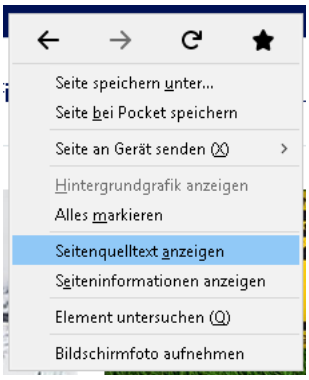
Das Internet ist eine wahre Fundgrube von Programmbeispielen. **w3schools** [1] und **HTML.net** [2] bieten Einführungen in **CSS** [3], **HTML** [4], **JS** [5] und **PHP** [6] .



Der Quellcode einer Webseite kann mit Klick der rechten Maustaste und Auswahl des Menüpuktes 'Seitenquelltext ansehen' mit Ctrl-A markiert und für die eigene Anwendung kopiert werden.

Auf der Homepage von **Bootstrap** [7] werden viele Beispiele angeboten. Darunter ist auch eine **Blog** [8]-Seite.

Eine Kopie des Quelltexts dieses Beispiels wird in der Datei '\resources\views\blogs\bs.blade.php' gespeichert.

Das Beispiel verwendet folgende Bibliotheken

- **Bootstrap** [9] ist ein freies Frontend-CSS-Framework. Es enthält auf HTML und CSS basierende Gestaltungsvorlagen für Typografie, Formulare, Buttons, Tabellen, Grid-Systeme, Navigations- und andere Oberflächen-gestaltungselemente sowie zusätzliche, optionale JavaScript-Erweiterungen. Gemäss Dokumentation benötigt Bootstrap jQuery.
- **JQuery** [10] ist eine schnelle, kleine, umfangreiche JavaScript Bibliothek. Mit jQuery wird das Arbeiten mit HTML, Behandeln von Events, Animation für die Mehrzahl der Internet-Browser bedeutend einfacher.
- **Font-Awesome** [11] bietet skalierbare vektorisierte Icons mit der Möglichkeit diese zu bearbeiten - Grösse, Farbe, Schatten – einfach alles was man mit CSS machen kann.

---

[1] https://www.w3schools.com/js/

[2] http://de.html.net/

[3] **CSS** (**C**ascading **S**tyle **S**heets) ist eine Stilsprache, die das Aussehen von HTML-Dokumenten definiert. CSS kann man z.B. zum Festlegen von Schriftarten, Farben, Rändern, Linien, Höhen, Breiten, Hintergrundbildern und vielen anderen Sachen benutzen. Da CSS-Dateien auf anderen CSS-Dateien aufbauen können, erhielten sie den den Beinamen Caskade. Die Erstellung von Web-Anwendungen ist ohne die Verwendung von CSS-Dateien kaum mehr vorstellbar.

[4] **HTML** (**H**yper**T**ext **M**arkup **L**anguage) ist eine textbasierte Sprache zur Strukturierung digitaler Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten.

[5] **JS** (**J**ava**S**cript) ist eine Sprache, die für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML und CSS zu erweitern.

[6] **PHP** (**P**ersonal **H**ome **P**age Tools – Hypertext Preprocessor) ist eine weit verbreitete und für den allgemeinen Gebrauch bestimmte Open Source-Skriptsprache, welche speziell für die Webprogrammierung geeignet ist und in HTML eingebettet werden kann.

[7] http://getbootstrap.com/docs/4.0/examples/

[8] Ein Blog ist ein auf einer Website geführtes und damit meist öffentlich einsehbares Tagebuch oder Journal, in dem mindestens eine Person, der Blogger, international auch Weblogger genannt, Aufzeichnungen führt, Sachverhalte protokolliert („postet“) oder Gedanken niederschreibt. Häufig ist ein Blog eine chronologisch sortierte Liste von Einträgen. Der Blogger ist Hauptverfasser des Inhalts, häufig sind die Beiträge aus der Ich-Perspektive geschrieben. Ein Blog bildet ein Medium zur Darstellung von Aspekten des eigenen Lebens und von Meinungen zu spezifischen Themen, je nach Professionalität bis in die Nähe einer Internet-Zeitung mit besonderem Gewicht auf Kommentaren. Oft sind auch Kommentare oder Diskussionen der Leser über einen Artikel möglich.

[9] http://getbootstrap.com/

[10] https://jquery.com/

[11] http://fontawesome.io/

- **ie10-viewport-bug-workaround.js** korrigiert eine Schwachstelle von Windows

Diese Pakete können durch Web-Anwendungen jeweils von entsprechenden Web-Seiten geladen oder lokal installiert werden.

Mit dem Laden der Pakete zur Ausführungszeit, greift die Anwendung immer auf die aktuellsten Pakete zu. Möglicherweise handelt man sich dabei Probleme ein, falls eine Aktualisierung nicht mehr mit der Web-Anwendung kompatibel ist oder die URL verändert wird. Mit der lokalen Variante greift mann immer auf die gleichen Pakete zu.

Mit **npm** [12] (**N**ode **P**ackage **M**anager) hat man Zugriff auf über 350.000 Open Source Pakete. Um **npm** brauchen zu können, muss **Node** [13] installiert sein. Dies kann mit

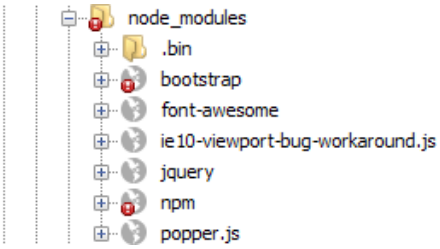
```
node –v
```

überprüft werden.

Auf der Webseite von npm ist angegeben, dass die im Beispiel verwendeten Bibliotheken mit

```
npm i npm                                    14
npm i jquery                                 15
npm i popper.js
npm i bootstrap
npm i font-awesome
npm i ie10-viewport-bug-workaround.js
```

installiert werden können. **npm** installiert die Pakete im Verzeichnis \node_modules.



Die Dateien

```
bootstrap.css
font-awesome.css
bootsstrap.js
ie10-viewport-bug-workaround.js
jquery.js
popper.js
```

müssen in die Verzeichnisse \css und \js und die Fonts in das Verzeichnis \fonts kopiert werden.

[12] https://www.npmjs.com/

[13] https://nodejs.org/de/

[14] **npm i npm** installiert die neuste Version von npm

[15] Bei der Aktualisierung von npm wird darauf hingewiesen, dass auch jquery und popper.js geladen werden muss,

Packagist [16] verwaltet ebenfalls Open-Source Pakete, welche mit **Composer** [17] installiert werden können.

Um gewisse Elemente dem persönlichen Geschmack anzupassen, wurde eine neue CSS-Datei erstellt.

\css\fjk.css

```
note10 {
    font-family: verdana;
    font-size: 10px;
}
note20 {
    font-family: verdana;
    font-size: 20px;
}
verdana {
    face:Verdana, Arial, Helvetica, sans-serif;
}
```

Damit kann die Schriftfamilie der Homepage festgelegt und die Elemente 'note10' und 'note20' benutzt werden, um Texte in unterschiedlicher Grösse darzustellen. Diese Datei kann nach Belieben angepasst oder weggelassen werden.

Das von LaravelCollective angebotene Paket unterstützt HTML und den Aufbau von Formulare. Das Paket kann mit

```
composer require laravelcollective/html
```

geladen werden. Dieser Aufruf von Composer ergänzt die Datei composer.json:

```
"require": {
    "php": ">=5.6.4",
    "laravel/framework": "5.4.*",
    "laravel/tinker": "~1.0",
    "laravelcollective/html": "^5.5"
},
```

Auf der Homepage von **LaravelCollective**[18] angegeben, dass die Datei config\app.php wie folgt zu ergänzen ist.

```
'providers' => [
  Collective\Html\HtmlServiceProvider::class,
],
'aliases' => [
  'Form' => Collective\Html\FormFacade::class,
  'Html' => Collective\Html\HtmlFacade::class,
],
```

Damit stehen nun alle Grundlagen für die Entwicklung von MyWeb lokal und optimal zur Verfügung.

Blade [19] ermöglicht neben anderem das Auslagern von Code und damit das Verwenden eines generellen Layouts. Blade-Dateien haben neu den Typ .blade.php.

---

[16] https://packagist.org/

[17] **Composer** ist ein anwendungsorientierter Paketmanager für die Programmiersprache PHP und ist im Laravel Framework integriert.

[18] https://laravelcollective.com/

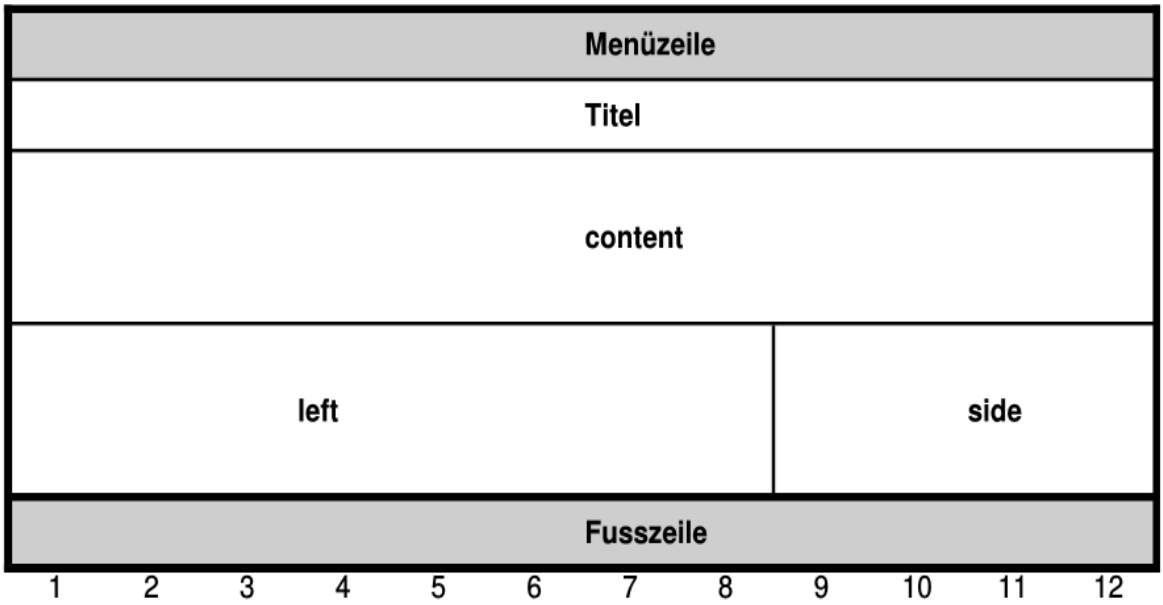
[19] https://laravel.com/docs/5.6/blade

Mit der Installation der Benutzeridentifikation wurde der Layout - /resources/views/layouts/app.blade.php eingeführt. Dieser Layouts wird für MyWeb angepasst.

\resources\views\layouts\master.blade.php

```
<!DOCTYPE html>
<html lang="de">
  <verdana>
    @include('layouts.header')
    <body>
      <div class="row">
        @include('layouts.nav')
        @yield('title')
        @yield('content')
      </div>
      <div class="row">
        <div class="col-md-8">
          @yield('left')
        </div>
        <div class="col-sm-4">
          @yield('side')
        </div>
      </div>
      <br>
      @include('layouts.footer')
      @include('layouts.scripts')
    </body>
  </verdana>
</html>
```

Bootstrap teilt eine Zeile in 12 Teile. Mit 'yield' wird Platz für einen Titel (title), die ganze Breite (content) oder zwei Spalten (left und side) reserviert. Der Layout gestaltet den Bildschirm von MyWeb wie folgt:



Mit Bootstrap kann ein Gebiet wiederum in 12 gleiche Stücke aufgeteilt werden.

Im Layout wird die Header-Section,

\resources\views\layouts\header.blade.php

```
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!-- CSRF Token -->
  <meta name="csrf-token" content="{{ csrf_token() }}">
  <title>{{ config('app.name', 'FJKarli') }}</title>
  <!-- Styles -->
  {{ Html::style('/css/app.css') }}
  {{ Html::style('/css/bootstrap.min.css') }}
  {{ Html::style('/css/font-awesome.min.css') }}
  {{ Html::style('/css/fjk.css') }}
</head>
```

und die Scripts

\resources\views\layouts\scripts.blade.php

```
<!-- JavaScript placed at the end of the document so the pages load faster  -->
{{ Html::script("/js/app.js") }}
{{ Html::script("/js/jquery.js") }}
{{ Html::script("/js/bootstrap.js") }}
{{ Html::script("/js/ie10-viewport-bug-workaround.js") }}
{{ Html::script("/js/popper.js") }}
```

nach \resources\views\layouts ausgelagert und dabei die neuen CSS und JS-Dateien eingefügt. Dabei wird mit {{ und }} LaravelCollective aktiviert.

Die Menüzeile wurde nach \resources\views\layouts\nav.blade.php ausgelagert.

```
<nav class="navbar navbar-default navbar-static-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
        data-target="#app-navbar-collapse" aria-expanded="false">
        <span class="sr-only">Toggle Navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="{{ url('/') }}">Home</a>
    </div>
    <div class="collapse navbar-collapse" id="app-navbar-collapse">
      <!-- Left Side Of Navbar -->
      <!-- Right Side Of Navbar -->
      <ul class="nav navbar-nav navbar-right">
        <!-- Authentication Links -->
        @guest
          <li><a href="{{ route('login') }}">Login</a></li>
          <li><a href="{{ route('register') }}">Registrieren</a></li>
        @else
          <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
              aria-expanded="false" aria-haspopup="true">
              {{ Auth::user()->name }} <span class="caret"></span>
            </a>
            <ul class="dropdown-menu">
              <li>
                <a href="{{ route('logout') }}"
                   onclick="event.preventDefault();
                         document.getElementById('logout-form').submit();">
                  <b>Logout</b>
                </a>
                <form id="logout-form" action="{{ route('logout') }}"
                      method="POST" style="display: none;">
                  {{ csrf_field() }}
                </form>
              </li>
            </ul>
          </li>
        @endguest
      </ul>
    </div>
  </div>
</nav>
```

Neu wird eine Fusszeile eingeführt:

\resources\view\layouts\footer.blade.php

```
<div class="panel-footer">
  F. J. Karli &copy; 2018
</div>
```

Webseiten verwenden verschiedene Felder, um Zeichenfolgen, Memos, Zahlen, usw. einzugeben.

Der dazu notwendige Code wird in das Verzeichnis '\resources\views\partials' ausgelagert.

Viele Felder sind - mit unterschiedlichen Bezeichnungen ($label) und Variablen ($value) - identisch.

\resources\views\partials\input_memo.blade.php

```
<div class="form-group">
  <div class="col-md-offset-1 col-md-3">{{ Form::label($label) }}</div>
  <div class="col-md-6">
    {{ Form::textarea($value, null,
      ['class' => 'form-control', 'rows' => '3', 'required']) }}
  </div>
</div>
```

Normalerweise ist das Feld für 'textarea' recht gross. Mit 'rows' => '3' kann die Höhe des Feldes reduziert werden.

\resources\views\partials\input_number.blade.php

```
<div class="form-group">
  <div class="col-md-offset-1 col-md-3">{{ Form::label($label) }}</div>
  <div class="col-md-6">
    {{ Form::number($value, null, ['class' => 'form-control', 'required']) }}
  </div>
</div>
```

\resources\views\partials\input_string.blade.php

```
<div class="form-group">
  <div class="col-md-offset-1 col-md-3">{{ Form::label($label) }}</div>
  <div class="col-md-6">
    {{ Form::text($value, null, ['class' => 'form-control', 'required']) }}
  </div>
</div>
```

Die Eingabe einer Mailadresse unterscheidet sich von andern Eingaben, da eine Mailadresse eine spezielle Form 'name@mailserver.TopLevelDomain' aufweisen muss. Der TopLevelDomain muss mindestens zwei Zeichen umfassen und ist meist identisch mit dem Ländercode – z. B. 'CH'. Gross- oder Kleinschreibung spielen keine Rolle.

\resources\views\partials\input_email.blade.php

```
<div class="form-group">
  <div class="col-md-offset-1 col-md-3">{{ Form::label('Mail-Adresse') }}</div>
  <div class="col-md-6">
    {{ Form::text('email', null,
      ['class' => 'form-control', 'required', 'type' => 'email']) }}
  </div>
</div>
```

Passwörter werden mit folgendem Feld erfasst:

\resources\views\partials\input_pwd.blade.php

```
<div class="form-group">
  <div class="col-md-offset-1 col-md-3">{{ Form::label('Passwort') }}</div>
  <div class="col-md-6">
    {{ Form::password('password', ['class' => 'form-control','required']) }}
  </div>
</div>
```

Werden Passwörter neu definiert, müssen sie zur Sicherheit zweimal eingegeben werden:

\resources\views\partials\input_pwdconfirm.blade.php

```
<div class="form-group">
  <div class="col-md-offset-1 col-md-3">
    {{ Form::label('Passwort wiederholen') }}
  </div>
  <div class="col-md-6">
    {{ Form::password('password_confirmation',
      ['class' => 'form-control','required']) }}
  </div>
</div>
```

Vielfach besteht die Möglichkeit, dass sich der Browser die Eingabedaten merken kann.

\resources\views\partials\input_rem.blade.php
```
<div class="form-group">
  <div class="col-md-6 col-md-offset-4">
    <div class="checkbox">
      <label>
        <input type="checkbox" name="remember" {{ old('remember') ? 'checked' : '' }}>
          Angaben speichern
      </label>
    </div>
  </div>
</div>
```

Eingaben werden normalerweise mit einer Schaltfläche abgeschlossen.

\resources\views\partials\submit.blade.php
```
<div class="form-group">
  <div class="col-md-6 col-md-offset-4">
    <br>
    {{ Form::submit($submitButtonText, ['class' => 'btn btn-primary']) }}
    <br>
  </div>
</div>
```

Auch diese Datei ist so ausgelegt, dass Schaltflächen mit unterschiedlchen Bezeichnungen ($submitButtonText) verwendet werden können.

Für Login besteht zusätzlich die Möglichkeit, einen Reset-Link anzufordern

\resources\views\partials\submit_login.blade.php
```
<div class="form-group">
  <div class="col-md-6 col-md-offset-4">
    {!! Form::submit('Login', ['class' => 'btn btn-primary']) !!}
    <a href="{{ route('password.request') }}"> Passwort-Reset-Link anfordern?</a>
  </div>
</div>
```

Viele Seiten sehen gleich aus. Neben dem Hauptspalte wird oft eine Seitenspalte abgebildet.

\resources\views\layouts\side.blade.php
```
<h4>Notizen</h4>
Diese Seite dient vor allem als Link zu:
<ol class="list-unstyled">
  <li><a href="http://Musik.FJKarli.ch">Musik</a></li>
</ol>
<h4>Weitere Links</h4>
<ol  class="list-unstyled">
  <li><a href="https://laravel.com/docs">Documentation</a></li>
  <li><a href="https://laracasts.com">Laracasts</a></li>
  <li><a href="https://laravel-news.com">News</a></li>
  <li><a href="https://forge.laravel.com">Forge</a></li>
  <li><a href="https://github.com/laravel/laravel">GitHub</a></li>
</ol>
```

Nun müssen die Dateien der Identifizierung wie folgt angepasst und gleichzeitig übersetzt werden.

\resources\views\auth\login.blade.php
```
@extends('layouts.master')
@section('content')
<div class="panel panel-default col-md-offset-1">
  <div class="panel-heading">Login</div>
  <div class="panel-body">
    {{ Form::open(['class' => 'form', 'route' => 'login']) }}
      @include('partials.input_email')
      @include('partials.input_pwd')
      @include('partials.input_rem')
      @include('partials.submit_login')
    {{ Form::close() }}
  </div>
</div>
```

\resources\views\auth\register.blade.php
```
@extends('layouts.master')
@section('content')
<div class="panel panel-default col-md-offset-1">
  <div class="panel-heading">Registrieren</div>
```

```
  <div class="panel-body">
    {{ Form::open(['class' => 'form', 'route' => 'register']) }}
      @include('partials.input_string', ['label' => 'Name Vorname', 'value' => 'name'])
      @include('partials.input_email')
      @include('partials.input_pwd')
      @include('partials.input_pwdconfirm')
      @include('partials.submit', ['submitButtonText' => 'Registrieren'])
    {{ Form::close() }}
  </div>
</div>
@endsection
```

\resources\views\auth\password\email.blade.php

```
@extends('layouts.master')
@section('content')
<div class="panel panel-default col-md-offset-1">
  <div class="panel-heading">Passwort-Reset-Link anfordern</div>
  <div class="panel-body">
    @include('layouts.session_message')
    {!! Form::open(['class' => 'form', 'route' => 'password.email']) !!}
      @include('partials.input_email')
      @include('partials.submit',
        ['submitButtonText' => 'Passwort-Reset-Link anfordern'])
    {!! Form::close() !!}
  </div>
</div>
@endsection
```

\resources\views\auth\password\reset.blade.php

```
@extends('layouts.master')
@section('content')
<div class="panel panel-default col-md-offset-1">
  <div class="panel-heading">Passwort setzen</div>
  <div class="panel-body">
    {!! Form::open(['class' => 'form', 'route' => 'password.request']) !!}
      <input type="hidden" name="token" value="{{ $token }}">
      @include('partials.input_email')
      @include('partials.input_pwd')
      @include('partials.input_pwdconfirm')
      @include('partials.submit',
        ['submitButtonText' => 'Neues Passwort übernehmen!'])
    {!! Form::close() !!}
  </div>
</div>
@endsection
```