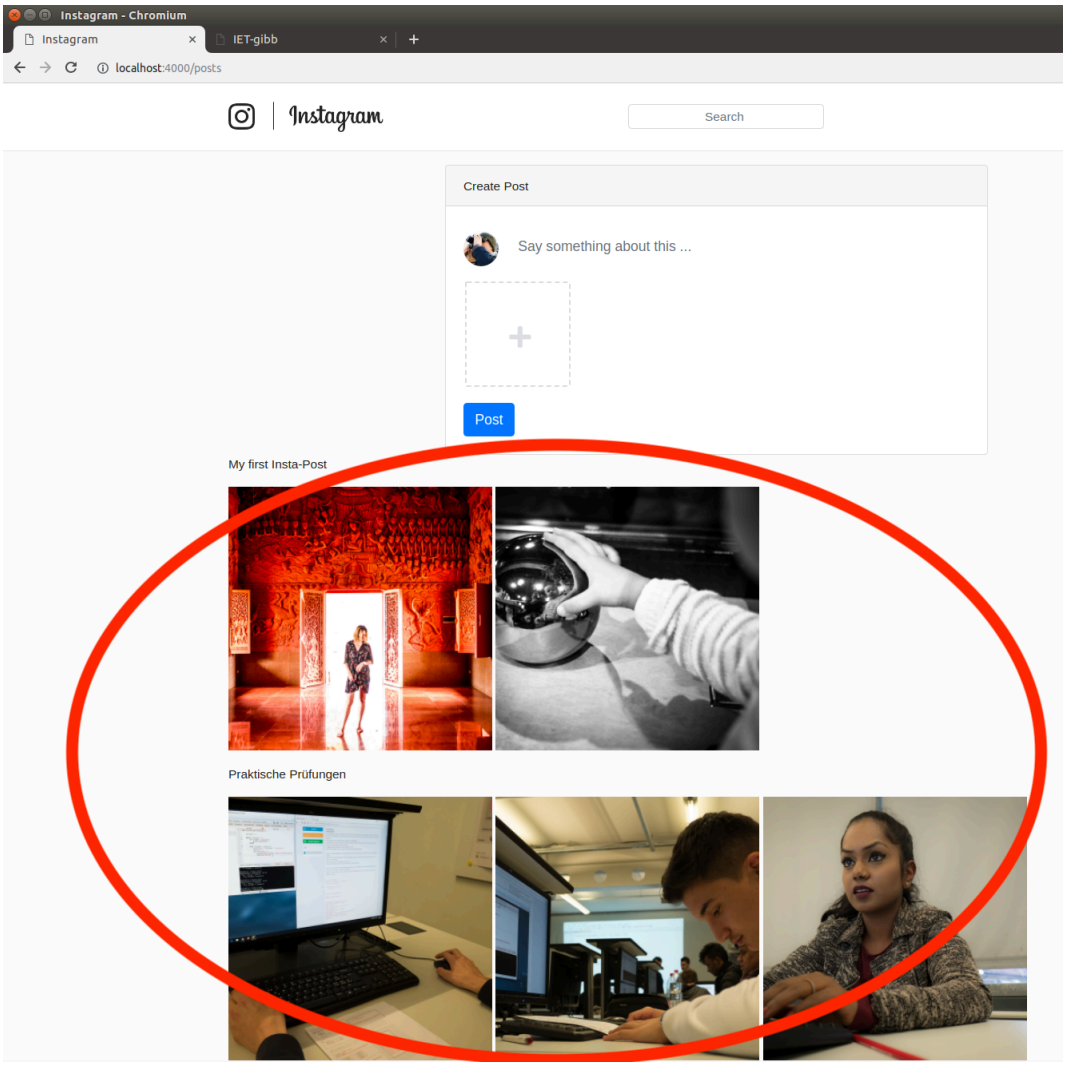
Sie können nur Bilder mit einer Grösse bis zu 1 MB hochladen. Finden Sie den entsprechenden Parameter und verändern Sie diesen, so dass 3 MB grosse Bilder angewandt werden können. Welche Datei und welcher Eintrag sind betroffen?

upload_post_images.js --> maxFilesize: 1,

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:

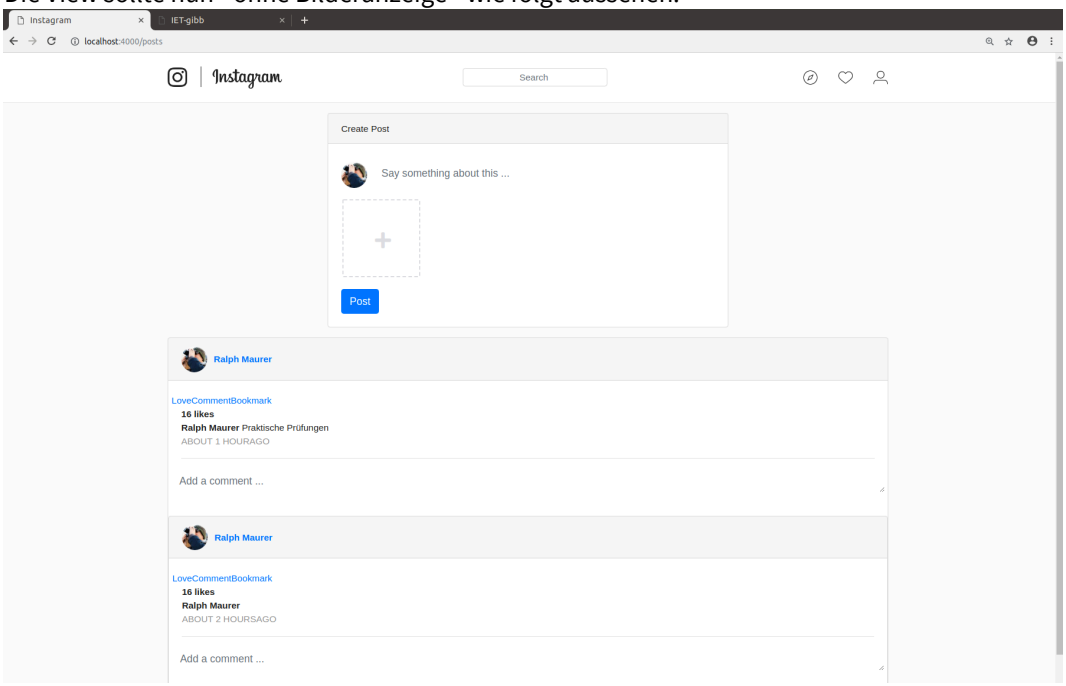
```
5   def index
6     @posts = Post.all.limit(10).includes(:photos, :user).order('created_at desc')
7     @post = Post.new
8   end
9
```

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

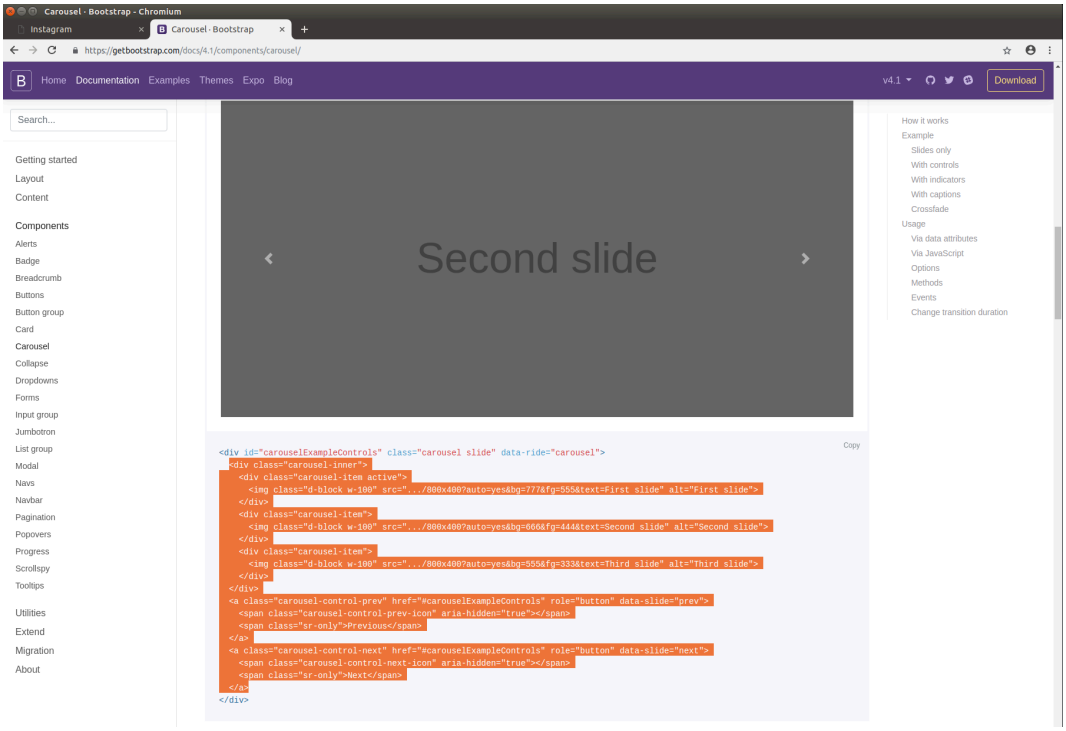
index.html.erb

```
<div class="d-flex flex-column align-items-center mt-3">
  <div class="col-xl-7 col-lg-8 col-md-10 col-sm-11 post-card">
    <div class="card">
      <div class="card-header"> Create Post </div>
      <div class="card-body">
        <%= form_for @post, :html => {:multipart => true, :class => "dropzone upload-images p-0 border-0"} do |f| %>
          <div class="form-group row mt-2">
            <div class="col-auto pr-0">
              <%= image_tag avatar_url(current_user), class: "post-author-icon" %>
            </div>
            <div class="col pl-0">
              <%= f.text_field :content, class: "form-control border-0",
                placeholder: "Say something about this ..." %>
            </div>
          </div>
          <div class="fallback"> <%= file_field_tag "images[]", type: :file, multiple: true %> </div>
          <div class="dz-message m-0"></div>
          <div class="dropzone-previews mb-3">
            <div class="upload-photos-icon">
              <i class="fa fa-plus fa-2x" aria-hidden="true" style="color:#dddfe2"></i>
            </div>
          </div>
          <%= f.submit "Post", class: "btn btn-primary" %>
        <% end %>
      </div>
    </div>
  </div>
</div>
<% @posts.each do |post| %>
  <div class="card mt-3 post">
    <div class="card-header d-flex align-items-center">
      <%= link_to user_path(post.user), class: "no-text-decoration" do %>
        <%= image_tag avatar_url(post.user), class: "post-author-icon" %>
      <% end %>
      <%= link_to user_path(post.user), class: "normal-color no-text-decoration",
        title: post.user.name do %>
        <strong><%= post.user.name %></strong>
      <% end %>
    </div>
    <div class="card-body">
      <div class="row actions">
        <a href="core-sprite love hide-text"> Love </a>
        <a href="core-sprite comment hide-text"> Comment </a>
        <a href="core-sprite bookmark hide-text"> Bookmark </a>
    </div>
    <div><strong><%= pluralize(16, "like")%></strong></div>
    <div>
      <span><strong><%= post.user.name %></strong></span>
      <span><%= post.content %></span>
    </div>
    <div class="light-color post-time"><%= time_ago_in_words(post.created_at).upcase %>AGO</div>
    <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"></textarea>
      </div>
    </form>
   </div>
  </div>
<%end%>
```

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



Für die Darstellung der Bilder nutzen wir wiederum 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).



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

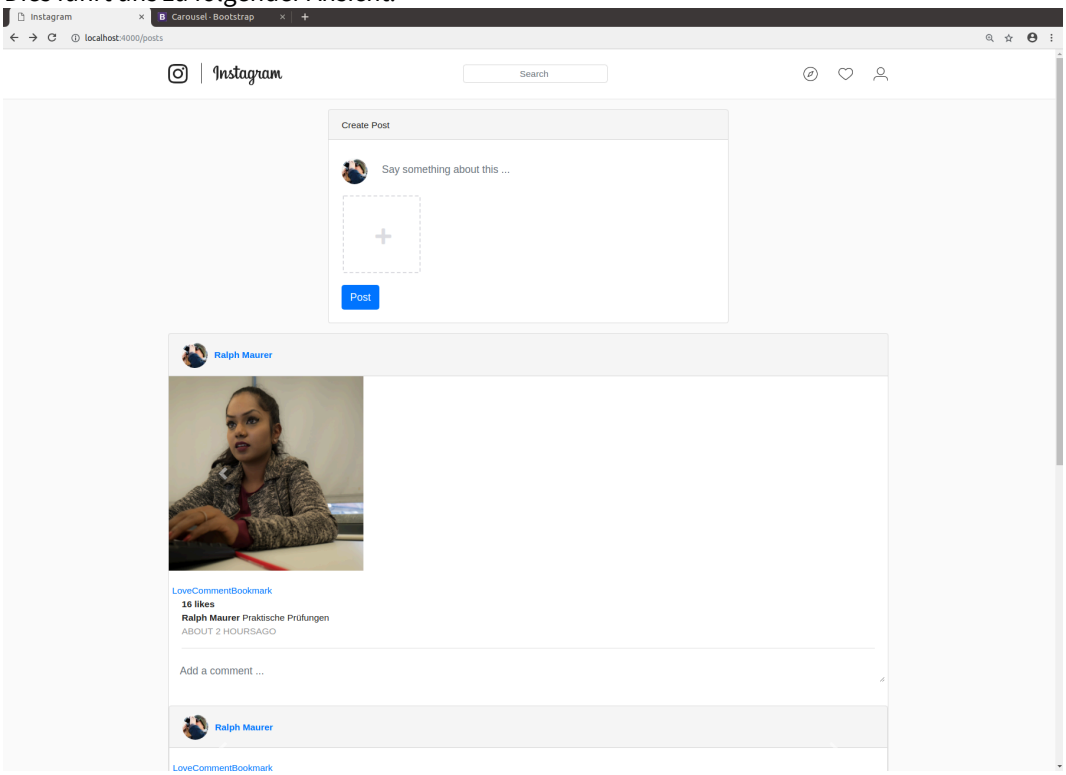
index.html.erb

```
            <%= f.submit "Post", class: "btn btn-primary" %>
          <% end %>
        </div>
      </div>
    </div>
  </div>
  <% @posts.each do |post| %>
    <div class="card mt-3 post">
      <div class="card-header d-flex align-items-center">
        <%= link_to user_path(post.user), class: "no-text-decoration" do %>
          <%= image_tag avatar_url(post.user), class: "post-author-icon" %>
        <% end %>
        <%= link_to user_path(post.user), class: "normal-color no-text-decoration",
          title: post.user.name do %>
          <strong><%= post.user.name %></strong>
        <% end %>
      </div>

      <div class="carousel-inner">
          <div class="carousel-item active">
            <img class="d-block w-100" src=".../800x400?auto=yes&bg=777&fg=555&text=First slide" alt="First slide">
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src=".../800x400?auto=yes&bg=666&fg=444&text=Second slide" alt="Second slide">
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src=".../800x400?auto=yes&bg=555&fg=333&text=Third slide" alt="Third slide">
          </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 class="card-body">
        <div class="row actions">
          <a href="core-sprite love hide-text"> Love </a>
          <a href="core-sprite comment hide-text"> Comment </a>
          <a href="core-sprite bookmark hide-text"> Bookmark </a>
      </div>
      <div><strong><%= pluralize(16, "like")%></strong></div>
      <div>
        <span><strong><%= post.user.name %></strong></span>
        <span><%= post.content %></span>
      </div>
      <div class="light-color post-time"><%= time_ago_in_words(post.created_at).upcase %>AGO</div>
      <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"></textarea>
        </div>
      </form>
     </div>
    </div>
  <%end%>
```

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

```
<% @posts.each do |post| %>
  <div class="card mt-3 post">
    <div class="card-header d-flex align-items-center">
      <%= link_to user_path(post.user), class: "no-text-decoration" do %>
        <%= image_tag avatar_url(post.user), class: "post-author-icon" %>
      <% end %>
      <%= link_to user_path(post.user), class: "normal-color no-text-decoration",
        title: post.user.name do %>
        <strong><%= post.user.name %></strong>
      <% end %>
    </div>

    <% if post.photos.size == 1%>
      <%= image_tag post.photos.first.image.url(:standard), class: "card_img_top" %>
    <% 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 %>
                <%= image_tag photo.image.url(:standard), class: "card_img_top" %>
              </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 %>

    <div class="card-body">
      <div class="row actions">
        <a href="core-sprite love hide-text"> Love </a>
        <a href="core-sprite comment hide-text"> Comment </a>
        <a href="core-sprite bookmark hide-text"> Bookmark </a>
      </div>
      <div><strong><%= pluralize(16, "like")%></strong></div>
      <div>
        <span><strong><%= post.user.name %></strong></span>
        <span><%= post.content %></span>
      </div>
      <div class="light-color post-time"><%= time_ago_in_words(post.created_at).upcase %>AGO</div>
      <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"></textarea>
        </div>
      </form>
     </div>
    </div>
  <%end%>
</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

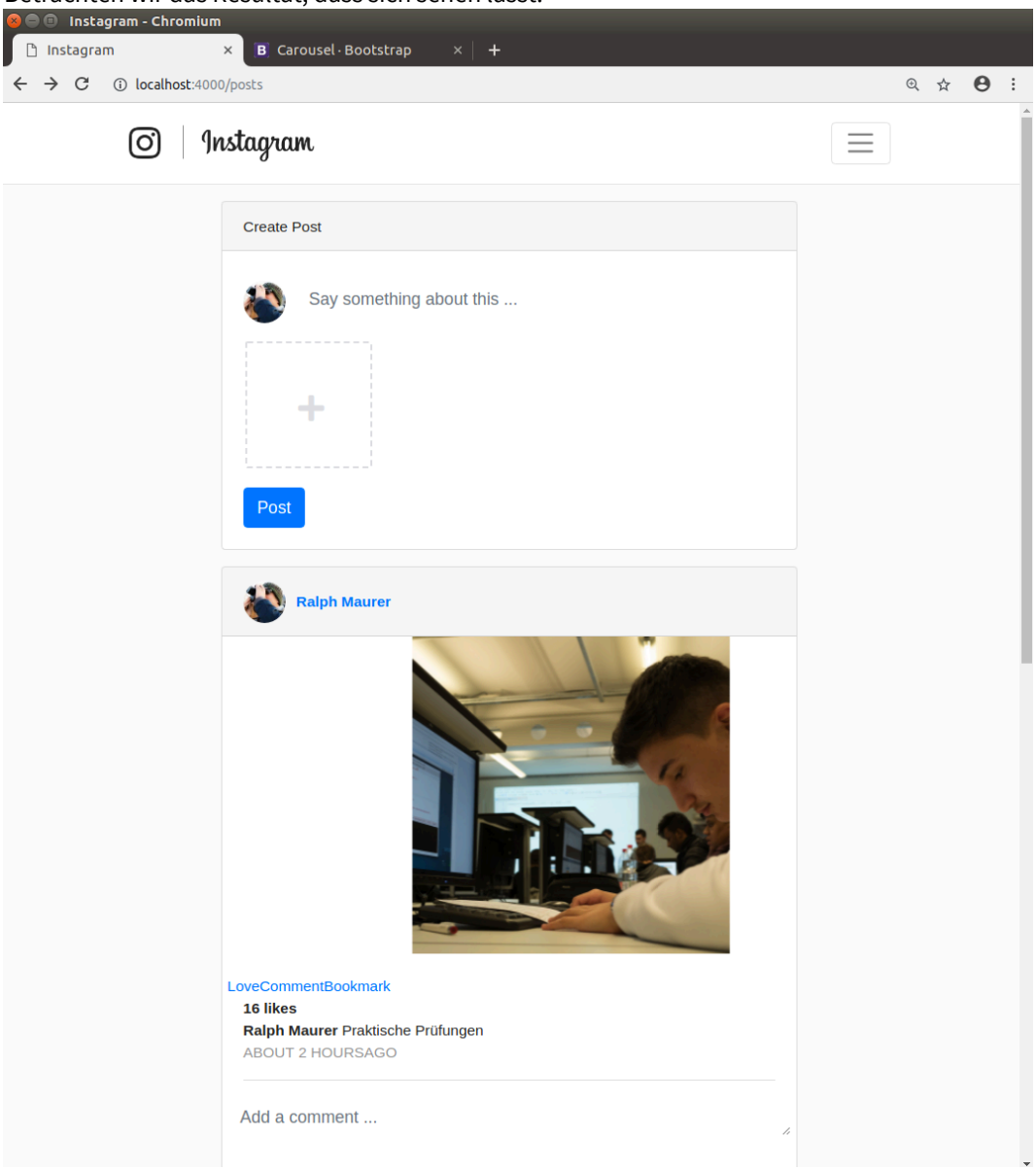
```
<% @posts.each do |post| %>
  <div class="card mt-3 post">
    <div class="card-header d-flex align-items-center">
      <%= link_to user_path(post.user), class: "no-text-decoration" do %>
        <%= image_tag avatar_url(post.user), class: "post-author-icon" %>
      <% end %>
      <%= link_to user_path(post.user), class: "normal-color no-text-decoration",
        title: post.user.name do %>
        <strong><%= post.user.name %></strong>
      <% end %>
    </div>

    <% if post.photos.size == 1%>
      <%= image_tag post.photos.first.image.url(:standard), class: "card_img_top" %>
    <% 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 %>
                <%= image_tag photo.image.url(:standard), class: "card_img_top" %>
              </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 %>

    <div class="card-body">
      <div class="row actions">
        <a href="core-sprite love hide-text"> Love </a>
        <a href="core-sprite comment hide-text"> Comment </a>
        <a href="core-sprite bookmark hide-text"> Bookmark </a>
    </div>
    <div><strong><%= pluralize(16, "like")%></strong></div>
    <div>
      <span><strong><%= post.user.name %></strong></span>
      <span><%= post.content %></span>
    </div>
    <div class="light-color post-time"><%= time_ago_in_words(post.created_at).upcase %>AGO</div>
    <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"></textarea>
      </div>
    </form>
   </div>
  </div>
<%end%>

```

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

```
<div class="d-flex flex-column align-items-center mt-3">
  <div class="col-xl-7 col-lg-8 col-md-10 col-sm-11 post-card">
    <div class="card">
      <div class="card-header"> Create Post </div>
      <div class="card-body">
        <%= form_for @post, :html => {:multipart => true, :class => "dropzone upload-images p-0 border-0"} do |f| %>
          <div class="form-group row mt-2">
            <div class="col-auto pr-0">
              <%= image_tag avatar_url(current_user), class: "post-author-icon" %>
            </div>
            <div class="col pl-0">
              <%= f.text_field :content, class: "form-control border-0",
                placeholder: "Say something about this ..." %>
            </div>
          </div>
          <div class="fallback"> <%= file_field_tag "images[]", type: :file, multiple: true %> </div>
          <div class="dz-message m-0"></div>
          <div class="dropzone-previews mb-3">
            <div class="upload-photos-icon">
              <i class="fa fa-plus fa-2x" aria-hidden="true" style="color:#dddfe2"></i>
            </div>
          </div>
          <%= f.submit "Post", class: "btn btn-primary" %>
        <% end %>
      </div>
    </div>
    <% @posts.each do |post| %>
    <%end%>
  </div>
</div>
```

Betrachten wir das Resultat, dass sich sehen lässt:



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

```
<div class="d-flex flex-column align-items-center mt-3">
  <div class="col-xl-7 col-lg-8 col-md-10 col-sm-11 post-card">
    <div class="card">
      <div class="card-header"> Create Post </div>
      <div class="card-body">
        <%= form_for @post, :html => {:multipart => true, :class => "dropzone upload-images p-0 border-0"} do |f| %>
          <div class="form-group row mt-2">
            <div class="col-auto pr-0">
              <%= image_tag avatar_url(current_user), class: "post-author-icon" %>
            </div>
            <div class="col pl-0">
              <%= f.text_field :content, class: "form-control border-0",
                placeholder: "Say something about this ..." %>
            </div>
          </div>
          <div class="fallback"> <%= file_field_tag "images[]", type: :file, multiple: true %> </div>
          <div class="dz-message m-0"></div>
          <div class="dropzone-previews mb-3">
            <div class="upload-photos-icon">
              <i class="fa fa-plus fa-2x" aria-hidden="true" style="color:#dddfe2"></i>
            </div>
          </div>
          <%= f.submit "Post", class: "btn btn-primary" %>
        <% end %>
      </div>
    </div>
    <div id="post">
      <%= render 'posts_list'%>
    </div>
  </div>
</div>
```

Kontrollieren Sie das Resultat.

Post View Button designen

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

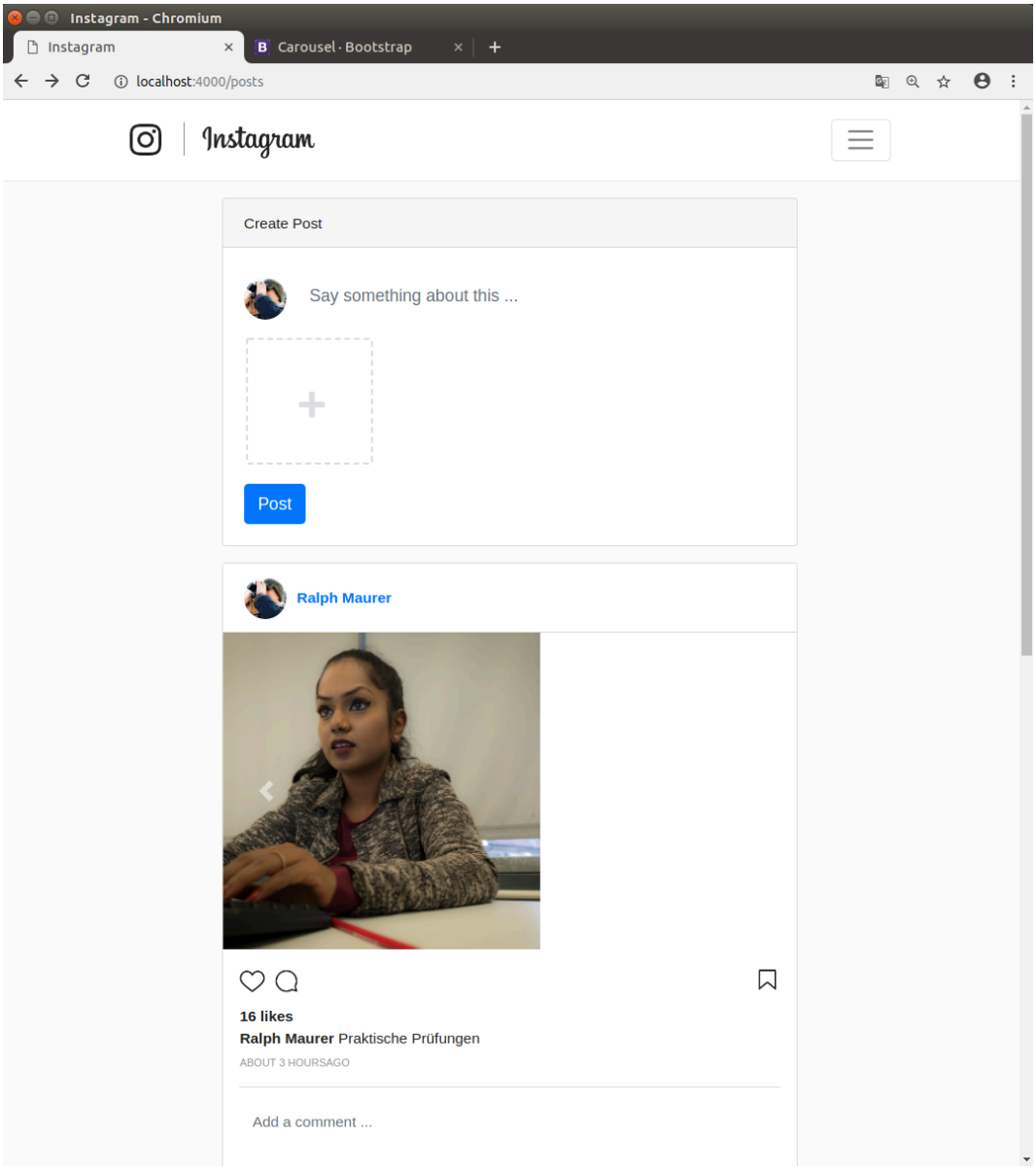
index.html.erb | application.scss | _posts_list.html.erb

```
// Customize post views
.post {
  .card-header {
    background-color: white;
  }
  .card-body {
    padding: 5px 16px 5px;
  }
  .actions {
    margin: 12px 0;
  }
  .post-time {
    margin-top: 5px;
    font-size: 10px;
  }
  .comment-input {
    font-size: 14px;
    resize: none;
  }
  .previous-btn {
    background-position: -281px -280px;
    height: 30px;
    opacity: 1;
    width: 30px;
  }
  .next-btn {
    background-position: -353px 0px;
    height: 30px;
    opacity: 1;
    width: 30px;
  }
}

.love {
  background-position: -325px -329px;
  height: 24px;
  width: 24px;
}
.loved {
  @extend .love;
  background-position: -200px -330px;
}
.comment {
  @extend .love;
  background-position: -258px -175px;
  margin-left: 8px;
}
.bookmark {
  @extend .love;
  background-position: -150px -355px;
}
.bookmarked {
  @extend .love;
  background-position: -353px -273px;
}
```

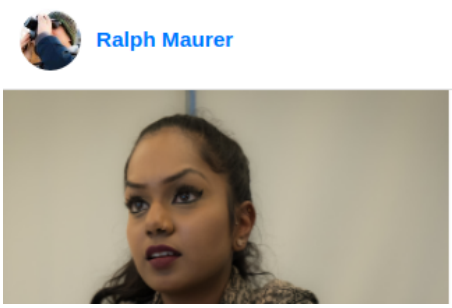
Damit das SCSS greift, müssen noch in der Partial-View `app/views/posts/_posts_list.html.erb` folgende Links für Love, Comment und Bookmark angepasst werden:

```
<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>
```

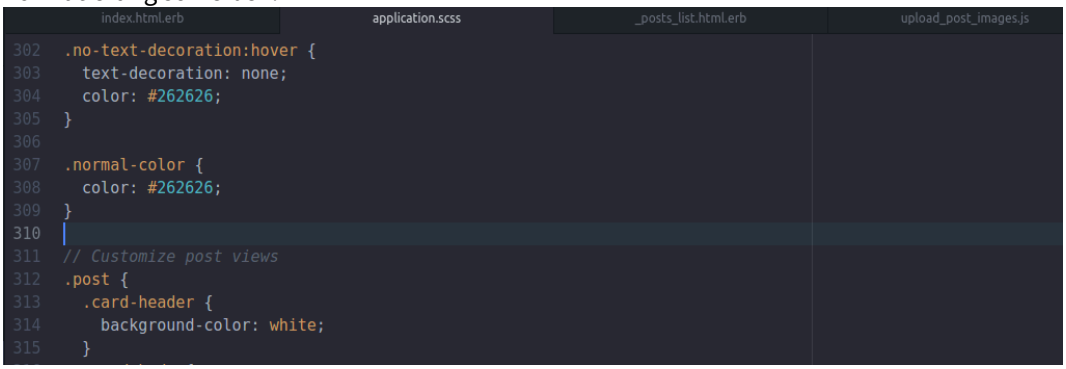
Die Button lassen sich nun sehen:



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

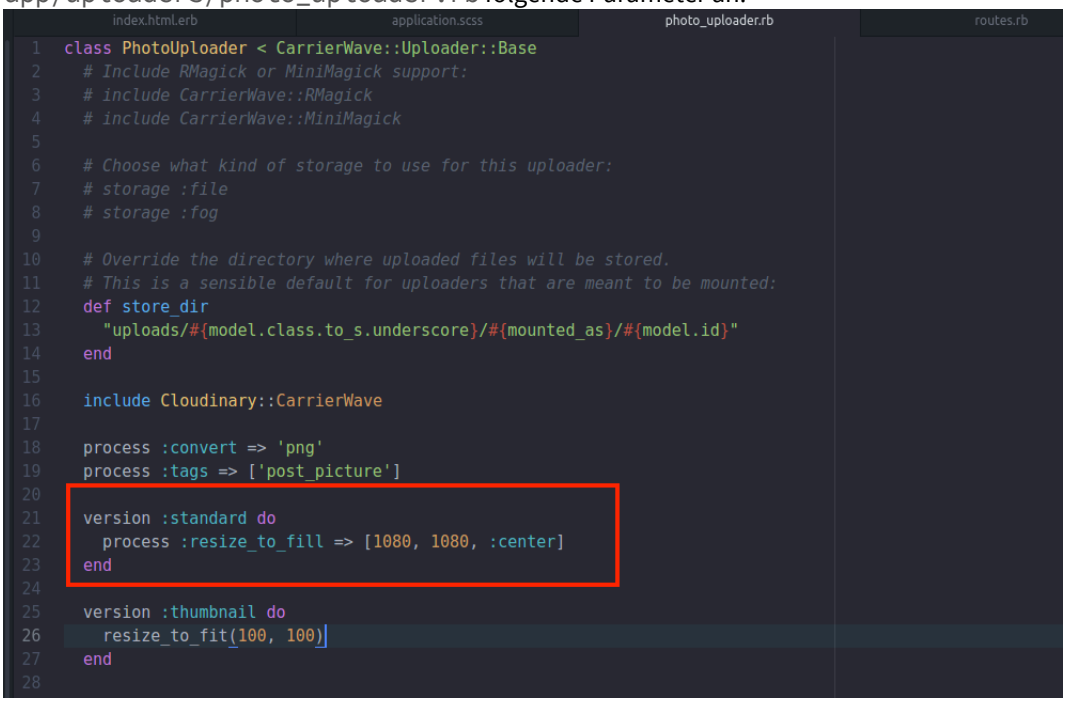


Erfassen Sie die zwei SCSS-Klassen `.no-text-decoration:hover` und `.normal-text`. Da die Klassen nicht in Post-View gebraucht werden, können Sie diese oberhalb der Post-View Formatierung schreiben.



Bildschärfe in Posts

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



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.