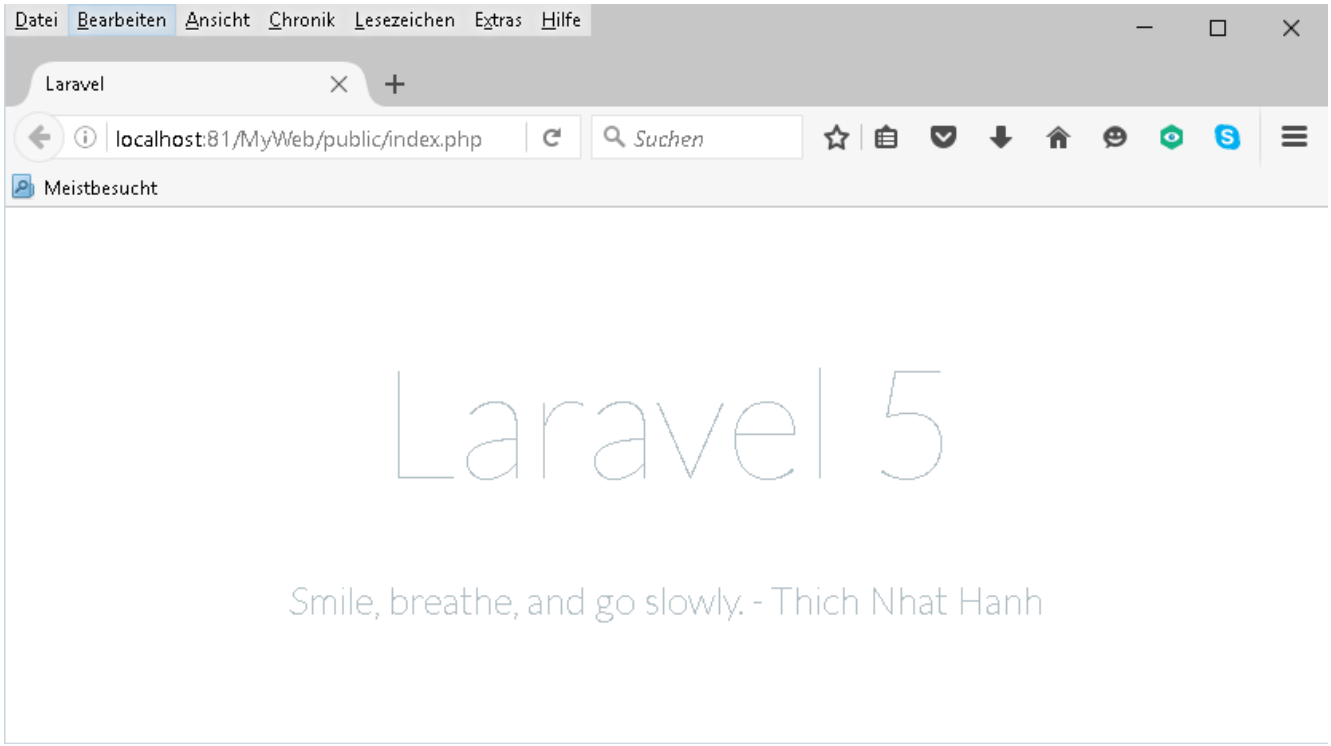
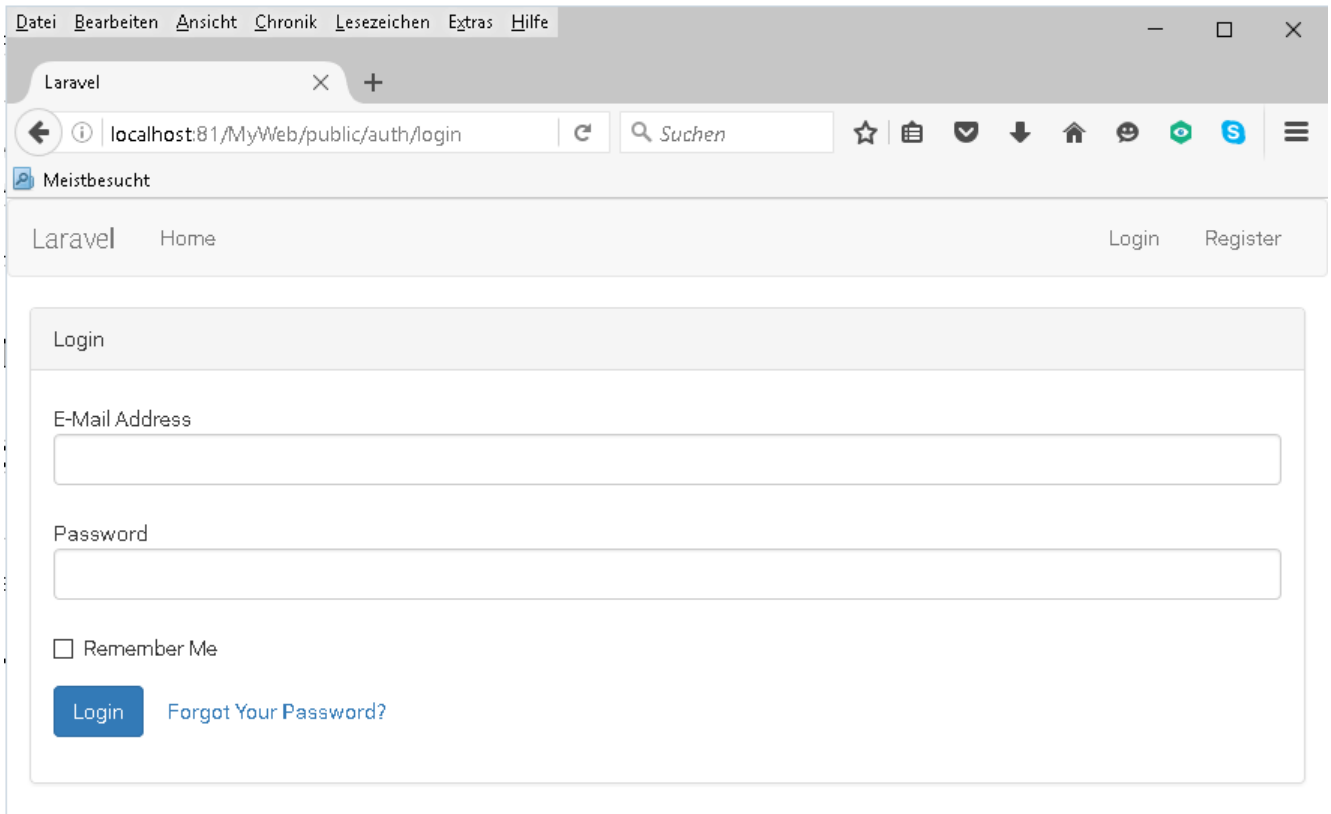
Wie schon erwähnt, erstellt Composer im Verzeichnis D:\Web\MyWeb eine funktionsfähige Web-Anwendung:

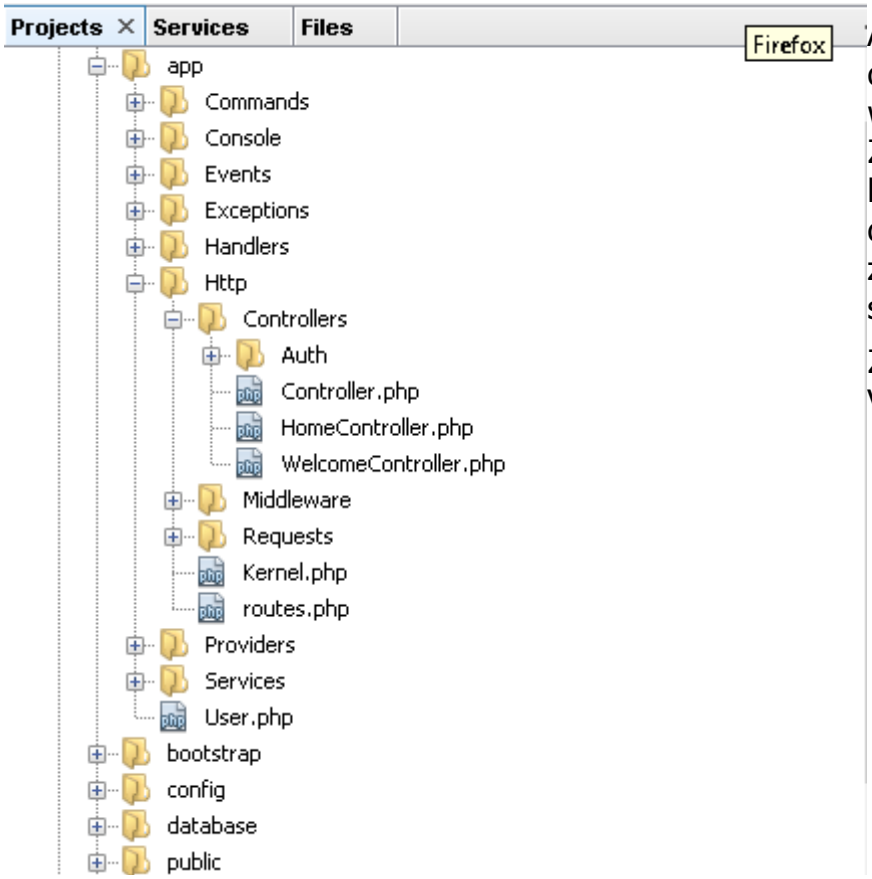


Ruft man die Anwendung mit http://localhost:81/MyWeb/public/home erscheint folgende Seite:



Wieso erscheint hier die Seite von public/auth/login?

Durch den Aufruf von Composer wurde folgende Verzeichnisstruktur erzeugt:



Auf den ersten Blick mag diese Struktur recht verwirrend sein. Im Lauf der Zeit wird sich der Nebel lichten, dies vor allem aus dem Grund, da die Verzeichnisstruktur von Laravel sehr logisch aufgebaut ist.

Zur Erklärung sind einige Verzeichnisse geöffnet.

Eingaben in Internet-Browser wie

http://localhost:81/MyWeb/public/

oder

http://localhost:81/MyWeb/public/home

werden in Laravel nach Angaben in der Datei app\Http\routes.php abgearbeitet. Der erste Teil 'http://localhost:81/MyWeb/public' wird vorausgesetzt und muss nicht angegeben werden.

```
<?php
...
Route::get('/', 'WelcomeController@index');
Route::get('home', 'HomeController@index');
Route::controllers([
  'auth' => 'Auth\AuthController',
  'password' => 'Auth\PasswordController',
]);
```

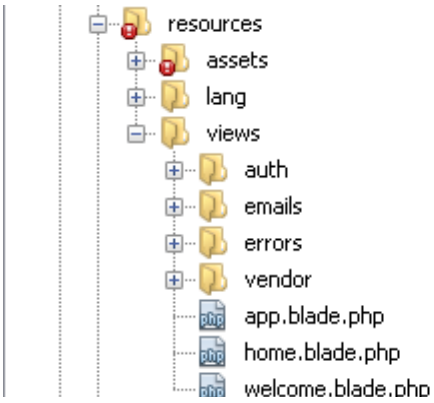
Wird die Startadresse '/' eingegeben, wird die Kontrolle wie in der Route angegeben an die Funktion index im WelcomeController übergeben.

Controller werden im Verzeichnis app\Http\Controllers\ gespeichert.

```
<?php namespace App\Http\Controllers;
class WelcomeController extends Controller {
  /*
  |--------------------------------------------------------------------------
  | Welcome Controller
  |--------------------------------------------------------------------------
  |
  | This controller renders the "marketing page" for the application and
  | is configured to only allow guests. Like most of the other sample
  | controllers, you are free to modify or remove it as you desire.
  |
  */
  /**
   * Create a new controller instance.
   *
   * @return void
   */
  public function __construct()
  {
    $this->middleware('guest');
  }
  /**
   * Show the application welcome screen to the user.
   *
   * @return Response
   */
  public function index()
  {
    return view('welcome');
  }
}
```

Im Kommentar ist angegeben, dass dieser Kontroller die Aktivitäten von allgemeinen Besuchern behandelt. So können beispielsweise allgemeine Seiten oder Seiten mit Werbung gesteuert werden. Die Überwachung geschieht mit dem Aufruf von $this->middleware('guest').

Views werden im Verzeichnis resources\views\ oder in Unterverzeichnissen davon gespeichert.



Die mit der Hauptadresse aufgerufene Funktion index ruft

```
return view('welcome');
```

die Willkommensseite welcome.blade.php auf. Zu bemerken ist, dass auch hier Pfadangaben (resources\views) und Dateierweiterungen (.blade.php) nicht angegeben werden müssen.

welcome.blade.php enthält neben einigen Formatierungsangaben die Ausgabe von Laravel 5 und einem speziellen Text.

```
<html>
  <head>
    <title>Laravel</title>
        <link href='//fonts.googleapis.com/css?family=Lato:100'
          rel='stylesheet' type='text/css'>
... Formatierungsangaben
  </head>
  <body>
    <div class="container">
      <div class="content">
        <div class="title">Laravel 5</div>
        <div class="quote">{{ Inspiring::quote() }}</div>
      </div>
    </div>
  </body>
</html>
```

Die Formatierungsangaben

```
    <style>
      body { ... }
      .container { ... } ...
    </style>
```

legen fest, wie die verschiedenen Klasse darzustellen sind.

Der Aufruf MyWeb/public/home bewirkt, wie in der folgenden Route angegeben

```
Route::get('home', 'HomeController@index');
```

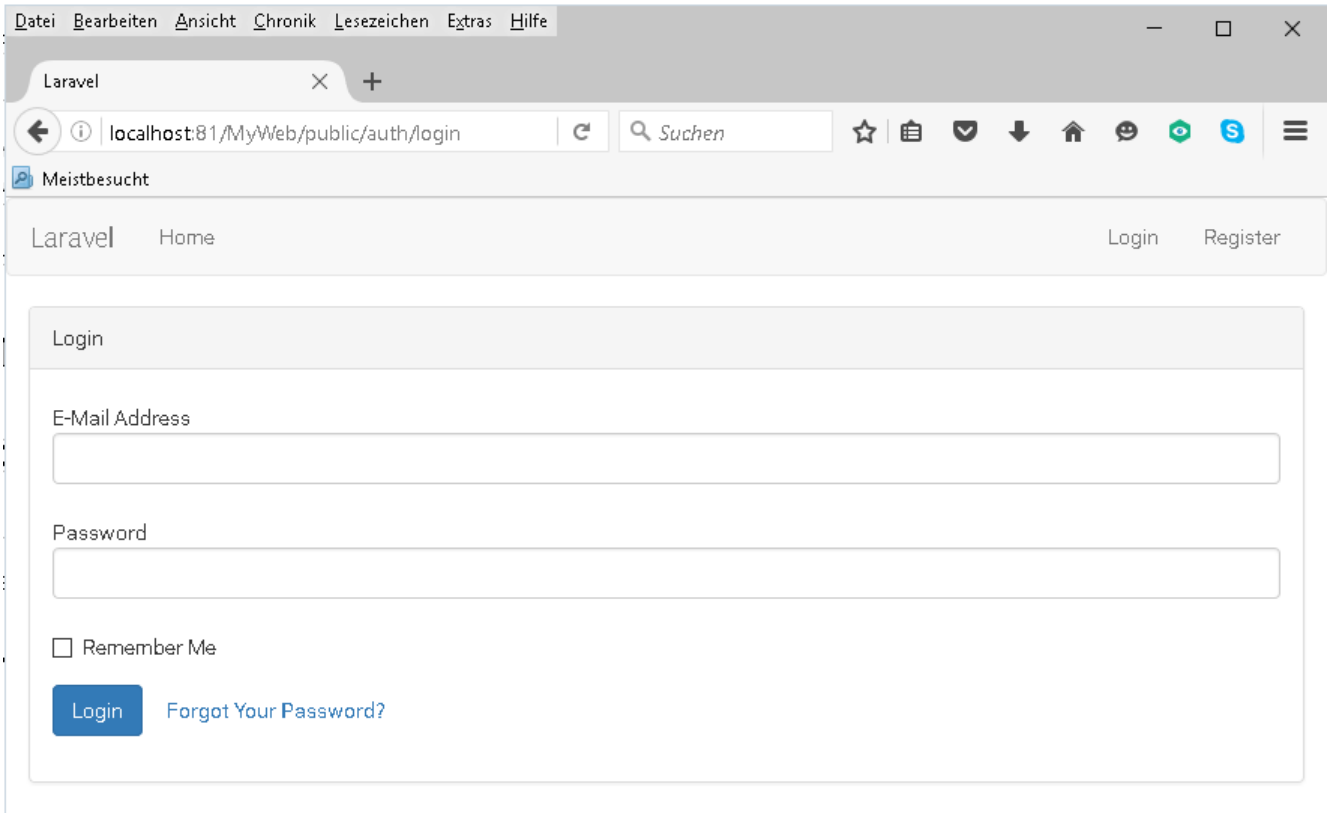
dass die Funktion index des HomeControllers aufgerufen wird:

```
<?php namespace App\Http\Controllers;
class HomeController extends Controller {
  public function __construct() {
    $this->middleware('auth');
  }
  public function index() {
    return view('home');
  }
}
```

Im Gegensatz zum WelcomeController wird der HomeController mit dem Aufruf $this->middleware('auth') erstellt. Damt wird ein „Wächter" erstellt, der die Berechtigung des Benutzers überprüft, bevor irgendwelche Aktionen in diesem Controller ausgeführt werden. Das heisst bevor die Funktion index – wie in der Route angegeben - ausgeführt werden kann, wird durch die Datei app/Http/Middleware/Authenticate.php überprüft, ob der aktuelle Benutzer authorisiert ist.

```
<?php namespace App\Http\Middleware;
use Closure;
use Illuminate\Contracts\Auth\Guard;
class Authenticate {
  protected $auth;
  public function __construct(Guard $auth) {
    $this->auth = $auth;
  }
  public function handle($request, Closure $next){
    if ($this->auth->guest()) {
      if ($request->ajax()) {
        return response('Unauthorized.', 401);
      } else {
        return redirect()->guest('auth/login');
      }
    }
    return $next($request);
  }
}
```

Handelt es sich beim Benutzer um einen Gast, wird anstelle der erwarteten Home-Seite die Login-Seite aufgerufen.



Der Kode der Login-Seite bietet einen interessanten Einstieg in den Web-Design. Im Gegensatz zu den Controller-Dateien (z. B. WelcomeController.php), haben Views-Dateien den Namenszusatz blade (z. B. welcome.blade.php). Diese Dateien verwenden Blade[1]. Blade unterstützt die Herstellung von Web-Seiten indem identische Programmsequenzen ausgelagert und einfach importiert werden können.

Web-Seiten enthalten in den meisten Fällen eine Vielzahl von übereinstimmenden Kode. Dieser kann dank Blade in eine Datei (resources/views/app.blade.php) ausgelagert werden.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Laravel</title>
  <link href="{{ asset('/css/app.css') }}" rel="stylesheet">
  <!-- Fonts -->
  <link href='//fonts.googleapis.com/css?family=Roboto:400,300' rel='stylesheet'
    type='text/css'>
</head>
```

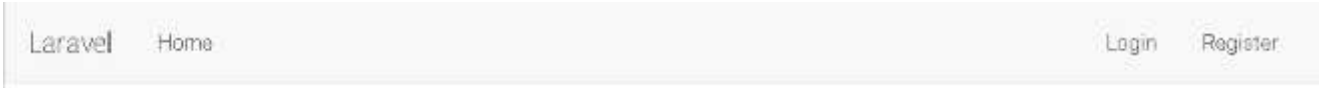
HTML-Seiten beginnen mit einer head-section. Mit dem Text – Laravel – wird angegeben, welcher als Titel der Web-Anwendung verwendet wird. Weiter werden die verwendeten Formatierungsangaben (CSS-Dateien) und Zeichensätze

[1] https://laravel.com/docs/5.0/templates#blade-templating

angegeben. Die Formatierungsangaben sind in der Datei public/css/app.css zusammengefasst.

In der body-section befindet sich die Anweisungen für die Darstellung der HTML-Seiten.

Die meisten Seiten einer Web-Anwendung haben eine Menüleiste, welche auf allen Seiten der Anwendung mehr oder weniger identisch ist.



Der zugehörige Kode wird im Anschnitt <nav> ... </nav> aufgeführt.

Hier der entsprechende Kode

```
<body>
  <nav class="navbar navbar-default">
    <div class="container-fluid">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed"
          data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
          <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="#">Laravel</a>
      </div>
      <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
        <ul class="nav navbar-nav">
          <li><a href="{{ url('/') }}">Home</a></li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
          @if (Auth::guest())
            <li><a href="{{ url('/auth/login') }}">Login</a></li>
            <li><a href="{{ url('/auth/register') }}">Register</a></li>
          @else
            <li class="dropdown">
              <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
               aria-expanded="false">{{ Auth::user()->name }}
               <span class="caret"></span>
              </a>
              <ul class="dropdown-menu" role="menu">
                <li><a href="{{ url('/auth/logout') }}">Logout</a></li>
              </ul>
            </li>
          @endif
        </ul>
      </div>
    </div>
  </nav>
```

Dazu ist zu bemerken, dass für Gäste die Schaltflächen Login und Register erscheinen, für authorisierte Benutzer jedoch der Name und ein drop-down Menü mit einer Schaltfläche für das Abmelden (Logout) erscheint. Dies wird mit der Blade-Anweisung @if (Auth::guest()) gesteuert.

Die Anweisung @yield wird durch die Sektion 'content' der aufrufenden Datei ersetzt.

```
  @yield('content')
  <!-- Scripts -->
  <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
  <script
    src="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.1/js/bootstrap.min.js">
  </script>
</body>
</html>
```

Abgeschlossen wird der Kode durch den Aufruf notwendiger JavaScript-Bibliotheken aus dem Internet.

Die Login-Seite **resource\views\auth\login.blade.php** erweitert nun die Datei app.blade.php und fügt anstelle von @yield('content') den Inhalt der Sektion 'content' ein.

```
@extends('app')
@section('content')
<div class="container-fluid">
  <div class="row">
    <div class="col-md-8 col-md-offset-2">
      <div class="panel panel-default">
        <div class="panel-heading">Login</div>
        <div class="panel-body">
          @if (count($errors) > 0)
            <div class="alert alert-danger">
              <strong>Whoops!</strong> There were some problems with your input.<br><br>
              <ul>
                @foreach ($errors->all() as $error)
                  <li>{{ $error }}</li>
                @endforeach
              </ul>
            </div>
          @endif
          <form class="form-horizontal" role="form" method="POST"
            action="{{ url('/auth/login') }}">
            <input type="hidden" name="_token" value="{{ csrf_token() }}">
            <div class="form-group">
              <label class="col-md-4 control-label">E-Mail Address</label>
              <div class="col-md-6">
                <input type="email" class="form-control" name="email"
                  value="{{ old('email') }}">
              </div>
            </div>
            <div class="form-group">
              <label class="col-md-4 control-label">Password</label>
              <div class="col-md-6">
                <input type="password" class="form-control" name="password">
              </div>
            </div>
            <div class="form-group">
              <div class="col-md-6 col-md-offset-4">
                <div class="checkbox">
                  <label>
                    <input type="checkbox" name="remember"> Remember Me
                  </label>
                </div>
              </div>
            </div>
            <div class="form-group">
              <div class="col-md-6 col-md-offset-4">
                <button type="submit" class="btn btn-primary">Login</button>
                <a class="btn btn-link" href="{{ url('/password/email') }}">
                  Forgot Your Password?</a>
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
@endsection
```

HTML-Seiten benötigen Formatierungsangaben in CSS-Dateien.Meist werden eigene Angaben mit Bootstrap[2] ergänzt. Die CSS-Anweisungen von Bootstrap sind in der Datei public/css/app.css integriert.

[2] http://getbootstrap.com/

Bootstrap ermöglich u. a. die Aufteilung einer Seite in Zeilen (col) und Spalten (row). Ein Feld wird in 12 Teile aufgeteilt.

```
<div class="container-fluid">
  <div class="row">

  </div>
</div>
```

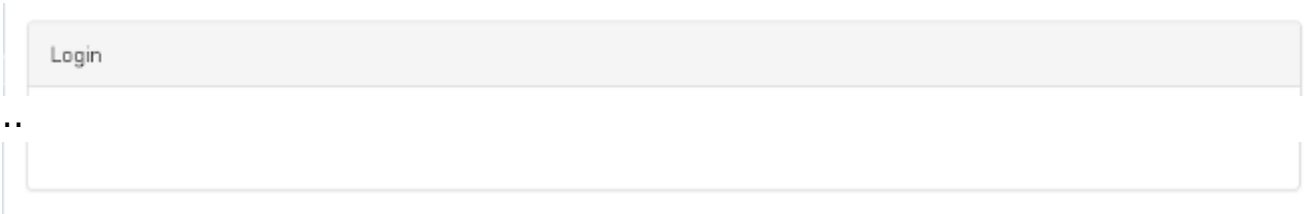
Damit wird ein Feld mit einer Spalte definiert.

In diesem Feld wird mit Abstand von 2 Teilen ein 8 Teile breites Gebiet mit der Überschrift 'Login' vorbereitet. Sollten bei der Eingabe Fehler bemerkt werden, wird eine Fehlermeldung ausgegeben.

```
<div class="col-md-8 col-md-offset-2">
  <div class="panel panel-default">
    <div class="panel-heading">Login</div>
    <div class="panel-body">
      @if (count($errors) > 0)
        <div class="alert alert-danger">
          <strong>Whoops!</strong> There were some problems with your input.<br><br>
          <ul>
            @foreach ($errors->all() as $error)
              <li>{{ $error }}</li>
            @endforeach
          </ul>
        </div>
      @endif
      <form class="form-horizontal" role="form" method="POST"
        action="{{ url('/auth/login') }}">
        <input type="hidden" name="_token" value="{{ csrf_token() }}">

eigentliches Formular

      </form>
    </div>
  </div>
</div>
```



..

Die Angaben zum Formular enthalten u. a. Angaben, was mit den Eingaben zu geschehen hat (method=“POST“) und wo die Eingaben verarbeitet werden sollen (action="{{ url('/auth/login') }}").

Zu beachten ist die Zeile

```
<input type="hidden" name="_token" value="{{ csrf_token() }}">
```

Damit wird eine verborgene Zeichenfolge in das Formular eingefügt und beim Absenden der Eingaben mitgegeben. Damit kann der empfangende Teil der Anwendung überprüfen, ob die Eingabe korrekt erfolgte.

Das eigentliche Formular enthält folgende Felder:


```
<div class="form-group">
  <label class="col-md-4 control-label">E-Mail Address</label>
  <div class="col-md-6">
    <input type="email" class="form-control" name="email"
      value="{{ old('email') }}">
  </div>
</div>
```

```
<div class="form-group">
  <label class="col-md-4 control-label">Password</label>
  <div class="col-md-6">
    <input type="password" class="form-control" name="password">
  </div>
</div>
```

```
<div class="form-group">
  <div class="col-md-6 col-md-offset-4">
    <div class="checkbox">
      <label>
        <input type="checkbox" name="remember"> Remember Me
      </label>
    </div>
  </div>
</div>
```

```
<div class="form-group">
  <div class="col-md-6 col-md-offset-4">
    <button type="submit" class="btn btn-primary">Login</button>
    <a class="btn btn-link" href="{{ url('/password/email') }}">
      Forgot Your Password?</a>
  </div>
</div>
```

Obschon alles vorbereitet ist, braucht es nun eigenen Aufwand, dass die Benutzeridentifizierung auch funktioniert.

Es braucht eine Datenbank, in welcher die Benutzer registriert werden und registrierte Benutzer identifiziert werden können.

Weiter braucht es Angaben, damit von der Web-Anwendung Mails versandt werden können.

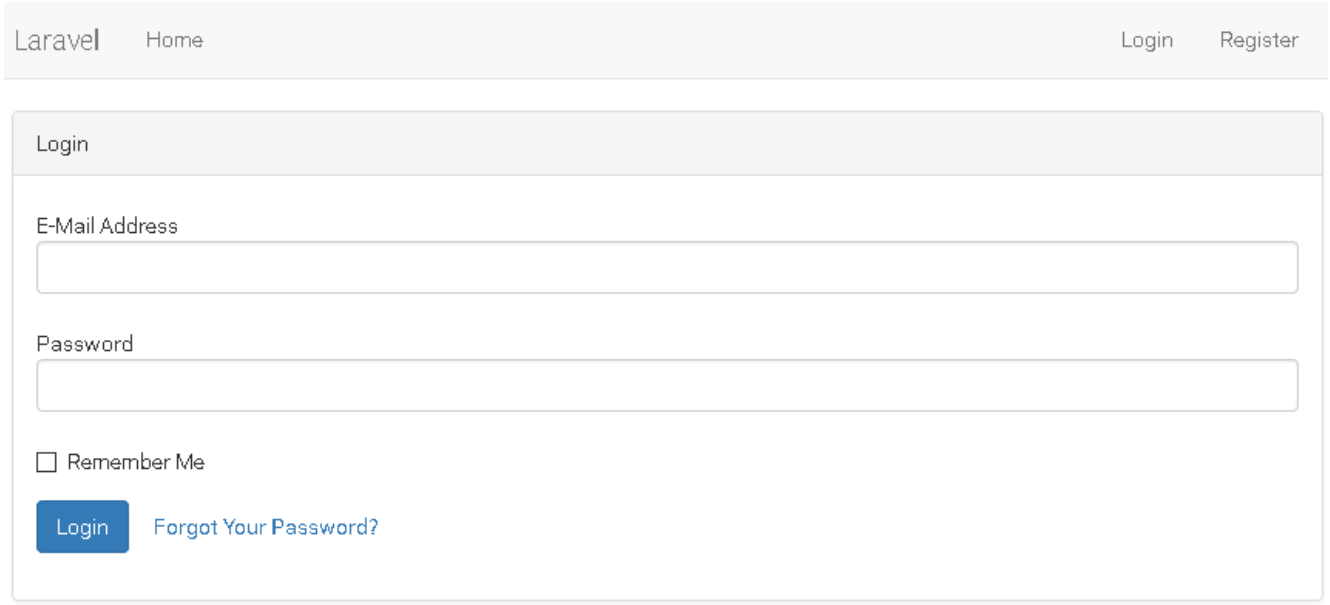
Abschliessend ist jedoch zu sagen, dass mit der Installation von Laravel eine fantastische Grundlage für eigene Entwicklungen unentgeltlich zur Verfügung steht und dies alles ohne eine einzige Zeile programmieren zu müssen.

Mit diesen Änderungen kann nun der erste Benutzer regitriert werden.

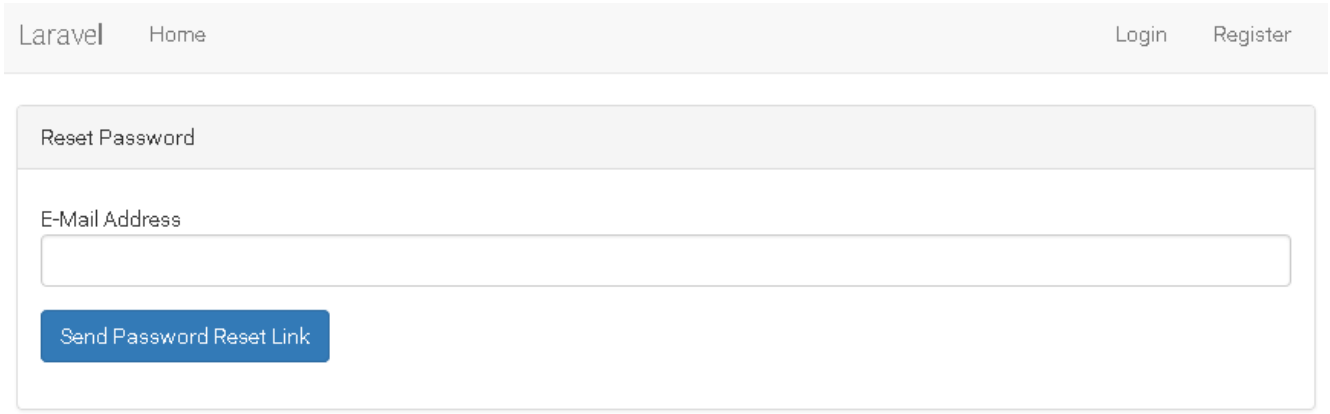
Nachdem nun die entsprechenden Felder ausgefüllt sind, werden die Daten nach dem Klick auf Register in die Datenbank eingetragen.

Unter dem Namen befindet sich eine Drop-Down-Liste, welche im Moment die Schaltfläche für das Abmelden enthält.

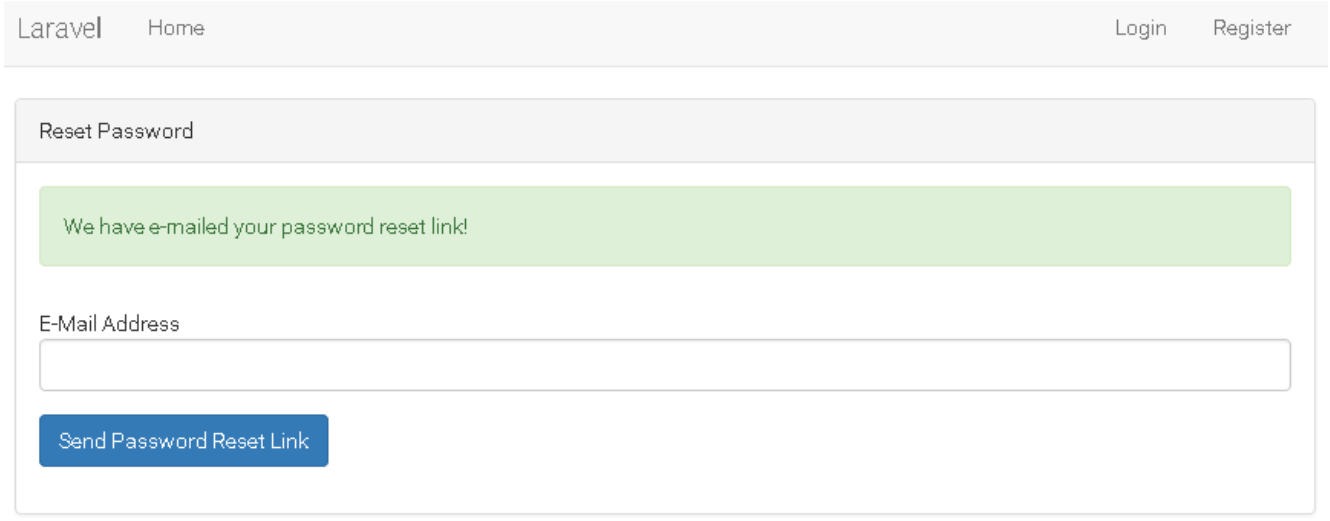
Nun kann man sich auf der Login-Seite anmelden.

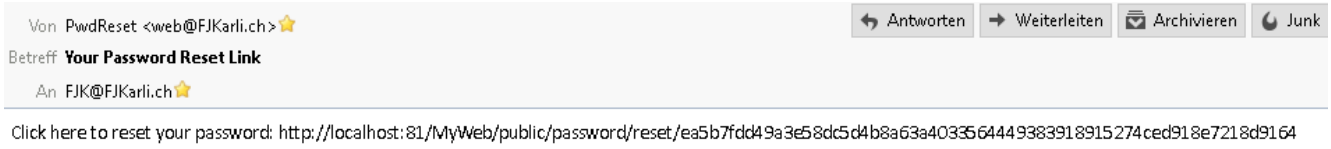


Mit Klick auf **Forgot Your Password** kann man sich einen Reset-Link zusenden lassen.



Mit einer regstrierten Mail-Adresse löst ein Klick auf eine Email aus.





Abschliessend folgende Bemerkung:

All diese Funktionalität dieser Web-Anwendung ist in der Installation von Laravel enthalten ohne dass nur eine einzige Zeile Kode geschrieben werden musste.

Einzig die Datenbank musste erstellt und Benutzer und Passwort sowie die Zugriffdaten für den Mail-Server in der .env-Datei festgehalten werden

Die Datei app/Http /routes.php ist sehr gut Dokumentiert.

In unserem Fall interessiert uns der Eintrag:

```
Route::get('/', function () {
    return view('welcome');
});
```

Ein Route gibt an, was zu geschehen hat, wenn eine Web-Adresse eingegeben wird.

Im Beispiel wird mit '/' angegeben, dass es sich um die Home Adresse innerhalb der Laravel-Anwendung.

Wird also diese Web-Adresse eingegeben wird die folgende Funktion ausgeführt und die Seite welcome.blade.php im Verzeichnis MyWeb\resources\views aufgerufen.

Da Laravel davon ausgeht, dass Views unter MyWeb\resources\views\ gespeichert sind, kann die Pfadangabe weggelassen werden. Das gleiche gilt für die Endungen .php oder .blade.php.