



MODUL 151

TEIL 4: INSTAGRAM NOTIFICATION USER PROFIL PAGE

Ralph Maurer

Inhaltsverzeichnis AB151-04

Modul 133: Instagram mit Rails bauen

Teil 4: Instagram User Profile Page

Inhaltsverzeichnis AB151-04.....	1
Modul 133: Instagram mit Rails bauen.....	1
Flash in Rails.....	2
Fancy Flash in Rails mit toastr.....	3
Aufgabe: Error Rails flash bei falscher Benutzerregistration	5
User Profile Page	6
Model User erweitern	6
routes.rb	6
users_controller.rb.....	7
show-View mit Bootstrap Modal	8
Gravatar für User Photo in show-View	8
link_to Profile Page.....	14
Edit User Page	15
Auftrag: Quicknote AB151-04.....	18

Flash in Rails

Rails bietet die Möglichkeit temporäre Variablen, normalerweise Meldungen (notification) zwischen Interaktionen (vor allem Actions), weiterzugeben. Alles was Sie im Flash ablegen, wird der nächsten Aktion übergegeben und dann wieder gelöscht. Dies ist eine grossartige Möglichkeit, Benachrichtigungen und Warnungen auszuführen, z. B. von «gem devise».

Nachdem Sie sich in unserer Rails App angemeldet haben, erhalten Sie eine Meldung mit dem Text "Signed in successfully." und verschwindet, wenn Sie zu einer anderen View wechseln oder einen Browserrefresh durchführen.

Nachrichten können unterschiedliche Typen haben:

- › success
- › notice
- › alert
- › error

Jede notification besteht aus einem Typen und einer Textnachricht.

In der Datei app/views/layouts/application.html.erb werden Flash wie folgt ausgeben:

```
<p class="notice"><%= notice %></p>
<p class="alert"><%= alert %></p>
```

Diese Schreibweise ist gleich wie

```
<p class="notice"><%= flash[:notice] %></p>
<p class="alert"><%= flash[:alert] %></p>
```

oder auch

```
<p class="notice"><%= flash['notice'] %></p>
<p class="alert"><%= flash['alert'] %></p>
```

Schauen Sie sich folgende Videos an:

sh-modules/iet-151/03_Arbeitsblaetter/AB151-04_flash_1.mp4
und AB151-04_flash_2.mp4.

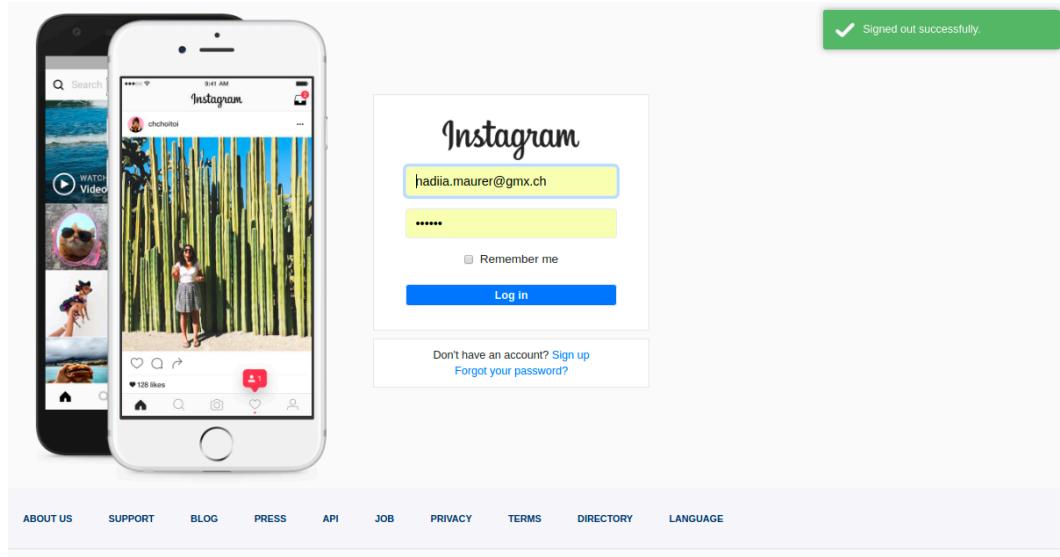
Fancy Flash in Rails mit toastr

Hier können Sie sich in die «gem toastr» vertiefen:

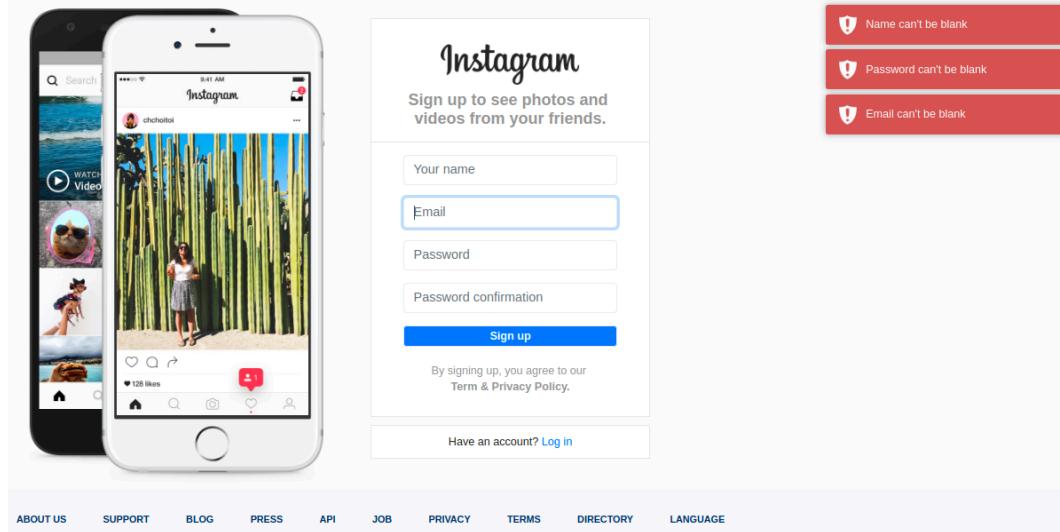
<https://github.com/tylernannon/toastr-rails>

toastr ist eine coole gem um flash notification in Rails zu gestalten:

Statt eine einfache Textmessage kommen die Nachrichten je nach Typ in einer entsprechenden Farbe oben rechts. Hier das Beispiel beim ausloggen:



oder hier bei einer leeren Benutzerregistration



Installieren Sie toastr:

- › Im gemfile
 `gem 'toastr-rails'`
 eintragen
- › In app/assets/javascripts/application.js
 `//= require toastr`
 eintragen
- › In app/assets/stylesheets/application.scss
 `@import "toastr";`
 eintragen

Führen Sie im Terminal den Befehl `bundle` aus und starten Sie den Server neu.

Schreiben Sie in die Datei app/views/layouts/application.html.erb folgenden Code und beschreiben Sie was passiert:

```
<script type="text/javascript">
    toastr.success("Wow, a success")
</script>
```

Sobald man die Seite aufruft, wird ein Toast angezeigt mit dem Text

"Wow, a success". Dieses geht nach ca 5sec wieder weg.

Da wir nie wissen, ob es mehrere Rails Flash Notification nach einer Action gibt, können wir diese iterieren und z.B. alle ausgeben:

```
<script type="text/javascript">
    <% flash.each do |key, value| %>
        toastr.<%= type %>('<%= value %>')
    <% end %>
</script>
```

Key ist succes, alert, notice oder error. Value beinhaltet den Text der Rails flash.

Passen Sie app/views/layouts/application.html.erb wie folgt an:

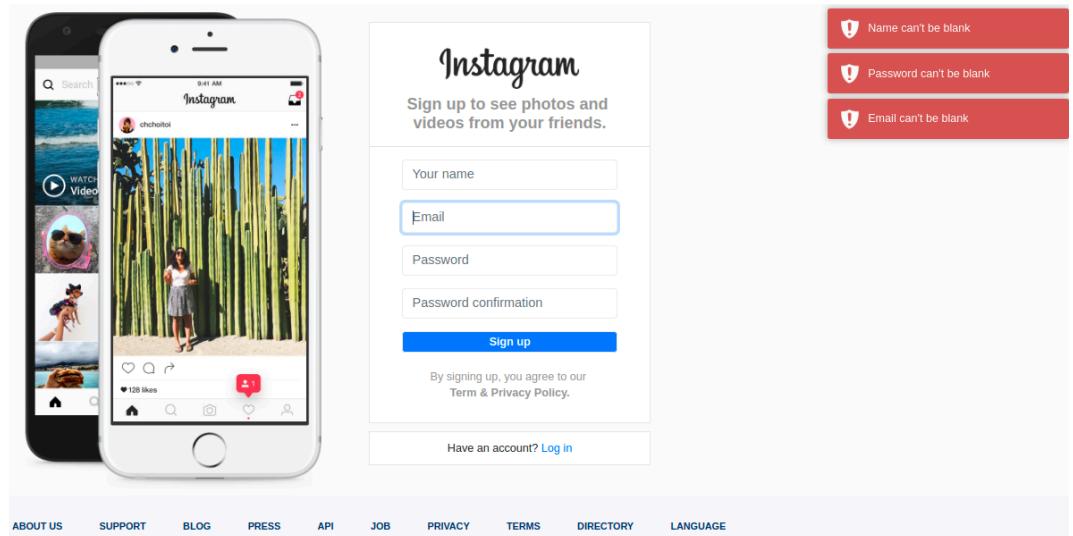
```
<!DOCTYPE html>
<html>
  <head>
    <title>Instagram</title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>

    <%= stylesheet_link_tag    'application', media: 'all', 'data-turbolinks-track': 'reload' %>
    <%= javascript_include_tag 'application', 'data-turbolinks-track': 'reload' %>
  </head>
  <body>
    <%= render 'shared/navbar' if current_user %>
    <%= yield %>
    <%= render 'shared/footer' %>
    <% if flash.any? %>
      <script type="text/javascript">
        <% flash.each do |key, value| %>
          <% type = key.to_s.gsub('alert', 'error').gsub('notice', 'success') %>
          toastr.<%= type %>('<%= value %>')
        <% end %>
      </script>
    <% end %>
  </body>
</html>
```

Aufgabe: Error Rails flash bei falscher Benutzerregistration

Erstellen Sie eine Partial-View app/views/shared/_devise.html.erb, die bei falscher Benutzerregistration in der richtigen View gerendert wird und nur Error-flashes ausgibt. Es gibt mehrere und einige sind zu erwähnen:

- › Name can't be blank
- › Email can't be blank
- › Password can't be blank
- › Password confirmation doesn't match
- › Password is too short
- › ...



User Profile Page

Model User erweitern

Erstellen sie eine Migration AddFieldsToUser, die der Tabelle users folgende Attribute hinzufügt:

```
:users, :website, :string
:users, :bio, :text
:users, :provider, :string
:users, :uid, :string
:users, :image, :string
```

Notieren Sie das Migrationsdatei und die notwendigen Kommandos für die Migration:

`rails g migration AddFieldsToUser website:string bio:text provider:string uid:string image:string`

`rails db:migrate`

V

```
class AddFieldsToUser < ActiveRecord::Migration[5.2]
  def change
    add_column :users, :website, :string
    add_column :users, :bio, :text
    add_column :users, :provider, :string
    add_column :users, :uid, :string
    add_column :users, :image, :string
  end
end
```

routes.rb

In der Datei routes.rb müssen wir nun einen folgenden Eintrag erstellen:

```
1 Rails.application.routes.draw do
2   root 'pages#home'
3   devise_for :users
4   # For details on the DSL available within this file, see http://guides.rubyonrails.org
5   resources :users, only: [:index, :show]
6 end
7
```

`resources :users` ist ein Shortcut für alle http-Verben und umfasst z.B. folgendes:

GET	/users /show
GET	/users
PATCH/PUT	/users
DELETE	/users
POST	/users

Mit der Einschränkung `:only: [:index, :show]` grenzen wir den Inhalt von `app/views/users/*.html.erb` auf die Views `index` und `show`. `index` ist zwar noch nicht erstellt aber wird folgen.

Das Rails Routing wird ausführlich unter <https://guides.rubyonrails.org/routing.html> erklärt.
Das HowTo ist auch unter `sh-modules/iet-151/03_Arbeitsblaetter/AB151-04_routing.pdf` zu finden.

users_controller.rb

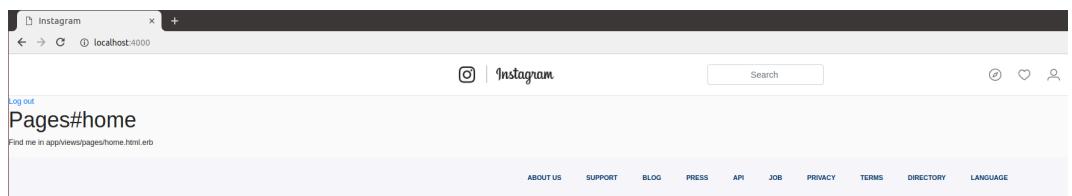
Erstellen Sie nun manuell (Datei neu) einen Controller mit Namen `app/controllers/user_controller.rb` und erfassen eine `show`-Methode welche einen bestimmten User mittels Attribut `id` findet und die Variable `@user` speichert und der `show`-View zur Verfügung stellt. Notieren Sie die `show`-Methode:

```
def show
  @user = User.find(params[:id])
end
```

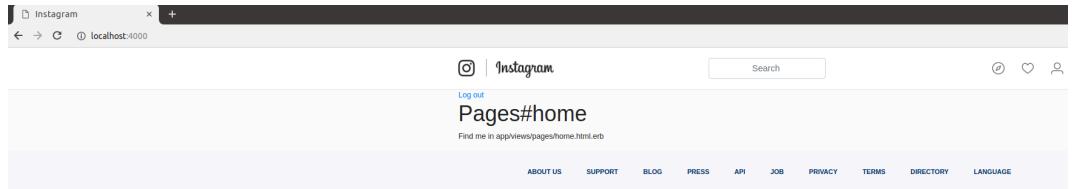
Da unserer View noch ein wenig unordentlich aussieht, wrappen wir den Inhalt von `yield` in `app/views/layouts/application.html.rb` in einen Container:

```
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' %>
9      <%= javascript_include_tag 'application', 'data-turbolinks-track': 'reload' %>
0    </head>
1    <body>
2      <%= render 'shared/navbar' if current_user %>
3      <div class="container">
4        <%= yield %>
5      </div>
6      <%= render 'shared/footer' %>
7      <% if flash.any? %>
8        <script type="text/javascript">
9          <% flash.each do |key, value| %>
0            <% type = key.to_s.gsub('alert', 'error').gsub('notice', 'success') %>
1            toastr.<%= type %>('<%= value %>')
2          <% end %>
3        </script>
4      <% end %>
5    </body>
6  </html>
```

Ohne container:



Mit container:

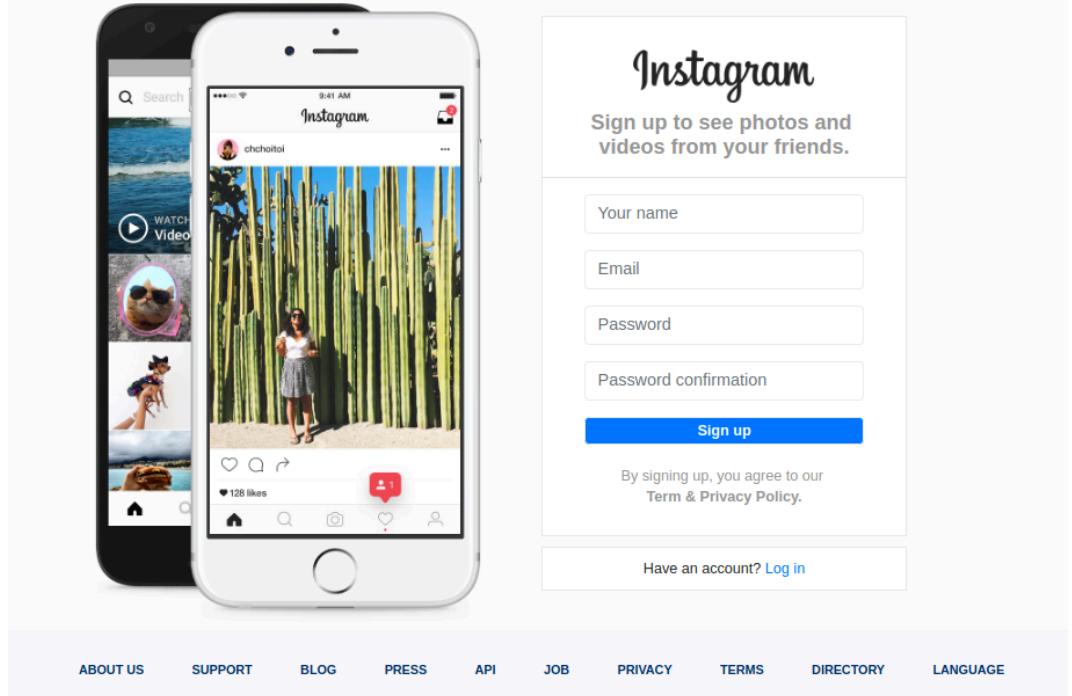


show-View mit Bootstrap Modal

Erstellen Sie eine manuelle View app/views/users/show.html.erb. In dieser View nutzen wir Bootstrap Modal. Mit Modal können wir fancy lightboxes, notifications oder Dialogboxen bauen. Alle Informationen zu Bootstrap Modal finden Sie unter:

<https://getbootstrap.com/docs/4.0/components/modal/>

Stellen Sie nun sicher, dass mindestens zwei besser drei Benutzer in unserer Insta-App erfasst sind:

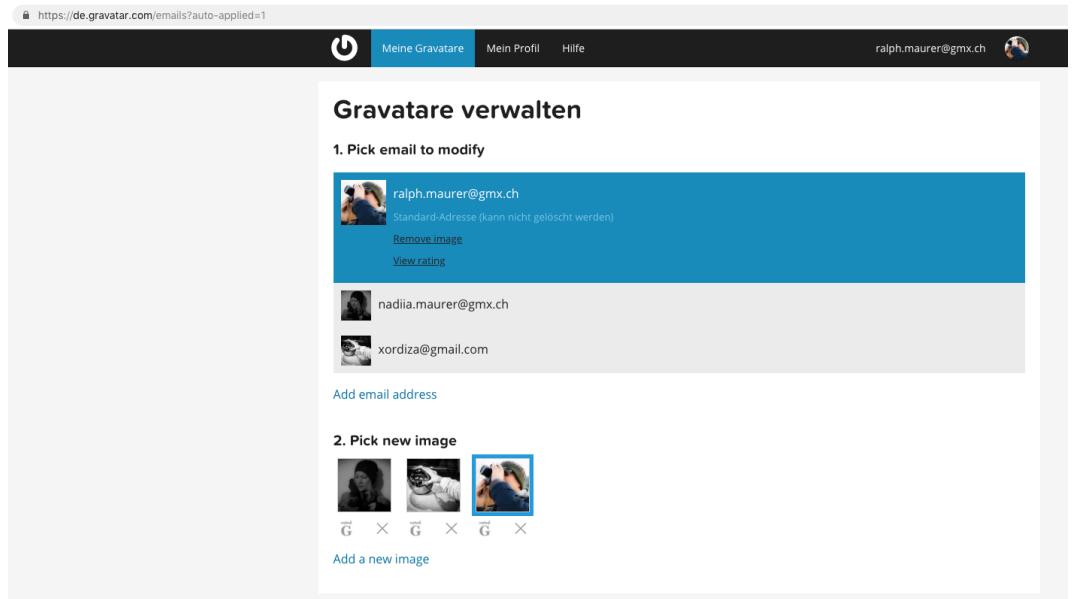


Die Benutzer müssen wir nun mit einem Profilbild versehen und dazu nutzen wir den Dienst von Gravatar.

Gravatar für User Photo in show-View

Gravatar ist ein Dienst, der Emailadressen mit Userbildern assoziiert. Registrieren Sie sich auf <https://de.gravatar.com/> und erfassen Sie die zwei Email Adressen der erfassten Benutzer in unserem Instagram. Fügen Sie geeignete Userbilder zu den Emailadressen hinzu. Gravatar

verlangt, dass Sie die Profilbilder nach Rudeness raten. Verzichten Sie auf Obszonitäten oder andere unpassende Bildinhalte.



The screenshot shows the Gravatar management interface at <https://de.gravatar.com/emails?auto-applied=1>. It displays a list of email addresses with their associated profile pictures. The first email, ralph.maurer@gmx.ch, is highlighted with a blue header and labeled as the standard address. Below it, there are other entries for nadiia.maurer@gmx.ch and xordiza@gmail.com. A link to add a new email address is visible. The second section, '2. Pick new image', shows three profile pictures with 'G' and 'X' icons below them, indicating options to upload or remove images.

Damit wir das Bild aus Gravatar anzeigen können, müssen wir einen Applicationhelper in Rails schreiben. Öffnen Sie herzu `app/helpers/application_helper.rb` und erstellen Sie folgende Methode:

```
1 module ApplicationHelper
2   def avatar_url user
3     gravatar_id = Digest::MD5::hexdigest(user.email).downcase
4     "https://www.gravatar.com/avatar/#{gravatar\_id}.jpg"
5   end
6 end
```

Erklären Sie in groben Zügen, was Sie geschrieben haben:

Die Methode verschlüsselt die Email des Users nach dem MD5 verfahren.

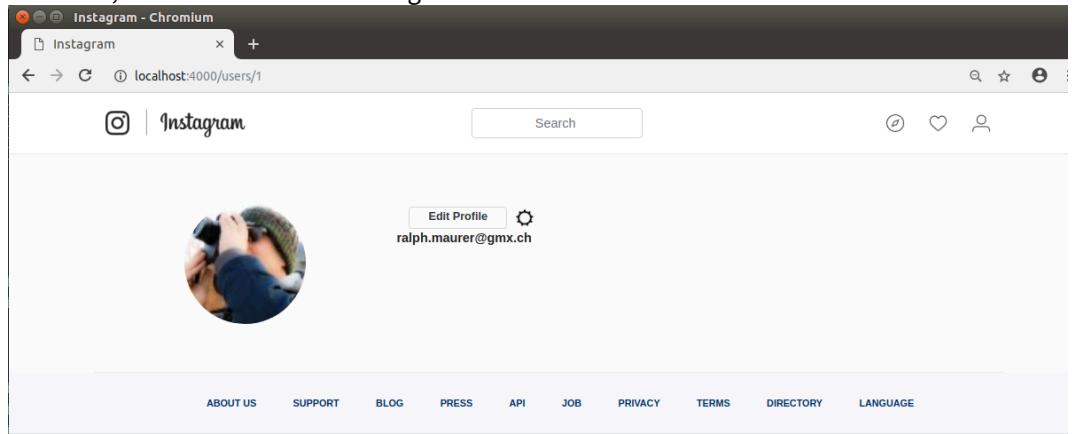
Die dabei entstehende ID wird dann an die URI gehängt.

luca@scherer.ml wird zu 88c74b205c52d1aa2c1db5126286f51a

<https://www.gravatar.com/avatar/88c74b205c52d1aa2c1db5126286f51a.jpg>

Als nächstes muss einiges an HTML und ERB geschrieben werden.

Ziel ist es, das die show-View wie folgt aussieht:



The screenshot shows a browser window titled 'Instagram - Chromium' displaying a user profile. The URL in the address bar is `localhost:4000/users/1`. The page features a large circular profile picture of a person. Below the picture are 'Edit Profile' and a gear icon. The email address `ralph.maurer@gmx.ch` is displayed. At the bottom of the page, there is a navigation bar with links for ABOUT US, SUPPORT, BLOG, PRESS, API, JOB, PRIVACY, TERMS, DIRECTORY, and LANGUAGE.

Beachten Sie das der Bereich Modal aus

<https://getbootstrap.com/docs/4.0/components/modal/> kopiert wurde und die CSS-Klassen leicht angepasst wurden.

```
application.html.erb           application_helper.rb | show.html.erb
1  <div class="row justify-content-md-center profile-wrapper">
2    <div class="col-md-4 text-center">
3      <%= image_tag avatar_url(@user), width: '152', height: '152', class: "round-img" %>
4    </div>
5    <div class="col-md-8">
6      <div class="row">
7        <p class="username"><%= @user.name %></p>
8        <% if @user == current_user %>
9          <%= link_to "Edit Profile", edit_user_registration_path, class: "btn btn-outline-dark common-btn edit-profile-btn" %>
10         <button type="button" class="core-sprite setting" data-toggle="modal" data-target="#exampleModal"></button>
11        <% end %>
12
13      <!-- Modal -->
14      <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
15        <div class="modal-dialog" role="document">
16          <div class="modal-content">
17            <div class="modal-header">
18              <h5 class="modal-title" id="exampleModalLabel">Settings</h5>
19              <button type="button" class="close" data-dismiss="modal" aria-label="Close">
20                <span aria-hidden="true">&times;</span>
21              </button>
22            </div>
23            <div class="list-group text-center">
24              <a href="#" class="list-group-item list-group-item-action">Change Password</a>
25              <%= link_to "Log out", destroy_user_session_path, method: :delete,
26                         class: "list-group-item list-group-item-action" %>
27              <a href="#" class="list-group-item list-group-item-action" data-dismiss="modal">Cancel</a>
28            </div>
29          </div>
30        </div>
31      </div>
32    </div>
33    <div class="row">
34      <p class="email"><%= @user.email %></p>
35    </div>
36  </div>
37  </div>
38
```

Erklären Sie kurz folgende Codeausschnitte:

```
...
<%= image_tag avatar_url(@user), width: '152', height: '152',
class: "round-img" %>
...
<% if @user == current_user %>
  <%= link_to "Edit Profile", edit_user_registration_path, class:
"btn btn-outline-dark common-btn edit-profile-btn" %>
  <button type="button" class="core-sprite setting" data-
toggle="modal" data-target="#exampleModal"></button>
<% end %>
...
<%= link_to "Log out", destroy_user_session_path, method: :delete,
class: "list-group-item list-group-item-action" %>
...
```

Ausschnitt 1: Es wird ein rundes Bild des Users angezeigt. Die URL des Bildes wird von `avatar_url()` zurückgegeben.

Ausschnitt 2: Falls ein Benutzer sein eigenes Profil anschaut, wird noch ein Settings-Knopf angezeigt.

Ausschnitt 3: Mit dem klicken auf dieses List-item wird die aktuelle Session verworfen.

Wo und wie können Sie Informationen zu dem 2. Parameter von `link_to` finden?

Im Folgenden ist der 2. Parameter
`edit_user_registration_path` und
`destroy_user_session_path`

```
...
<% if @user == current_user %>
  <%= link_to "Edit Profile", edit_user_registration_path, class:
  "btn btn-outline-dark common-btn edit-profile-btn" %>
    <button type="button" class="core-sprite setting" data-
  toggle="modal" data-target="#exampleModal"></button>
<% end %>
...
<%= link_to "Log out", destroy_user_session_path, method: :delete,
  class: "list-group-item list-group-item-action" %>
```

mit "rails routes" werden die Verben, Controller und Methoden jedes **Prefixes** angezeigt.

Wir bauen die benötigten SCSS-Klassen in
app/assets/stylesheets/application.scss

```
173 .profile-wrapper {  
174   padding: 60px 0;  
175   border-bottom: 1px solid #efefef;  
176  
177   .username {  
178     font-size: 32px;  
179     line-height: 40px;  
180     font-weight: 200;  
181   }  
182  
183   .email {  
184     font-size: 16px;  
185     font-weight: bold;  
186   }  
187 }  
188  
189 .round-img {  
190   border-radius: 50%;  
191 }  
192  
193 .common-btn {  
194   height: 28px;  
195   line-height: 24px;  
196   padding: 0 24px;  
197   border-color: #dbdbdb;  
198   font-size: 14px;  
199 }  
200  
201 .edit-profile-btn {  
202   margin: 5px 0 0 15px;  
203   font-weight: bold;  
204 }  
205  
206 .setting {  
207   margin: 7px 0 0 10px;  
208   background-color: transparent;  
209   background-position: -125px -355px;  
210   height: 24px;  
211   width: 24px;  
212   border: none;  
213 }  
214 .setting:hover {  
215   cursor: pointer;  
216 }
```

Wenn alles fehlerfrei gemacht wurde, sollten Sie nun folgendes Resultat mit den erfassten Benutzern prüfen können.

1. Benutzer (eingeloggt): <http://localhost:4000/users/1>

A screenshot of a web browser window showing an Instagram profile. The title bar says "Instagram - Chromium". The address bar shows "localhost:4000/users/1". The page header includes the Instagram logo and a search bar. The main content area displays a circular profile picture of a person wearing a green beanie and blue jacket. To the right of the picture are "Edit Profile" and a gear icon, followed by the email "ralph.maurer@gmx.ch". Below the profile picture is a navigation bar with links: ABOUT US, SUPPORT, BLOG, PRESS, API, JOB, PRIVACY, TERMS, DIRECTORY, and LANGUAGE.

2. Benutzer: <http://localhost:4000/users/2>

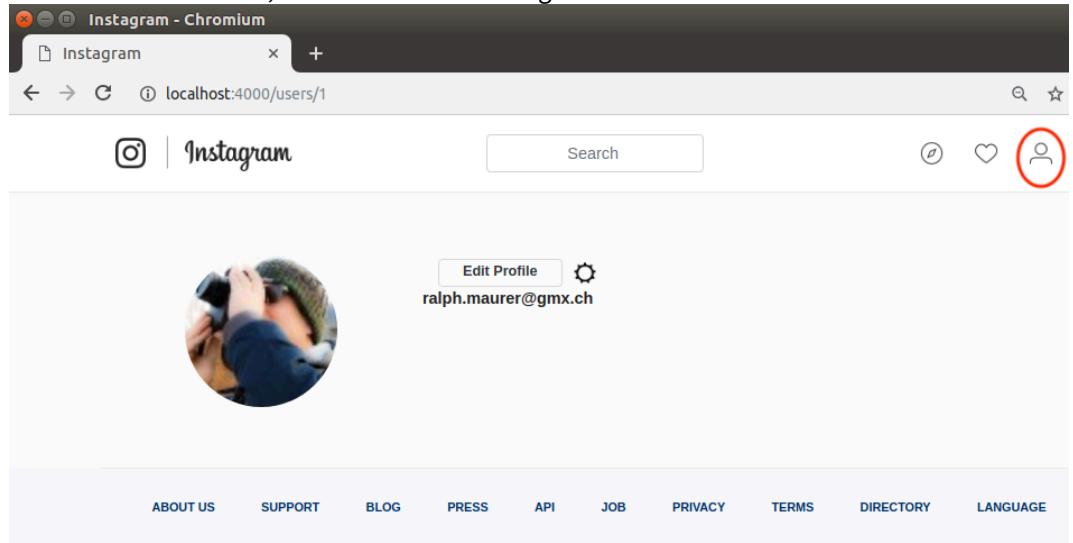
A screenshot of a web browser window showing an Instagram profile. The title bar says "Instagram - Chromium". The address bar shows "localhost:4000/users/2". The page header includes the Instagram logo and a search bar. The main content area displays a circular profile picture of a person wearing a dark beanie. To the right of the picture is the name "nadiia" and the email "nadiia.maurer@gmx.ch". Below the profile picture is a navigation bar with links: ABOUT US, SUPPORT, BLOG, PRESS, API, JOB, PRIVACY, TERMS, DIRECTORY, and LANGUAGE.

3. Benutzer: <http://localhost:4000/users/3>

A screenshot of a web browser window showing an Instagram profile. The title bar says "Instagram - Chromium". The address bar shows "localhost:4000/users/3". The page header includes the Instagram logo and a search bar. The main content area displays a circular profile picture of a person's hands holding a small object. To the right of the picture is the name "Xor" and the email "xordiza@gmail.com". Below the profile picture is a navigation bar with links: ABOUT US, SUPPORT, BLOG, PRESS, API, JOB, PRIVACY, TERMS, DIRECTORY, and LANGUAGE.

link_to Profile Page

Als nächstes wollen wir, dass der Link Profile Page in der Navbar funktioniert:



Die Datei `-navbar.html.erb` wird wie folgt angepasst:

A screenshot of a code editor showing the `_navbar.html.erb` file. The code is a template for a Bootstrap navbar. A red oval highlights the line of code where the `link_to` statement is located, which generates a link to the user's profile page. The code editor also shows a sidebar with project files and a preview pane.

```
Project: Instagram
File: _navbar.html.erb
Content:
1 <nav class="navbar navbar-expand-lg navbar-light">
2   <div class="container">
3     <a class="navbar-brand core-sprite hide-text" href="#">Navbar</a>
4     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation"><span class="navbar-toggler-icon"></span></button>
5     <div class="collapse navbar-collapse id="navbarSupportedContent">
6       <form class="form-inline my-2 my-lg-0 ml-md-auto">
7         <input class="form-control mr-sm-2 text-center common-btn" type="search" placeholder="Search" aria-label="Search">
8       </form>
9       <ul class="navbar-nav ml-md-auto">
10         <li class="nav-item">
11           <a class="nav-link core-sprite explore-icon hide-text" href="#">Explore</a>
12         </li>
13         <li class="nav-item">
14           <a class="nav-link core-sprite notification-icon hide-text" href="#">Notification</a>
15         </li>
16         <li class="nav-item">
17           <a class="nav-link core-sprite profile-icon hide-text" href="<%= link_to "Profile", user_path(current_user) %>">Profile</a>
18         </li>
19       </ul>
20     </div>
21   </div>
22 </div>
23 </div>
24 </nav>
```

Edit User Page

Ein User-Profile soll einfach veränderbar sein. Devise hat uns eine Vorlage unter app/views/devise/registrations/edit.html.erb geliefert:

The screenshot shows a web browser window with the URL `localhost:4000/users/edit`. The page is titled "Edit User". It contains fields for "Email" (set to "xordiza@gmail.com"), "Password" (left blank), "Password confirmation" (left blank), and "Current password" (left blank). Below these fields is a "Update" button. A large "Cancel my account" button is prominently displayed. Underneath it, there's a link "Unhappy?" followed by a "Cancel my account" button. At the bottom of the page, there are links for "ABOUT US", "SUPPORT", "BLOG", "PRESS", "API", "JOB", "PRIVACY", "TERMS", "DIRECTORY", and "LANGUAGE".

Als erstes ergänzen Sie den application_controller.rb wie folgt:

```
devise_parameter_sanitizer.permit(:account_update, keys: [:name, :bio, :website])
```

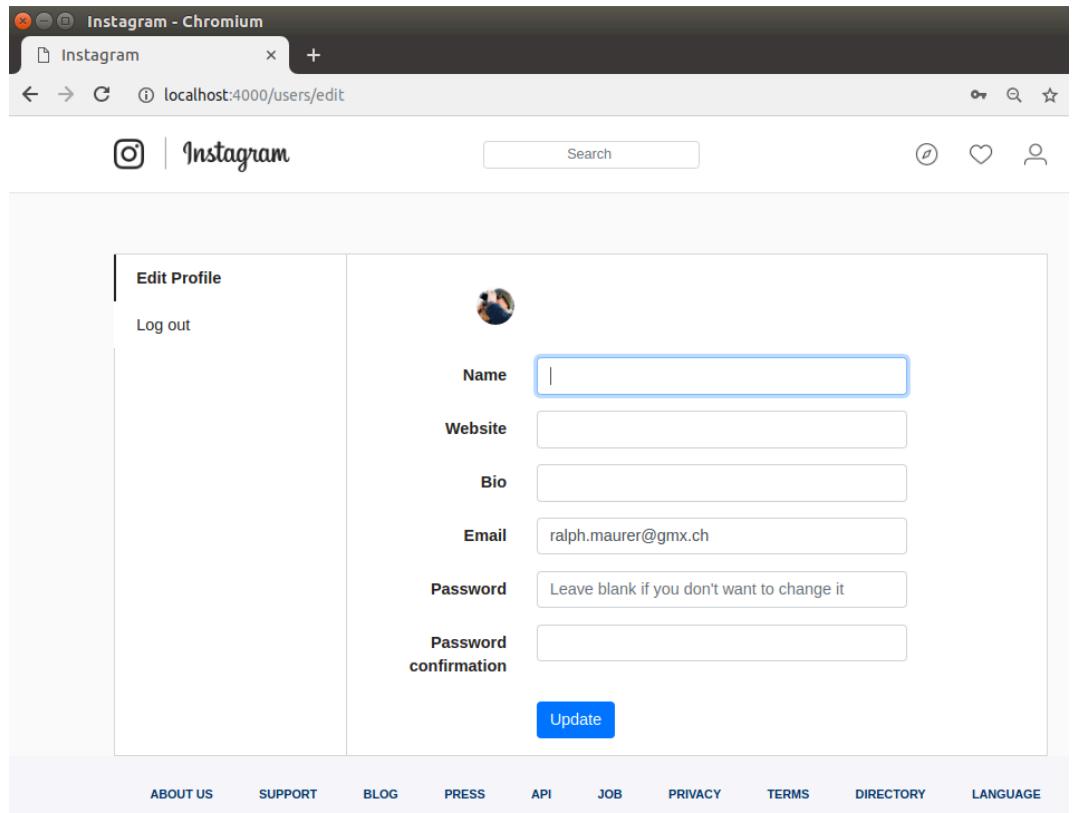
Ergänzen Sie app/views/devise/registrations/edit.html.erb und app/assets/stylesheets/application.scss

Die Vorlagen finden Sie unter sh-modules/iet-151/03_Arbeitsblaetter/AB151-04_edit.html.rb sowie:

```
.edit-profile-wrapper {  
    margin: 60px 0;  
  
.left-col {  
    border: 1px solid #dbdbdb;  
    padding: 0;  
    background-color: white;  
  
.list-group-item {  
    border: none;  
    color: #262626;  
    border-radius: 0;  
    font-size: 16px;  
    border-left: 2px solid transparent;  
}  
  
.list-group-item:first-child {  
    font-weight: bold;  
    border-left-color: #262626;  
}
```

```
.list-group-item:not(:first-child):hover {  
    border-left-color: #dbdbdb;  
}  
  
.right-col {  
    border: 1px solid #dbdbdb;  
    border-left: none;  
    background-color: white;  
  
.username-img {  
    margin: 32px 0;  
  
.col-sm-9 {  
    font-size: 20px;  
    font-weight: 400;  
    line-height: 20px;  
}  
}  
  
.form-group {  
    font-size: 16px;  
}  
}
```

Damit erhalten wir ein schönes und anschauliches Resultat:



The screenshot shows a browser window titled "Instagram - Chromium" displaying a profile edit page. The URL in the address bar is "localhost:4000/users/edit". The page has a header with the Instagram logo and a search bar. On the left, there's a sidebar with "Edit Profile" and "Log out" options. The main content area contains fields for "Name" (with a placeholder), "Website", "Bio", "Email" (containing "ralph.maurer@gmx.ch"), "Password" (with a note: "Leave blank if you don't want to change it"), and "Password confirmation". A blue "Update" button is at the bottom right of the form. Below the form is a navigation bar with links: ABOUT US, SUPPORT, BLOG, PRESS, API, JOB, PRIVACY, TERMS, DIRECTORY, and LANGUAGE.

Versuchen Sie nun etwas zu verändern. Was stellen Sie fest:

Es wird nichts geändert und man wechselt auf /users

Lösen Sie das Problem mit Hilfe von:

<https://github.com/plataformatec/devise/wiki/How-To:-Allow-users-to-edit-their-account-without-providing-a-password>

The screenshot shows a GitHub repository page for 'plataformatec / devise'. The 'Wiki' tab is active. The main content area displays a guide titled 'How To: Allow users to edit their account without providing a password'. The sidebar on the right is titled 'Pages 133' and lists several topics such as 'Home', 'Bug reports', 'Callbacks', 'Contributing', etc.

How To: Allow users to edit their account without providing a password

Julien Micio edited this page on 26 Feb · 70 revisions

By default, Devise allows users to change their password using the `:registerable` module. But sometimes, developers want to create other actions that allow the user to change their information without requiring a password. The best option in this case is to create your own controller, that belongs to your application, and provide an edit and update actions, as you would do for any other resource in your application.

Keep in mind though to be restrictive in the parameters you allow to be changed. In particular, you likely want to permit just user data fields and avoid e-mail, password and such information to be changed:

```
params(:user).permit(:first_name, :last_name, :address)
```

An alternative solution would be to simply override the `update_resource` method in your registrations controller as follows:

```
# app/controllers/registrations_controller.rb
class RegistrationsController < Devise::RegistrationsController
  protected
    def update_resource(resource, params)
      resource.update_without_password(params)
    end
end
```

"Attention": This way of using this method does not allow the user to change his password.

When using this solution you will need to tell devise to use your controller in `routes.rb`:

```
devise_for :users, controllers: { registrations: 'registrations' }
```

Finally, if you have generated views using devise, remove the corresponding form input for 'current_password' in `app/views/devise/registrations/edit.html.erb`.

Das Editieren der Profil Page muss funktionieren ☺

The screenshot shows a browser window displaying an Instagram profile edit page. The URL is `localhost:4000/users/edit`. The page has a sidebar with 'Edit Profile' and 'Log out' options. The main content area contains fields for 'Name', 'Website', 'Bio', 'Email' (with value `ralph.maurer@gmx.ch`), 'Password' (with placeholder 'Leave blank if you don't want to change it'), and 'Password confirmation'. A blue 'Update' button is at the bottom. The footer includes links for 'ABOUT US', 'SUPPORT', 'BLOG', 'PRESS', 'API', 'JOB', 'PRIVACY', 'TERMS', 'DIRECTORY', and 'LANGUAGE'.

Auftrag: Quicknote AB151-04

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_AB151-04.zip*

Das Archiv beinhaltet:

- › **Quicknote:** Klasse_Name_Vorname_QN_AB151-04.pdf
- › **Code:** show.html.erb, application.html.erb, application.scss, application_helper.rb, registrations_controller.rb, users_controller.rb, edit.html.rb, routes.rb

Zusammenfassung AB151-04

Die Quicknote soll eine kurze prägnante Zusammenfassung des Dokuments AB151-04 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:

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