Meilenstein 1 – Infrastruktur

Ziel ist das Aufsetzen des Projekts „Abalo“. Das Aufsetzen umfasst 1) das Installieren der Datenbank PostgreSQL, 2) das Anlegen des Projekts auf Basis des Laravel Frameworks sowie 3) das Implementieren eines ersten Datenbankschemas mit Hilfe von Migrationsdateien.

Aufgabe 1

Richten Sie Ihre ausgewählte IDE für die Entwicklung von PHP-Projekten ein.

Aufgabe 2

Installieren Sie sich lokal eine PostgreSQL Datenbank mindestens Version 11. Gerne können Sie auch eine aktuellere Version nutzen.

Die Datenbank sowie eine Installationsanleitung finden Sie unter:

https://www.postgresql.org

Starten Sie die Datenbank und richten Sie über ein Administrationswerkzeug Ihrer Wahl einen Useraccount mit dem Namen *dev* und einem Passwort ein.
Erzeugen Sie eine Datenbank mit dem Namen „abalo“ in UTF-8 Enkodierung und legen Sie darin das Schema „public“ an, falls dieses noch nicht existiert.

Aufgabe 3

Einrichtung. Richten Sie ein neues Projekt in Ihrer IDE ein: abalo. Richten Sie das Projekt „abalo“ mit Laravel über composer ein. Nennen Sie das Projekt „abalo“. Starten Sie das Projekt zum Test über `artisan serve` (oder mit `php artisan serve`).

Aufgabe 4

Richten Sie die IDE ein. Richten Sie das Projekt „Abalo“ in Ihrer IDE so ein, so dass Sie das Projekt komfortabel direkt darüber starten können.
Installieren Sie hilfreiche Plugins der IDE, z.B. für die komfortable Bearbeitung von Konfigurationen (wie composer.json)

Aufgabe 5 (optional)

Testverbindung. Wir führen einen Test unseres Setups durch, damit wir später direkt mit der eigentlichen Anwendungsentwicklung beginnen können. Dafür legen wir eine Tabelle mit Testdaten in der Datenbank an und zeigen diese Daten im Webbrowser an.

a) Legen sie die folgende Tabelle an.

ab_testdata

Attribut	Typ	Kommentar
id	int8	Primärschlüssel
ab_testname	varchar (80), not null, unique	Testname eines Artikels

b) Tragen Sie die folgenden Daten in die Tabelle ab_testdata ein.

id	ab_testname
1	Fotokamera
2	Blitzlicht

c) Schreiben Sie ein Model „AbTestData“ unter Verwendung von Eloquent, das die Daten der Tabelle ab_testdata auslesen und beschreiben kann.

d) Legen Sie die Route /testdata an, die über einen Aufruf des Controllers AbTestDataController die Daten über das Model „AbTestData“ ausliest und ausgibt.

Aufgabe 6

Anmeldung. Ziel ist die Umsetzung der Simulation einer Anmeldung. Im späteren Verlauf des Praktikums benötigen wir Daten der Benutzer:innen, wie ID oder Anmeldename (oft E-Mail), was wir gewöhnlich durch eine Anmeldung erhalten. Dazu legen wir uns einen Controller an, der eine Anmeldung simuliert, damit uns die notwendigen Daten später zur Verfügung stehen.
Integrieren Sie dafür einen bestehenden Controller, der einen automatischen Login für eine/n bestimmte Nutzer:in durchführen soll. Verwenden Sie als Controller den zur Verfügung gestellten Controller „AuthController.php“. Tragen Sie die notwendigen Einträge in die web.php ein. Sie erhalten die zwei Dateien enthalten in der auth.zip. Entfernen Sie vor dem Entpacken das „.sec“ bei der auth.zip.:

- AuthController.php
- web.php

Der AuthController legt lediglich serverseitig einige Daten in die PHP-Session. Schauen Sie sich den AuthController an.

Hinweis: Wir entwickeln keine vollständige Anmeldung für Abalo, da wir dies bereits in DBWT kennengelernt und umgesetzt haben. Sie sind jedoch frei eine vollständige Anmeldung unter Nutzung von Laravel umzusetzen.

Woche 2

Empfehlung VL: Schemamigration

Aufgabe 7

Schemamigration. Erzeugen Sie die folgende Datenstruktur in der Datenbank „abalo“ unter Einsatz von Migrationsdateien in Laravel. Setzen Sie notwendige Fremdschlüssel und sonstige geforderte Nebenbedingungen.

Hinweis: uint8 steht für einen 8-Byte Unsigned (Big-)Integer.

ab_user

Attribut	Typ	Kommentar
id	uint8	Primärschlüssel
ab_name	varchar (80), not null, unique	Name
ab_password	varchar (200), not null	Passwort
ab_mail	varchar (200), not null, unique	E-Mail-Adresse

ab_article

Attribut	Typ	Kommentar
id	uint8	Primärschlüssel
ab_name	varchar (80), not null	Name
ab_price	int, not null	Preis in Cent
ab_description	varchar (1000), not null	Beschreibung, die die Güte oder die Beschaffenheit näher darstellt. Wird durch den „Ersteller“ (ab_user) gepflegt
ab_creator_id	uint8, not null	Referenz auf den/die Nutzer:in, der den Artikel erstellt hat und verkaufen möchte
ab_createdate	TIMESTAMP, not null	Zeitpunkt der Erstellung des Artikels

ab_articlecategory

Attribut	Typ	Kommentar
id	uint8	Primärschlüssel
ab_name	varchar (100), not null, unique	Name
ab_description	varchar (1000), nullable	Beschreibung

Attribut	Typ	Kommentar
ab_parent	uint8, nullable	Referenz auf die mögliche Elternkategorie. Artikelkategorien sind hierarchisch organisiert. Eine Kategorie kann beliebig viele Kind Kategorien haben. Eine Kategorie kann nur eine Elternkategorie besitzen. NULL, falls es keine Elternkategorie gibt und es sich um eine Wurzelkategorie handelt.

ab_article_has_articlecategory

Attribut	Typ	Kommentar
id	uint8	Primärschlüssel
ab_articlecategory_id	uint8, not null	Referenz auf eine Artikelkategorie
ab_article_id	uint8, not null	Referenz auf einen Artikel
	UNIQUE	ab_articlecategory_id, ab_article_id

Aufgabe 8

Übersicht. Erstellen Sie zur Datenstruktur unter Aufgabe 7 zur Übersicht ein ER-Diagramm nach Chen in elektronischer Form. (Verwenden Sie z.B. draw.io)

Aufgabe 9

Laden von Daten. Wir laden initial Daten in die aufgebaute Datenstruktur von Abalo. Sie erhalten die drei Dateien (enthalten in **data.zip**):

- articlecategory.csv
- article.csv
- user.csv

a) Laden Sie mit Hilfe eines DataSeeders mit dem Namen „DevelopmentData“ Daten in die Datenbank.

b) Führen Sie mögliche notwendige Konvertierungen an den Daten in PHP durch. Verändern Sie die Daten nicht direkt in der csv-Datei!

Hinweis: CSV-Daten können auf unterschiedliche Arten in die Datenbank geladen werden. Recherchieren Sie eine Möglichkeit, wie über PHP eine CSV-Datei komfortabel eingelesen werden kann.

Aufgabe 10

Artikelübersicht. Als erste Funktionalität mit Nutzen für Kund:innen entwickeln wir eine Übersicht von angebotenen Artikeln.

a) Implementieren Sie eine Artikelübersicht unter dem Endpunkt /articles/, wo Nutzer:innen nach Artikeln suchen können. Der Suchbegriff soll nur über die

Abfrage der URL gesteuert werden (via &search=) und das Attribut ab_article.ab_name in der Datenbank abfragen. Der Artikel soll dann als Treffer gelten, wenn das Suchwort auch nur teilweise in ab_name enthalten ist. Verwenden Sie eine Suche, die keine Groß- und Kleinschreibung berücksichtigt (nicht case-sensitive). Verwenden Sie kein CSS. Zeigen Sie die Artikel und deren Daten in einer Tabelle an. Ist kein Suchbegriff gesetzt sollen alle Artikel dargestellt werden.

b) Sie erhalten die Bilder zu den Artikeln (articleimages.zip). Zeigen Sie das hinterlegte Bild des Artikels. Ein Artikel besitzt ein Bild, wenn ein zugehöriges Bild mit <ab_article.id>.{png, jpg} hinterlegt ist.

Aufgabe 11

Massendaten. Für später geplante Performancetests der Anwendung sollen bereits jetzt sehr viele Benutzer:innen in der Datenbank zur Verfügung stehen. Schreiben Sie eine geeignete Struktur zur Erzeugung von Massendaten, wobei Sie mindestens 10000 zufällige Benutzer:innen in Tabelle ab_user erzeugen.

Führen Sie die weitere Entwicklung allerdings ohne die mögliche große Anzahl an Datensätzen in der Tabelle ab_user fort.

Aufgabe

Abgabe. Laden Sie Ihre Ergebnisse als ZIP in unter

https://git.fh-aachen.de/praktika/dbwt2

in den Ordner des aktuellen Semesters und Ihres Teams in den Branch „main“ hoch. Der Name des ZIPs soll sein:

M1.zip

Das ZIP soll alle Ergebnisse des Meilensteins (Antworten zu Freitextaufgaben, Grafiken, Quelltexte, …) enthalten. Fügen Sie zusätzlich ein README ein, in der Sie Vor-, Nachname und Matrikelnummer der beteiligten Teammitglieder hinterlegen. Eine Abgabe einer Person für das Team reicht aus. Die letzte Abgabe vor dem Abgabezeitpunkt (Samstag, 14:00 Uhr) zählt.