



# **MODUL 151**

## **TEIL 3: INSTAGRAM AUTHENTICATION VIEW**

Ralph Maurer

# Inhaltsverzeichnis AB151-03

## Modul 133: Instagram mit Rails bauen

### Teil 3: Instagram Authentication View

Inhaltsverzeichnis AB151-03.....	1
Modul 133: Instagram mit Rails bauen.....	1
Attribut Benutzernamen erfassen .....	2
Login-View «sign_in» anpassen .....	4
Menu ausblenden bei logout .....	4
“dummy-phone” aus Carousel in Login-View «sign_in» einbauen.....	5
Carousel <form_for> in Login-View «sign_in» einbauen .....	8
<footer> in in Login-View «sign_in» einbauen .....	12
Login-View «sign_up» anpassen.....	13
Auftrag: Quicknote AB151-03.....	16

# Attribut Benutzernamen erfassen

Betrachten wir die Datenbank, die «devise gem» generiert hat:

The screenshot shows a database schema browser with the 'Schema' tab selected. Under the 'main' database, the 'Tabellen' (Tables) section contains three entries: 'ar\_internal\_metadata', 'schema\_migrations', and 'users'. The 'users' table is highlighted with an orange selection bar. The 'Columns' section lists eight columns: id, email, encrypted\_password, reset\_password\_token, reset\_password\_sent\_at, remember\_created\_at, created\_at, and updated\_at. Below the columns, there are sections for 'Indices' (2), 'Foreign Keys' (0), and 'Triggers' (0). The 'Sichten' (Views) and 'Systemkatalog' (System Catalog) sections are also visible. To the right of the schema browser is a large grid area where data can be viewed or edited. Below the schema browser is a toolbar with various icons. At the bottom of the interface, there is a status bar with the text 'Abfrage OK' and '1 Zeile(n) zurückgegeben'.

2.4.1 :001 > User.connection

....

2.4.1 :002 > User

```
=> User(id: integer, email: string, encrypted_password: string,
reset_password_token: string, reset_password_sent_at: datetime,
remember_created_at: datetime, created_at: datetime, updated_at:
datetime)
```

## Das Attribut Benutzername fehlt!

Fügen Sie dieses Attribut mit einer Migration mit Namen `AddNameToUser` hinzu. Rails nutzt Namenskonventionen und wenn Sie Add«Attributnamen»To«Tabellennamen» anwenden, passiert so einiges automatisch. Besonders wenn Sie noch Attributname und Datentyp bei der Migrationserstellung im Namen angeben.

```
1 class AddNameToUser < ActiveRecord::Migration[5.2]
2   def change
3     add_column :users, :name, :string
4   end
5 end
6
```

Wie lautet die Generierung der Migration, so dass automatisch die obige Migrationsklasse generiert wird? Erstellen Sie die Migration und führen Sie diese aus. Notieren Sie die notwendigen Kommandos:

A blank lined notebook page with a small pen icon in the top right corner, intended for the student to write their notes.

Wir wollen nun den Usernamen nutzen und validieren. Erstellen Sie in der Datei `instagram/models/user.rb` eine Validierung, die prüft,

- › ob der Benutzername angegeben wurde
- › sowie die max. Länge nicht 50 Zeichen überschreitet.

Notieren Sie die Validierung:



A large rectangular area with four horizontal lines for writing. In the top right corner, there is a small icon of a pen.

Die «devise gem» lässt aktuell nur die Email-Adresse als Benutzername zu. Um dies zu ändern müssen wir `instagram/controllers/application_controller.rb` ändern.



Beachten Sie, dass der folgende Code «devise gem» spezifisch ist und einfach übernommen werden kann.

Wollen Sie mehr wissen oder etwas nachschlagen, finden Sie hier die «devise gem»-Documentary: <https://www.rubydoc.info/gems/devise/Devise>

```
application_controller.rb
1 class ApplicationController < ActionController::Base
2   protect_from_forgery with: :exception
3
4   before_action :configure_permitted_parameters, if: :devise_controller?
5
6   protected
7
8   def configure_permitted_parameters
9     devise_parameter_sanitizer.permit(:sign_up, keys: [:name])
10    devise_parameter_sanitizer.permit(:account_update, keys: [:name])
11  end
12 end
```

Achten Sie darauf, den Code korrekt zu übernehmen oder die Rails-App wird nicht mehr funktionieren.

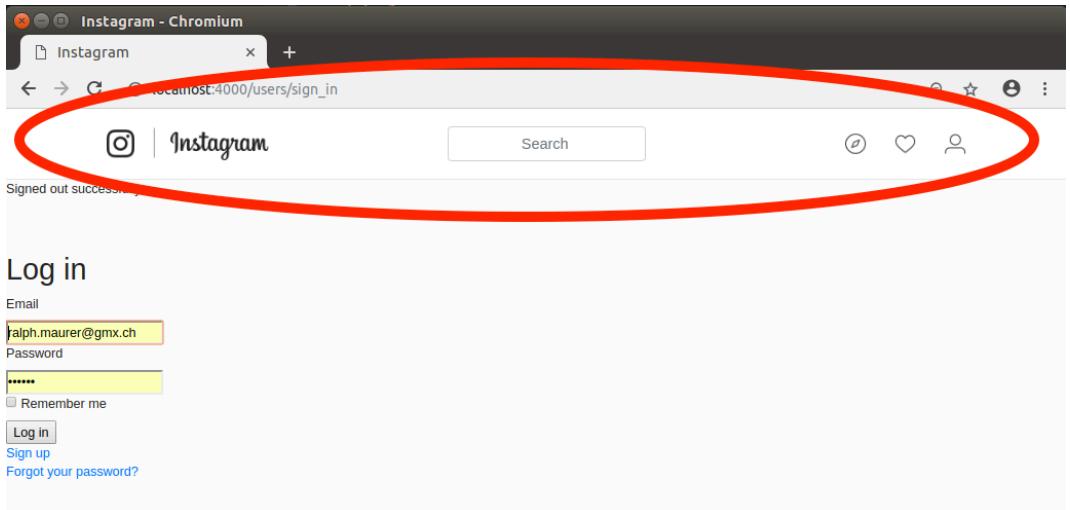
Sie können alles testen indem Sie den Rails Server neustarten. Mögliche Syntaxfehler werden genau beschrieben und können korrigiert werden.

# Login-View «sign\_in» anpassen

Die Login-View «sign\_in» ist wie bekannt unter [http://localhost:4000/users/sign\\_in](http://localhost:4000/users/sign_in) aufrufbar.

## Menu ausblenden bei logout

Bevor wir nicht eingeloggt sind, soll das obere Menu verschwinden. Dazu biete uns «devise gem» die Variable `current_user`. Mit der Abfrage `if current_user` wird geprüft, ob eine valide Authentifikation durch einen Benutzer stattgefunden hat.



Bauen Sie die Abfrage in der richtigen view am richtigen Ort ein. Notieren Sie die View und die Zeile mit der Abfrage:

Notepad icon

---

---

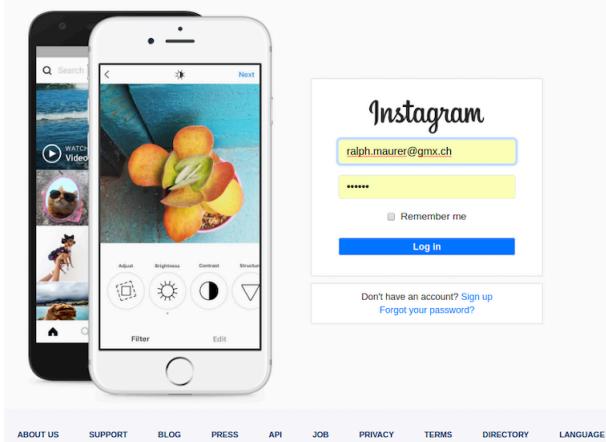
---

Wir passen nun die Login-View an, was einiges an Schreibaufwand verursacht. Öffnen Sie `/app/view/devise/sessions/new.html.erb` und kommentieren Sie alles aus. Einige Teile können später wiederverwendet werden (`ctrl + /`).

```
<!-- <h2>Log in</h2>
<%= form_for(resource, as: resource_name, url:
session_path(resource_name)) do |f| %>
  <div class="field">
    <%= f.label :email %><br />
    <%= f.email_field :email, autofocus: true, autocomplete: "email" %>
  </div>
  ...
<% end %>
<%= render "devise/shared/links" %> -->
```

# “dummy-phone” aus Carousel in Login-View «sign\_in» einbauen

Die Sign-in View soll in etwa so aussehen:



In der linken Seite platzieren wir ein «dummy-phone», dessen Display mit einem Image-Slider gestaltet wird. Auf der rechten Seite kommt die Registrierung. Passen Sie die Datei /app/view/devise/sessions/new.html.erb wie folgt an:

```
new.html.erb | application.scss
1  <div class="container">
2    <div class="row">
3      <div class="col-lg-6 landing-left">
4        <div class="dummy-phone">
5          <div class="screen-shot">
6            <div class="carousel.carousel-fade" data-ride="carousel">
7              <div class="carousel-inner">
8                <div class="carousel-item active">
9                  <%= image_tag "screenshot1.jpg", height: '427', width: '240' %>
10               </div>
11               <div class="carousel-item">
12                 <%= image_tag "screenshot2.jpg", height: '427', width: '240' %>
13               </div>
14               <div class="carousel-item">
15                 <%= image_tag "screenshot3.jpg", height: '427', width: '240' %>
16               </div>
17               <div class="carousel-item">
18                 <%= image_tag "screenshot4.jpg", height: '427', width: '240' %>
19               </div>
20               <div class="carousel-item">
21                 <%= image_tag "screenshot5.jpg", height: '427', width: '240' %>
22               </div>
23             </div>
24           </div>
25         </div>
26       </div>
27     </div>
28   <div class="col-lg-6 landing-right">
29
30
31     </div>
32   </div>
33 </div>
34
35 <!-- <h2>Log in</h2>
36 <%= form_for(resource, as: resource_name, url: session_path(resource_name)) do
37   <div class="field">
38     <%= f.label :email %><br />
39     <%= f.email_field :email, autofocus: true, autocomplete: "email" %>
```

Ergänzen Sie die SCSS Datei  
/app/assets/stylesheets/applications.scss

```
74 // Landing
75 .landing-left {
76   .dummy-phone {
77     background-image: image-url("dummy-phone-2x.png");
78     background-repeat: no-repeat;
79     background-position: right 0;
80     background-size: 454px 618px;
81     position: relative;
82     height: 618px;
83     margin-left: -35px;
84     margin-right: -15px;
85   }
86   .screen-shot {
87     position: absolute;
88     top: 99px;
89     right: 63px;
90
91     .item {
92       position: absolute;
93     }
94   }
95 }
```

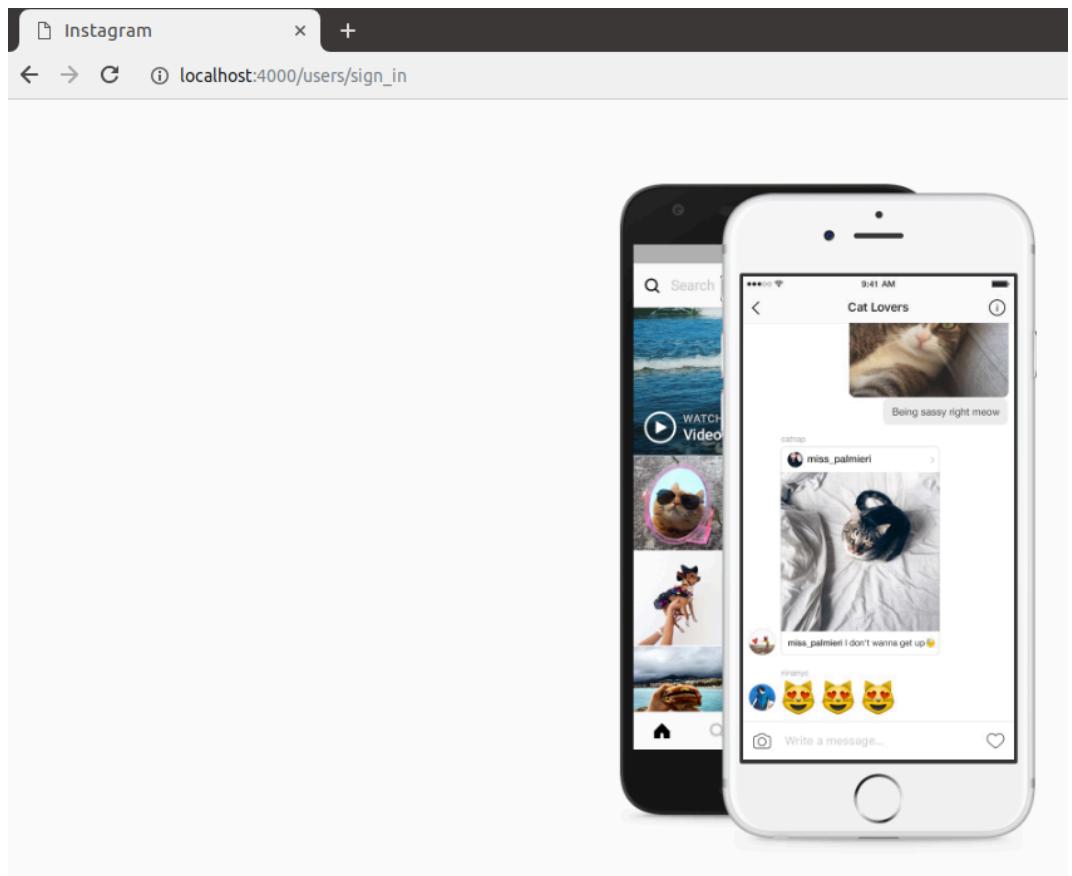
Was fällt Ihnen auf? Stichwort SCSS, CSS und «nesting»?

```
.landing-left {
  .dummy-phone {
    ...
  }
  .screen-shot {
    ...
    .item {
      ...
    }
  }
}
```



A large rectangular area for handwriting practice, featuring horizontal ruling lines. In the top right corner, there is a small icon of a pen.

Schauen wir uns das Resultat kurz im Browser an:



Welches Attribut wechselt die Bilder im «dummy-phone»?

---

---

---

Wie ist im SCSS ein wechselndes Bild im Slider benannt?

---

---

---

In welchen SCSS-Klassen oder in welcher SCSS-Klasse werden die Bilder des Sliders positioniert?

---

---

---

Welche SCSS-Klassen unterteilen die linke «dummy-phone» und später rechte «die Registrierung» der Seite?

---

---

---

Wo wird der Bildrahmen genauer der Hintergrund des «dummy-phone» definiert?

---

---

---

## Carousel <form\_for> in Login-View «sign\_in» einbauen

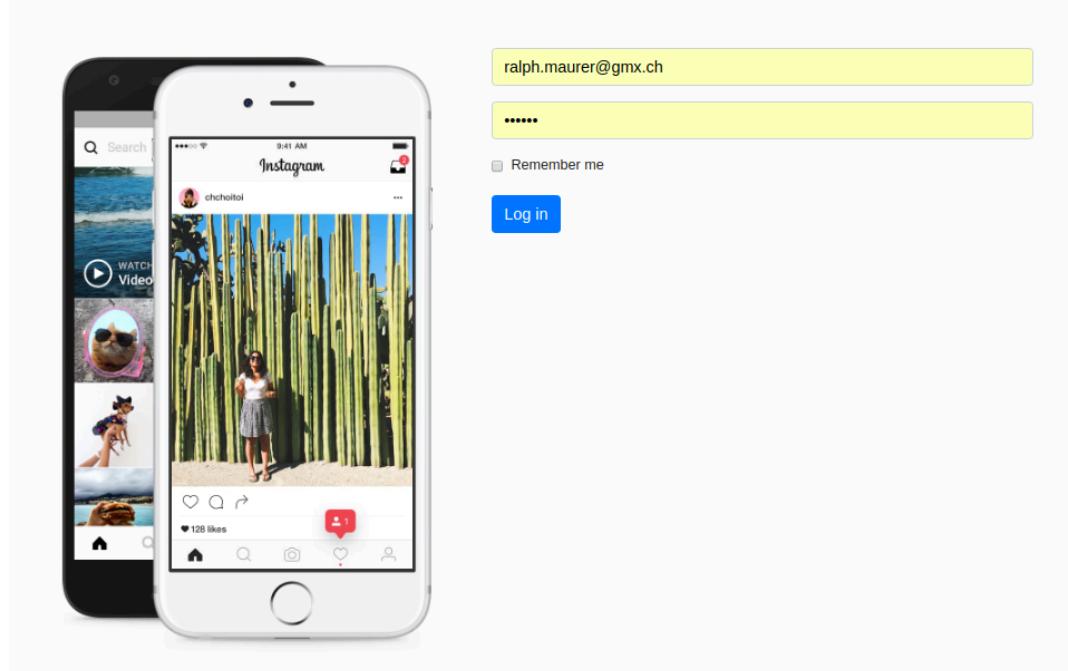
Jetzt können Sie den auskommentierten Bereich in /app/view/devise/sessions/new.html.erb wiederverwenden und anpassen. Fügen Sie den Code in

```
<div class="col-lg-6 landing-right">  
...  
</div>
```

ein und machen Sie die Anpassung mit SCSS-Klassen von Bootstrap. Die angewandten Klassen sind unter <https://getbootstrap.com/docs/4.0/components/forms/> genauer erklärt.

```
<div class="form-login box">  
  <h3 class="core-sprite brand-name-img"></h3>  
  <%= form_for(resource, as: resource_name, url: session_path(resource_name)) do |f| %>  
  
    <div class="form-group">  
      <%= f.email_field :email, autofocus: true, placeholder: "Email", class: "form-control" %>  
    </div>  
  
    <div class="form-group">  
      <%= f.password_field :password, autocomplete: "off", placeholder: "Password", class: "form-control" %>  
    </div>  
  
    <% if devise_mapping.rememberable? -%>  
      <div class="form-check">  
        <%= f.check_box :remember_me, class: "form-check-input" %>  
        <%= f.label :remember_me, class: "form-check-label" %>  
      </div>  
      <br>  
    <% end -%>  
  
    <div class="actions">  
      <%= f.submit "Log in", class: "btn btn-primary" %>  
    </div>  
    <br>  
  <% end %>  
  
</div>
```

Sie sollten nun folgende Browseransicht erhalten:



Wir beginnen unsere Authentifikation mit

```
<div class="form-login box">
    <h3 class="core-sprite brand-name-img"></h3>
```

Zu gestalten. Hierzu müssen wir .box und .brand-name-img im SCSS definieren.  
Ergänzen Sie /app/assets/stylesheets/applications.scss um folgende zwei SCSS Klassen:

```
.box {
    background-color: #fff;
    border: 1px solid #e6e6e6;
    border-radius: 1px;
    padding: 10px 0;
    margin: 0 0 10px;
}

.brand-name-img {
    background-position: 0 0;
    height: 51px;
    width: 175px;
}
```

Das gleiche muss mit der ganzen rechten Seite genauer mit der SCSS-Klasse .landing-right gemacht werden:

```
.landing-right {
    .form-login {
        width: 350px;

        form {
            padding: 0 40px;

            .btn-primary {
                width: 100%;
                line-height: 14px;
                font-size: 14px;
                font-weight: 600;
            }

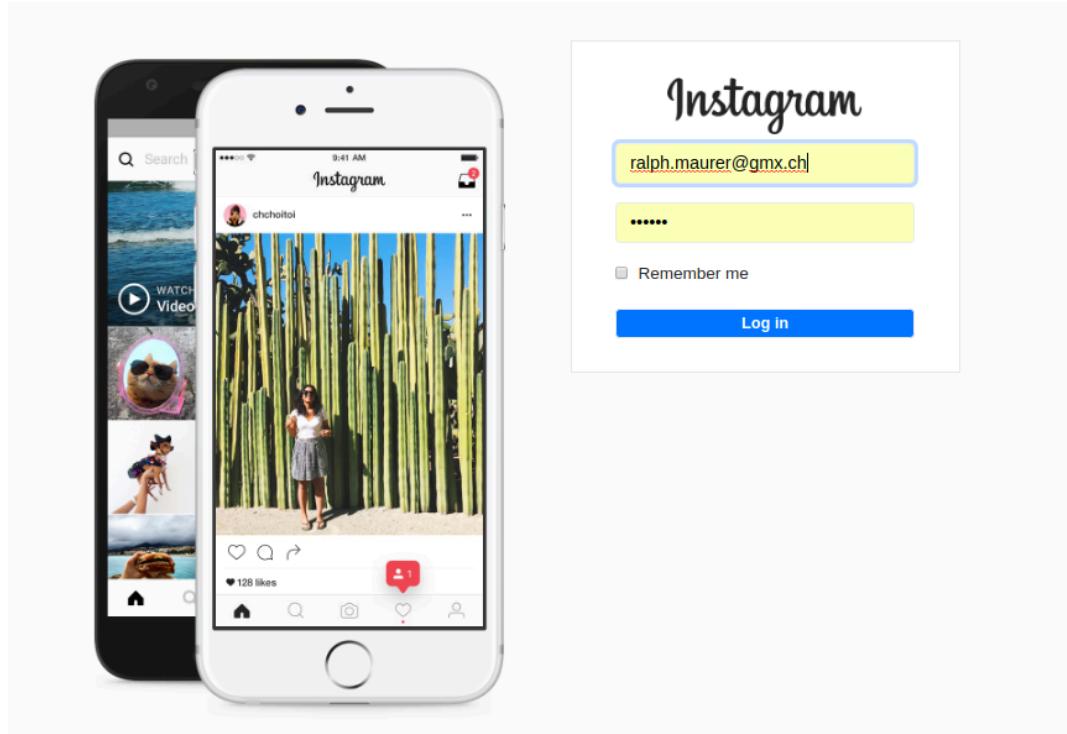
            input {
                border: 1px solid #efefef;
            }

            .form-check-label {
                padding: 0;
                font-size: 15px;
                color: #262626;
            }
        }

        .brand-name-img {
            margin: 22px auto 8px;
        }
    }

    .redirect-links {
        width: 350px;
    }
}
```

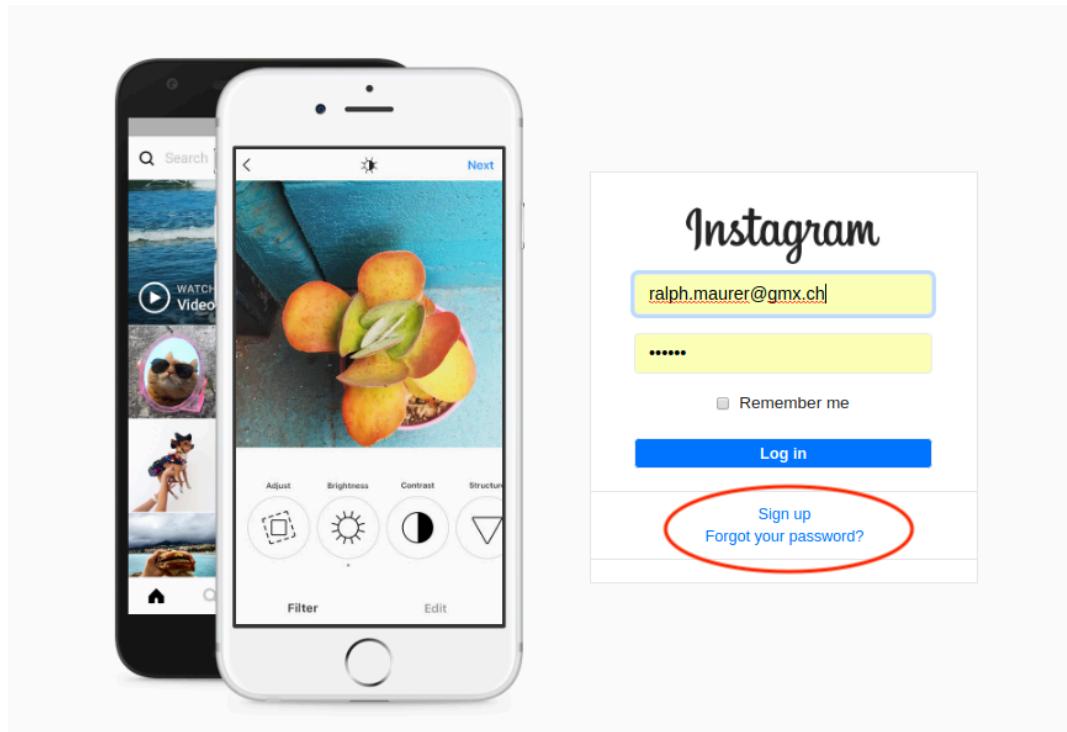
Ihre Login-View sollte nun so aussehen, wenn Sie alles richtig gemacht haben; und das Resultat lässt sich sehen:



Zentrieren wir noch die Elemente des <div>-Containers mit der Klasse .landing-right horizontal und vertikal:

```
<div class="col-lg-6 landing-right text-center d-flex align-items-center">
```

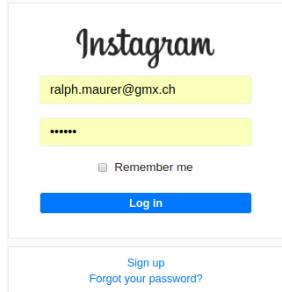
Die Links [Sign up](#) und [Forgot your password?](#) können wir als Partial-View mit Hilfe der «gem devise» einblenden.



Fügen Sie diesen Code nach dem <form\_for> ein

```
<div class="redirect-links box">
  <%= render "devise/shared/links"%>
</div>
```

Zwischen dem Formular und den Links [Sign up](#) und [Forgot your password?](#) bauen wir eine Begrenzung mit getrennten <div>-Containern ein:



The screenshot shows a login form for Instagram. At the bottom right of the form area, there is a separate white rectangular box containing two links: "Sign up" and "Forgot your password?".

```
<div class="col-lg-6 landing-right text-center d-flex align-items-center">
  <div>
    <div class="form-login box">
      <h3 class="core-sprite brand-name-img"></h3>
      <%= form_for(resource, as: resource_name, url: session_path(resource_name)) do |f| %>
        <div class="form-group">
          <%= f.email_field :email, autofocus: true, placeholder: "Email", class: "form-control" %>
        </div>

        <div class="form-group">
          <%= f.password_field :password, autocomplete: "off", placeholder: "Password", class: "form-control" %>
        </div>

        <% if devise_mapping.rememberable? -%>
          <div class="form-check">
            <%= f.check_box :remember_me, class: "form-check-input" %>
            <%= f.label :remember_me, class: "form-check-label" %>
          </div>
          <br>
        <% end -%>

        <div class="actions">
          <%= f.submit "Log in", class: "btn btn-primary" %>
        </div>
        <br>
      <% end %>
    </div>
    <div class="redirect-links box">
      <%= render "devise/shared/links" %>
    </div>
  </div>
</div>
```

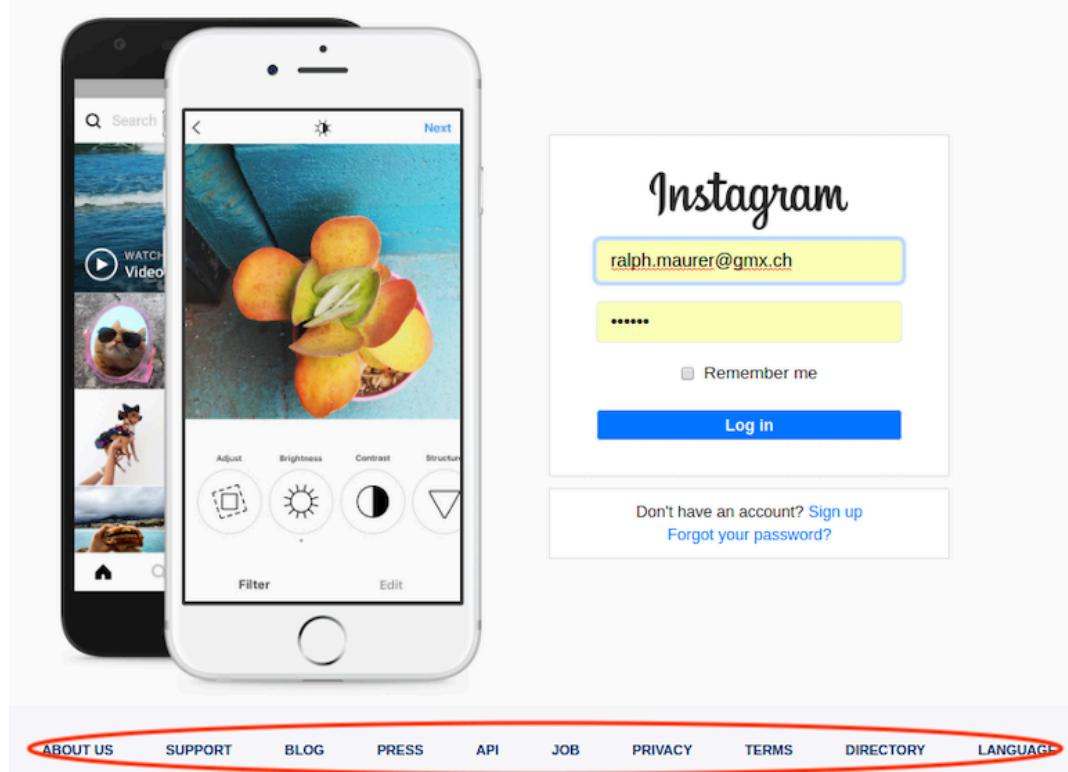
Damit die Links einwenig passender aussehen, ergänzen wir die «gem devise» Seite app/views/devise/shared/\_links.html.erb

```
new.html.erb         links.html.erb       application.css
1  <%- if controller_name != 'sessions' %>
2    <%= "Have an account? "%>
3    <%= link_to "Log in", new_session_path(resource_name) %><br />
4  <% end -%>
5
6  <%- if devise_mapping.registerable? && controller_name != 'registrations' %>
7    <%= "Don't have an account? "%>
8    <%= link_to "Sign up", new_registration_path(resource_name) %><br />
9  <% end -%>
10
11 <%- if devise_mapping.recoverable? && controller_name != 'passwords' && controller_name != 'registrations' %>
12   <%= link_to "Forgot your password?", new_password_path(resource_name) %><br />
13 <% end -%>
14
15 <%- if devise_mapping.confirmable? && controller_name != 'confirmations' %>
16   <%= link_to "Didn't receive confirmation instructions?", new_confirmation_path(resource_name) %><br />
17 <% end -%>
18
19 <%- if devise_mapping.lockable? && resource_class.unlock_strategy_enabled?(:email) && controller_name != 'unlocks' %>
20   <%= link_to "Didn't receive unlock instructions?", new_unlock_path(resource_name) %><br />
21 <% end -%>
22
23 <%- if devise_mapping.omniauthable? %>
24   <%- resource_class.omniauth_providers.each do |provider| %>
25     <%= link_to "Sign in with #{OmniAuth::Utils.camelize(provider)}", omniauth_authorize_path(resource_name, provider) %><br />
26   <% end -%>
27 <% end -%>
```

## <footer> in in Login-View «sign\_in» einbauen

Erstellen Sie eine Partial-View für den footer. Rendern Sie den footer einmalig, so dass er trotzdem in jeder View zu sehen ist.

Das Endresultat muss wie folgt aussehen:



HTML-Vorschlag für den footer:

```
<footer>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <button class="navbar-toggler" type="button" data-
    toggle="collapse" data-target="#navbar">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbar">
      <div class="navbar-nav mx-auto">
        <a href="" class="nav-item nav-link" href="#">ABOUT US</a>
        <a href="" class="nav-item nav-link" href="#">SUPPORT</a>
        <a href="" class="nav-item nav-link" href="#">BLOG</a>
        <a href="" class="nav-item nav-link" href="#">PRESS</a>
        <a href="" class="nav-item nav-link" href="#">API</a>
        <a href="" class="nav-item nav-link" href="#">JOB</a>
        <a href="" class="nav-item nav-link" href="#">PRIVACY</a>
        <a href="" class="nav-item nav-link" href="#">TERMS</a>
        <a href="" class="nav-item nav-link" href="#">DIRECTORY</a>
        <a href="" class="nav-item nav-link" href="#">LANGUAGE</a>
      </div>
    </div>
  </nav>
</footer>
```

CSS-Vorschlag für den footer:

```
footer .nav-item {  
    color: #003569 !important;  
    font-size: 12px;  
    font-weight: bold;  
}
```

1. In welcher Datei haben Sie den Code für den footer als Partial-View erfasst?
2. Wo wird die Datei mit dem footer gerendert?
3. Wie lautet der Befehl zum rendern?



## Login-View «sign\_up» anpassen

Der ganze Code von der Login-View «sign\_in» soll wiederverwendet werden. Deshalb erstellen Sie nun eine Partial-View mit dem «dummy-phone» Teil aus /app/view/devise/sessions/new.html.erb. Benennen Sie die Partial-View \_dummy\_phone.html.erb.

1. In welcher Datei haben Sie den Code für den «dummy-phone» erfasst?
2. Wo wird die Datei mit dem «dummy-phone» für die Login-View «sign\_in» und «sign\_up» gerendert?
3. Wie lautet der Befehl zum rendern?



Die Login-View «sign\_up» sollte nun so aussehen:

The image shows a smartphone on the left displaying the Instagram mobile application. The screen shows a feed of photos and a camera viewfinder. To the right of the phone is a separate sign-up form for Instagram. The form has a title 'Instagram' and a sub-instruction 'Sign up to see photos and videos from your friends.' It contains four input fields: 'Your name', 'Email', 'Password', and 'Password confirmation'. Below these fields is a large blue 'Sign up' button. At the bottom of the form, there is a note: 'By signing up, you agree to our [Term & Privacy Policy](#)'. At the very bottom, there is a link: 'Have an account? [Log in](#)'. At the very bottom of the entire composite image, there is a navigation bar with links: ABOUT US, SUPPORT, BLOG, PRESS, API, JOB, PRIVACY, TERMS, DIRECTORY, and LANGUAGE.

Damit der Text «*Sign up to see photos and videos from your friends.*» oder «*By signing up, you agree to our Term & Privacy Policy.*» leicht eingefärbt aussieht, wird folgende SCSS Klasse definiert

```
.light-color {  
    color: #999;  
}
```

und hier angewandt:

```
<h5 class="light-color"><strong>Sign up to see photos and <br>  
videos from your friends.</strong></h5>
```

```
<p class="light-color">By signing up, you agree to our  
<br><strong>Term & Privacy Policy.</strong></p>
```

Des Weiteren sorgen folgende SCSS Elemente für ein Fade-Effekt im Slider innerhalb des dummy-phone:

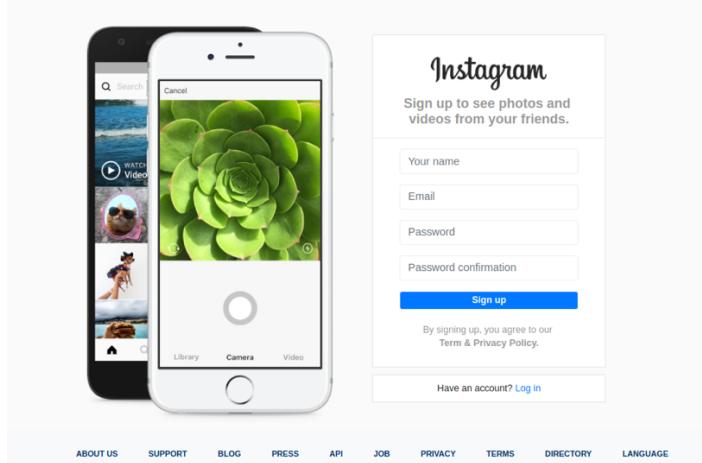
```
.carousel-fade {  
    .carousel-item.active {  
        opacity: 1;  
        display: block;  
    }  
  
    .carousel-item {  
        transition: opacity 0.6s ease-in-out;  
        opacity: 0;  
        position: relative;  
        display: block;  
    }  
  
    .carousel-item:not(.active) {  
        position: absolute;  
        top: 0px;  
    }  
}
```

Sie können den rotmarkierten Teil aus der Login-View «sign\_in» kopieren:

```
1  <div class="container">  
2    <div class="row">  
3      <div class="col-lg-6 landing-left">  
4        <%= render "devise/shared/dummy_phone"%>  
5      </div>  
6      <div class="col-lg-6 landing-right text-center d-flex align-items-center">  
7        <div>  
8          <h2>Sign up</h2>  
9  
10         <%= form_for(resource, as: resource_name, url: registration_path(resource_name)) do |f| %>  
11           <%= devise_error_messages! %>  
12  
13           <div class="field">  
14             <%= f.label :email %><br />  
15             <%= f.email_field :email, autofocus: true, autocomplete: "email" %>  
16           </div>  
17  
18           <div class="field">  
19             <%= f.label :password %>  
20             <% if @minimum_password_length %>  
21               <em>(<%= @minimum_password_length %> characters minimum)</em>  
22             <% end %><br />  
23             <%= f.password_field :password, autocomplete: "new-password" %>  
24           </div>  
25  
26           <div class="field">  
27             <%= f.label :password_confirmation %><br />  
28             <%= f.password_field :password_confirmation, autocomplete: "new-password" %>  
29           </div>  
30  
31           <div class="actions">  
32             <%= f.submit "Sign up" %>  
33           </div>  
34         <% end %>  
35  
36         <%= render "devise/shared/links" %>  
37  
38     </div>  
39   </div>  
40 </div>
```

iet-gibb  
AB151-03  
Seite 15/16

Passen Sie nun /app/view/devise/registrations/new.html.erb so an,  
dass Sie der Vorgabe entspricht:



# Auftrag: Quicknote AB151-03

Alle Aufträge des Typen Quicknote sind Bestandteil der Bewertung. Erstelle Sie eine Quicknote von max. 4 A4-Seiten und geben Sie diese in PDF Form gemäss Zeitangabe der Lehrperson ab. Angedacht sind max. 4 Lektionen seit Abgabe dieses Dokumentes.

**Titel der Quicknote:** Klasse\_Name\_Vorname\_QN\_AB151-03.zip

**Das Archiv beinhaltet:**

- › **Quicknote:** Klasse\_Name\_Vorname\_QN\_AB151-03.pdf
- › **Code:** application.scss, .../sessions/new.html.erb, .../registrations/new.html.erb, \_footer.html.erb, \_navbar.html.erb, \_links.html.erb aus devise

Was beinhaltet die Quicknote AB151-03?

## Zusammenfassung AB151-03

Die Quicknote soll eine kurze prägnante Zusammenfassung des Dokuments AB151-03 beinhalten und einen Überblick über die vorgestellten Techniken, Methoden und Konzepte beinhalten.

Achten Sie darauf, dass alle wesentliche Themen vollständig erwähnt und wenn möglich in Zusammenhang gebracht sind.

Beantworten Sie folgende Fragen zu Anwendungszweck und Selbstreflexion:

### Anwendungszweck:

1. Wie und wo können die vorgestellten Techniken, Methoden und Konzepte in einer Rails-App angewandt werden?
2. Was sind Vorteile und was sind Nachteile?

### Selbstreflexion:

3. Was habe ich gelernt?
4. Was hat mich behindert?
5. Was habe ich nicht verstanden?
6. Was kann ich beim Studium besser machen?

### Abschliessende Reflexion über das Gelernte:

Schreiben Sie eine persönliche Schlussfolgerung über den Lerninhalt in 2-3 Sätzen.