Begrüssung

Willkommen zu meiner Präsentation über meine Individuelle praktische Arbeit. Ich werde hier die Vorgehensweise und Umsetzung meiner IPA vorstellen. Diese fand über den Zeitraum von zehn Tagen vom 13. März bis zum 28. März statt.

Agenda | Auftrag

Die Präsentation ist folgendermassen aufgebaut:

Zuerst werde ich die Vorgaben und den Auftrag grob beschreiben.

Dann werde ich Aufzeigen wie ich von Anfang bis ende Vorgegangen bin, um den Auftrag umzusetzen.

Es folgt eine kurze Auswertung, wo ich die Vorgaben mit dem finalen Produkt vergleiche.

Zum Schluss noch mein persönliches Fazit und was ich aus dieser Arbeit mitnehme.

Projektauftrag

Hier ist was die Applikation können sollte. Es sollte Ein Verwaltungssystem erstellt werden, in dem Shops und ihre Produkte erfasst werden können. Aus diesen Daten sollen dann Bestellungen mit Produkten erstellt werden können. Dadurch soll eine Übersicht für ausgeführte Bestellungen vorhanden sein.

Funktionalität

Benutzer können sich mit einer E-Mailadresse Registrieren und Anmelden, was ihnen Zugang auf alle Daten gibt. Zur Authentifizierung wird ein Bearer-Token verwendet und die Kommunikation findet in JSON statt. Es existiert kein Frontend aber eine Dokumentation mit Swagger.

Mittel

Zur Umsetzung wurden folgende Mittel und Technologien eingesetzt.

Was ist XAMPP

Als lokale Server-Umgebung wurde XAMPP verwendet. XAMPP ist eine Abkürzung für Cross-Plattform, Apache, MySQL, PHP, Perl. Dadurch wurde also Apache als Webserver, MySQL als Datenbank und PHP als Programmiersprache zur Verfügung gestellt.

https://www.educba.com/what-is-xampp/

Was ist Laravel

Anstatt Vanilla-PHP wurde das Framework Laravel benutzt. Laravel bietet diverse nützliche Tools.

Die wichtigsten davon sind die MVC-Struktur von der hier nur der Model-Controller Teil benutzt wird, da wir keine Views haben,

Artisan ist das Comand-Line-Interface, das von Laravel bereitgestellt wird, damit können Migrationen durchgerührt werden und Boiler Plate-Code erzeugt werden,

Eloquent ist das ORM oder Object-Relational-Mapper, es verbindet die Modelle mit der Datenbank

Sanctum ist ein Authentisierungs System das Laravel bietet und eine einfache Integration von Bearer-Token erlaubt

https://builtin.com/software-engineering-perspectives/laravel

Was ist eine API

Die Umsetzung als API oder Application Programming Interface, es ist eine Sammlung von Definitionen und Protokollen, für das Erstellen von Software. Es erlaubt es ein Produkt oder Service bereit zu stellen, ohne zu wissen wie diese später implementiert werden. In diesem Fall eine Bestellungsverwaltung. Die Endpoints, die erstellt wurden, können von einer Mobile-App, einer Web-App oder auch von einer anderen API verwendet werden.

edhat.com/en/topics/api/what-are-application-programming-interfaces

JSON steht für JavaScript Object Notation und ist ein gängiges Format für das Speichern und Transportieren von Daten.

Was ist Swagger

Swagger ist Open Source und bietet Regeln, Spezifikationen und Tools für die Entwicklung und Beschreibung von RESTful-APIs. Das Swagger-Framework ermöglicht es Entwicklern, interaktive, maschinen- und menschenlesbare API-Dokumentation zu erstellen.

Das hier benutzte L5-Swagger ist eine auf Laravel angepasste Version.

https://www.techtarget.com/searchapparchitecture/definition/Swagger

Agenda | Vorgehen

Nun zum Vorgehen.

IPERKA

Bei meiner Vorgehensweise folgte ich der IPERKA Projekt-Methode. Diese stellt sich aus den sechs Schritten Informieren, Planen, Entscheiden, Realisieren, Kontrollieren und Auswerten zusammen.

Informieren

Im ersten Schritt der Informations-Phase, ging es hauptsächlich darum kleinere Unklarheiten zu beheben. Wie zum Beispiel, ob jeder Benutzer auf alle Daten jedes Benutzers zugreifen darf. Da ich mit den verwendeten Technologien schon weitestgehend vertraut war, hielt sich der Recherchier Bedarf aber in Grenzen.

Planen

Zeitplan (sieht man ok)

Die gesamte Planung für die Umsetzung der IPA, wurde in einem Zeitplan festgehallten. Wobei der geschätzte Aufwand in Orange und der tatsächliche Aufwand in Grün dargestellt ist. Wie man sehen kann, hatte ich eine grössere Fehleinschätzung, beim Punkt Testkonzept erstellen.

Klassendiagramm (sieht man ok)

Das Klassendiagramm sollte in der Dritten Normalform sein und musste bestimmte Werte in bestimmten Tabellen enthalten.

Aktivitätsdiagramm (sieht man nicht)

In diesem Aktivitätsdiagramm ist abgebildet, wie ein neues Produkt erstellt wird. Man sieht es hier zwar nicht so gut, ich möchte aber einfach die Schritte aufzeigen, die in dieser Anwendung bei jedem Request nötig sind.

Schritt 1, Route-Model-Binding

Der erste Schritt ist optional. Falls im URL eine ID mitgegeben wurde, wird mit dieser von Laravel das entsprechende Model aus der Datenbank geholt. Schlägt das fehl, wird eine Fehlermeldung zurückgegeben. Diese Funktionalität heisst Route Model Binding und ich werde dies später noch einmal aufgreifen.

Schritt 2, Validierung

Im zweiten Schritt werden alle im Post Request mitgegebenen Daten validiert.

Schritt 3, File Upload

Da Produkte ein Bild enthalten können, ist dieser zusätzliche Schritt nötig, um einen separaten File Upload zu machen.

Schritt 4, Datenbankspeicherung / Antwort

Am Ende dieses Vorganges, wird ein neuer Eintrag in der Datenbank erstellt.

Schritt 5, Fehlermeldung

Falls zu irgendeinem Zeitpunkt in diesem Prozess fehlerhafte oder ungültige Daten erkannt werden, muss eine Fehlermeldung generiert und zurückgegeben werden.

Aufbaudiagramm (sieht man die Details nicht)

Hier sieht man den Ablauf, einer Interaktion mit der API vereinfacht dargestellt.

Im Router werden die URLs definiert, die von aussen angesprochen werden können.

Diese Routes korrespondierend zu einer Methode im Controller.

Die Methoden akzeptieren als Parameter einen bestimmten Request Typ. Zum Beispiel einen Produkt Update Request. In diesem Request ist definiert, was alles Enthallten sein muss. Dinge wie Shop ID, Namen und Bild URL. Hier werden diese Daten auch gleich validiert.

In der Controller Methode wird dann eine Resource erzeugt. Diese definiert das Format der Antwort und wie dessen Daten präsentiert werden.

Zum Schluss wird die Resource als JSON-Response zurückgegeben.

Entscheiden

In der Entscheidungsphase kamen mir die Laravel-Konventionen, die genauen Vorgaben der Aufgabenstellung und meine Erfahrung mit den verwendeten Technologien zugute. So gab es nicht so viele Entscheidungen, die getroffen werden mussten.

Entscheidungs-Matrix mit Scout

Bei den Entscheidungen, die getroffen werden mussten, habe ich versucht so systematisch wie möglich vorzugehen. Dazu benutzte ich Tabellen mit gewichteten Werten, um die Optionen zu vergleichen.

Realisieren

Nach der ausführlichen Planungsphase ging es schlussendlich an die Realisationsphase.

Migrationen (Vorgabe)

Modell (Vorgabe)

Request mit Fehlermeldung (Vorgabe)

Ressource (Vorgabe)

Routen mit Route-Model-Binding

404 Fehlermeldung

Controller

Swagger-Annotation

Swagger Produkt

~~Kontrollieren~~

Auswerten

Vorgaben mit Produkt vergleichen

Fazit

Reflexion