



MODUL 151

TEIL 2: INSTAGRAM BASIC AUTHENTIFICATION PARTIAL NAVBAR

Ralph Maurer

Inhaltsverzeichnis AB151-02

Modul 133: Instagram mit Rails bauen

Teil 2: Basic Authentication und Partial Navbar

Inhaltsverzeichnis AB151-02	1
Modul 133: Instagram mit Rails bauen	1
Was bauen wir?	2
Basic Authentifikation für unser Rails Instagram.....	2
Navbar mit partial view	5
Bootstrap 4 und JQuery installieren.....	5
Controller Pages mit View home erstellen	6
Instagram Menu erstellen	7
CSS-Klasse .core-sprite	12
Instagram Menu layouts	14
Partial-view für <nav> erstellen	16
<%= link_to %>	17
login / logout.....	18
Auftrag: Quicknote AB151-02	19

Was bauen wir?

In Modul 151 wollen wir in Rails Instagram mit vielen Grundfunktionalitäten nachbauen. Wir fokussieren uns auf:

- › einen möglichst vollständigen Nachbau von einem persönlichen Instagram
- › mit mehreren Benutzern
- › und einem vertieften Model und CRUD Verständnis.
- › Wir wenden Bootstrap CSS Framework an und implementieren
- › Features mit JavaScript und AJAX aus JQuery.

Das Fallbeispiel Instagram als Rails-App basiert auf den Arbeiten von Truong Nguyen und ist bei UdeMy publiziert.

Leider ist in einem so umfangreichen Projekt viel vorgegeben. Einige Recherchen können selber gemacht werden. Der Autor geht davon aus, dass wir Ideen ein erstes Mal umsetzen und die Vertiefung im Selbststudium geschieht.

Basic Authentifikation für unser Rails Instagram

Damit wir das Rad nicht neu erfinden müssen, soll eine Authentifizierungskomponente genauer ein gem für unsere Rails-App zum Einsatz kommen.

Das angewandte gem heisst Devise und die Installation wird nun schrittweise vorgestellt:



Tragen Sie am Schluss des gemfile in das Kommando
gem 'devise' ein

Was ist das gemfile und wieso müssen Sie diesen Eintrag erstellen?

1. Wie installiert man ein gem?

Die Installation wird mit den Kommandos aus Frage 2 gefolgt von

```
rails generate devise:install
```

abgeschlossen.

Nach der Installation von devise werden Sie aufgefordert einige manuelle Konfigurationen vorzunehmen. Beachten Sie vorerst nur die Punkte 1, 3 und 4:

=====

Some setup you must do manually if you haven't yet:

1. Ensure you have defined default url options in your environments files. Here is an example of default_url_options appropriate for a development environment

```
in config/environments/development.rb:

config.action_mailer.default_url_options = { host:
'localhost', port: 3000 }
```

In production, :host should be set to the actual host of your application.

2. Ensure you have defined root_url to *something* in your config/routes.rb.

For example:

```
root to: "home#index"
```

3. Ensure you have flash messages in app/views/layouts/application.html.erb.

For example:

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

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

4. You can copy Devise views (for customization) to your app by running:

```
rails g devise:views
```

=====

Nach Abschluss der Punkte 1, 3 und 4 geben Sie folgendes Kommando im Terminal ein:

```
rails g devise User
```

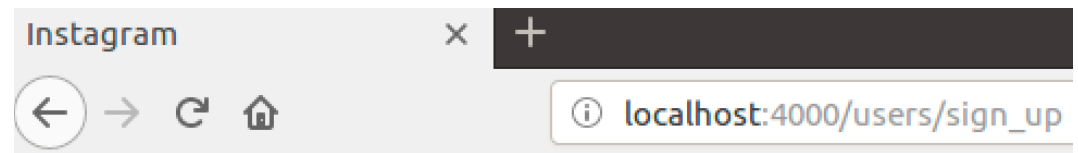
Dieses Kommando generiert die ein Model User und eine Migration
NNNNNNNNNNNNNNNN _device_create_users.rb

Notieren Sie hier das Kommando für die Migrationausführung und führen Sie diese gleich aus:



Testen Sie nun die Seite

http://localhost:4000/users/sign_up



Sign up

Email

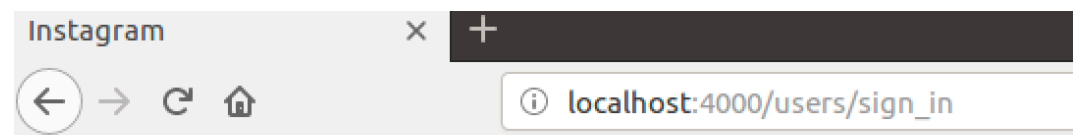
Password (6 characters minimum)

Password confirmation

[Log in](#)

und erfassen Sie einen Benutzer. Checken Sie, ob die Authentifizierung funktioniert.

http://localhost:4000/users/sign_in



Log in

Email

Password

☐ Remember me

[Sign up](#)

[Forgot your password?](#)

Kontrollieren Sie die Datenbank users und notieren Sie Attribute und weitere Beobachtungen:

Navbar mit partial view

Bootstrap 4 und JQuery installieren

Wir arbeiten mit Bootstrap 4. Hierzu ergänzen wir das gemfile wie folgt:

```
gem 'bootstrap', '~> 4.0.0'
gem 'jquery-rails'
```

Benennen Sie die Datei `app/assets/stylesheets/application.css` in `app/assets/stylesheets/application.scss` um.

Löschen Sie nun den Inhalt von `app/assets/stylesheets/application.scss` und schreiben Sie folgende Zeile am Anfang der in die Datei `app/assets/stylesheets/application.scss`:

```
@import "bootstrap";
```

In der Datei `app/assets/javascripts/application.js` muss nun JQuery und Bootstrap eingebunden werden. Ergänzen Sie wie `app/assets/javascripts/application.js` folgt:

```
// This is a manifest file that'll be compiled into application.js,
// which will include all the files
// listed below.
//
// Any JavaScript/Coffee file within this directory,
// lib/assets/javascripts, or any plugin's
// vendor/assets/javascripts directory can be referenced here using
// a relative path.
//
// It's not advisable to add code directly here, but if you do,
// it'll appear at the bottom of the
// compiled file. JavaScript code in this file should be added after
// the last require_* statement.
//
// Read Sprockets README
// (https://github.com/rails/sprockets#sprockets-directives) for
// details
// about supported directives.
//= require rails-ujs
//= require activestorage
//= require turbolinks
//= require jquery3
//= require popper
//= require bootstrap-sprockets
//= require_tree .
```

Controller Pages mit View home erstellen


Als nächstes muss ein Controller Pages mit einer View home erstellt werden. Wie lautet das entsprechende Kommando?



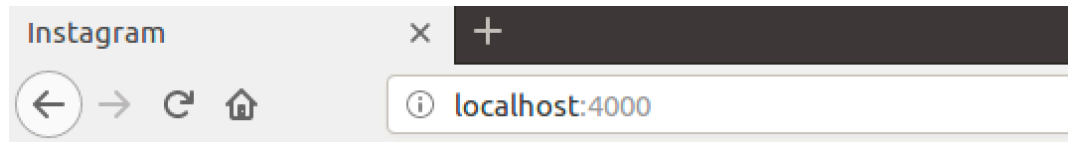
Das angezeigte Resultat nach der Controller-Erstellung sieht wie folgt aus:
Running via Spring preloader in process 29014

```
Running via Spring preloader in process 29014
create  app/controllers/pages_controller.rb
route   get 'pages/home'
invoke  erb
create  app/views/pages
create  app/views/pages/home.html.erb
invoke  test_unit
create  test/controllers/pages_controller_test.rb
invoke  helper
create  app/helpers/pages_helper.rb
invoke  test_unit
invoke  assets
invoke  coffee
create  app/assets/javascripts/pages.coffee
invoke  scss
create  app/assets/stylesheets/pages.scss
```

Verändern Sie `app/config/routes.rb` so, dass die soeben erstellte View als Startseite definiert ist:



Starten Sie den Rails Server neu und kontrollieren Sie die neue Startseite (Root of web in Rails). Sie sollten folgendes unter <http://localhost:4000/> erhalten:

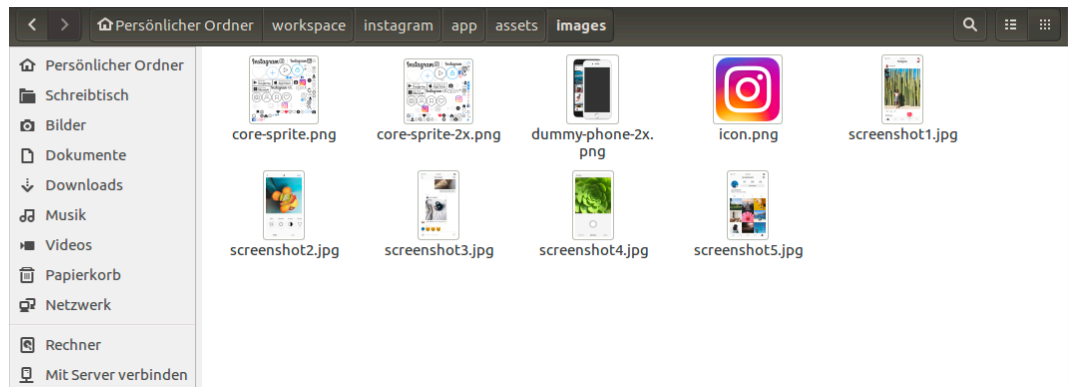


Pages#home

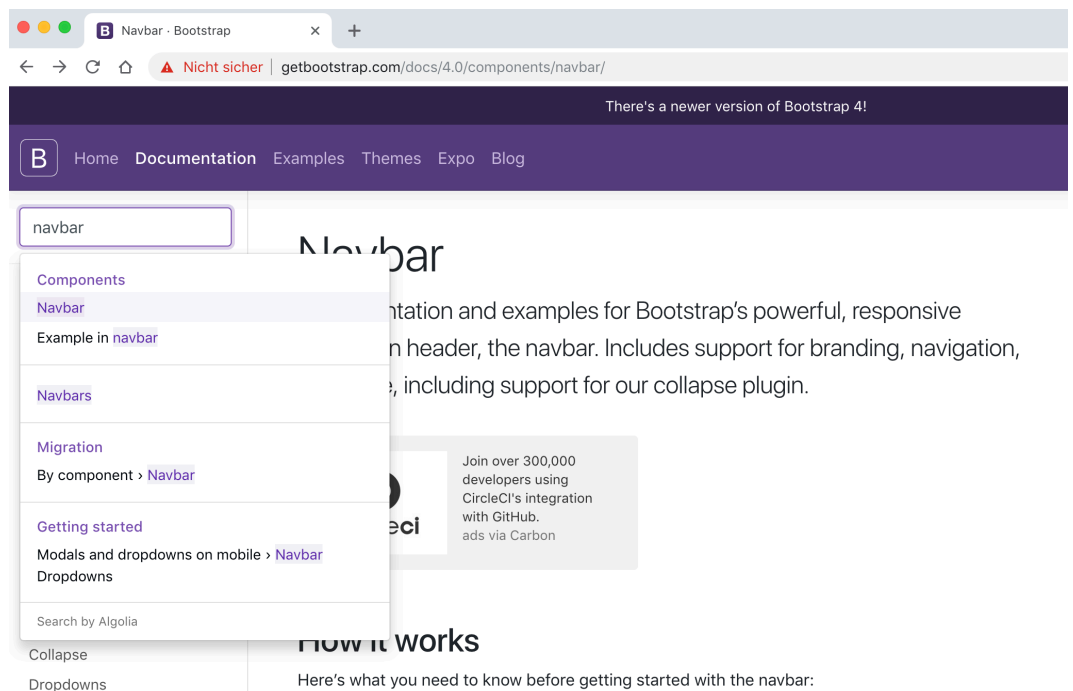
Find me in `app/views/pages/home.html.erb`

Instagram Menu erstellen

Kopieren Sie alle Bilder aus dem Ordner sftp://sftp.iet-gibb.ch/shmodules/iet-151/03_Arbeitsblatter/instagram_files/images in `workspace/instagram/app/assets/images`



Gehen Sie nun auf <http://getbootstrap.com/docs/4.0/components/navbar/> und suchen Sie nach navbar



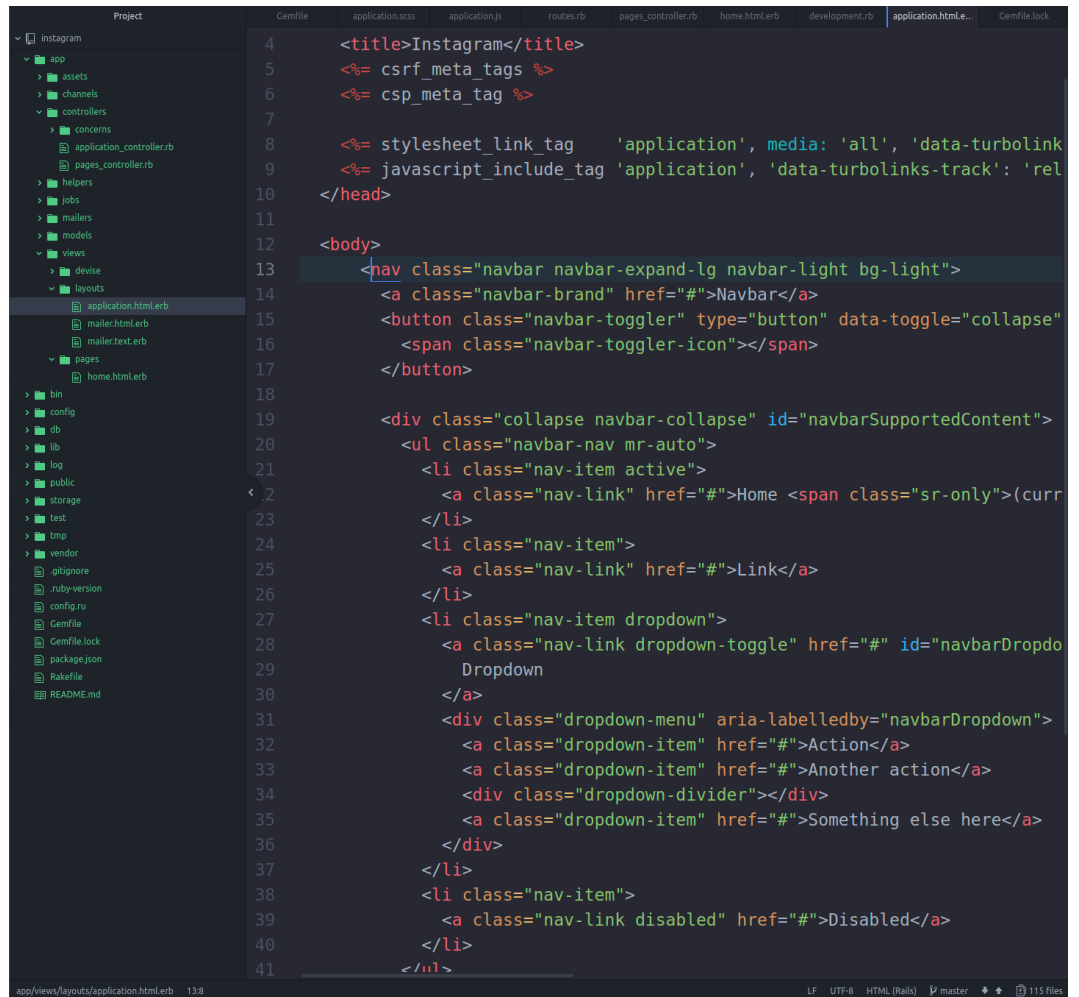
Kopieren Sie den HTML-Code

```
<nav>
```

...

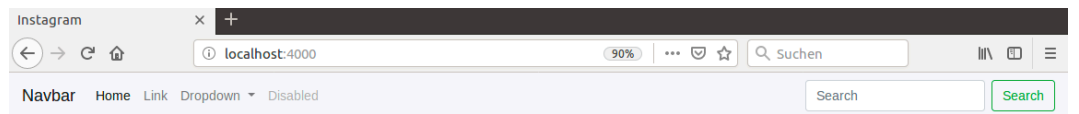
```
</nav>
```

Und fügen Sie diesen in die Datei `app/views/layouts/application.html.erb` gleich nach dem `<body>` Element.



```
4 <title>Instagram</title>
5 <%= csrf_meta_tags %>
6 <%= csp_meta_tag %>
7
8 <%= stylesheet_link_tag 'application', media: 'all', 'data-turbolink
9 <%= javascript_include_tag 'application', 'data-turbolinks-track': 'rel
10 </head>
11
12 <body>
13 <nav class="navbar navbar-expand-lg navbar-light bg-light">
14 <a class="navbar-brand" href="#">Navbar</a>
15 <button class="navbar-toggler" type="button" data-toggle="collapse"
16 <span class="navbar-toggler-icon"></span>
17 </button>
18
19 <div class="collapse navbar-collapse" id="navbarSupportedContent">
20 <ul class="navbar-nav mr-auto">
21 <li class="nav-item active">
22 <a class="nav-link" href="#">Home <span class="sr-only">(curr
23 </li>
24 <li class="nav-item">
25 <a class="nav-link" href="#">Link</a>
26 </li>
27 <li class="nav-item dropdown">
28 <a class="nav-link dropdown-toggle" href="#" id="navbarDropdo
29 Dropdown
30 </a>
31 <div class="dropdown-menu" aria-labelledby="navbarDropdown">
32 <a class="dropdown-item" href="#">Action</a>
33 <a class="dropdown-item" href="#">Another action</a>
34 <div class="dropdown-divider"></div>
35 <a class="dropdown-item" href="#">Something else here</a>
36 </div>
37 </li>
38 <li class="nav-item">
39 <a class="nav-link disabled" href="#">Disabled</a>
40 </li>
41 </ul>
```

Kontrollieren Sie die navbar im Browser:



Pages#home

Find me in `app/views/pages/home.html.erb`

Wir passen nun die Navigation genauso an, wie Sie in Instagram nutzt.

1. Entfernen Sie im obersten `<nav>`-Element das die CSS-Klasse `bg-light`. Damit entfernen wir den hellgrauen Hintergrund im Menu.
2. Kopieren Sie das `<form>`-Element und fügen Sie es unterhalb des `<button>`-Elementes in das `<div>`-Element ein:

```
<body>
  <nav class="navbar navbar-expand-lg navbar-light">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" ...
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" ...>
      <form class="form-inline my-2 my-lg-0">
        <input class="form-control mr-sm-2" type="search" ...>
        <button ... type="submit">Search</button>
      </form>
```

3. Löschen Sie folgende Elemente aus der Liste:

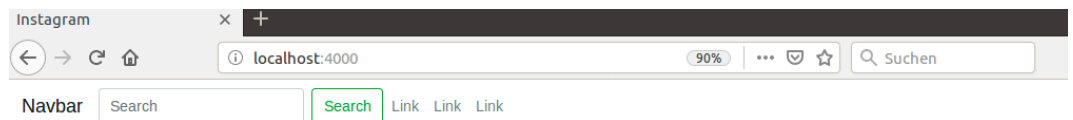
```
<li class="nav-item active">
  <a class="nav-link" href="#">Home <span class="sr-
only">(current)</span></a>
</li>
```

```
<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
    Dropdown
  </a>
  <div class="dropdown-menu" aria-labelledby="navbarDropdown">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="#">Something else here</a>
  </div>
</li>
<li class="nav-item">
  <a class="nav-link disabled" href="#">Disabled</a>
</li>
```

4. Ergänzen Sie den Code mit drei Links so dass am Schluss der Body wie folgt aussieht:

```
<body>
  <nav class="navbar navbar-expand-lg navbar-light">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <form class="form-inline my-2 my-lg-0">
        <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
      </form>
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
      </ul>
    </div>
  </nav>
  <p class="notice"><%= notice %></p>
  <p class="alert"><%= alert %></p>
  <%= yield %>
</body>
```

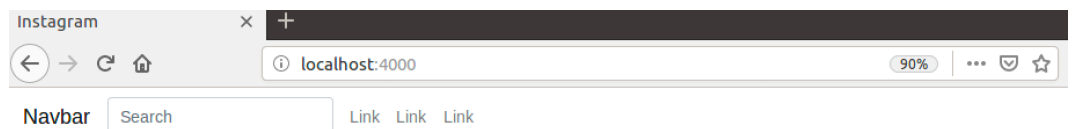
Wenn alles geklappt hat, sollte das Resultat der Startseite so aussehen:



Pages#home

Find me in app/views/pages/home.html.erb

Entfernen Sie nun den grünen Search-Button:



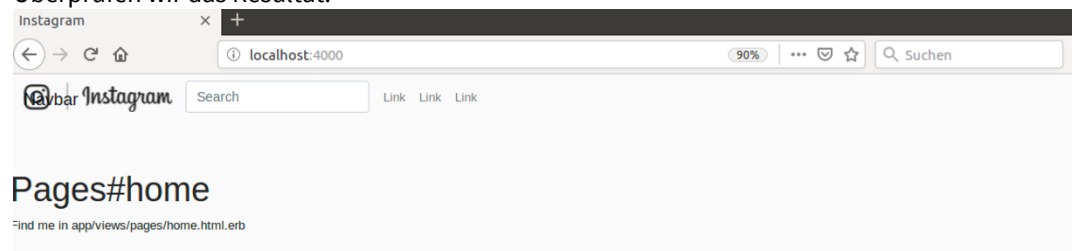
Pages#home

Find me in app/views/pages/home.html.erb

- Ergänzen Sie die Zeile mit dem <a>-Element mit core-sprite hide-text:
Navbar
- Die CSS Klassen existieren noch nicht deshalb müssen wir diese in app/assets/stylesheets/application.scss wie folgt ergänzen:

```
1  @import "bootstrap";
2
3  body {
4    background-color: #fafafa;
5    font-size: 14px;
6    color: #262626;
7  }
8
9  .core-sprite{
10   background-image: image-url("core-sprite-2x.png");
11   background-size: 405px 379px;
12   background-repeat: no-repeat;
13 }
14
15 .navbar-brand{
16   background-position: -176px 0;
17   height: 35px;
18   width: 176px;
19 }
```

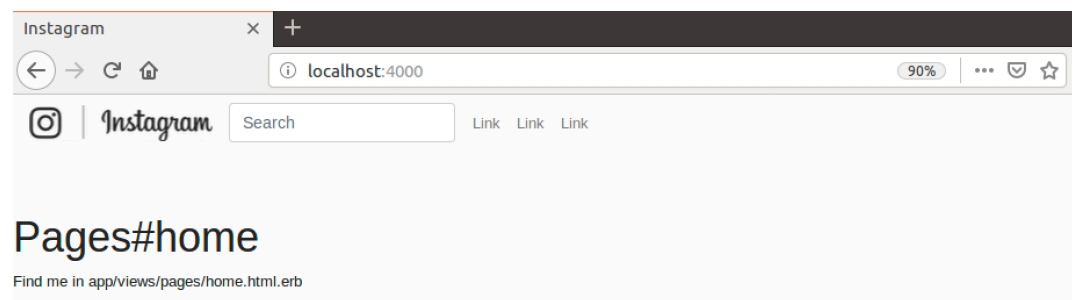
Überprüfen wir das Resultat:



7. Wir sehen nun, dass die navbar. Mit dem Instagram Logo kollidiert. Hierzu müssen wir im CSS noch `hide-text` definieren.

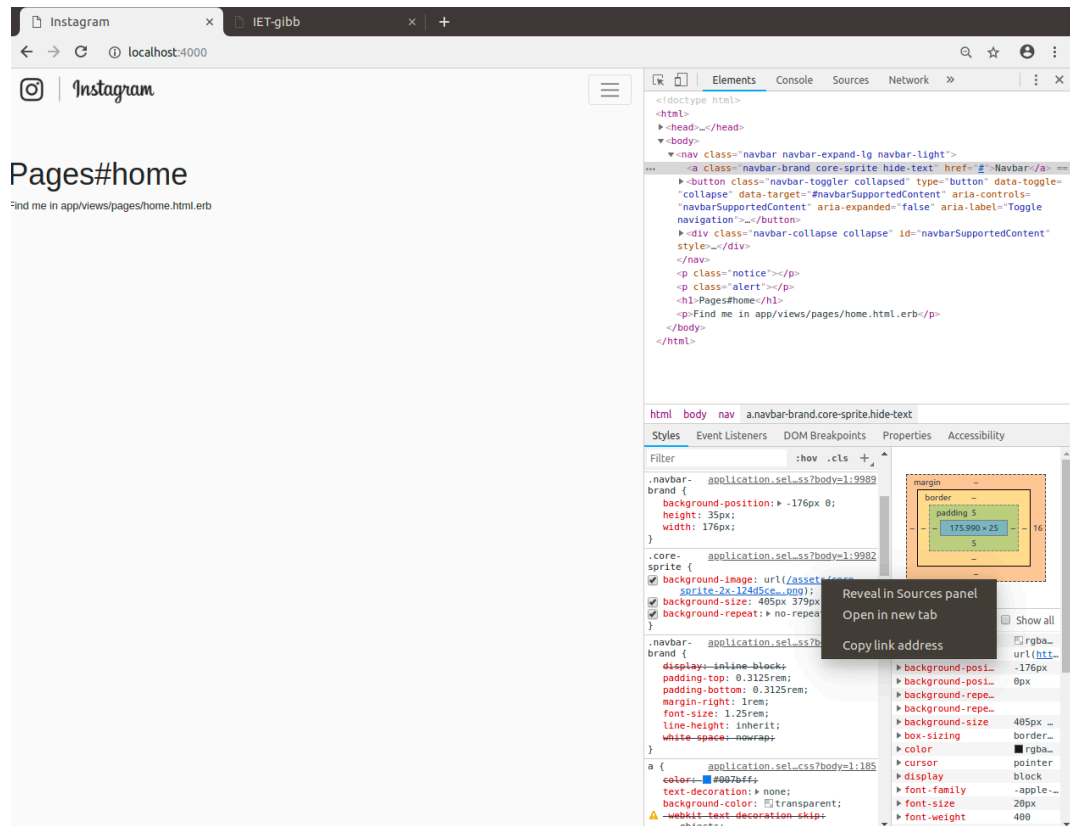
```
.hide-text{
  display: block;
  overflow: hidden;
  text-indent: 100%;
  white-space: nowrap;
}
```

Somit erreichen wir folgendes Resultat:

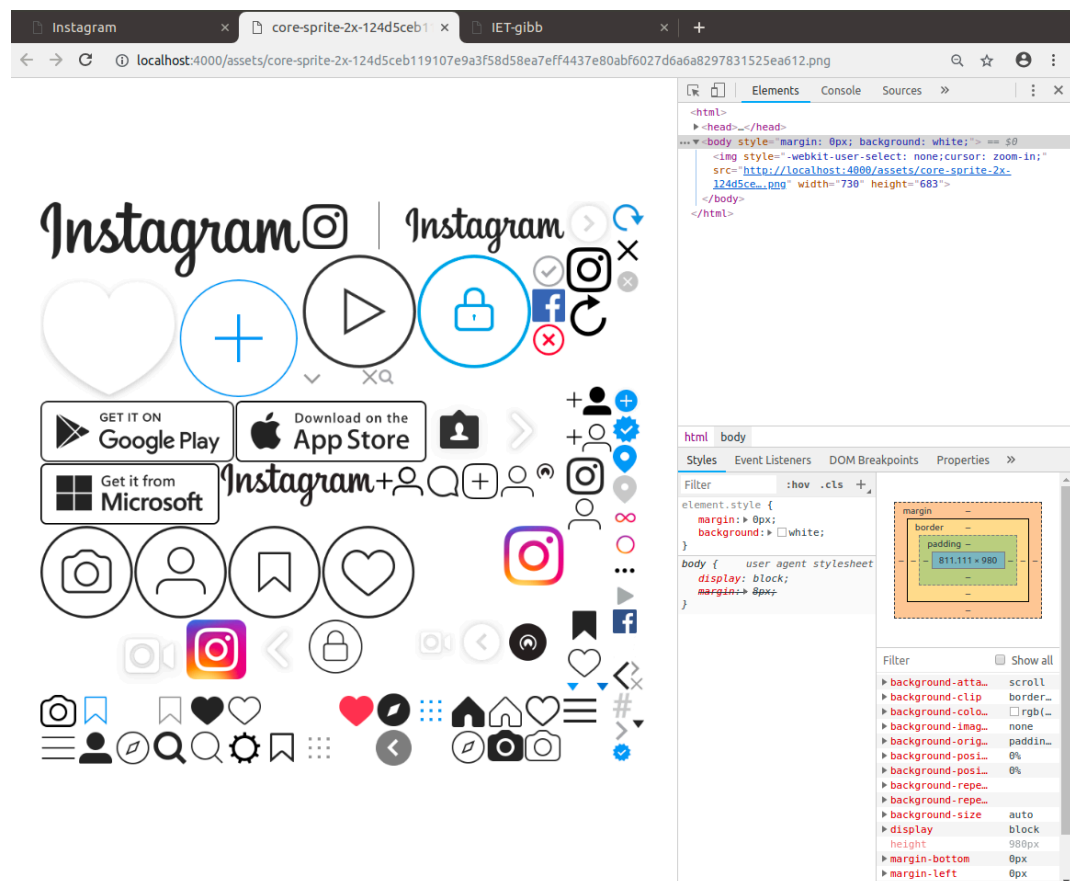


CSS-Klasse .core-sprite

Untersuchen wir kurz die CSS-Klasse `core-sprite`. Dazu öffnen wir den Browser und untersuchen die Webseite. Suchen Sie die CSS-Klasse `core-sprite` und öffnen Sie das Bild `/assets/core-sprite-2x-124d5ceb119107e9a3f58d58ea7eff4437e80abf6027d6a6a8297831525ea612.png` in einem neuen Tab.



Wir sehen nun ein «wildes» Bild:



Erklären Sie die Funktionsweise dieses Bildes anhand der CSS-Klassen `core-sprite`, `navbar-brand` und `hide-text`:



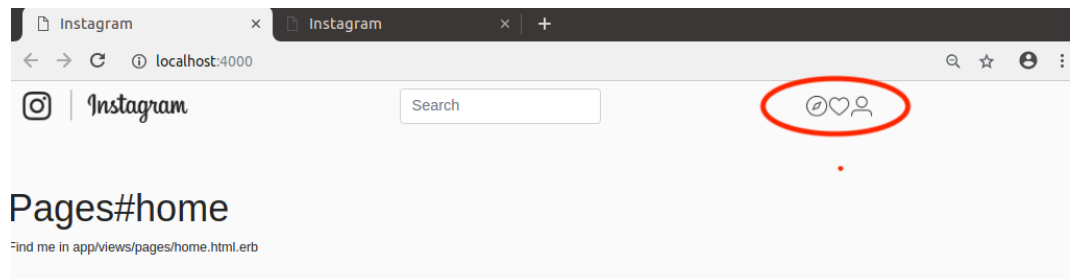
Instagram Menu layouten

Damit unser Menu langsam das Gesicht von Instagram erhält, müssen wir einige CSS-Klassen im Formular und den Links hinterlegen:

- › <form> bekommt die CSS-Klasse `ml-md-auto`
- › Das erste <a>-Element bekommt die CSS-Klassen `core-sprite`, `explore-icon` und `hide-text`. Der Link soll Explore heissen.
- › Das zweite <a>-Element bekommt die CSS-Klassen `core-sprite`, `notification-icon` und `hide-text`. Der Link soll Notification heissen.
- › Das dritte <a>-Element bekommt die CSS-Klassen `core-sprite`, `profile-icon` und `hide-text`. Der Link soll Profile heissen.

```
<nav class="navbar navbar-expand-lg navbar-light">
  <a class="navbar-brand core-sprite hide-text" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <form class="form-inline my-2 my-lg-0 ml-md-auto">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
    </form>
    <ul class="navbar-nav mr-auto ml-md-auto">
      <li class="nav-item">
        <a class="nav-link core-sprite explore-icon hide-text" href="#">Explore</a>
      </li>
      <li class="nav-item">
        <a class="nav-link core-sprite notification-icon hide-text" href="#">Notification</a>
      </li>
      <li class="nav-item">
        <a class="nav-link core-sprite profile-icon hide-text" href="#">Profile</a>
      </li>
    </ul>
  </div>
</nav>
```

Nun müssen wir noch die entsprechenden CSS-Klassen definieren, die in `app/assets/stylesheets/application.scss` formuliert werden. Suchen Sie die richtige Startkoordinate sowie die notwendigen Breiten und Höhen, damit folgende Icons erscheinen.



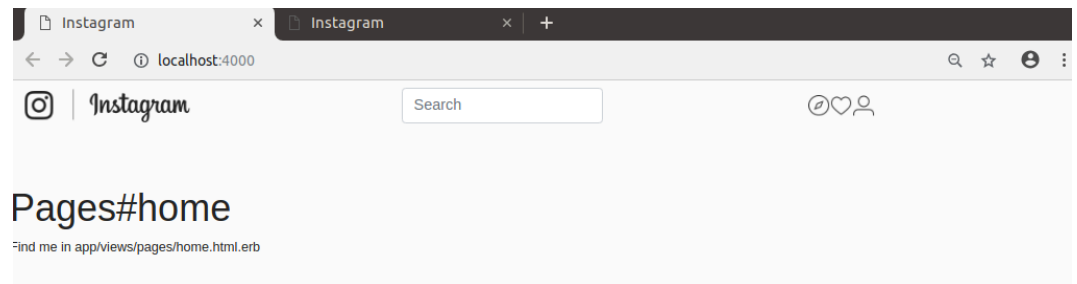
```
.explore-icon {
  background-position: _____;
  height: 24px;
  width: 24px;
}

.notification-icon {
  background-position: _____;
  height: 24px;
  width: 24px;
}

.profile-icon {
  background-position: _____;
  height: 24px;
  width: 24px;
}
```

Tipp: Sollten die Icons nicht vollständig den Text überdecken, ändern Sie in der CSS-Klasse `.hide-text` folgendes Attribut `text-indent: 110%`; von 100% auf 110% oder mehr.

Die Navigation sollte nun so aussehen:

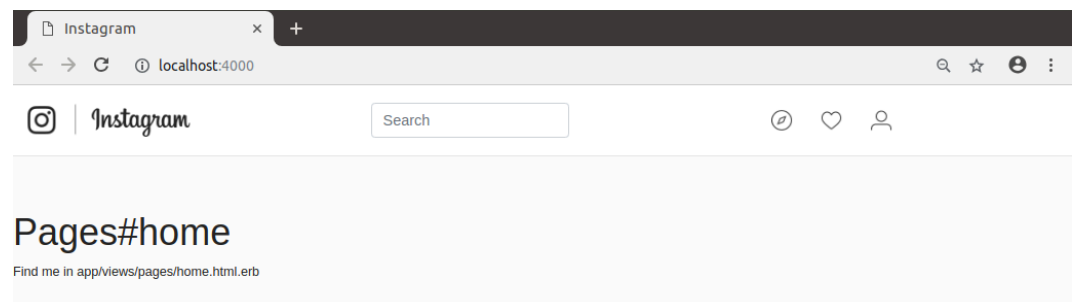


Leider sind die drei icons noch Nahe bei einander und wir schaffen nun ein wenig mehr Platz mit der CSS Erweiterung:

```
.nav-item:not(first-child){  
    margin-left: 30px;  
}
```

Mit der CSS-Klasse `.navbar` können wir einen schönen Rand um unser Menu gestalten:

```
.navbar {  
    background-color: #fff;  
    border-bottom: 1px solid rgba(0,0,0,.0975);  
    height: 77px;  
}
```



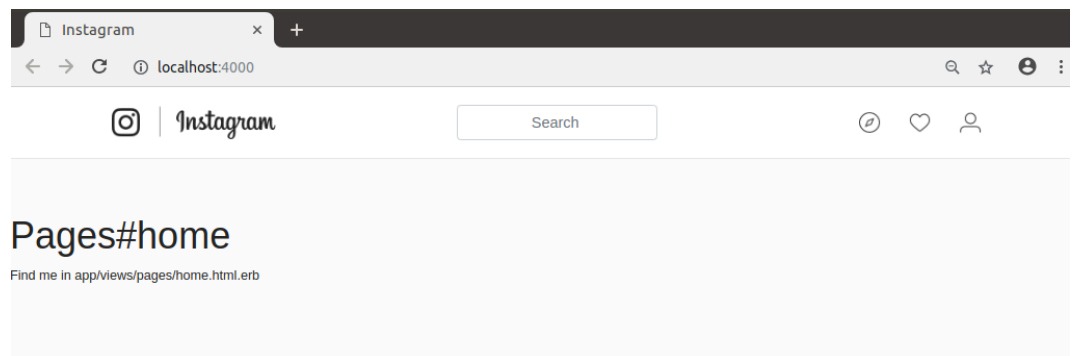
Damit wir das Menu noch besser im Instagram-Style ausrichten können, müssen wir um unser Menu einen Container definieren:

```
<body>  
    <nav class="navbar navbar-expand-lg navbar-light">  
        <div class="container">  
            <a class="navbar-brand core-sprite hide-text"  
href="#">Navbar</a>  
            ...  
        </div>  
    </nav>  
    <p class="notice"><%= notice %></p>  
    <p class="alert"><%= alert %></p>  
    <%= yield %>  
</body>
```


Als nächstes zentrieren wir den Text Search im Suchfeld mit `text-center` und löschen `Margin-Right Auto` aus der unsortierten Liste ``.

```
<nav class="navbar navbar-expand-lg navbar-light">
  <div class="container">
    <a class="navbar-brand core-sprite hide-text" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-cont
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <form class="form-inline my-2 my-lg-0 ml-md-auto">
      <input class="form-control mr-sm-2 text-center" type="search" placeholder="Search" aria-label="Search">
    </form>
    <ul class="navbar-nav ml-md-auto">
      <li class="nav-item">
        <a class="nav-link core-sprite explore-icon hide-text" href="#">Explore</a>
      </li>
      <li class="nav-item">
        <a class="nav-link core-sprite notification-icon hide-text" href="#">Notification</a>
      </li>
      <li class="nav-item">
        <a class="nav-link core-sprite profile-icon hide-text" href="#">Profile</a>
      </li>
    </ul>
  </div>
</div>
</nav>
```

Im Browser muss das Resultat nun so aussehen:



Partial-view für `<nav>` erstellen

Damit die Datei `app/views/layouts/application.html.erb` nicht unendlich wächst, können wir nun das Menu auslagern. Erstelle Sie folgenden Ordner und die Datei ohne Controller `app/views/shared/_navbar.html.erb`. Die Kennzeichnung Unterstrich `«_»` definiert eine Partialseite und deshalb sprechen wir von einer Navbar mit Partial View.

Kopieren Sie das HTML-Element `<nav>` aus

`app/views/layouts/application.html.erb` und fügen Sie es in die Datei

`app/views/shared/_navbar.html.erb`. Danach löschen Sie `<nav>` aus

`app/views/layouts/application.html.erb`.

Die Integration der Partialseite erfolgt durch folgenden Ruby Code:

```
<%= render 'shared/navbar' %>
```

Der Unterstrich muss nicht wie im Dateinamen angegeben werden. Rails versteht, dass mit `<%= render 'shared/navbar' %>` die Partialseite `_navbar.html.erb` aufgerufen werden muss.

<%= link_to %>

Als nächstes wollen wir mit dem Symbol



einen Link auf die Rootseite <http://localhost:4000> unserer Instagram Applikation erstellen. Ersetzen Sie in der Datei `app/views/shared/_navbar.html.erb` folgende Zeile

```
<a class="navbar-brand core-sprite hide-text" href="#">Navbar</a>
```

mit

```
<%=link_to "Icon", root_path, class: "navbar-brand core-sprite  
hide-text"%>
```

Als nächstes bearbeiten wir die View `app/views/pages/home`:

```

1  <%= link_to "Log out", destroy_user_session_path, method: :delete%>
2  <h1>Pages#home</h1>
3  <p>Find me in app/views/pages/home.html.erb</p>
4

```

1. Erklären Sie die 1. Zeile. Verwenden dazu den Befehl `rails routes` im Terminal.
2. Wohin geht der Link?

login / logout

Ergänzen Sie den Controller `pages/home`

```
1 class PagesController < ApplicationController
2   def home
3     if !user_signed_in?
4       redirect_to new_user_session_path
5     end
6   end
7 end
```

Die Methode `home` checkt, ob ein Benutzer nicht eingeloggt ist. Ist dies der Fall, wird er auf die Loginseite geleitet.

Testen Sie die Startseite, erfassen Sie einem Benutzer, loggen Sie sich ein und melden sich wieder ab.

Bei Fehlermeldungen suchen Sie selber eine Lösung oder sprechen Sie sich mit dem Nachbarn ab. Falls Sie gar keine Lösung finden, wenden Sie sich an die Lehrperson und erwähnen Sie das Problem in der Quicknote.

Auftrag: Quicknote AB151-02

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_AB151-02.pdf*

Beispiel: INF17R_Maurer_Ralph_QN_AB151-02.pdf

Was beinhaltet die Quicknote AB151-02?

Zusammenfassung AB151-2

Die Quicknote soll eine kurze prägnante Zusammenfassung des Dokuments AB151-02 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?

Lösungen der Aufgaben:

Komprimieren Sie Ihre Rails-App und geben Sie diese mit der Quicknote ab.

Name des Archivs: *Klasse_Name_Vorname_Insta_AB151-02.zip*

Beispiel: INF17R_Maurer_Ralph_Insta_AB151-02.zip

Abschliessende Reflexion über das Gelernte:

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