



**MODUL 151**

TEIL 3: INSTAGRAM

AUTHENTICIATION VIEW

Ralph Maurer

# Inhaltsverzeichnis AB151-03

## Modul 133: Instagram mit Rails bauen

Teil 3: Instagram Authentication View

[Inhaltsverzeichnis AB151-03 1](#_Toc15962)

[Modul 133: Instagram mit Rails bauen 1](#_Toc15963)

[Attribut Benutzernamen erfassen 2](#_Toc15964)

[Login-View «sign\_in» anpassen 4](#_Toc15965)

[Menu ausblenden bei logout 4](#_Toc15966)

[“dummy-phone” aus Carousel in Login-View «sign\_in» einbauen 5](#_Toc15967)

[Carousel <form\_for> in Login-View «sign\_in» einbauen 8](#_Toc15968)

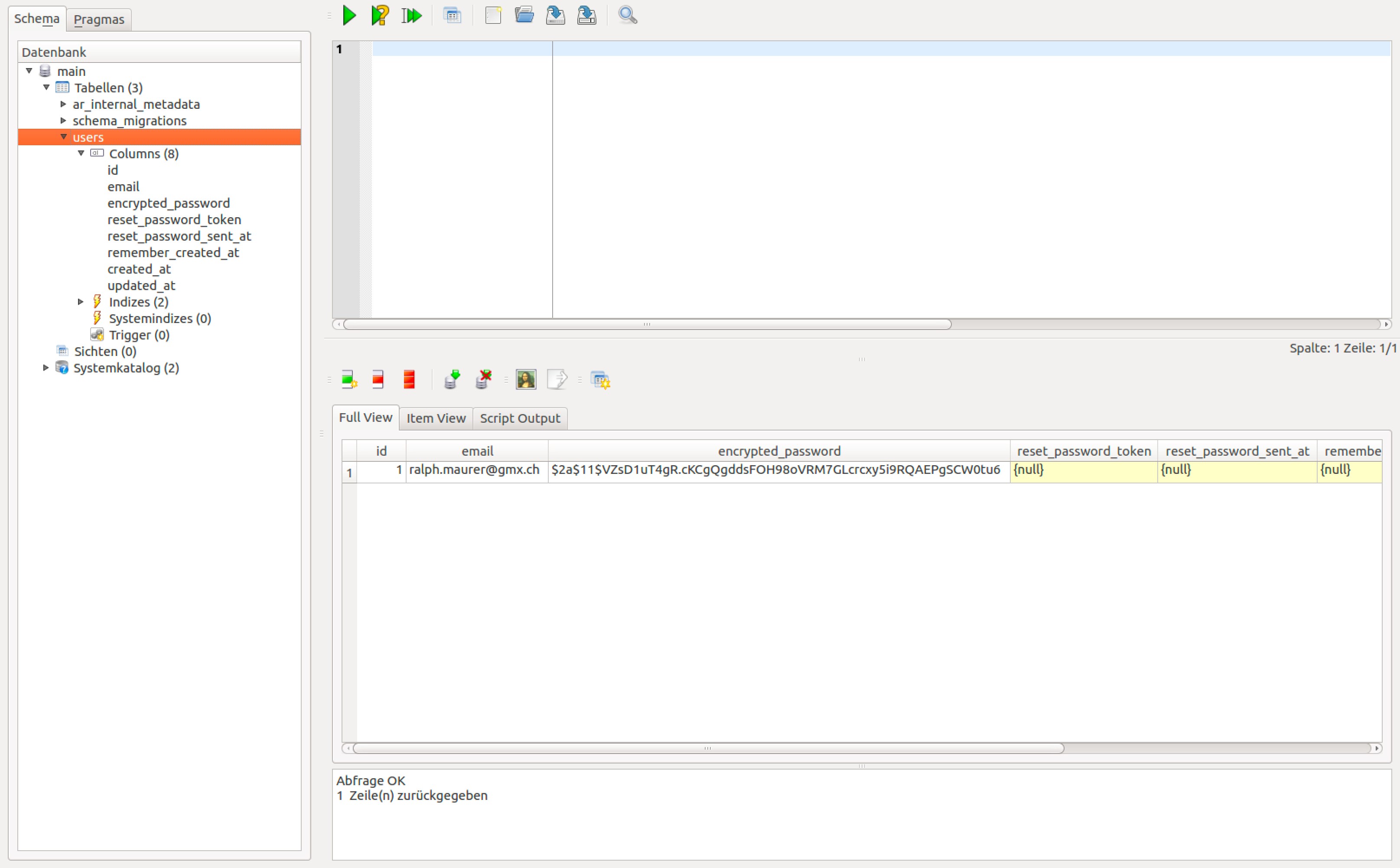
[<footer> in in Login-View «sign\_in» einbauen 12](#_Toc15969)

[Login-View «sign\_up» anpassen 13](#_Toc15970)

[Auftrag: Quicknote AB151-03 16](#_Toc15971)

# Attribut Benutzernamen erfassen

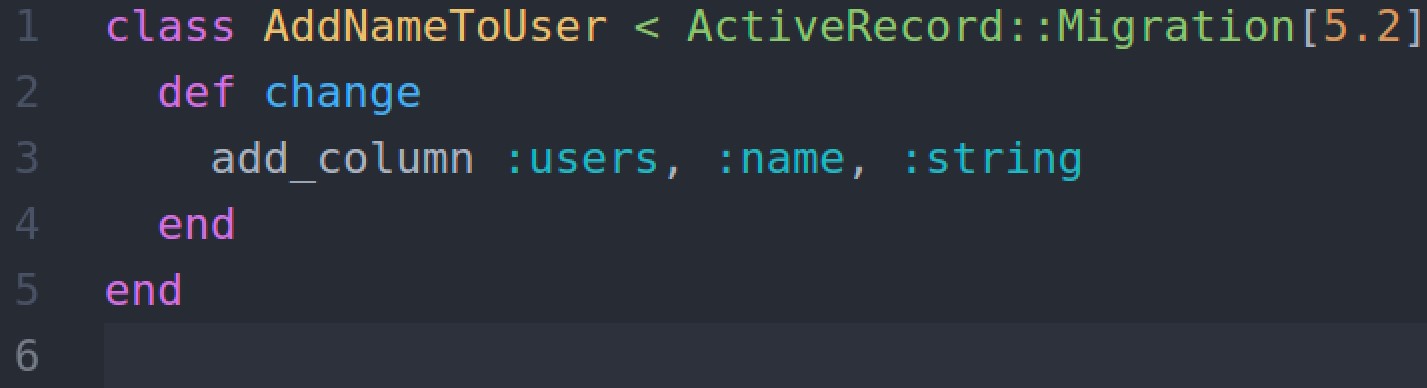
Betrachten wir die Datenbank, die «devise gem» generiert hat:



|  |
| --- |
| 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.



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:

|  |
| --- |
|  |
| Rails generate migration AddNameToUser |
|  |
|  |

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

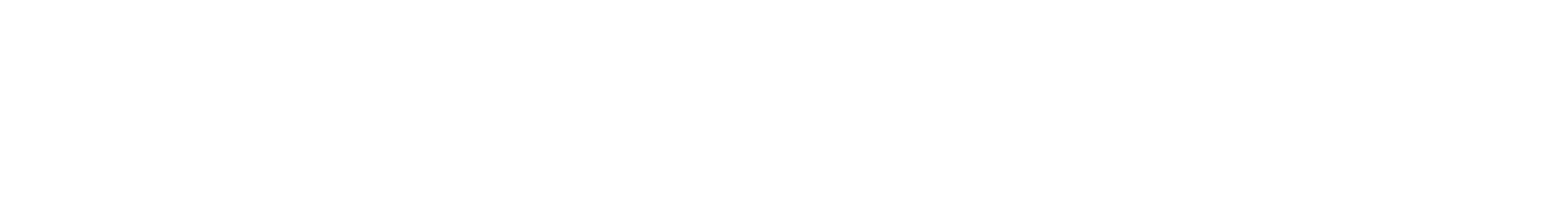
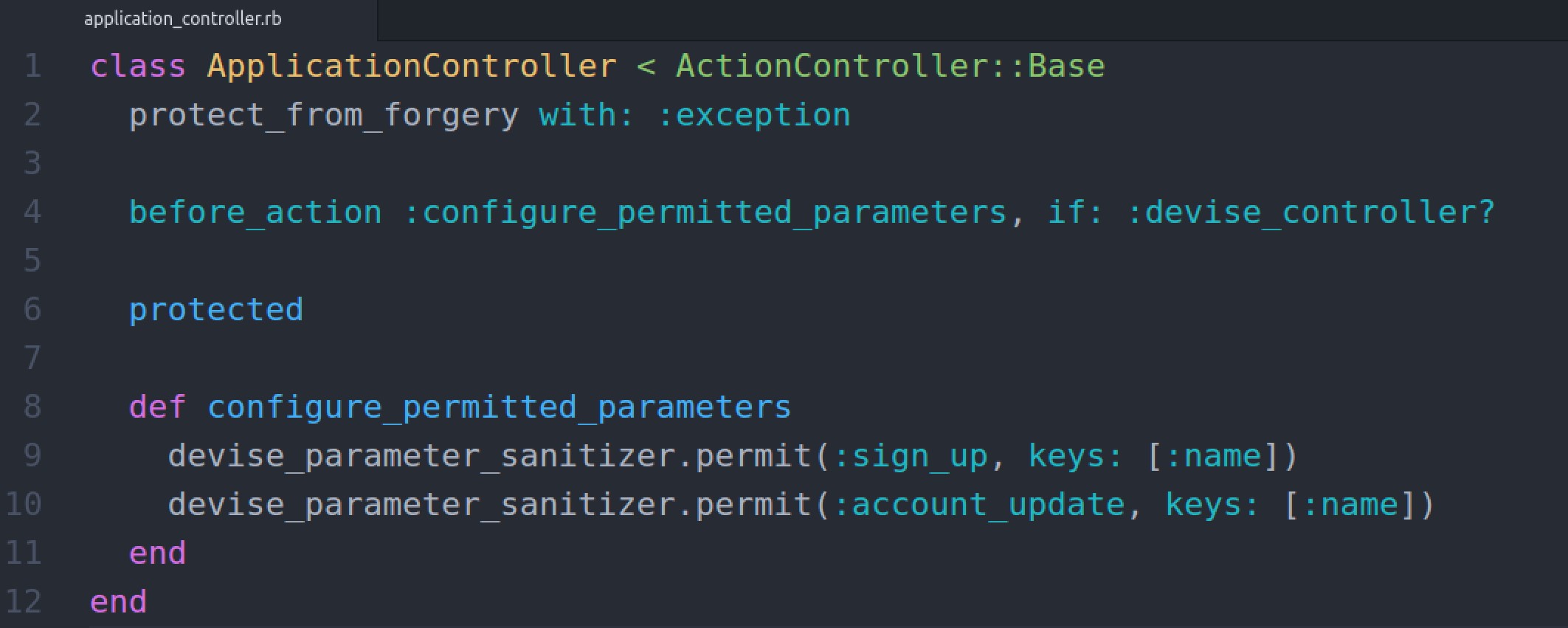
› ob der Benutzername angegeben wurde

› sowie die max. Länge nicht 50 Zeichen überschreitet.

Notieren Sie die Validierung:

validates :name, confirmation: { minimum: 1, maximum: 50}

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

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

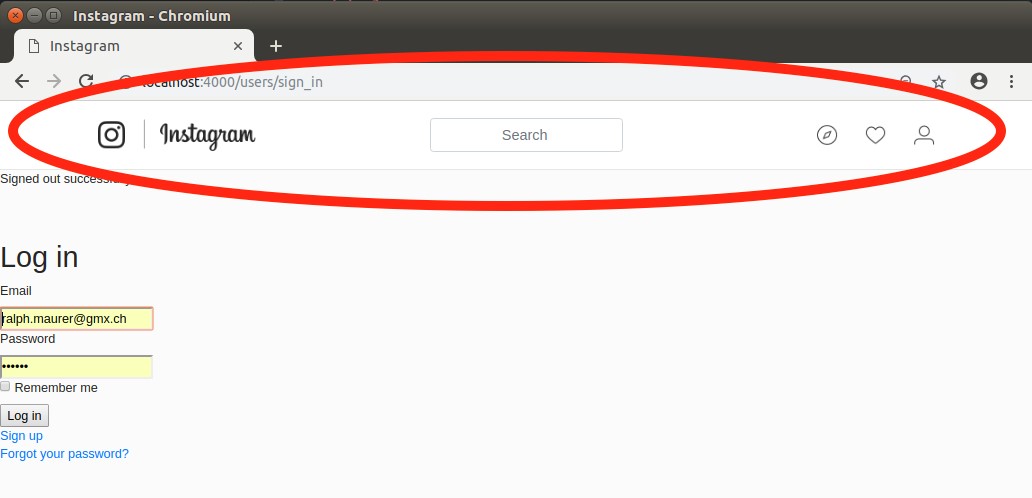
Sie können alles testen indem Sie den Rails Server neustarten. Mögliche Syntaxfehler werden genau beschrieben und können korrigiert werden.

# Login-View «sign\_in» anpassen

Die Login-View «sign\_in» ist wie bekannt unter http://localhost:4000/users/sign\_in 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:

|  |
| --- |
|  |
| Application.html.erb |
| <% if current\_user %>  <%=render 'shared/navbar'%>  <% end %> |
|  |

Wir passen nun die Login-View an, was einiges an Schreibaufwand verursacht. Öffnen Sie

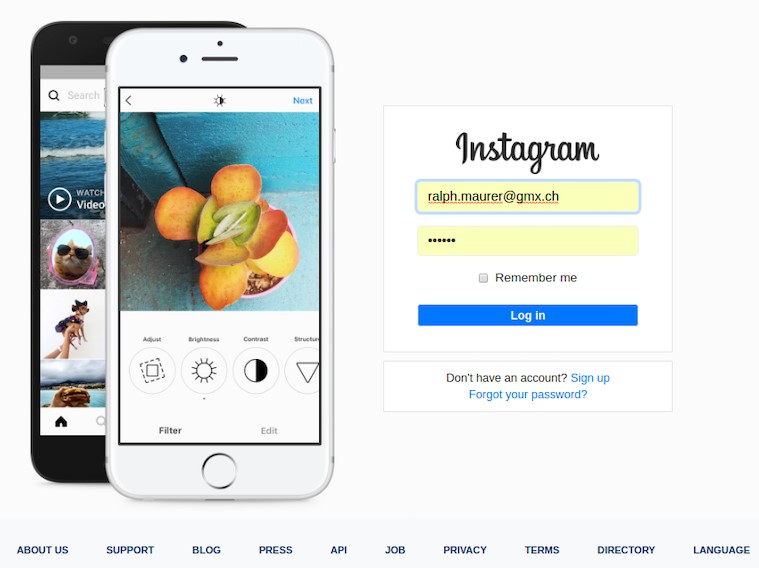
/app/view/devise/sessions/new.html.erb

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:



Ergänzen Sie die SCSS Datei

/app/assets/stylesheets/applications.scss



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

.landing-left { .dummy-phone { ...

}

.screen-shot { ...

.item { ...

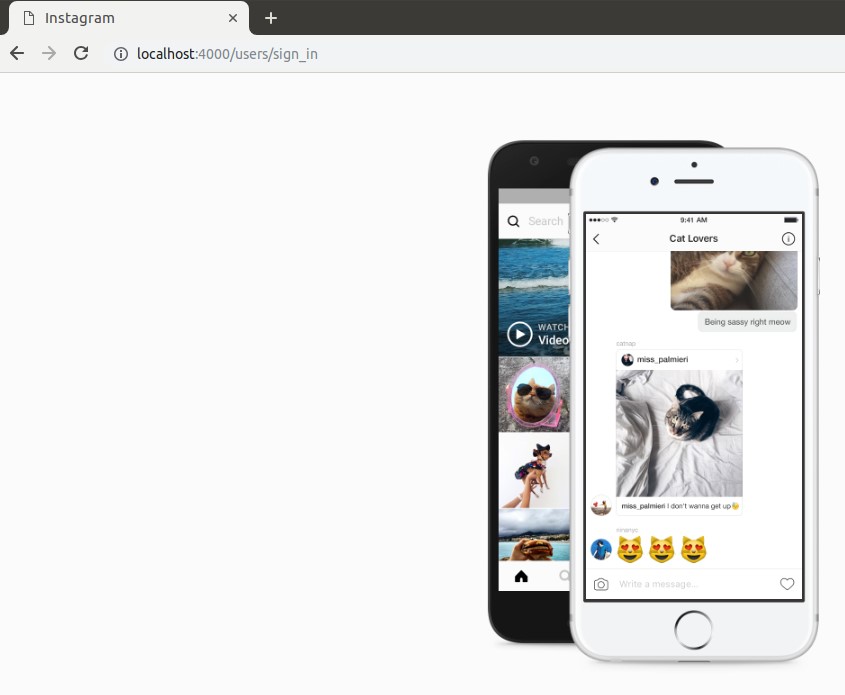
}

}

}

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

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

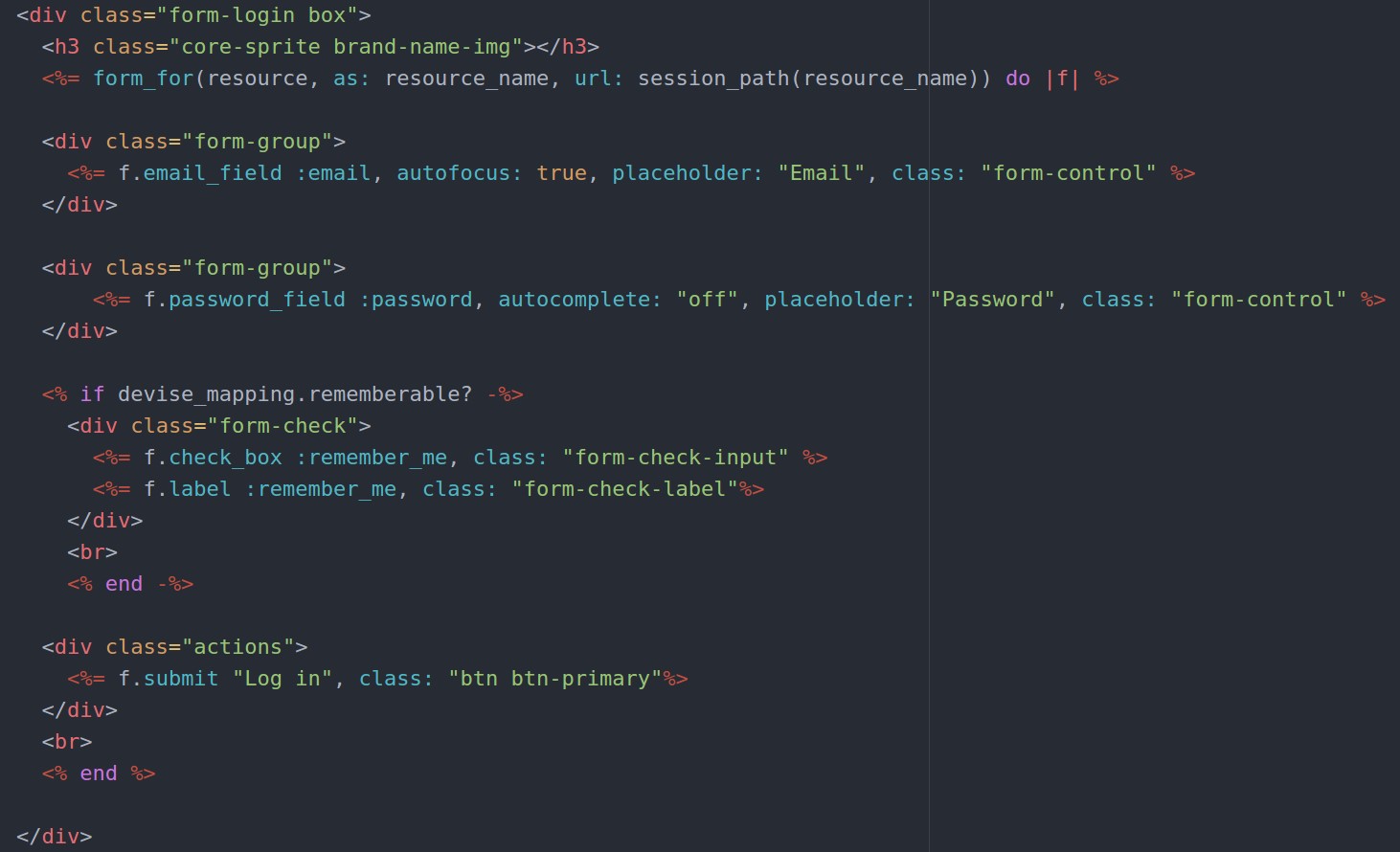
wiederverwenden und anpassen. Fügen Sie den Code in

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

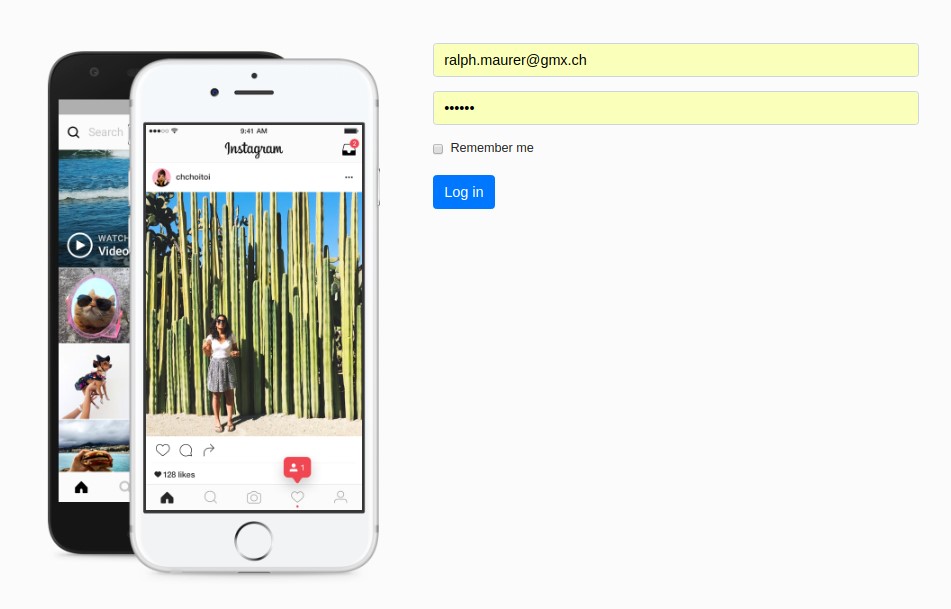
…

</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.



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

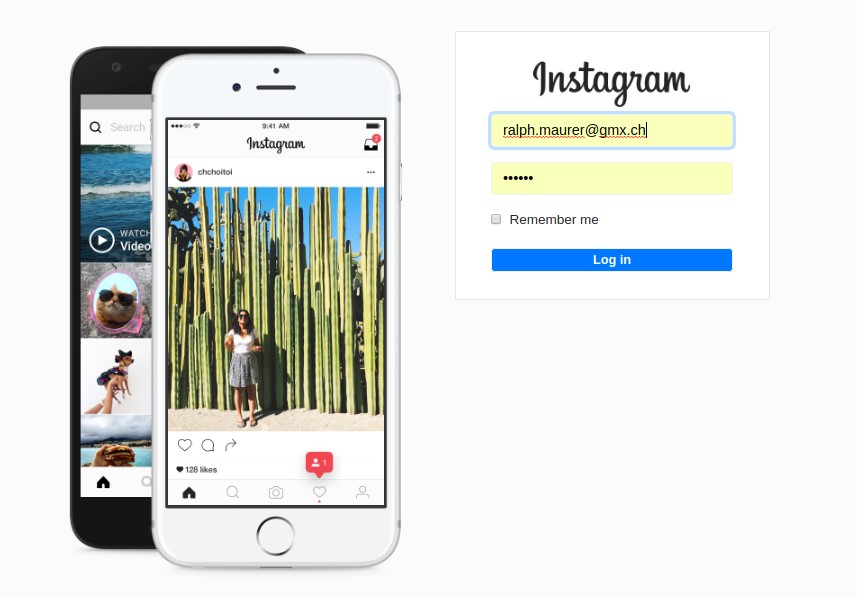
Klassen:



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



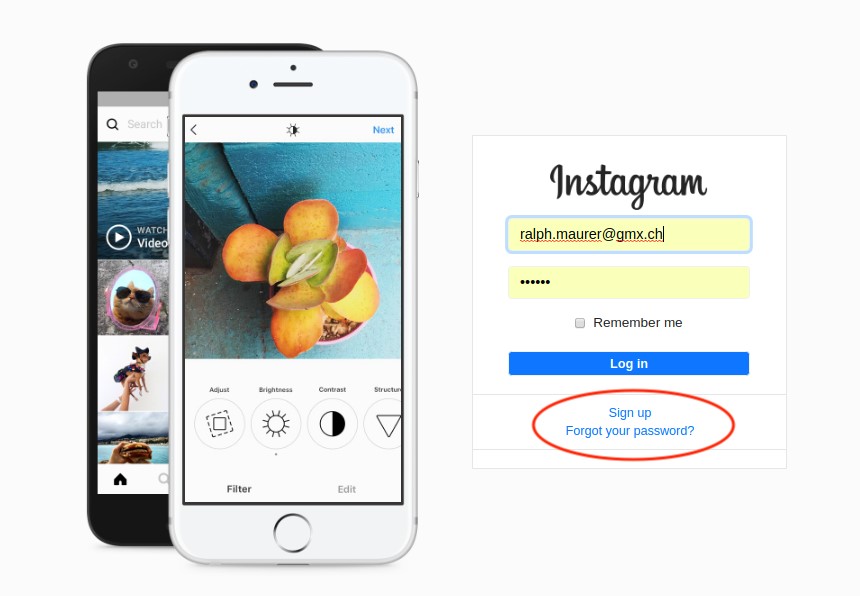
Ihre Login-View sollte nun so aussehen, wenn Sie alles richtig gemacht haben; und das Resultat lässt sich sehen:



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-itemscenter">

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

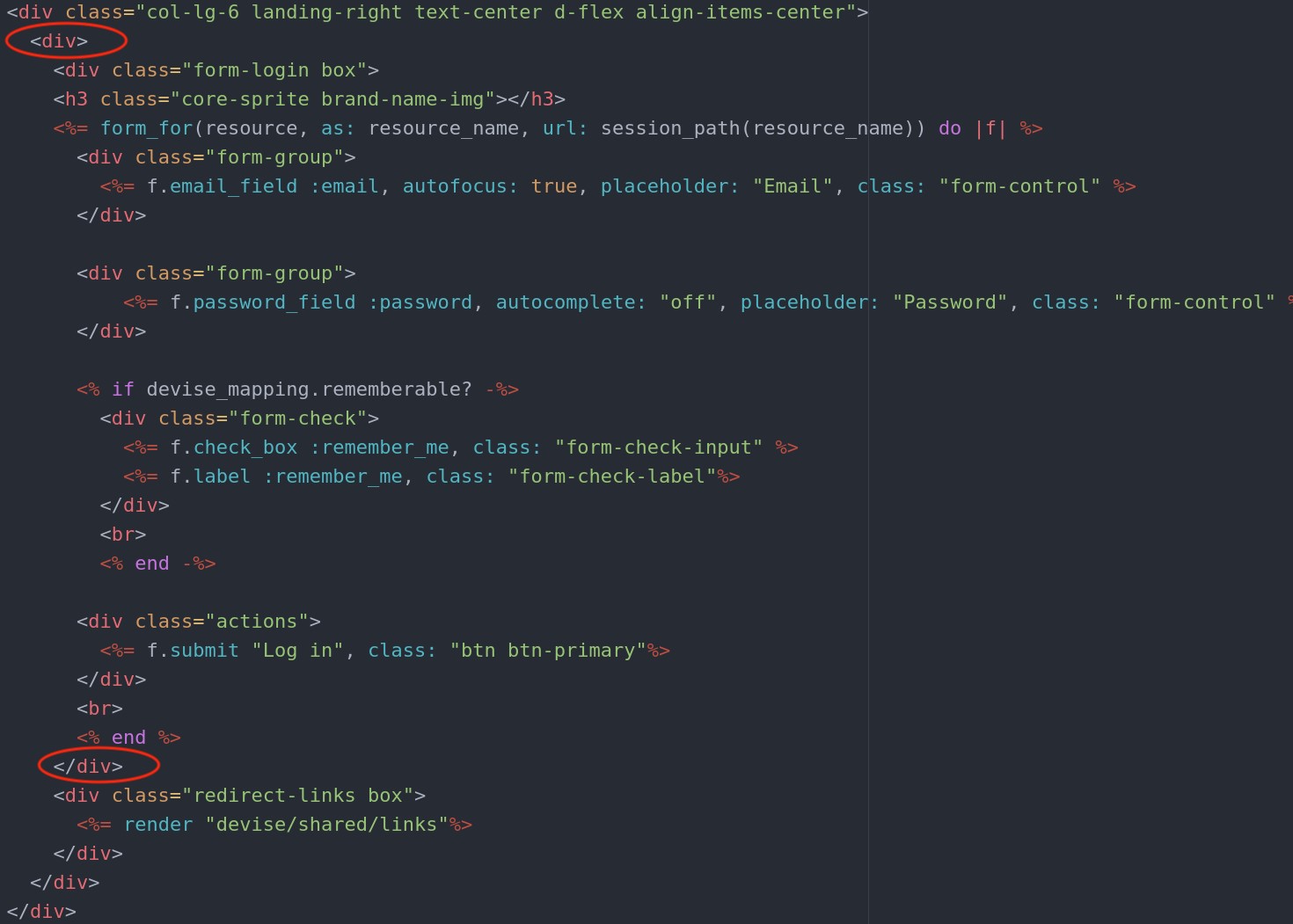
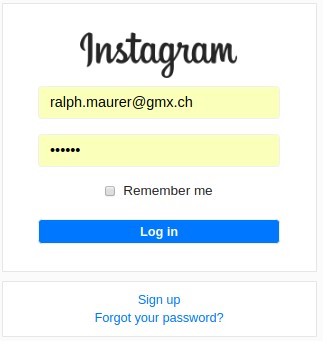


Fügen Sie diesen Code nach dem <form\_for> ein

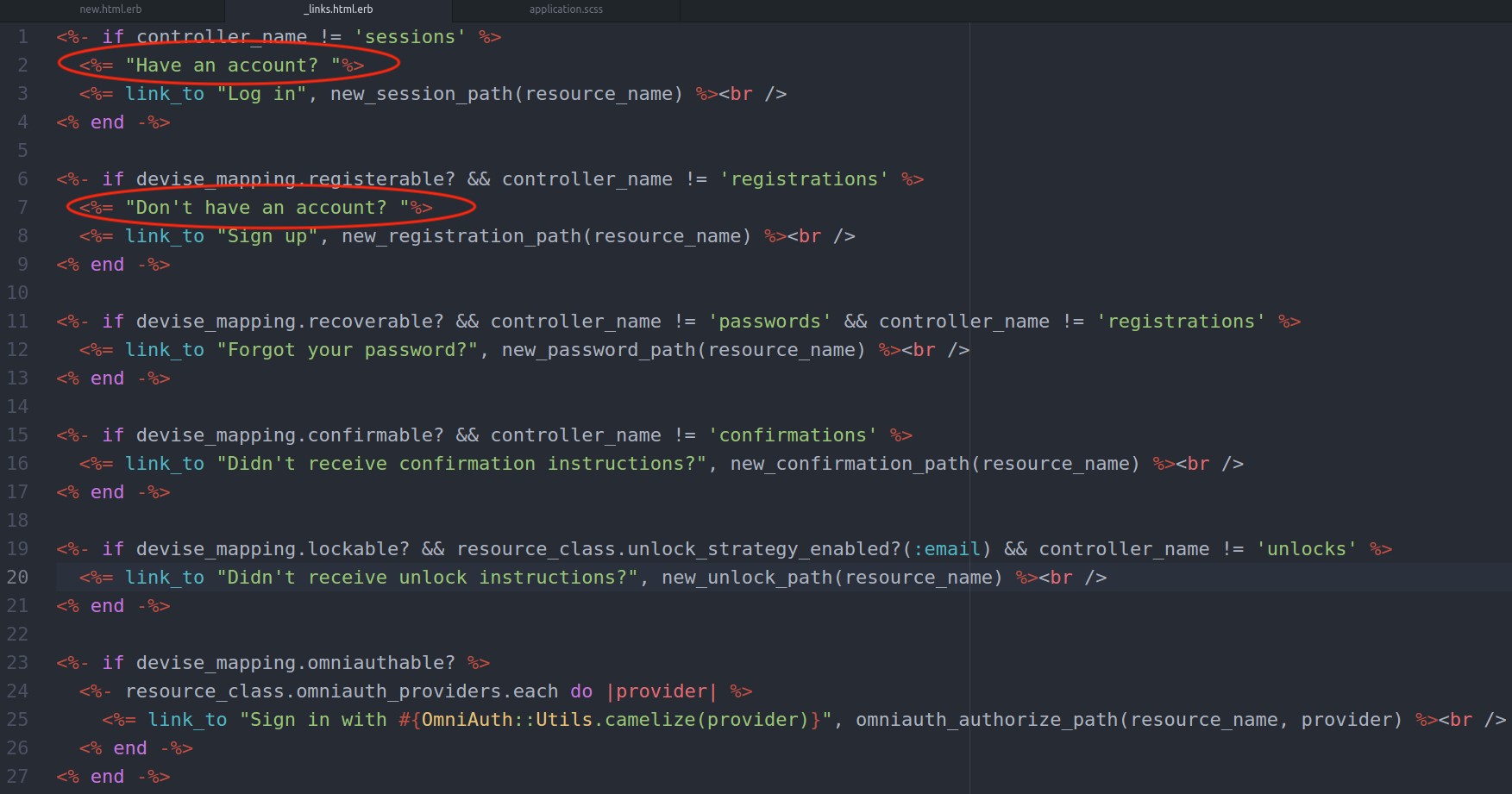
<div class="redirect-links box">

<%= 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:

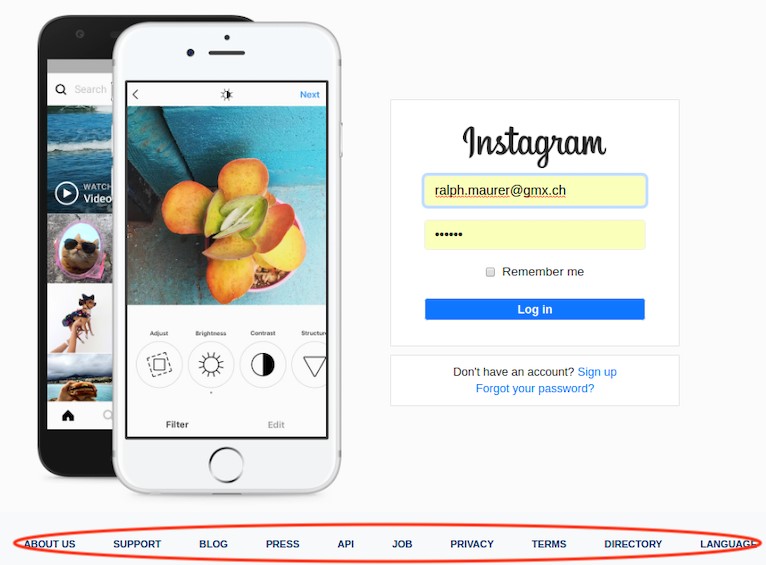


Damit die Links einwenig passender aussehen, ergänzen wir die «gem devise» Seite app/views/devise/shared/\_links.html.erb



## <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" datatoggle="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

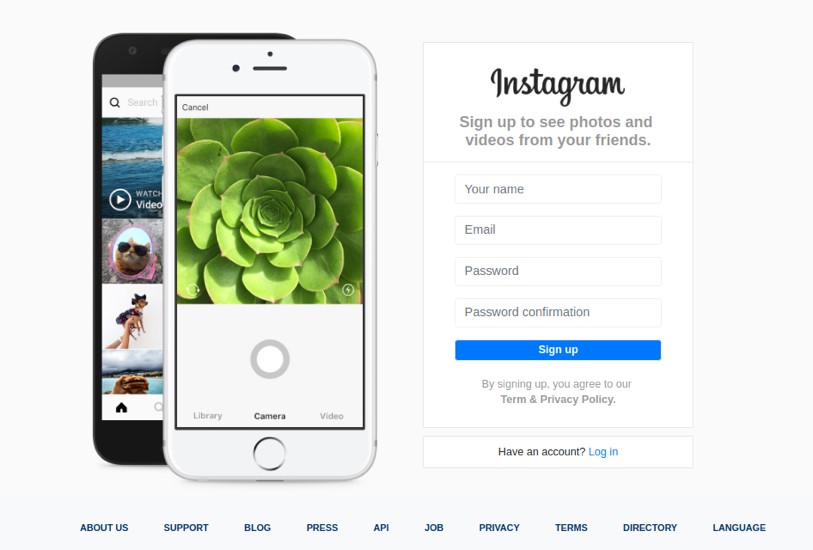
Sie nun eine Partial-View mit dem «dummy-phone» Teil aus

/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?

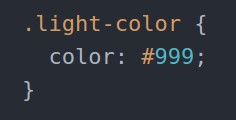
|  |
| --- |
|  |
|  |
|  |
|  |

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



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



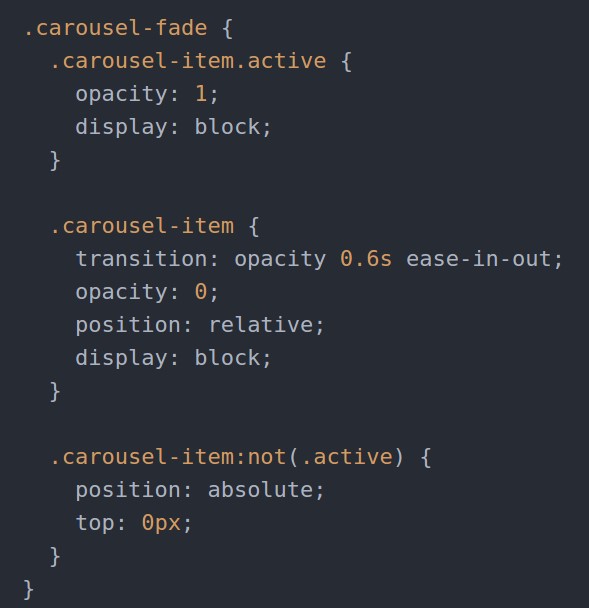
und hier angewandt:

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

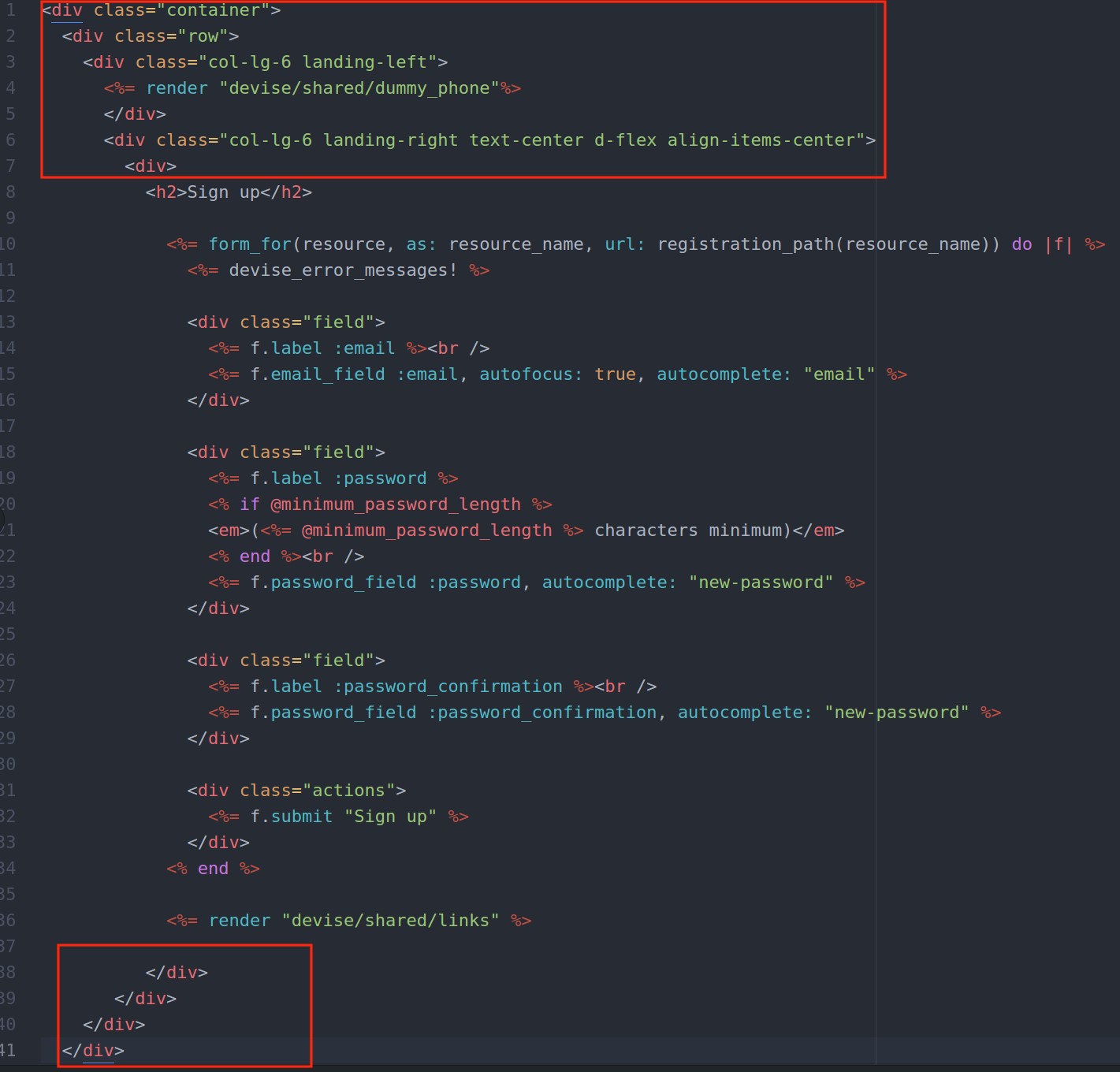
<p class="light-color">By signing up, you agree to our

<br><strong>Term &amp; Privacy Policy.</strong></p>

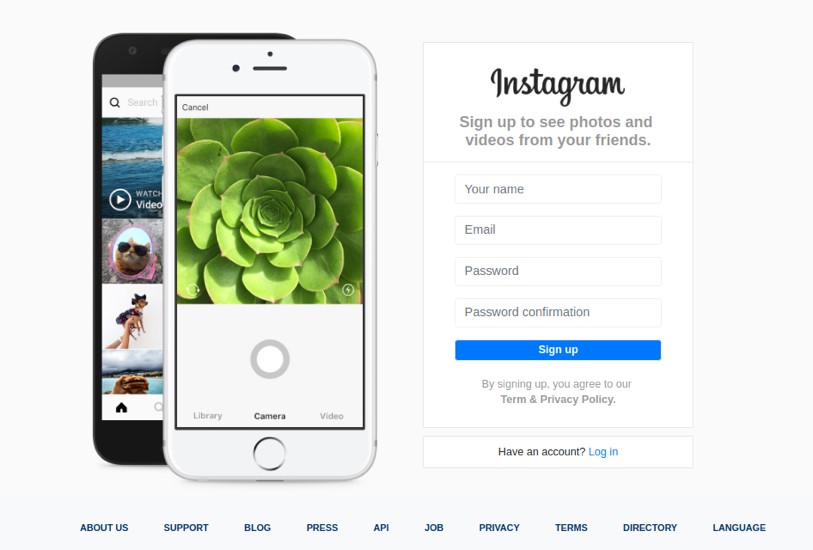
Des Weiteren sorgen folgende SCSS Elemente für ein Fade-Effekt im Slider innerhalb des dummy-phone:



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



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:**

› **Quichknote:** *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:

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

Abschliessende Reflexion über das Gelernte:

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