Projektaufgabe Laravel Shop  
Zu erstellen ist ein Micro-Shop. Browsen soll im Vordergrund stehen.

Fake-Produkte und Kategorien sollen aber auch mit den typischen CRUD-Formularen geändert werden. Dummy-Daten-Befüllung soll möglich sein. Mit Bildchen.

Kursteilnehmer

Mario Salzer

Kurs: FI-36-AE

Datum: 2014-01-18 bis -01-19

Aufgabe

* Erstellung eines Micro-Shops
* MySQL-DB + Laravel + PHP
* DB-Design, Presentationsstruktur, Doku Vorbereitung
* Setup via composer & laravel tools
* Optional: CSS -- Thema wird: Horror-Shop (aus alter TV-Serie)

Vorbereitung / Planung

* Planungsdoku (this is happening now!!!), Format: Word
* Alles recht trivial, von daher wieder schwierig implizite Details als „Planung“ zu erkennen
* Versionsverwaltung nicht relevant.
* Lokaler Projektordner wird c:\xampp\htdocs\laravel\shop (ggf. Paralleltest via Apache und Laravels [php -S](https://www.php.net/manual/en/features.commandline.webserver.php))
* Datenbank in myAdmin wird vermutlich als „laravelshop“ angelegt (mit bestehendem User+PW, da ohne Relevanz). Im Prinzip ginge auch SQLite weil Laravel und QueryBuilder das eh wieder abstrahieren.
* DB-Tabellen werden warscheinlich:
  + **articles**

|  |  |  |
| --- | --- | --- |
| catid | INT FOREIGN KEY | 1-5 |
| title | TEXT/VARCHAR | „verfluchte Kaffeetasse“ |
| desc | TEXT | „zum Trinken und so“ |
| img | VARCHAR | **url** |
| img\_thumb | VARCHAR | **url** |

* + **categories**

|  |  |  |
| --- | --- | --- |
| id | INT PRIMARY KEY | 1-5 |
| name | TEXT/VARCHAR | „tassen“ |
| desc | TEXT | „Küchentools“ |
| icon | VARCHAR | Unicode symbol |

Änderungen: img\_\* werden URLs statt lokale Pfade/Dateien, icon für die Kategorie war in Aufgabe nicht gefordert. Aber wo wir halt einmal dabei sind…

Faulheit: article.`catid` mapt zu category.id

Standardfelder von Laravel (id, created\_at, updated\_at) nicht extra aufgeführt.

* Laravel-Migrationen jeweils eine create\_<TABELLE>\_table. Und eine Modifikations-Migration (change\_article\_table\_add\_key) für article.catid zu FOREIGN KEY. (Zuerst als INT angelegt, und später dann als FOREIGN KEY. - Mal sehen ob das funktioniert………………)
* Ansichten/Views werden benötigt:
  + Layout (mit @section „main“)
  + \_\_simplecss\_\_ statt Bootstrap
  + article/index, ./view, ./change (kombiniert create+edit), und delete?
  + category/index, .view, ./change
  + Ordner public/img/ vermutlich redundant, wenn URLs verwendet werden
  + Optional: components/select.blade.php für ein <x-select>im Formular
* CRUD/REST-Controller werden benötigt:
  + ArticlesController -mrc
  + CategoriesController -mrc
* Kombinierte Erstellung via -mrc oder -a durch make:model

<https://stackoverflow.com/questions/43187735/laravel-create-model-controller-and-migration-in-single-artisan-command>

In routes/web dementsprechend als Route::resource(…, \App\Http\Controller\Xy::class)? wars glaub ich.

* SQL-Abfragen, beispielshalber für **article** hier:
  + Index: SELECT articles.\* FROM articles JOIN categories ON catid
  + Mit Filter: WHERE catid=sql@($\_REQUEST->catid)
  + Mit Eloquent wird es demnach irgendwie (Anm.d.Red.: das ist schon nicht mehr Planung hier, sondern Blind-Programming) in der Form:

$a = Article::join(„category“, „article.catid“, „=“, „category.id“)->orderBy(„created\_at“, „desc“)

If (isset($request[„catid“])) {

$a = $a->where(„catid“, „=“, $req\_cat\_id);

}

…. [„view\_var“ = $a->get(), …]

* (Extras: versionierte DB in Laravel sollte doch angeblich schon eingebaut sein? Recherche auf Implementierungs-Phase verschoben…)
* Factories:
  + Eine für article, eine für category
* Faker: wird weniger benutzt. War in Aufgabe nur eine Anregung. Das Tool selbst ist überschaubar nützlich - in Anbetracht der genierierten Textausschnitte (für Nutzernamen mag es okay sein, ansonsten keiner Promotion würdig).

Stattdessen werden für img-URLs einige Vorgaben in nem Array vorgehalten. (Quelle: vermutlich einfach Google Search. Womöglich keine distincten \_thumbs)

Produkte und Beschreibungen zusammengewürfelt.

Vielleicht für Beschreibungen LoremIpsum verwenden.

Auch für Kategorien vielleicht einfacher eine Liste mit Beispielwerten oder base64/rand-substr.

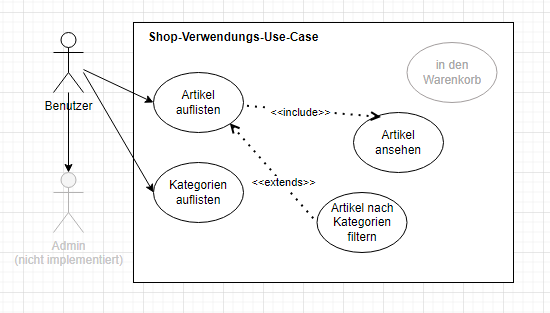
* Alberne Variablennamen werden diesmal nicht verwendet. (Laravel nutzt PSR/composer, dummen Framework-Autoloader, ist daher partiell PHP-aversiv.)
* Optional: Möglicherweise Vereinfachung der Kategorien-Ansicht. Es reicht eine Tabelle in views/category/index mit Edit-Btn, keine Detail-Ansicht nur für Name?
* Zusatzaufgabe: Kategorien-Liste wird im footer auf der views/index untergebracht.

Die Filterung kann im ArticlesController@index erfolgen, wenn $request->catid vorhanden ist. Damit könnte die Startseitenroute Route::get(„/“) im Prinzip auch den selben Callback und damit View verwenden (article/index).

* Testen: nochmal prüfen
  + Artikel-Liste, Kategorien-Liste
  + Artikel-Details, Bilderansicht, etc.
  + Artikel-Ändern/Löschen/Neu
* ToDo/Checkliste diesmal am Ende, nicht in Planungsphase
* Paketierung als ZIP soll Boilerplate auslassen, respektive .gitignore-Vorgaben (/.phpunit.cache /node\_modules /public/build /public/hot /public/storage /storage/\*.key /vendor .env .env.backup .env.production .phpunit.result.cache Homestead.json Homestead.yaml auth.json npm-debug.log yarn-error.log /.fleet /.idea /.vscode) auslassen.
* Via Linux-cmd trivial: zip msalzer-shop-laravel.zip -r [-x@.gitignore](mailto:-x@.gitignore) shop/

UML Use Case / Activity Diagram

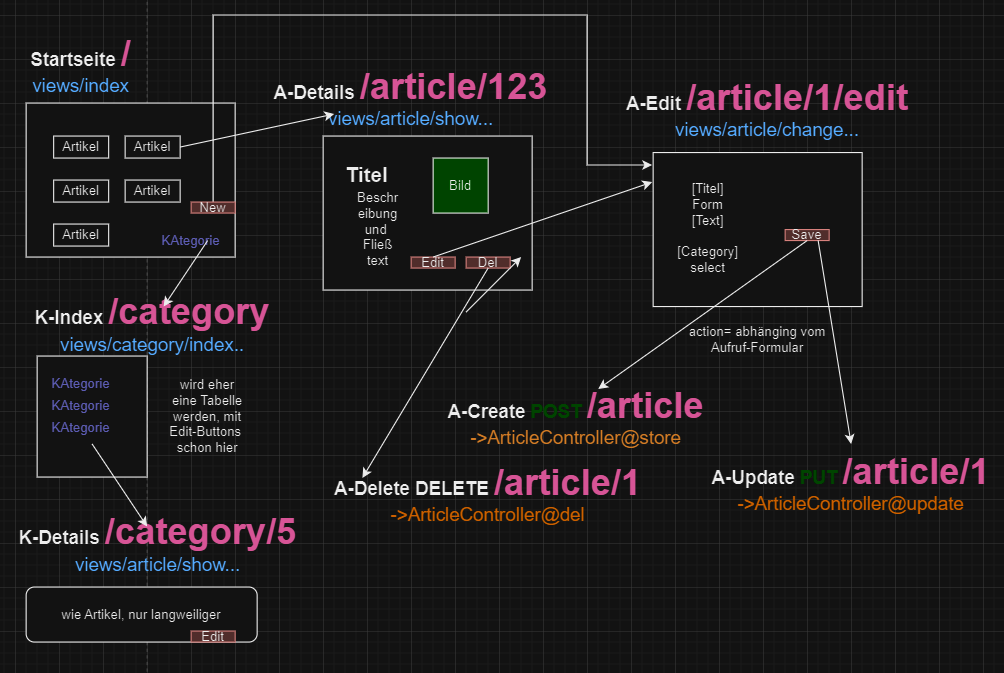
Streng genommen ist das weder Use Case noch Activity-konform. Aber füllt halt die Planungs-Word-Datei….



Routen (und teilweise Layout)

Formatierung der einzelnen Seiten (insbesondere Kategorien) noch nicht abschließend. Kann sich ändern. Nicht Teil der Planung, weil kein Mediendesign-Kurs.

Routen entsprechen so oder so den REST-Defaults. Lediglich kombiniertes Formular (edit+new) weicht ab (in Routen nur vage erwähnt, aber action= und @method variieren abhängig von $request/$id).



Bilderliste

<https://openclipart.org/image/800px/266006>

als Preview-Fallback erstmal.

Liste wird eher live erstellt, in Implementierungsphase für database/seeding.

Research

Falls die Zeit reicht:

* Einfacherere Update-Methoden:

<https://kinsta.com/blog/laravel-crud/>

$request->all()

* Flash-Messages? ->with(„success“, …)
* Lorem Picsum:

<https://picsum.photos/>

erstmal als Fallback-Bilder

<https://picsum.photos/$rand/$rand>

* Powershell-Shortcuts (notepad $PROFILE)
  + function artisan() { php artisan $args }
  + function tinker() { artisan tinker $args }

----------------------------------------------------------------

Implementierung / Steps

PS C:\xampp\htdocs\laravel> **composer create-project laravel/laravel shop**

Creating a "laravel/laravel" project at "./shop"

Installing laravel/laravel (**v10.3.2**)



PS C:\xampp\htdocs\laravel\shop> **artisan make:model Article -a**

Hat zu viel erstellt!

**rm** app\Policies\ArticlePolicy.php app\Http\Requests\UpdateArticleRequest.php app\Http\Requests\StoreArticleRequest.php app\Http\Requests\StoreArticleRequest.php

Obendrein die ExtraRequests wieder in Standard umgewandelt

PS C:\xampp\htdocs\laravel\shop> **artisan make:model Category -mrc**

PS C:\xampp\htdocs\laravel\shop> **artisan make:factory CategoryFactory**

**artisan make:migration alter\_article\_table\_add\_catid\_foreign\_key**

Sehr beschreibend

Erste Routen vorbereitet

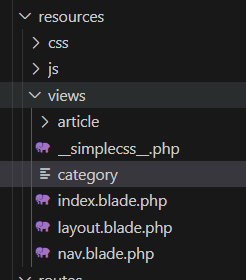
Route::resource(

    "article",

    \App\Http\Controllers\ArticleController::class

);

Und Layout/Struktur:



DB

shop/database/migrations/2024\_01\_18\_125957\_create\_articles\_table.php

            $table->string("title", 100);

            $table->text("desc");

            $table->string("img", 160);

            $table->string("img\_thumb", 160);

shop/database/migrations/2024\_01\_18\_130123\_create\_categories\_table.php

            $table->string("name", 40);

            $table->string("desc", 200);

            $table->string("icon", 16);

shop/database/migrations/2024\_01\_18\_130845\_alter\_article\_table\_add\_catid\_foreign\_key.php

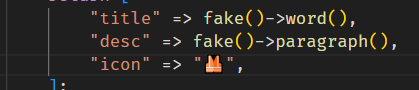
            $table->foreignId("catid")->references('id')->on('categories');

**artisan migrate** war gleich erfolgreich

Sieht gut aus in phpMyAdmin/MariaDB.

Factories

Für die Kategorien Fabrik waren wir dann doch etwas faul:



Die Artikel-Factory ist abhängig von populierten Kategorien:

            // Kategories müssen schon bestehen!!!!

            "catid" => \App\Models\Category::select("id")->inRandomOrder->limit(1)->get(1)->id,

Via **artisan tinker**:

* \App\Models\Category::factory()->count(5)->create();funktioniert

Für Artikel-Factory nochmal geändert:

 "catid" =>  DB::table("categories")->select("id")->inRandomOrder()->limit(1)->first()->id,

* \App\Models\Article::factory()->count(5)->create();lief dann, aber mit undefined key warnings (vmtl array rand) --- Anm.d.Red.: $art[… count(**$art**))

Erster Versuch mit article.index indizierte die Images waren doch nicht ganz random, daher nochmal mit /id/$rand ergänzt.

ControllerTime

Bei den Controllern ist interessant wie artisan die jeweils angelegt hatte. Mit -mrc werden klassische Parameter erstellt, während -a Article mit function show(Article $article) statt $string id angelegt wurde. Scheint aber keinen Unterschied zu machen, und ist stellenweise marginal kürzer.

Alternative Verwendung

Im Prinzip unterscheiden sich ArticleController und CategoryController auch nur in den Formularen und Feldnamen. Ansonsten größtenteils dupliziert. Außer dass CategoryController->store+update die einfache fill-Funktion verwenden (kein Fremdschlüssel, keine Validierung zwingend / Admin-Fkt und Laravel-Exception hinreichend.)

    function save\_req($category, $request)

    {

        $category->fill($request->all());

Views

Es gibt ein paar Überlappungen. category.show wird über article.index abgebilder (welches nach Kategorie filtert). Daher kein QUERY\_STRING-handlign in der index-Liste (die ansonstent article.index entspricht)

<x-select>

artisan make:component select (shop/app/View/Components/select.php) um eine Formular-Funktion abzukürzen. Im Prinzip machbar, aber $attributes ist in programmatischen View-Components doch nicht verfügbar. Zwar mit Doppel-String-Context für :name=, aber name=“id“ geht auch. Bonus-HTML-Attribute funktionieren aber nicht.

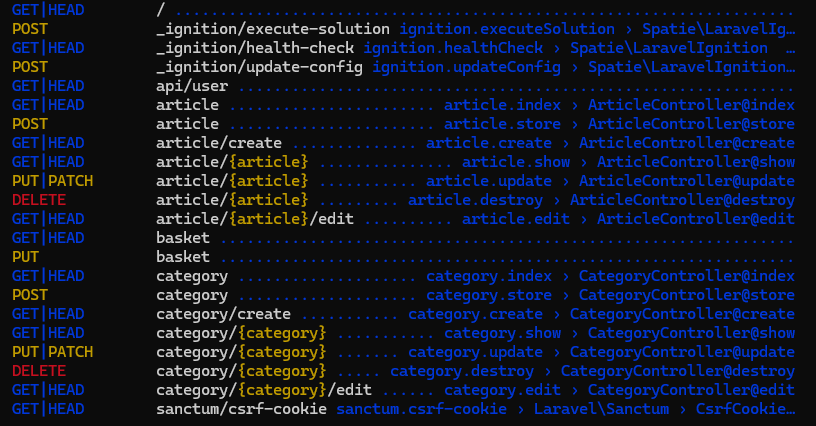
 <x-select :name="'catid'" :options="$cats" :default="$article->catid" />

Mglw. wird ->withAttributes auch nie aufgerufen (GH:laravel/framework/issues/31653)

Flash messages

->withSuccess(„…“) wird einfach im layout (zwischen nav und main) manuell abgefragt.

Routing

Scheint vollständig: 

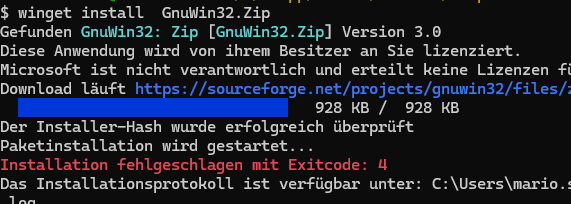
Final Test

* "DROP DATABASE `laravelshop`
* artisan migrate => ERR SQLSTATE[22001]: String data, right truncated: 1406 Data too long for column 'desc' at row 1 -> nochmal factory geändert
* nochmal DROP und artisan migrate - ✓
* \App\Models\Category::factory()->count(7)->create(); - ✓
* \App\Models\Article::factory()->count(25)->create(); - ✓
* \App\Models\Article::factory()->count(10)->create(); - ✓
* Ansehen ✓, Editieren ✓, Neu ✓, Löschen ✓k, Warenkorb ✓

Caveats (Anm.d.Red.: euphemistic way to say bugs)

* DELETE FROM categories bringt einen Fehler, bei nicht-leeren Kategorien (bei neuen Dummy-Kategorien geht es). Das ist hier vorsätzlich dass kein onDelete(„CASCADE“) in der Migration eingetragen war. Ist zwar aus Sicht von Datenbankexperten nicht die richtige Herangehensweise, aber in der Praxis würd ich hier auch ***niemalsnimmernicht*** automatisierte Löschungen aktivieren.

SQLSTATE[23000]: Integrity constraint violation: 1451 Cannot delete or update a parent row: a foreign key constraint fails (`laravelshop`.`articles`, CONSTRAINT `articles\_catid\_foreign` FOREIGN KEY (`catid`) REFERENCES `categories` (`id`))

* Standard-ZIP lässt sich nicht installieren (entwendete Admin-Rechte)  
  

Daher packen mit exakter .gitignore nicht garantierbar

* Stattdessen mit tar cvzf ../msalzer-laravel-shop.tgz --exclude-from=.gitignore2 .  
  (angepasste .gitignore mit ./-pfaden statt /-vroot ref)
* Für Kategorien ist der DELETE-Link gewollt nur auf der Edit-Seite, für Artikel hingegen schon in der Details-Ansicht verfügbar.
* Für das ORDER BY created\_at wurde das Stylesheet nochmal angepasst auf flexbox-Anzeige statt columns:3 (neueste Artikel in erster Zeile, statt Leiste-links-runterwärts)
* Stylesheet und Anpassungen nur im Dark Mode getestet
* Footer-Kategorie-Links nur auf Startseite, so gewollt (ursprünglich wars immer aktiv), und ohne Dropdown (ansonsten wärs $request->get() und index-view angepasst)
* Warenkorb-Preise sind random, nur Anzahl in Session, weil DB articles diesmal nicht zu stark angepasst (keine `price`-Spalte)

ToDo/Checkliste

|  |  |
| --- | --- |
| Planung | ✓ |
| MVCs überlegen | ✓ |
| REST-Controller-Routing | ✓ |
| Foreign Keys | ✓ |
| Anlegen via phpMyAdmin | ✓ |
| Testdaten | ✓ |
| Tabellendesign | ✓ |
| Via migrate anlegen | ✓ |
| Nach Datum absteigend sortiert | ✓ |
| Link+Bild | ✓ |
| Artikel Bearbeiten+Löschen+Neu | ✓ |
| Artikel-Ansicht mit **allen** infos | ✓ |
| Kategorien-Filter | ✓ |
| ZIP | ✓ |
| Last-Minute-DB-Export | ✓ |

