CRUD Entwicklung mit Laravel

CRUD steht für folgende Aktionen: Create, Read, Update, Delete. Dies entspricht den Anforderungen eines redaktionellen Systems zur Verwaltung von Daten. Wir versuchen das hier am Beispiel eines Datenbestands von Autoren darzustellen.
Tabelle Autoren:

Gemäß **MVC** benötigen entsprechend Model, Views und Controller.

- Model „Author“ (in app/Models)
- Controller „AuthorController“ (in app/Http/Controllers)
- Views (in resources/views)
- ausserdem Routen Definitionen (in routes/web)

Model, Controller und eine Request-Klasse zur Validierung von Formulardaten erstellen wir mittels Artisan-Kommando auf der Konsole (`php artisan`).

Model Klasse „**Author**“ erstellen:

```
php artisan make:model Models/Author
```

```php
<?php
// app/Models/Author.php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Author extends Model
{
    protected $fillable = ['firstname','lastname'];
    protected $appends = ['name'];
    public $timestamps = false;

    /**
     * @return \Illuminate\Database\Eloquent\Relations\HasMany
     */
    public function movies() {
        return $this->hasMany(Movie::class);
    }

    /**
     * @return string
     */
    public function getNameAttribute() {
        return $this->firstname . ' ' . $this->lastname;
    }
}
```

Request Klasse „**AuthorRequest**“ erstellen:

```
php artisan make:request AuthorRequest
```

```php
<?php
// app/Http/Request/AuthorRequest.php
namespace App\Http\Requests;

use Illuminate\Support\Facades\Auth;
use Illuminate\Foundation\Http\FormRequest;

class AuthorRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     * @return bool
     */
    public function authorize()
    {
        return Auth::check() && Auth::user()->is_admin;
    }

    /**
     * Get the validation rules that apply to the request.
     * @return array
     */
    public function rules()
    {
        return [
            'firstname' => 'required|alpha',
            'lastname'  => 'required|alpha',
        ];
    }

    /**
     * Get the validation messages that apply to the rules.
     * @return array
     */
    public function messages()
    {
        return [
            'firstname.required' => 'Bitte einen Vornamen angeben!',
            'firstname.alpha'    => 'Der Vornamen darf nur aus Buchstaben bestehen!',
            'lastname.required'  => 'Bitte einen Nachnamen angeben!',
            'lastname.alpha'     => 'Der Nachnamen darf nur aus Buchstaben bestehen!',
        ];
    }
}
```

Diese Klasse wird von folgenden AuthorController Funktionen als Request Parameter benutzt:

- store(`AuthorRequest $request`) für Neueintrag
- update(`AuthorRequest $request, …`) für Update

(Siehe hierzu weiter unten zum AuthorController.)

Beide Funktionen (Methoden) dienen als Formular-Actions und übernehmen die Validierung der Formular-Daten. Ist die Validierung erfolgreich, werden die Daten gespeichert. Ansonsten erfolgt ein Redirect zur Formular Seite inklusive Fehlermeldung entsprechend der im Validator definierten Regeln und Messages.

Controller Klasse „**AuthorController**“ erstellen:

```
php artisan make:controller --model=Models/Author AuthorController
```

```php
<?php
// app/http/Controller/AuthorController.php
namespace App\Http\Controllers;

use App\Models\Author;
use App\Http\Requests\AuthorRequest;

class AuthorController extends Controller
{
    // listen ansicht
    public function index()
    {
        $data = Author::paginate(15);
        return view('authors', compact('data'));
    }
    // einzel ansicht
    public function show(Author $author)
    {
        return view('author', compact('author'));
    }

    // anzeige formular für neueintrag
    public function create()
    {
        return view('admin.author.create');
    }

    // neueintrag speichern
    public function store(AuthorRequest $request)
    {
        Author::create($request->validated());
        return redirect()->route('author.list');
    }

    // anzeige update formular
    public function edit(Author $author)
    {
        return view('admin.author.edit', compact('author'));
    }

    // update eines datensatzes
    public function update(AuthorRequest $request, Author $author)
    {
        $author->update($request->validated());
        return redirect()->route('author.list');
    }

    // löschen eines datensatzes
    public function destroy(Author $author)
    {
        $countMovies = $author->movies->count();
        $msg = <<< MSG
Der Autor kann nicht gelöscht werden, da ihm noch $countMovies Movie(s) zugegordnet
sind!
MSG;
        if($countMovies > 0) {
            return back()
                ->with('alert', 'success')
                ->with('message', $msg)
            ;
        }
        $author->delete();
        return redirect()->route('author.list');
    }
}
```

Views (Blade-Templates) für die Anzeige der Daten, Formulare

Grundlayout, in das sich alle Page-Views einbetten via **@extends:**
resources/views/layouts/default.blade.php:

```
<!doctype html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>{{ config('app.name', 'Laravel') }}</title>
    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}" defer></script>
    <!-- Fonts -->
    <link rel="dns-prefetch" href="//fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet">
    <!-- Styles -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
</head>
<body>
    <div id="app">
        <nav class="navbar navbar-expand-md navbar-light bg-white shadow-sm">
            <div class="container">
                @yield('top-navigation')
            </div>
        </nav>
        <main class="py-4">
            <div class="container">
                <div class="row justify-content-center">
                    <div class="col-md-12">
                        <div class="card">
                            <div class="card-header">
                                @yield('header')
                            </div>
                            @if(session('alert'))
                                <x-alert type="{{ session('alert') }}" message="{{
session('message') }}" />
                            @endif
                            <div class="card-body">
                                @yield('body')
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </main>
    </div>
</body>
</html>
```

View der tabellarische Ansicht der Autoren

resources/views/authors.blade.php:

```
@extends('layouts.default')

@section('header')
    Meine Autoren
@endsection

@section('body')
    <div class="my-2">
        <a href="{{ route('author.create') }}"
           class="btn-sm btn-primary" role="button">Create Autor</a>
    </div>
    {{ $data->links() }}
    <table class="table table-striped table-responsive">
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Anzahl Movies</th>
            <!-- anzeige nur für eingeloggte user -->
            @auth
                <th colspan="2"><br></th>
            @endauth
        </tr>
        @foreach($data as $item)
            <tr>
                <td>{{ $item->id }}</td>
                <td><a class="nav-link" href="{{ route('author.show', ['author' =>
$item]) }}">
                        {{ $item->name }}</a></td>
                <td align="center">{{ $item->movies->count() }}</td>
                <!-- anzeige nur für eingeloggte user -->
                @auth
                    <td><a href="{{ route('author.edit', ['author' => $item]) }}"
                           class="btn-sm btn-primary" role="button">Edit</a></td>
                    <td><a href="{{ route('author.destroy', ['author' => $item]) }}"
                           class="btn-sm btn-danger delsoft"
role="button">Delete</a></td>
                @endauth
            </tr>
        @endforeach
    </table>
@endsection
```

Einzelansicht Autor

resources/views/author.blade.php

```
@extends('layouts.default')

@section('header')
    Meine Autor
@endsection

@section('body')
    ID: {{ $author->id }}<br>
    Name: {{ $author->name }}<br>
    Movies: {{ $author->movies->count() }}<br>
    <ol>
        @foreach($author->movies as $item)
            <li>{{ $item->title }}</li>
        @endforeach
    </ol>
@endsection
```

Formular View für Neueintrag eines Autors

resources/views/admin/author/create.blade.php:

```
@extends('layouts.default')

@section('header')
    Autor erstellen
@endsection

@section('body')
    <div>
        <a href="{{ route('author.list') }}"
           class="btn-sm btn-primary" role="button">Zurück</a>
    </div>
    <div class="my-2">
        <form method="post" action="{{ route('author.store') }}">
            @csrf
            <!-- see: app/Providers/AppServiceProvider.php
            -> boot function for setting components aliases
            //-->
            <x-inp.text name="firstname" label="Vorname" />
            <x-inp.text name="lastname" label="Nachname" />
            <x-inp.submit name="speichern" value="speic hern" />
        </form>
    </div>
@endsection
```

Formular View für Update eines bestehenden Autors

resources/views/admin/author/edit.blade.php:

```
@extends('layouts.default')

@section('header')
    Autor ID {{ $author->id }}
@endsection

@section('body')
    <div>
        <a href="{{ route('author.list') }}"
           class="btn-sm btn-primary" role="button">Zurück</a>
    </div>
    <div class="my-2">
        <form method="post" action="{{ route('author.update', compact('author')) }}">
            @csrf
            <!-- see: app/Providers/AppServiceProvider.php
            -> boot function for setting components aliases
            //-->
            <x-inp.text name="firstname" label="Vorname" :value="$author->firstname" />
            <x-inp.text name="lastname" label="Nachname" :value="$author->lastname" />
            <x-inp.submit name="speichern" value="speichern" />
        </form>
    </div>
@endsection
```

Für Formular Felder wurden entsprechende View-Componenten erstellt via:

```
php artisan make:component Form/Input/{FormElementType}
```

und in den Form-Templates benutzt (z.b: <x-inp.text name='name' :value='$value' />)

Und schliesslich Routen definieren

in routes/web.php:

```
// author routen
Route::name('author.') //
    ->prefix('author')
    ->group(function ($route) {
        // anzeige formular für neueintrag
        $route->get('/create', 'AuthorController@create')
            ->name('create')
            ->middleware('auth') // Authentifizierung erforderlich
        ;
        // neueintrag speichern
        $route->post('/store', 'AuthorController@store')
            ->name('store')
            ->middleware('auth') // Authentifizierung erforderlich
        ;
        // anzeige update formular
        $route->get('/edit/{author}', 'AuthorController@edit')
            ->name('edit')
            ->middleware('auth') // Authentifizierung erforderlich
        ;
        // eintrag löschen
        $route->get('/destroy/{author}', 'AuthorController@destroy')
            ->name('destroy')
            ->middleware('auth') // Authentifizierung erforderlich
        ;
        // bestehenden eintrag speichern
        $route->post('/update/{author}', 'AuthorController@update')
            ->name('update')
            ->middleware('auth') // Authentifizierung erforderlich
        ;
        // keine Authentifizierung erforderlich
        // anzeige liste
        $route->get('/list', 'AuthorController@index')->name('list');
        // anzeige einzelner eintrag
        $route->get('/autor/{author}', 'AuthorController@show')->name('show');
    });
```

Wir definieren hier eine Gruppe von (Author) Routen. Jede Route hat einen Namen. z.B: „create“. Durch den Gruppen-Namen 'author.' wird dieser Name zu „author.create“. Über diesen Namen, kann ich mir in anderen Templates entsprechende URL's generieren lassen.
Die Angabe des Gruppen-Prefix bewirkt, daß aus der Route '/create' => 'author/create' wird, also URL/author/create, statt URL/create aufgerufen werden muß.
Die Angabe von „`middleware('auth')`“ bewirkt, daß für die Benutzung dieser Route ein User eingeloggt sein muß. Wenn das nicht der Fall ist, wird man zur Login Seite umgeleitet (per redirect)
.