Raffaele Senften
Inf 17f

Quicknote

Attribute erfassen

Möchte man ein Attribut nachträglich in die Datenbank hinzufügen, kann man dies mit einer Migration ganz einfach lösen. Erstellt man eine Migration nach dem Schema «AddAttributnameToTabellenname», dann wird schon fast alles automatisch erstellt. In die soeben erstellte Migration muss nur noch das Kommando «add_column :tabellenname, :attributname, :datentyp» hinzugefügt werden. Migrieren kann man das Ganze auf der Konsole mit dem Befehl «rails db:migrate».

Anwendung
Dank einer Migration nach einem Namenskonzept, kann man ein Attribut zu einer bestehenden Tabelle hinzufügen. Dies kann nützlich sein, wenn man ein Attribut vergessen hat.

Vor- und Nachteile
+Falls man ein Attribut vergessen wird, kann man dieses Attribut sehr einfach nachträglich eintragen.
- Nachteile gibt es meiner Meinung nach keine.

Validierung

Im Arbeitsblatt können wir die gelernten Validierungsmethoden anwenden. Wie zum Beispiel: *presence:* oder *length:* .

Anwendung
Eine Validierung kann sehr nützlich sein, wenn man z.B eine maximale Länge definieren will.

Vor- und Nachteile
+Durch die Validierung kann man die Länge eines Attributes steuern und somit kann man Fehler vermeiden.
- Nachteile gibt es meiner Meinung nach keine.

Devise Gem

Das Devise Gem lässt nur die E-Mail Adresse als Benutzernamen zu. Um dies zu ändern muss man im Application Controller einen Eintrag machen.

Eine Devise Gem Dokumentation findet man unter folgendem Link:
https://www.rubydoc.info/gems/devise/Devise

Anwendung
Mit dem Devise Gem kann man die Login-Art(E-Mail oder Benutzername) einstellen.

Vor- und Nachteile
+Mit diesem Gem kann man die Login-Art ändern.
- Nachteile gibt es meiner Meinung nach keine.

Sign_in

Das Devise Gem bietet uns sehr viele Variablen an, darunter *current_user*. Mit der Abfrage *if current_user* kann sehr einfach und schnell geprüft werden, ob eine valide Authentifikation von

einem Benutzer stattgefunden hat. Mit dieser Abfrage kann man z.B das Navbar erst dann anzeigen lassen, wenn der Benutzer angemeldet ist. Mit der Klasse «carousel» von Bootstrap kann man einen Image-Slider sehr leicht aufbauen. Um dies zu tun, muss man nur bei einem Div das Attribut «data-ride» auf «carousel» setzen.

Anwendung
Mit der Variabel *current_user*, kann man überprüfen, ob eine valide Authentifikation stattgefunden hat. Mit der Carousel-Funktion kann man verschieden Bilder nacheinander darstellen lassen, wie ein Carousel.

Vor- und Nachteile
+Mit *current_user* kann man sehr einfach Elemente ausblenden, solange man nicht eingeloggt ist.
+Die Carousel-Funktion ist sehr einfach zu verstehen und sie funktioniert sehr gut.
- Nachteile gibt es meiner Meinung nach keine.

SCSS und CSS Nesting

Der Haupt-Unterschied zwischen SCSS und CSS ist, dass bei SCSS eine grössere Verschachtelung möglich ist, die den Code viel Lesbarer macht.

Anwendung
SCSS verfügt über ein besseres Nesting als CSS, das es besser lesbar macht.

Vor- und Nachteile
+Wie schon erwähnt, ist der Code mit SCSS lesbarer als mit CSS.
- Nachteile gibt es meiner Meinung nach keine.

Bootstrap Form Classes

Bootstrap bietet viele Klassen, um eine Form zu stylen

Unter folgendem Link werden viele Klassen von Bootstrap genauer definiert:
https://getbootstrap.com/docs/4.0/components/forms

Anwendung
Bootstrap biete uns viele verschiedene SCSS Klassen an, um unsere App zu designen.

Vor- und Nachteile
+Bootstrap vereinfacht das Design von Apps.
- Man muss die Bootstrap Klassen zuerst kennenlernen und verstehen, wie man sie braucht.

Redirect Links

Das Devise Gem stellt uns eine Partial View mit verschiedenen Links zur Verfügung.

```
<%- if devise_mapping.registerable? && controller_name != 'registrations' %>
  <%= "Don't have an account? "%>
  <%= link_to "Sign up", new_registration_path(resource_name) %><br />
<% end %>
```

Anwendung
Mit der zur Verfügung gestellten Partial View mit Links zu anderen Seiten, kann man diesen immer und überall in seinem Projekt wiederverwenden.

Vor- und Nachteile
+Die Partial-View ist schon erstellt und man kann sie sehr einfach gebrauchen.
- Nachteile gibt es meiner Meinung nach keine.

Partial View

In diesem Arbeitsblatt verwenden wir viele Partial Views, da wir Code mehrmals verwenden und so die Code-Teile nur einmal aufgerufen werden müssen.

Anwendung
Es wurden die Partial Views «Footer» und «Dummy-Phone» erstellt, damit wir den Code nur einmal schreiben müssen, und er für mehrere Verfügbar ist.

Vor- und Nachteile
+Partial-Views sind sehr praktisch um den Code auszulagern und mehrmals zu verwenden.
- Nachteile gibt es meiner Meinung nach keine.

Selbstreflexion

Was habe ich gelernt?
Ich habe sehr vieles gelernt.

- Wie man ein Attribut nachträglich erstellt.
- Wie man Bootstrap verwendet.
- Wie man verschiedene Bilder nacheinander darstellen lässt.
- Wie man *current_user* verwendet
- Wie die SCSS Struktur aufgebaut ist

Wie bin ich vorgegangen beim Lernen bzw. Ausführen des Auftrages?
Wenn ich eine Aufgabe des Auftrages gelesen habe, habe ich diesen gleich ausgeführt, bevor ich an dem Auftrag weiterlas.

Was waren die Schwierigkeiten, wie konnte ich diese lösen?
Schwierigkeiten gab es kein. Der Auftrag war sehr gut beschrieben.

Was habe ich nicht verstanden bzw. was konnte ich nicht lösen?
Ich habe alles verstanden und konnte alles lösen.

Was kann ich nächstes Mal besser machen?
Meiner Meinung nach habe ich alles gut gemacht und finde auch, dass ich das nächste Mal alles gleich machen kann.

Lösungen der Aufgaben

Erstellung der Migration und Ausführung dieser Migration

- $ *rails generate migration AddNameToUser name_string*
- $ *rails db:migrate*

Anzeigen des Navbars erst, wenn man eingeloggt ist
Folgendes muss im File «app/views/shared/_navbar.html.erb» eingefügt warden:
Zu oberst im File kommt: *<% if current_user %>*
Zu unterst im File kommt: *<% end %>*

Was fällt ihnen auf? Stichwort SCSS, CSS un «nesting»?
Diese Antwort wurde im Text Gelb markiert.

Welches Attribut wechselt die Bilder im Dummy-Phone?
Data-ride=»carousel»

Mit welchen Bootstrap- und SCSS-Klassen werden die Bilder des Sliders positioniert?
.landing-left {... .dummy-phone {...} .screen-shot {... .item {…}}}

Welche SCSS-Klassen unterteilen die Seite in links (Dummy-Phone) und in rechts (Registrierung)?
classs=»col-lg-6 landing-left» class=”col-lg-6 landing-right”

Wo wird der Hintergrund des Dummy-Phone definiert (das Mobiltelefon Gehäuse)?
.landing-left { .dumm-phone {backgreound-image: image-url(“dummy-phone-2x.png”) …}}

In welcher Datei haben Sie den Code für den Footer als Partial-View erfasst?
app/views/shared/_footer.html.erb

Wo wird die Datei mit dem Footer gerendert?
app/views/layouts/application.html.erb

Wie lautet der Befehl zum Rendern?
<%=render ‘shared/footer’ %>

In welcher Datei haben Sie den Code für das Dummy-Phone erfasst?
app/views/devise/shared/_dummy_phone.html.erb

Wo wird die Datei mit dem Dummy-Phone für die Views «sign_in» und «sign_up» gerendert?
app/views/devise/sessions/new.html.erb
app/views/devise/registrations/new.html.erb

Wie lautet der Befehl zum rendern dieser Datei?
<%=render ‘devise/shared/dummy-phone’ %>

Abschliessende Reflexion

Alles in allem habe ich sehr viel Neues gelernt. Dieses Arbeitsblatt ist sehr gut beschrieben und auch sehr gut strukturiert. Ich kann mir das gelernte sehr gut merken und werde es in der Zukunft auch wiederverwenden. Mit jedem Arbeitsblatt das ich löse, lerne ich etwas Neues über Rails.

Screenshots

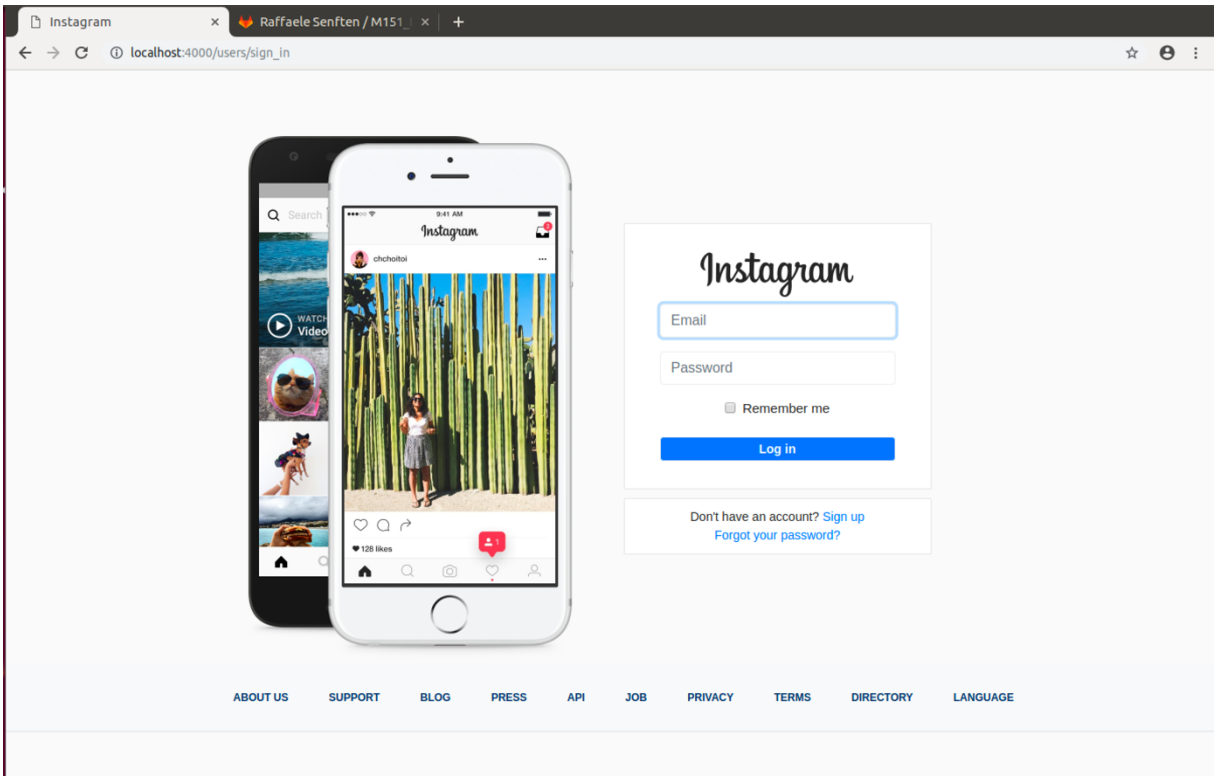

Abbildung 1: Login-View

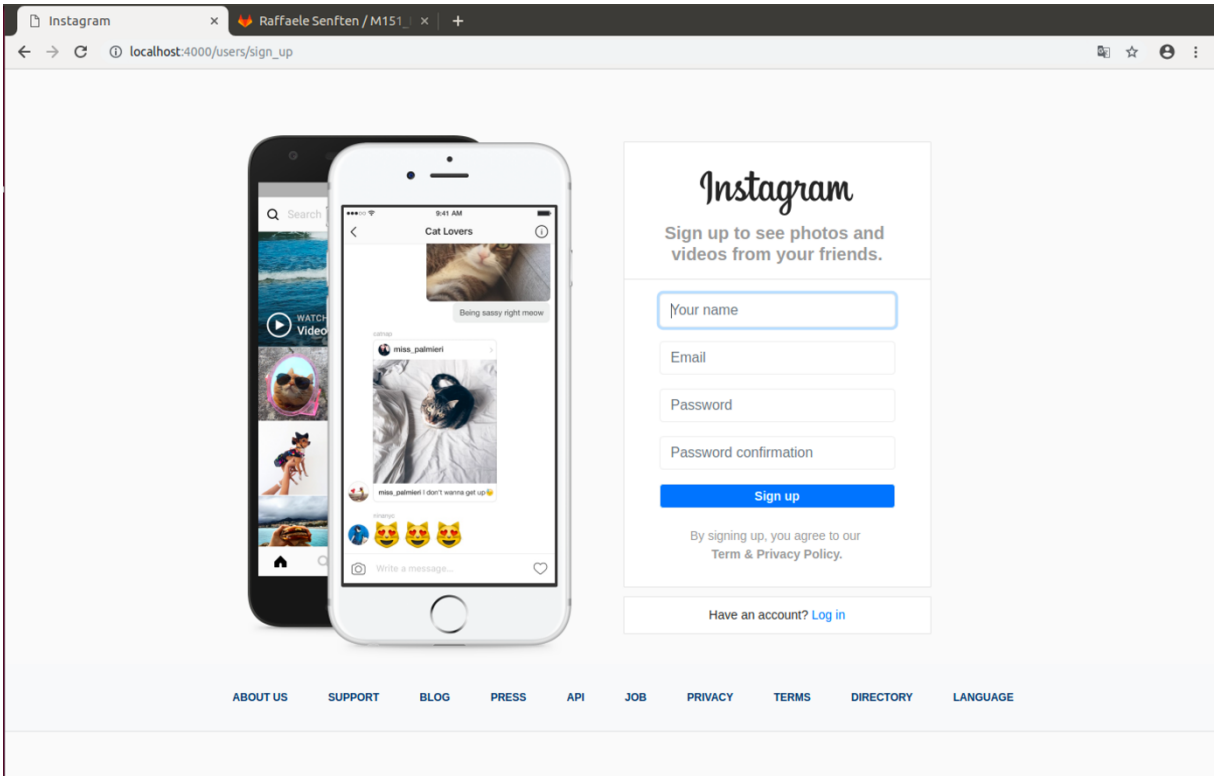

Abbildung 2: Registration-View