Das PHP-Framework Laravel im Vergleich mit Symfony

Sebastian Dominik Schmid

BACHELORARBEIT

Nr. 1310238061-A

eingereicht am
Fachhochschul-Bachelorstudiengang

Medientechnik und -design

in Hagenberg

im Februar 2016

Diese Arbeit entstand im Rahmen des Gegenstands

Web Applications

im

Wintersemester 2015/16

Betreuer:

DI Martin Harrer

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 4. Februar 2016

Sebastian Dominik Schmid

Inhaltsverzeichnis

Kurzfassung

Diese Bachelorarbeit befasst sich mit dem PHP-Framework Laravel, das kurz nach seiner Entstehung bereits auffallend populär wurde.

Dazu werden zuerst einige Grundlagen erläutert, die Frameworks allgemein betreffen. Außerdem werden Aspekte aufgezeigt, die für eine Verwendung von Frameworks in der Webentwicklung sprechen und somit bei der Auswahl entscheidend sein können. Dazu zählen eine einheitliche Struktur nach dem MVC-Prinzip, effiziente Entwicklung durch Automatisierung typischer Abläufe sowie Sicherheit durch Validierung.

Daraufhin wird Laravel als das derzeit beliebteste PHP-Framework vorgestellt. Dabei wird besonders auf das Versprechen einer klaren, verständlichen Syntax und eines angenehmeren Programmierens eingegangen. Außerdem fallen ausführliche Dokumentationen und die Videoplattform Laracasts als Lerngrundlagen auf.

Für einen konkreten Vergleich wird Symfony herangezogen, welches sich bereits lang bewährt hat und zudem die Grundlage Laravels darstellt. Ein theoretischer Vergleich zeigt zunächst Ähnlichkeiten im Aufbau aber auch konkrete Unterschiede, besonders bei der Datenbankanbindung.

Als Praxisvergleich wird ein einfaches Forum in beiden Frameworks erstellt. Dabei fallen besonders beim Routing und im Bereich des Models Unterschiede auf.

Insgesamt lässt sich sagen, dass Laravel auf effiziente Implementierung ausgelegt ist und trotz seiner Übersichtlichkeit recht viel Flexibilität bietet. Außerdem ist die Syntax gut lesbar und aufgrund der durchgehenden Verwendung von PHP einheitlicher. Symfony setzt hingegen auf freiere Gestaltung der Struktur und Wiederverwendung eigener Implementierung durch Bundles. Jedoch ist die Konfiguration aufwendiger und die Syntax durch unterschiedliche Formate und unabhängige Komponenten weniger übersichtlich als in Laravel.

Abstract

This bachelor thesis addresses the PHP framework Laravel, which became strikingly popular in the few years of its existance.

First of all some fundamentals about frameworks in general are explained. Moreover advantages of using a framework in web development are shown, which are also important points in the decision between different frameworks. Examples are the consistent structure of MVC, automation of typical procedures to develop more efficiently and validation by the framwork for higher security.

After that, Laravel as the currently most popular PHP framework is presented. It pledges clear and understandable syntax and a pleasent programming experience. Also the detailed documentation and Laracasts, a platform for video tutorials, attract attention as learning aids.

Symfony is chosen for a comparison, because it is widely used and Laravel is based on it. A theoretical comparison shows similarities in the overall structure but also differences, e. g. in database handling.

Then a simple forum is created in both frameworks to compare them practically. Differences are most noticeable in routing and dealing with the data model.

All in all Laravel is focused on efficiency and clearness but still flexible. Moreover its syntax is easy to read and more consistent, because PHP is used for everything. Symfony provides more freedom in structuring the project and reusing bundles of your own or other developers. However, configuration is more complex and the syntax is not as clear, because different formats and independent components are used.

Kapitel 1

Einleitung

Das relativ neue Framework *Laravel* erlangte auffällig schnell extreme Beliebtheit unter den PHP-Entwicklern weltweit. Was diesen Erfolg ausmacht und in welchen Fällen Laravel tatsächlich Vorteile bringt, ist Gegenstand dieser Arbeit. Zunächst folgt ein Überblick über Zielsetzung, Relevanz und Aufbau.

1.1 Zielsetzung und Fragestellung

Diese Arbeit soll beschreiben, worin die generellen Gründe für die Verwendung eines PHP-Frameworks in der Webentwicklung bestehen, und auf dieser Grundlage die speziellen Vorteile Laravels aufzeigen. In einem Praxisvergleich sollen die prägnantesten Unterschiede zum Framework *Symfony* erfasst werden, welches als Grundlage für die Entwicklung Laravels verwendet wurde. Unter Einbeziehung der Ergebnisse wird insgesamt untersucht, was mögliche Gründe für die Popularität Laravels sind. Die konkrete Fragestellung lautet demnach folgendermaßen:

Wo liegen besondere Vorteile des PHP-Frameworks Laravel und in welchen Situationen ist sein Einsatz sinnvoll? Wodurch unterscheidet sich Laravel von seiner Konkurrenz und Grundlage Symfony? Welche Gründe gibt es für die schnelle Verbreitung und die große Beliebtheit des Frameworks?

1.2 Relevanz der Arbeit

Zunächst soll die Arbeit einen Überblick über Funktionsumfang, Aufbau und Verwendung von Laravel bieten. Außerdem kann sie als Entscheidungshilfe dienen, wenn der Einsatz Laravels oder generell eines PHP-Frameworks in Erwägung gezogen wird. Relevant sind die Ergebnisse demnach besonders für Neueinsteiger, die bisher andere Systeme oder überhaupt kein Framework verwendet haben. Ein Abgleich der Versprechungen mit den Erfahrungen

im Praxiseinsatz und die Untersuchung weiterer Erfolgsfaktoren sollen eine reflektierte Entscheidung, unabhängig vom Verbreitungsgrad, ermöglichen.

1.3 Aufbau der Arbeit

Als Einstieg dient eine Beschreibung von Frameworks im Allgemeinen sowie speziell in der Webentwicklung mit PHP. Dabei werden zusätzlich Grundlagen wie das MVC-Prinzip oder typische Abläufe in Webapplikationen erläutert und gezeigt, welche Vorteile der Einsatz von Frameworks bietet.

Darauf folgt ein erster Überblick über Laravel. Darin wird der Erfolg beschrieben und untersucht, durch welche Versprechungen auf der offiziellen Website sich das Framework von anderen absetzen soll.

In einem theoretischen Vergleich werden dann Struktur, Funktionsumfang und Handhabung von Laravel und Symfony gegenübergestellt.

Auf diesen Grundlagen findet schließlich ein Praxisvergleich statt. Dabei wird ein einfaches Frage-Antwort-Forum in beiden Frameworks identisch implementiert und die Unterschiede in den wichtigsten Punkten herausgearbeitet.

Zusammengefasst werden alle Ergebnisse in einem Fazit-Kapitel. Dabei wird die Theorie mit der Praxis verglichen und Schlussfolgerungen aus den vorherigen Abschnitten gezogen.

Kapitel 2

Grundlegendes zu Web-Frameworks

Dieses Kapitel soll als Einstieg einen kurzen Überblick über das Konzept von Frameworks geben und aufzeigen, warum und unter welchen Umständen der Einsatz Vorteile bringt. Nach einer allgemeinen Einführung in die Thematik wird erläutert, wie Frameworks speziell in der Webentwicklung eingesetzt werden. Anhand verschiedener Aspekte der serverseitigen Entwicklung soll zudem gezeigt werden, warum sich die Verwendung eines PHP-Frameworks wie Laravel grundsätzlich lohnt.

2.1 Frameworks im Allgemeinen

Der Begriff Framework bezeichnet in diesem Kontext ein Grundgerüst, aus dem verschiedenste Applikationen entwickelt werden können. Es gibt meist die Teile der Funktionalität vor, die in vielen verschiedenen Anwendungsfällen nötig sind und ohne Framework jedes Mal auf dieselbe Weise implementiert werden müssten. Das betrifft oft Schnittstellen zu anderen Programmen oder die Kompatibilität mit unterschiedlichen Plattformen und Betriebssystemen. Der Entwickler beginnt somit nicht bei Null, sondern baut die Applikation auf einem bestehenden Gerüst auf, indem er ausschließlich neue und für den jeweiligen Fall spezifische Teile hinzufügt.

Frameworks kamen bereits Anfang der 1980er Jahre zur Verwendung, als auch die objektorientierte Programmierung immer populärer wurde. Die Idee ergab sich aus der Bestrebung, Klassen möglichst so aufzubauen, dass sie später in anderen Anwendungen wiederverwendet werden können. Jedoch unterscheidet sich ein Framework von einer reinen Bibliothek aus Klassen und Funktionen durch weitere Eigenschaften, wie Ralph E. Johnson und Brian Foote 1988 im Zuge ihrer Definition erläuterten [2, Seite 4]:

> A framework is a semi-complete application. A framework provi-

> des a reusable, common structure to share among applications. Developers incorporate the framework into their own application and extend it to meet their specific needs. Frameworks differ from toolkits by providing a coherent structure, rather than a simple set of utility classes.

Frameworks geben also zusätzlich zu ihrer Funktionalität eine gewisse Struktur vor. In diesem Zusammenhang spielt auch die Funktionsweise eine wichtige Rolle, die sich grundlegend von reinen Bibliotheken unterscheidet. Nach dem Prinzip *Inversion of Control* werden nicht Komponenten des Frameworks aufgerufen, sondern der eigene Programmcode durch das Framework. Die eigenen Programmteile werden ausgeführt, sobald sie benötigt werden. Alles Weitere wird durch das Framework übernommen.

Aus den genannten Eigenschaften ergeben sich mehrere Vorteile, die bereits im Allgemeinen für den Einsatz von Frameworks sprechen. Der wichtigste Punkt ist die Ersparnis an Aufwand und Arbeitszeit. Da grundlegende Funktionen bereits vorhanden sind, die sonst bei jedem Projekt erneut Arbeit in Anspruch nehmen würden, bleibt mehr Zeit für das Wesentliche. Der Entwickler beschäftigt sich nur mit der Funktionalität, die das Produkt letztendlich ausmacht und es von anderen Applikationen unterscheidet.

Doch nicht nur die Effizienz wird gesteigert, sondern ebenso die Qualität. Frameworks werden meist von vielen Entwicklern kontinuierlich optimiert, sodass sie im Zweifel stabiler und sicherer sind als selbst geschriebene Programme. Man erspart sich also nicht nur die eigene Arbeitskraft, sondern erhält im Gegenzug die Leistung und Erfahrung aller Entwickler, die hinter dem Framework stehen.

Weitere Vorteile bietet die einheitliche Struktur, besonders, wenn gemeinsam an einem Projekt gearbeitet wird. Ist man einmal mit einem Framework vertraut, findet man sich schneller in Projekten zurecht, die ebenfalls darauf aufbauen.

2.2 Bedeutung von Frameworks in der Webentwicklung

Speziell in der Webentwicklung werden Frameworks häufig und in verschiedenen Bereichen eingesetzt. Sowohl serverseitig als auch clientseitig gibt es bestimmte Abläufe und Funktionsweisen, die in fast jeder Anwendung zu finden sind. Zusätzlich gibt es für viele Spezialgebiete wiederum Frameworks, die beispielsweise rein für den Aufbau einer Websocket-Verbindung oder zur Anzeige von grafischen Objekten im Browser – teilweise innerhalb eines anderen Frameworks – verwendet werden. Wie sich die bereits beschriebenen Vorteile eines Frameworks speziell im Webbereich auswirken, wird nun anhand einiger wichtiger Aspekte näher erläutert.

2.2.1 MVC

Das MVC-Prinzip beschreibt eine Struktur, die in vielen Bereichen der Softwareentwicklung verbreitet ist. Viele Frameworks in der Webentwicklung legen eine MVC-Struktur zugrunde. Manche sind wiederum flexibler, bieten aber ebenfalls Möglichkeiten, nach diesem Prinzip zu arbeiten.

Trygve Reenskaug, Erfinder dieser Programmiertechnik, fasst das Ziel in seinem Artikel *The model-view-controller (mvc) its past and present* von 2003 folgendermaßen zusammen [1]:

> MVC was conceived in 1978 as the design solution to a particular problem. The top level goal was to support the user's mental model of the relevant information space and to enable the user to inspect and edit this information.

Die Abkürzung MVC steht für drei getrennte Aufgabenbereiche innerhalb einer Applikation: Model, View und Controller. Das Model umfasst alle Daten, deren Strukturierung, Speicherung und Verwaltung. In serverseitigen Frameworks betrifft das beispielsweise den Datenbankzugriff. Die Präsentation dieser Daten gegenüber dem Benutzer ist Aufgabe des Views. In diesen Bereich fallen Templates für HTML-Dokumente, die an den Client gesendet werden. Der Controller wiederum erlaubt es dem Benutzer, die Daten im Model zu manipulieren. Am Server entspricht das der Programmlogik, die durch einen eingehenden Request gestartet wird.

Durch eine Aufteilung in diese logischen Bereiche wird der Webapplikation eine Struktur verliehen. Sie wird übersichtlicher, skalierbar und leichter zu warten. Deshalb ist das MVC-Prinzip eine wichtige Grundlage für die weiteren Betrachtungen in dieser Arbeit.

2.2.2 Konzentration auf das Wesentliche

Der besondere Nutzen von Frameworks besteht jedoch nach wie vor in der Wiederverwendung von Funktionalität. In Webapplikationen gibt es eine Vielzahl von Abläufen, die grundlegend sind und immer wieder auf ähnliche Weise implementiert werden müssen. Um eine möglichst hohe Effizienz bei der Entwicklung zu erzielen, sollte man sich jedoch auf die spezifischen Bereiche konzentrieren. Anhand eines abstrakten Beispiels soll nun aufgezeigt werden, wie umfassend Frameworks den typischen Ablauf in einer Webapplikation unterstützen.

Der Prozess – hier der Aufruf einer Webseite (siehe Abb. 2.1) – beginnt, indem ein Request des Clients – hier ein Browser – am Server eintrifft. Dieser Request muss interpretiert werden, um zu entscheiden, was genau zu tun ist. Bereits hier kann ein Framework große Teile des Problems lösen. Nur das Routing, also die Zuordnung zwischen der URL im Request und der jeweils auszuführenden Aktion, ist anwendungsspezifisch. Daraufhin könnte

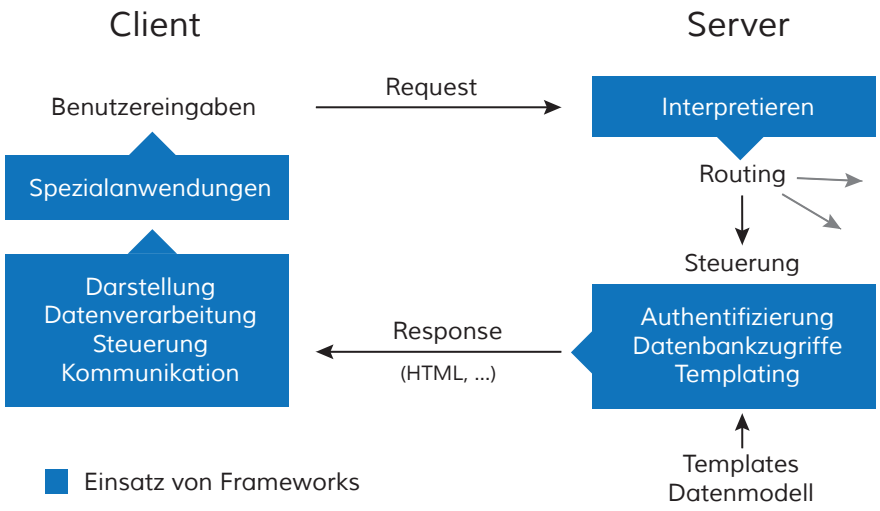


Abbildung 2.1: Typischer Kreislauf einer Webapplikation und mögliche Einsatzbereiche für Frameworks.

eine Authentifizierung nötig sein, um zu prüfen, ob der Client als Nutzer angemeldet ist. Auch das ist eine Standardaufgabe in vielen Applikationen, die ein Framework übernehmen kann. Als nächstes wird die HTML-Seite erstellt und mit für den Nutzer spezifischen Daten bestückt. Wenn diese aus einer Datenbank bezogen werden, kann das Framework den Zugriff abstrahieren und erleichtern. Es kann auch das Erstellen des HTML-Dokuments durch den Einsatz von Templates einfacher machen. Wird dann die Response, also die Antwort des Servers inklusive HTML, an den Client geschickt, übernimmt der clientseitige Teil der Applikation. Auch hier können JavaScript-Frameworks eingesetzt werden, die gerade bei umfangreicheren Anwendungen Vorteile bringen. Sie trennen unter anderem die Daten von der Darstellung und erleichtern die weitere Kommunikation mit dem Server, etwa über eine REST-Schnittstelle.

Übrig bleiben also tatsächlich nur Punkte, die auch in der Konzeption der neuen Anwendung eine Rolle spielen: Welche Funktionalität kann der Nutzer wie abrufen? Welche Daten werden gespeichert und wie werden sie manipuliert? Und wie werden die Ergebnisse letztendlich dem Nutzer präsentiert? Alle weiteren Aufgaben, die keine dieser Fragen betreffen, werden zu großen Teilen bereits durch Frameworks implementiert. Hier zeigt sich außerdem der Sinn der MVC-Struktur, da die Bereiche, auf denen nun der Fokus liegt, genau den drei Komponenten dieses Musters entsprechen.

2.2.3 Einfachere Zusammenarbeit

Übliche Skriptsprachen in der Webentwicklung wie PHP oder JavaScript geben wenige Limitierungen vor und lassen großen Spielraum in der Gestaltung der Codestruktur. Das führt zu einer Vielzahl an unterschiedlichen Herangehensweisen und Projektstrukturen, je nach Einsatzgebiet und Vorlieben der Entwickler. Was grundsätzlich nützliche Flexibilität bringt, kann jedoch zum Problem werden, sobald mehrere Menschen an einem Projekt arbeiten. Sie müssen sich auf gewisse Strukturen und Normen einigen, um nicht mit der Zeit den Überblick zu verlieren. Doch auch damit bleibt es für Neueinsteiger schwierig, sich in einem bestehenden Projekt zurechtzufinden.

Frameworks geben oft eine klare Struktur vor oder helfen zumindest dabei, konsistent zu bleiben. Mit der Wahl eines passenden Frameworks erübrigen sich somit viele Diskussionen über den grundlegenden Aufbau und einheitliche Strukturen. Die Wartbarkeit wird erhöht, da man auch ältere Projekte schneller wieder durchschaut, in denen ein bekanntes Framework zum Einsatz kam. Auch neue Kollegen finden sich schneller zurecht, wenn sie mit dem vorhandenen System bereits Erfahrung haben.

Jedoch kommt es hier sehr auf die richtige Wahl an. Die Struktur ist ein wichtiger Aspekt, der auch beim Vergleich im Vorhinein mit einbezogen werden sollte. Manche Frameworks geben einen relativ strikten Rahmen vor, andere sind extrem flexibel. Zudem ist der Verbreitungsgrad ein wichtiger Faktor, wenn es darum geht, den Einstieg für neue Entwickler zu erleichtern. Ein absolutes Nischenprodukt bringt also in Bezug auf diesen Punkt weniger Vorteile. Dieser Aspekt sollte aber nicht über die tatsächliche Funktionalität des Frameworks gestellt werden. Wenn aus Prinzip immer auf das gleiche System gesetzt wird, kann es vorkommen, dass es in Projekten eingesetzt wird, für die es ungeeignet ist und im Vergleich zu Alternativen eher Nachteile bringt.

2.3 PHP-Frameworks

Da die Thematik dieser Arbeit in der serverseitigen Entwicklung mit PHP liegt, sind noch einige Punkte zu erwähnen, die speziell PHP-Frameworks betreffen. Zunächst folgt jedoch ein Überblick über die Skriptsprache selbst und eine kurze Demonstration, wie sie durch den Einsatz eines Frameworks aufgewertet werden kann.

PHP ist nach vielen Jahren immer noch die am weitesten verbreitete serverseitige Skriptsprache. Nach Angaben von W3Techs [10] verwenden im Dezember 2015 über 80% aller Websites, deren serverseitige Sprache bekannt ist, PHP. In Anbetracht der nach W3Techs [11] relativ konstanten Entwicklung steht die Relevanz dieser Technologie also außer Frage.

Seit Rasmus Lerdorf die Skriptsprache 1995 für seine persönliche Homepage entwarf, wurde sie kontinuierlich weiterentwickelt. Unter der Leitung

Programm 2.1: Ausgabe des aktuellen Datums mit PHP in der einfachsten Form.

```
1 <p>Heute ist der <?php echo date('d.m.Y') ?>.</p>
```

Programm 2.2: Hier lautet die Ausgabe z. B. „Heute ist Freitag, der 11.12.2015.“. Der Code ist bereits schwieriger nachzuvollziehen.

```
1 <?php
2 function printDay() {
3   $days = array('0' => 'Sonntag', '1' => 'Montag', '2' => 'Dienstag', '3
      ' => 'Mittwoch', '4' => 'Donnerstag', '5' => 'Freitag', '6' => '
      Samstag');
4   $dayNumber = date('w');
5   echo $days[$dayNumber];
6 }
7 ?>
8 <p>Heute ist <?php printDay() ?>, der <?php echo date('d.m.Y') ?>.</p>
```

von Zend Technologies Ltd. entstand daraus bald eine komplexe Programmiersprache mit unzähligen Bibliotheken. PHP ist im kleinen Rahmen einsetzbar, um nur einzelne Teile eines HTML-Dokuments mit wenig Aufwand dynamisch zu gestalten. Gleichzeitig bietet es aber den vollen Funktionsumfang, um auch komplexe Webapplikationen zu realisieren. Das zeigt sich unter anderem darin, dass die bekanntesten Content-Management-Systeme wie *WordPress*, *Drupal* oder *Typo3* auf PHP setzen. Auch weltbekannte Plattformen verwenden laut W3Techs [11] PHP für ihre komplexen Applikationen, darunter Facebook, Twitter und Wikipedia.

2.3.1 PHP ohne Framework

Der Einstieg in PHP stellt keine große Hürde dar, denn mit wenig Code können bereits sinnvolle Ergebnisse erzielt werden. Programm 2.1 zeigt, wie das heutige Datum inline innerhalb des HTML-Dokuments generiert wird. Durch Abgrenzung mit `<?php` und `?>` kann an beliebiger Stelle ein Abschnitt mit PHP-Code eingefügt werden.

Werden die dynamischen Inhalte komplexer, bietet es sich an, Code in eigene Funktionen auszulagern. In einem einfachen Fall wie in Programm 2.2 kann damit die Übersichtlichkeit gewahrt bleiben. Es ist jedoch bereits zu erkennen, dass ein HTML-Dokument mit integrierten PHP-Abschnitten bei zunehmender Komplexität schnell unübersichtlich wird. Das Beispiel zeigt schließlich nur einen Ausschnitt des entsprechenden kompletten HTML-Dokuments.

Programm 2.3: Inhalt der Template-Datei in Laravel.

```
1 <p>Heute ist {{$day}}, der {{$date}}.</p>
```

Programm 2.4: Inhalt des Controllers in Laravel.

```
1 $days = array('0' => 'Sonntag', '1' => 'Montag', '2' => 'Dienstag', '3'
      => 'Mittwoch', '4' => 'Donnerstag', '5' => 'Freitag', '6' => '
      Samstag');
2 return view('heute', [
3   'day' => $days[date('w')],
4   'date' => date('d.m.Y')
5 ]);
```

Um lange unstrukturierte Dokumente zu vermeiden, kann PHP-Code in mehrere Dateien gegliedert werden, die dann per `require ...` eingefügt werden. Ein weiterer Schritt wäre, direkt objektorientiert zu programmieren und somit Klassen und Methoden zu erstellen, die im gesamten Projekt zum Einsatz kommen können. Spätestens in dieser Situation bietet es sich an, ein geeignetes PHP-Framework in Erwägung zu ziehen.

2.3.2 Templating

Um beim Thema der Übersichtlichkeit zu bleiben, soll nun mit dem sogenannten *Templating* der am schnellsten ersichtliche Vorteil des Framework-Einsatzes gezeigt werden. Template-Engines können auch eigenständig verwendet werden, um die gleichen Effekte zu erzielen. Doch sie stellen einen wichtigen Bestandteil vieler PHP-Frameworks dar und bilden zudem den View in einer MVC-Struktur, die ganzheitlich umgesetzt noch mehr Potenzial bietet.

Das Prinzip dabei ist eine Trennung des Views, also der HTML-Struktur, von der PHP-Logik. Dazu wird eine Template-Datei erstellt, die nur das HTML-Dokument enthält. Bereiche, die dynamisch erzeugt werden sollen, enthalten Platzhalter mit Variablennamen. Wird nun der Absatz aus Beispiel 2.2 in Laravel umgesetzt, entspricht der Inhalt des Templates Beispiel 2.3. Das Dokument ist wieder leichter zu lesen und Änderungen von Markup oder Satzbau lassen sich einfacher durchführen.

Der PHP-Teil liegt nun im Controller, wo auch das Template aufgerufen wird. Die Variablen werden in einem assoziativen Array an das Template übergeben. Beispiel 2.4 zeigt den Code, der dem vorherigen Beispiel entsprechend den Namen des Wochentags und das aktuelle Datum ermittelt.

Nun kann man den View anpassen, ohne sich mit der tatsächlichen Berechnung der Inhalte beschäftigen zu müssen. Gleiches gilt für eine Ände-

rung der Berechnung, die nun von der Darstellung entkoppelt ist. Hier wird bereits die Arbeitsweise mit Frameworks ansatzweise deutlich. Durch das MVC-Prinzip bildet sich eine klare Struktur heraus, während jedoch weiterhin ausschließlich der Inhalt implementiert werden muss. Die Koordination wird vom Framework übernommen.

2.3.3 Sicherheit

Einen weiteren, speziell in der serverseitigen Entwicklung wichtigen Bereich bildet das Thema Sicherheit. Es muss damit gerechnet werden, dass der Client jeden erdenklichen Request schicken kann, im schlimmsten Fall mit schadhaftem Inhalt. Um das Backend möglichst sicher zu gestalten, sodass es für den Produktivbetrieb geeignet ist, muss ein nicht unerheblicher Aufwand betrieben werden.

Sobald Werte, die vom Client stammen, dynamisch verarbeitet werden bieten sich Möglichkeiten, den Server zu manipulieren. Das kann bereits durch URL-Parameter geschehen oder auch über den Inhalt eines POST-Requests oder einer hochgeladenen Datei. Bei unzureichend gesicherten Applikationen kann ein Angreifer damit auf Inhalte zugreifen, die nicht für ihn bestimmt sind, Daten per SQL-Injection verändern, Spam-Mails versenden oder das System mit schadhaften Inhalten sabotieren. „Keinesfalls darf ein Programm Werte aus einer GET, POST oder COOKIE-Quelle direkt verwenden“ [4], warnt deshalb Kristian Köhntopp auf der FAQ-Seite von PHP. Alle Daten, die der Client theoretisch manipulieren kann, müssen vollständig geprüft werden. Jede Stelle, an der vergessen wird zu validieren, stellt eine Sicherheitslücke dar. Deshalb müssen an jeder kritischen Stelle erneut alle möglichen Risiken bedacht und so vollständig wie möglich eliminiert werden.

Da PHP-Frameworks üblicherweise Prozesse wie das Auslesen von Requests und den Datenbankzugriff für den Entwickler abstrahieren, erübrigen sich einige grundlegende Sicherheitsfragen, da Überprüfungen und Formatierungen standardmäßig durchgeführt werden. Zusätzlich werden einfachere Möglichkeiten der Validierung von Benutzereingaben zur Verfügung gestellt, wie sich auch bei den hier diskutierten Systemen zeigen wird. Es kommt jedoch trotzdem auf die Qualität und Aktualität des verwendeten Frameworks an, weshalb auch dieses Thema bei der Wahl ein wichtiges Kriterium darstellt.

2.4 Zusammenfassung

Ein Framework ist im allgemeinen Sinn eine Rahmenstruktur für Applikationen. Es implementiert bereits häufig benötigte Funktionalität und gibt eine Ordnung zur besseren Übersichtlichkeit und Wartbarkeit vor. Im Webbereich sind Frameworks von besonderem Nutzen, da sie Standardaufgaben

im typischen Kreislauf zwischen Request und Response übernehmen. Die spezifischen Bereiche der Applikation werden üblicherweise nach dem MVC-Prinzip strukturiert, wodurch das Projekt besser skalierbar und die Zusammenarbeit erleichtert wird. Speziell für die serverseitige PHP-Entwicklung wurden diese Vorteile exemplarisch anhand des Templating demonstriert. Außerdem wurde das Thema Sicherheit in Grundzügen dargestellt, da es im Backend eine wichtige Rolle spielt und Frameworks dabei einen wichtigen Beitrag leisten können. Auf dieser Grundlage kann nun der Gegenstand dieser Arbeit, das PHP-Framework Laravel, beurteilt und verglichen werden.

Kapitel 3

Das PHP-Framework Laravel

Laravel ist ein PHP-Framework, dessen Entwicklung 2011 von Taylor Otwell initiiert wurde. Nachdem es zunächst als privates Projekt entstanden war, entschied sich Otwell ein Jahr darauf, es als Open-Source-Framework zu veröffentlichen. Schon bald erfuhr das Projekt große Beliebtheit und eine große Community entstand. Zusätzlich wird es mittlerweile von verschiedenen Unternehmen gefördert, unter anderem Cartalyst und UserScape.

Laravel basiert auf der beliebten MVC-Struktur, was den Umstieg von anderen bekannten Frameworks erleichtert. Besondere Ähnlichkeit besteht zu *Symfony*, da dieses lang bewährte Framework die Ausgangsbasis bildet. In diesem Kapitel wird nun untersucht, was Laravel speziell auszeichnet und welche grundlegenden Features möglicherweise für den Erfolg mitverantwortlich sind.

3.1 Der Erfolg von Laravel

Das Framework erfuhr schnell großen Zuspruch und wurde innerhalb weniger Jahre zum scheinbar beliebtesten PHP-Framework. So ergab eine Erhebung von *Beebom* im Februar 2015 [3], dass Laravel mit 22,7% gemessen an *GitHub-Stars* am beliebtesten sei, gefolgt von Symfony mit 15,1%.

Ähnliche Ergebnisse zeigt auch eine Sitepoint-Umfrage von März 2015 zu den beliebtesten PHP-Frameworks [8]. Hier liegen ebenfalls Laravel und Symfony an der Spitze, wobei die Beliebtheit von Laravel am Arbeitsplatz um rund 50%, im Privatgebrauch sogar rund 100% größer ist als die von Symfony. In den meisten Ländern belegt Laravel den ersten Platz und auch in allen Altersgruppen ab 18 Jahren liegt es vorne.

Diese Ergebnisse sind besonders bemerkenswert, wenn man bedenkt, dass Laravel erst seit 2012 auf dem Markt ist. Symfony hingegen ist seit weitaus längerer Zeit etabliert. Möglicherweise profitiert Laravel aber nicht zuletzt von dieser populären Basis, da der Umstieg von Symfony recht nahe liegt und das Projekt in der großen Symfony-Community diskutiert wird.

3.2 Versprechen und Besonderheiten

Auf der offiziellen Website von Laravel [6] fällt zuerst der Slogan „The PHP Framework For Web Artisans“ auf. Außerdem ist die Rede von „beautiful code“ und „elegant applications“. Diese Wortwahl zeigt deutlich die Intention von Laravel und den Anspruch, im Gegensatz zu rein technischen Ansätzen der Konkurrenz, die Erfahrung des Programmierens selbst zu verbessern. Hartmut Schlosser fasst das in einem Beitrag bei *entwickler.de* [7] so zusammen:

> Das Laravel Framework sieht sich als Versuch, alle schmerzhaften Punkte aus der Web-Entwicklung durch Abstraktionen zu eliminieren und diese in eine elegante Syntax zu gießen, die Spaß macht.

Die Selbstpräsentation auf der eigenen Website zeigt das hochprofessionelle Marketing, mit dem neue Anwender überzeugt werden sollen. Das beginnt bei ansprechenden Texten mit dem Versprechen des künstlerischen Anspruchs und wird von passendem Design und wenigen, prägnanten Beispielen unterstrichen. Abgesehen davon finden sich jedoch auch neben ansprechender Syntax einige klare Argumente, mit denen sich Laravel hervorheben will.

Besondere Erwähnung findet *Eloquent*, eine ORM-Komponente zur Abstraktion von Datenbankanbindungen. Sie wurde eigens für Laravel entwickelt und bildet somit ein wichtiges Alleinstellungsmerkmal im Vergleich zu anderen Frameworks. Die Zusammenarbeit im Team soll durch unkomplizierte Datenbankmigrationen erleichtert werden. Das Datenmodell wird wie bei vergleichbaren Systemen in Form von Model-Klassen angelegt, aus denen automatisch Datenbank-Tabellen erzeugt werden. Zugriffe werden durch Methoden abstrahiert und erleichtert. Besonderes Merkmal soll auch hier wieder die klare und einfache Syntax sein.

Auch das Routing soll besonders leicht und übersichtlich funktionieren. Außerdem gibt es eine *Queue-Library*, die besonders zeitintensive Prozesse auslagert und damit die Performanz der Applikation erhöhen soll. Für das häufig wiederkehrende Problem der Authentifizierung steht ebenso eine Lösung bereit, mit der es „out of the box“ möglich sei, schnell Zugriffsrechte zu definieren [6].

Außerhalb des reinen Entwicklungsprozesses gibt es weitere Produkte, die den gesamten Prozess vereinfachen sollen. Der Webhoster *Forge* wird vorgestellt, der speziell auf Laravel-Anwendungen ausgelegt ist. Mit *Homestead* wird eine komplett auf Laravel-Entwicklung zugeschnittene Virtual Machine angeboten.

Nicht zuletzt wird auf die ausführliche Dokumentation und die Lernvideoplattform *Laracasts* hingewiesen. Somit liegt der Fokus sehr deutlich

auf Vereinfachung, schnellem Einstieg und einem angenehmen Entwicklungsprozess. Die genannten Features werden im weiteren Verlauf bei genaueren Untersuchungen und Vergleichen besonders berücksichtigt.

3.3 Dokumentation und Laracasts

Ein wahrscheinlich wichtiger Grund für die schnelle Verbreitung ist die umfangreiche und klar strukturierte Dokumentation. Von schrittweisen Erklärungen für Anfänger bis zu Detailbeschreibungen wird hier der Funktionsumfang von Laravel weitgehend abgedeckt. Zusätzlich ist auch der Code an den meisten Stellen extrem ausführlich und verständlich kommentiert. Für das schnelle Wachstum der Community nach der Open-Source-Veröffentlichung dürfte das ein wesentlicher Grund gewesen sein.

Um sich als Anfänger mit Laravel vertraut zu machen oder fortgeschrittene Best-Practices zu erlernen, bietet sich aber nicht nur die Dokumentation an. *Laracasts* [12] ist eine Videoplattform, auf der Tutorials speziell zu Laravel aber auch anderen dafür relevanten Webtechnologien angeboten werden. Die von Jeffrey Way produzierten Lehrvideos erfreuen sich großer Beliebtheit und werden dem Nutzer schon auf der Homepage von Laravel als Informationsquelle empfohlen. Für andere Frameworks gibt es zwar ebenso Videotutorials auf allgemeinen Plattformen, eine spezialisierte Lernplattform ist aber ein exklusiver Vorteil von Laravel.

Solche Angebote erleichtern den Einstieg und senken so die Hürde, von gewohnten Frameworks umzusteigen. Die Erlernbarkeit kann durchaus bei der Wahl eines Frameworks mit einbezogen werden. In diesem Punkt besitzt Laravel mit seiner Dokumentation und Laracasts einen klaren Vorteil.

3.4 Symfony als Grundlage

Die Grundlage von Laravel ist das bereits seit 2005 existierende Framework Symfony. Hinter dem Open-Source-Produkt steht das Unternehmen *SensioLabs* und der Initiator Fabien Potencier. Nach eigenen Angaben auf der Symfony-Website [14] gibt es über 300.000 Symfony-Entwickler und das Framework wird pro Monat über 5 Millionen mal heruntergeladen. Es ist seit Jahren weit verbreitet und besetzt heute noch Spitzenpositionen in den Statistiken (siehe 3.1). Außerdem bildet es die Grundlage für das bekannte Content-Management-System *Drupal*.

Laravel wird als Projekt auch auf der Website von Symfony [5] präsentiert, was die enge Verwandtschaft der beiden Systeme zeigt. Dort wird es als Weiterentwicklung von Symfony mit Fokus auf einfacher, ausdrucksstarker Syntax und Erleichterung typischer Aufgaben beschrieben. Außerdem verwendet Laravel einige Kernkomponenten aus dem Symfony-Projekt. Aufgrund dieser Verwandtschaft und weil es sich um die beiden derzeit er-

folgreichsten PHP-Frameworks handelt, wird Symfony in dieser Arbeit zum Vergleich herangezogen.

Kapitel 4

Theoretischer Vergleich von Laravel und Symfony

Dieses Kapitel dient als Vorbereitung für den Praxisvergleich, in dem die Besonderheiten von Laravel im Vergleich zu Symfony ermittelt werden sollen. Dabei folgt zunächst ein grundlegender Vergleich von Handhabung und Struktur anhand einer erstmaligen Installation. Außerdem werden im Voraus die Komponenten und der Funktionsumfang laut Dokumentation verglichen.

4.1 Installation und erste Schritte

Bei beiden Frameworks ist ein neues, lauffähiges Projekt schnell installiert. Die Anleitungen dazu sind einfach und finden sich auf der jeweiligen Website. Während Laravel zur Standardinstallation auf *Composer* setzt, kann Symfony mit einem Befehl in der Konsole heruntergeladen werden. Allerdings ist Composer in beiden Fällen nützlich, um weitere Komponenten schnell und einfach zu installieren.

4.1.1 Symfony

Hier wird zunächst eine Installationsdatei namens Symfony mittels PHP-Befehl heruntergeladen. Danach können mit der Eingabe von `symfony new [name]` leere Symfony-Projekte generiert werden.

Im Projektordner steht dem Entwickler dann die Datei *Console* für einige nützliche Konsolenbefehle zur Verfügung. Sie liegt ab Version 3.0 im Ordner */bin*, bei früheren Versionen in */app*. Mit dem Befehl `php bin/console server:run` kann beispielsweise der Server gestartet werden. Ferner können unter anderem Elemente generiert, Datenbanken verwaltet und der Cache geleert werden.

4.1.2 Laravel

Ein neues Laravel-Projekt kann mit *Composer* auf die übliche Weise mit `composer create-project laravel/laravel [name]` erstellt werden. Es entsteht eine sofort lauffähige, leere Applikation.

Alternativ kann die speziell auf Laravel zugeschnittene Virtual Machine Homestead verwendet werden. Sie beinhaltet viele gängige Programme und Systeme, die bei der PHP-Entwicklung benötigt werden, und bietet zudem den Vorteil der absoluten Unabhängigkeit von Betriebssystemen und Umgebungen. Homestead verwendet entweder *VirtualBox* oder *VMware* in Kombination mit *Vagrant* und wird selbst als Vagrant-Box und Composer-Paket installiert.

Das Gegenstück zu *Console* bei Symfony bildet hier die Datei *Artisan*, die im Grunde ähnliche Funktionen anbietet. Es können ebenso Elemente generiert und verschiedene Werkzeuge aufgerufen werden. Als Unterschied fallen hier die Befehle für Datenbankmigrationen und die Queue-Library auf – zwei Spezialitäten von Laravel, wie bereits in 3.2 bei Betrachten der Laravel-Website auffiel. Der Befehl, um den Server zu starten, lautet hier `php artisan serve`.

4.2 Struktur

Beide Frameworks sind nach dem MVC-Prinzip strukturiert. Die Controller befinden sich in dafür vorgesehenen Ordnern, ebenso die Templates an anderer Stelle. Interne Logik von Framework-Komponenten befindet sich im Ordner *Vendor*. Der Einstiegspunkt für den Zugriff auf die Website liegt bei Symfony im Ordner *web*, bei Laravel in *public*. Ebenso steht in beiden Systemen ein eigener Ordner für Testdateien bereit, der jeweils schon ein Beispiel enthält. Beiden gemeinsam ist ein Hauptordner namens *App*, der allerdings verschiedene Zwecke erfüllt.

Ein Unterschied, der den Aufbau der Applikation betrifft, ist die Art der Konfiguration. Von Routing-Regeln und MVC-Beziehungen über Datenbank-Anbindungen bis hin zu spezifischen Einstellungen in allen Bereichen stehen dem Entwickler Konfigurationsdateien zur Verfügung. Symfony lässt dafür meistens die Wahl zwischen PHP, XML, YAML und Annotationen in den Klassen selbst. Laravel hingegen enthält in erster Linie PHP-Dateien, die mit mehrzeiligen Kommentaren zu jedem einstellbaren Wert sehr ausführlich dokumentiert sind.

Weitere Unterschiede ergeben sich durch die Strukturierung von Symfony-Applikationen in *Bundles*. Damit können in sich abgeschlossene Teile der Applikation als unabhängige Einheit wiederverwendet werden. Auch das Framework selbst besteht aus Bundles für die einzelnen Komponenten und ebenso einem Bundle für die Kernfunktionalität. Wenn mit der Konsole ein Bundle automatisch generiert wird, setzt Symfony einen eigenen Namespace

und erzeugt Einträge in den Konfigurationsdateien. Alle Einstellungen, auch die Art der Konfigurationen, können pro Bundle individuell gewählt werden. Nach der erstmaligen Installation existiert bereits ein *AppBundle* mit einem Beispiel-Controller. Bei Laravel sieht die Standardstruktur einen Ordner für alle Controller sowie andere Ordner für Routing und Models vor. Eine Aufteilung in unabhängige Komponenten ist in der Standardstruktur nicht vorgesehen.

4.3 Komponenten und Kompatibilität

Die Gemeinsamkeit der beiden Frameworks besteht vor allem darin, dass Laravel einige Kernkomponenten von Symfony verwendet. Wichtige Beispiele dafür sind *HttpKernel* und *HttpFoundation*, die zentrale Bestandteile beider Frameworks sind. Auch die interne Funktionsweise des Routings basiert in beiden Fällen auf der gleichen Komponente. Außerdem werden weitere Komponenten für verschiedene spezifische Aufgaben verwendet, beispielsweise *Translation* zur Erstellung von mehrsprachigen Websites. Alle gemeinsam genutzten Komponenten sind bei der Laravel-Projektvorstellung auf der Symfony-Website [5] aufgeführt.

Unterschiede liegen hauptsächlich in den Bereichen Templating und Datenbankabstraktion, wo für Laravel von Grund auf eigene Komponenten entwickelt wurden. Als Template-Engine setzt Laravel auf *Blade*. Es kann jedoch ebenso *Twig* verwendet werden, welches bei Symfony zum Einsatz kommt. Für Datenmodelle und Datenbankzugriffe ist bei Laravel *Eloquent* verantwortlich, Symfony verwendet *Doctrine*.

Das Symfony-Projekt ist in erster Linie eine Sammlung von Komponenten, die jeweils auch eigenständig eingesetzt werden können. Das Framework bündelt sie letztendlich in einem Gesamtpaket. Dieses Konzept hat den Vorteil, dass einzelne Komponenten theoretisch auch in einem Laravel-Projekt eingesetzt werden können.

Kapitel 5

Praxisvergleich anhand eines einfachen Testprojekts

Für einen Praxistest der beiden Frameworks wird nun eine Webapplikation möglichst identisch einmal mit Symfony und dann mit Laravel implementiert. Das Ziel des Projekts ist einerseits, die generellen Vorteile der Frameworks aufzuzeigen und herauszufinden, in welchen Bereichen sie einen besonderen Nutzen aufweisen. Andererseits soll durch den Vergleich beider Systeme untersucht werden, wo mögliche Gründe für die besondere Popularität Laravels liegen.

Berücksichtigt werden dabei in erster Linie Aufgaben, die bei fast jeder Webapplikation früher oder später gefragt sind. Installation, Konfiguration und Routing sind obligatorische Bereiche in jedem Projekt. Wenn es sich um Webseiten handelt und nicht etwa um eine reine Schnittstelle, ist auch Templating ein wichtiger Punkt. Außerdem fallen Formulare, Datenbankzugriffe und Benutzerauthentifizierung ins Gewicht, da in vielen Fällen Benutzereingaben verarbeitet und gespeichert werden müssen. Das sind die Kernaufgaben eines Frameworks und auch der Hauptgrund, ein solches einzusetzen.

Spezielle Tricks oder Erweiterungen werden nicht eingesetzt und es besteht kein Anspruch auf die effizienteste Implementierung. Stattdessen zählt die Lernkurve bei der ersten Einarbeitung und der nötige Aufwand, um ohne Vorkenntnisse in dem jeweiligen Framework eine saubere und voll funktionsfähige Implementierung zu erzielen.

5.1 Grundfunktionalität eines Forums als Testprojekt

Aufgrund dieser Überlegungen wurde als Testprojekt ein einfaches Frage-Antwort-Forum gewählt. Implementiert wird die Grundfunktionalität, so-

Abbildung 5.1: Hauptseite mit einer Liste aller Fragen. Da der Benutzer nicht eingeloggt ist, wird hier das Formular für neue Fragen nicht angezeigt.

dass angemeldete Benutzer Fragen stellen sowie auf Fragen anderer Benutzer antworten können. Damit wird das gesamte Spektrum der zu untersuchenden Themen abgedeckt, wodurch weitere Funktionen, wie etwa das Bearbeiten eigener Fragen, für diesen Zweck nicht notwendig sind. Sie ließen sich aber bei einer Weiterentwicklung auf gleiche Weise hinzufügen.

Um zunächst einen Überblick über die Applikation zu geben, folgt eine Auflistung der einzelnen Seiten mit einer kurzen Beschreibung der enthaltenen Funktionen. Als Hauptseite dient die

Übersicht (siehe Abb. 5.1), auf der eine Liste vorhandener Fragen, jeweils mit Betreff, Name des Erstellers und Datum, angezeigt wird. Außerdem ist für angemeldete Benutzer ein Formular sichtbar, mit dem sie eine neue Frage erstellen können. Diese wird dann in einer

Detailansicht (siehe Abb. 5.2) mit dem vollständigen Text angezeigt. Darunter befindet sich eine Liste aller Antworten und ebenso ein Formular zum Erstellen einer neuen Antwort. Außerdem gibt es eine

Login-Seite zum Anmelden und analog dazu eine

Registrierungsseite zum Erstellen neuer Benutzerprofile. Auf der Seite

Profil wird vorerst ausschließlich der eigene Benutzername ausgegeben. Sie ist nur für angemeldete Benutzer zu erreichen. Das

Impressum dient als Beispiel für eine statische Seite.

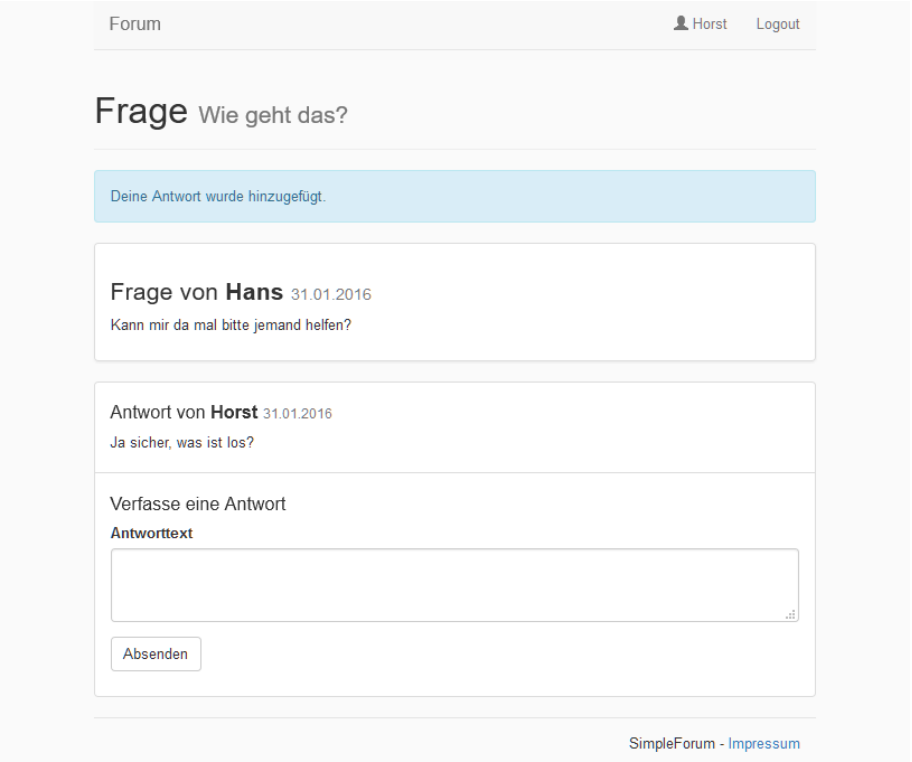


Abbildung 5.2: Detailansicht einer Frage mit Antwort und einem Formular.

5.2 Projektstart

Wie bereits in Abschnitt 4.1 beschrieben, ist die Installation beider Systeme sehr einfach. Für das Symfony-Projekt wird die Version 3.0 gewählt, bei Laravel handelt es sich um Version 5.2. Beide Systeme sind von Beginn an lauffähig und weisen bereits Konfigurationen auf, die in den meisten Fällen für die Entwicklung ausreichen sollten. Zusätzlich ist vorerst nur eine Anpassung der Datenbank-Verbindungsdaten nötig.

Hier zeigt sich bereits in beiden Fällen, dass der Einsatz der Frameworks keinen zusätzlichen Aufwand bei der Einrichtung eines neuen Projekts bedeutet. Dem Entwickler wird sogar die Arbeit abgenommen, eine sinnvolle Struktur zu entwerfen, da die vorgegebene Grundapplikation für Projekte fast jeder Größe einsetzbar ist. Es kann also sofort mit dem Erstellen von Routes und Controllern begonnen werden.

Eine Besonderheit bei Symfony ist an dieser Stelle die Aufteilung in *Bundles*, wie bereits in 4.2 beschrieben. Von Beginn an ist ein *AppBundle* enthalten, das für die eigenen Implementierungen gedacht ist und auch hier verwendet wird.

Als Grundlage für die Entwicklung dieses Projekts in Symfony wurden

Programm 5.1: Routing-Information in der Annotation des Controllers zur Anzeige einer Frage.

```
1 class QuestionController extends Controller
2 {
3   /**
4   * @Route("/question/{id}", name="question")
5   */
6   public function showAction(Request $request, $id)
7   {
8     // ...
```

die Anleitungen im *Symfony Book* [9] herangezogen. Bei Laravel dienen die offizielle Dokumentation und die Einführung in Laravel 5 bei *Laracasts* [13] als Informationsquellen. Um das Projekt möglichst repräsentativ umzusetzen, wird die Implementierung so nah wie möglich an diesen offiziellen Empfehlungen gehalten.

5.3 Routing

Obligatorisch in jeder Webapplikation ist das Definieren von Routes, wodurch festgelegt wird, wie auf einen eingehenden Request reagiert werden soll. Dabei haben Symfony und Laravel unterschiedliche Herangehensweisen.

Grundsätzlich besteht eine Route-Definition aus einem String, der dem Pfad in der URL entspricht, und der Information, welche Funktion ausgeführt werden soll, wenn ein Request mit diesem Pfad eintrifft. Optional, aber oft nützlich, ist die Übertragung von Variablen als Route-Parameter, was in beiden Frameworks auf gleiche Weise funktioniert. In diesem Fall führen URLs mit `/question/{id}`, wobei *{id}* der ID einer Frage in der Datenbank entspricht, zur Detailansicht genau dieser Frage. Die Syntax mit geschwungenen Klammern wird in beiden Frameworks genau so verwendet.

Die Controller-Funktion bekommt ein Request-Objekt übergeben, das mit zahlreichen Methoden ausgestattet ist, um Informationen über die Anfrage zu beziehen. Zusätzlich werden die Route-Parameter übergeben, um dynamisch darauf reagieren zu können.

Routes können mit eindeutigen Namen versehen werden, um in den verschiedenen Templates und Controllern einfacher darauf zu referenzieren. Das hat den Vorteil, dass bei einer Änderung der URL nicht alle Referenzen geändert werden müssen.

5.3.1 Routing per Annotation in Symfony

In Symfony gibt es verschiedene Möglichkeiten, Routes zu definieren. Allerdings muss die gewählte Art für das gesamte Bundle konsistent verwendet werden. Empfohlen wird das Routing per Annotation, das deshalb in diesem Beispiel verwendet wird. Dabei werden vor jeder Controller-Funktion in der Annotation die entsprechenden Informationen angegeben (siehe Programm 5.1). Eine sinnvolle Alternative bietet die Definition in einer YAML-Datei. Ein Vorteil dieser Lösung wäre die Übersichtlichkeit, da alle Routes an einer Stelle aufgelistet werden. Dafür wäre wiederum der Controller in einer anderen Datei. Eine Liste aller Routes der gesamten Applikation lässt sich zumindest in jedem Fall mit `php bin/console debug:router` einsehen, falls es einmal notwendig wird.

5.3.2 Zentrales PHP-Routing in Laravel

In Laravel werden alle Routing-Informationen zentral in der Datei *routes.php* definiert. Dabei werden Methoden je nach Art des Requests (GET, POST etc.) aufgerufen und der Pfad sowie der Name des entsprechenden Controllers in einer @-Notation als Argumente übergeben. Alternativ kann als zweites Argument eine anonyme Funktion definiert werden, was sich für sehr einfache Controller anbietet.

Das Routing mit PHP in Laravel ist kompakt und übersichtlich, bietet aber gleichzeitig zusätzliche Flexibilität. Bevor der Controller ausgeführt wird, kann sogenannte *Middleware* eingesetzt werden, also Funktionen, die den Request ihrem Einsatzzweck entsprechend bearbeiten und dann an den Controller weitergeben. Das ist bei sich wiederholenden Aufgaben sinnvoll, wie beispielsweise der Authentifizierung. Außerdem können Gruppen erstellt werden, die auf alle enthaltenen Routes die gleiche Middleware anwenden oder die Pfade mit einem Präfix versehen (siehe Programm 5.2).

Laravel ist im Routing somit auf den ersten Blick flexibler und bietet alle gewohnten Möglichkeiten der Programmierung, da mit PHP gearbeitet wird. Ob die zentrale Definition der Annotationsschreibweise von Symfony vorzuziehen ist, hängt jedoch von persönlichen Vorlieben ab. Ein Vorteil ist aber, dass anhand der sprechenden Syntax die Funktionsweise gut durchschaubar ist und schnell ein komplexes, aber systematisches Routing realisiert werden kann.

5.4 Erzeugen der HTML-Seiten

Für das dynamische Erzeugen von HTML stellen beide Frameworks eine eigene Template-Engine bereit. Die Funktionsweisen von *Twig* (Symfony) und *Blade* (Laravel) sind fast identisch, auch wenn sich die Syntax unterscheidet.

Ein Vorteil von Templates, der schnell deutlich wird, ist eine Art Ver-

Programm 5.2: Eine Route-Gruppe in Laravel. Route::auth() importiert an dieser Stelle alle Routes für die Authentifizierung, also Login, Registrieren etc.

```
1 Route::group(['middleware' => ['web']], function () {
2
3   Route::get('/question/{question}', 'QuestionController@showQuestion')
4   ->name('question');
5
6   Route::post('/question/{question}', 'QuestionController@newAnswer');
7
8   // ..
9 });
```

erbung, wie sie aus der Objektorientierten Programmierung bekannt ist. In diesem Projekt wird beim Erstellen des HTML schnell deutlich, dass etwa zwei Drittel des Dokuments auf allen Seiten beinahe identisch sind. Diese Teile werden zunächst in ein Layout-Template gegeben, das daraufhin den gesamten *Head* und fixe Bereiche im *Body* wie zum Beispiel Navigation und Fußzeile enthält (siehe Programm 5.3). Dann werden Bereiche festgelegt, die von abgeleiteten Templates überschrieben werden können. Die weiteren Templates werden nun von diesem Layout abgeleitet und überschreiben nur noch die Bereiche, die tatsächlich abweichen. Diese Funktionalität ist in beiden Engines gleichermaßen gegeben.

Aufgerufen werden die Templates im Controller in beiden Systemen auf die gleiche Weise mit einer Funktion, die als Argumente den Namen des Templates und ein assoziatives Array der anzuzeigenden Variablen erhält. Bei Symfony lautet die Funktion `$this->render('template.html.twig', ['Schlüssel' => 'Wert'])`, bei Laravel `view('template', ['Schlüssel' => 'Wert'])`.

5.4.1 Symfony und Twig

Die Standard-Template-Engine in Symfony ist Twig, es können jedoch auch andere verwendet werden. Die Templates sind einfache HTML-Dokumente mit der Dateiendung *.html.twig*, die mit Platzhaltern für Variablen und Anweisungen wie *if* oder *foreach* in geschwungenen Klammern versehen werden. Einige davon sind in Programm 5.4 ersichtlich.

Etwas ungewohnt ist dabei die von PHP abweichende Schreibweise, um auf Variablen zuzugreifen. Ansonsten ist die Syntax jedoch gut an jeder Stelle innerhalb des HTML einsetzbar und in sich konsistent. Eine Besonderheit sind außerdem *Pipes*, mit denen Variablen umformatiert werden können.

Programm 5.3: Ausschnitt aus dem Layout-Template in Laravel (Blade). Mit @yield('content') wird der Inhalt aus dem abgeleiteten Template eingebunden.

```
1 <!-- davor Head und Navigation -->
2 </nav>
3 <div class="page-header">
4   <h1>@section('headline'){{ isset($title) ? $title : 'Forum' }}@show </
      h1>
5 </div>
6 @if(Session::has('notice'))
7   <div class="alert alert-info" role="alert">
8     {{ Session::get('notice') }}
9   </div>
10 @endif
11
12 @yield('content')
13
14 <footer class="modal-footer">
15 <!-- Fußzeile -->
```

5.4.2 Laravel und Blade

Laravel setzt auch in seinen Views auf PHP. Templates können entweder als reine PHP-Datei, die Variablen mit `<?php echo $foo; ?>` einbindet, oder als Blade-Templates erstellt werden. Laravel erkennt beides automatisch.

Die Ausgabe von Variablen erfolgt wie bei Twig mit geschwungenen Klammern. Anweisungen verwenden jedoch eine andere Syntax, zum Beispiel `@if` oder `@yield`. Diese Schreibweise ist nicht ganz so problemlos zu verwenden wie die von Twig, da unmittelbar kein Zeichen folgen darf. Nach einem `@endif` muss beispielsweise ein Leerzeichen oder ein Zeilenumbruch folgen. Ansonsten ist die Funktionalität jedoch identisch (siehe Programm 5.5).

Als Vorteil zeigt sich auch hier wieder der konsequente Einsatz von PHP-Syntax. Somit kann auf Variablen genauso zugegriffen werden wie etwa im Controller. Theoretisch kann jeglicher PHP-Code verwendet werden, um zum Beispiel Arrays zu sortieren oder Datumsausgaben zu formatieren. Wichtig ist dabei jedoch, dass der Entwickler selbst aufpasst, keine Controller-Logik sondern nur anzeigespezifischen Code in den View einzubauen.

5.5 Datenbankzugriff

Das sogenannte *Object-Relational-Mapping (ORM)* ermöglicht es, ein Datenmodell in Form von PHP-Klassen zu definieren, um unabhängig von be-

Programm 5.4: Das Twig-Template für die Übersichtsseite mit der Liste aller Fragen. Zunächst wird vom Layout abgeleitet und dann der Bereich mit dem Inhalt überschrieben. Darin wird in einer For-Schleife über alle Fragen iteriert und einzelne Variablen jeder Frage ausgegeben. Um die Ausgabe *1 Antworten* zu vermeiden, wird eine If-Bedingung eingesetzt. Für bessere Übersichtlichkeit wurden weitere Inhalte sowie Bootstrap-Klassen entfernt.

```
1 {% extends 'layout.html.twig' %}
2 {% block content %}
3   {% for q in questions %}
4     <a href="{{ path('question', {'id': q.id}) }}">
5       <h3>{{ q.subject }}</h3>
6       <p>
7         <strong>{{ q.user.username }}</strong>
8         am {{ q.timestamp|date('d.m.Y') }} -
9         {{ q.answers.count }}
10        Antwort{% if q.answers.count != 1 %}en{% endif %}
11      </p>
12    </a>
13  {% endfor %}
14  <!-- Formular aus Platzgründen entfernt -->
15 {% endblock %}
```

stimmten Datenbanksystemen zu sein. Bei Änderungen an den Daten werden Methoden der PHP-Objekte aufgerufen, während die ORM-Komponente im Hintergrund die entsprechenden Datenbankzugriffe ausführt. Der Umstieg auf ein anderes Datenbanksystem bedeutet somit im Optimalfall nur eine kleine Änderung in einer Konfigurationsdatei.

In diesem Projekt sieht das Datenmodell die folgenden Tabellen in der Datenbank bzw. Klassen im ORM-Model vor.

User repräsentiert den eingeloggten Nutzer,

Question eine Frage mit Inhalt und Beziehung zum verfassenden Nutzer und

Answer eine Antwort auf eine bestimmte Frage, ebenfalls mit Beziehung zu ihrem Verfasser.

Beide Frameworks abstrahieren die Datenbankzugriffe mit ORM (siehe 4.3). Allerdings beruhen beide Systeme auf unterschiedlichen Prinzipien, wie die Implementierung des Forums zeigt. Demonstriert wird dies nun exemplarisch mit dem Hinzufügen einer Antwort.

5.5.1 Symfony und Doctrine

In Symfony wird das Model zentral in einer *Entity-Klasse* definiert. Diese enthält alle Felder als Variablen sowie die entsprechenden Getter- und Setter-Methoden. Eigenschaften wie Datentypen oder Beziehungen werden

Programm 5.5: Das Blade-Template für die Übersichtsseite wie in Programm 5.4. Unterschiede sind vor allem die Schreibweise mit @ und die PHP-Syntax. Für bessere Übersichtlichkeit wurden auch hier weitere Inhalte sowie Bootstrap-Klassen entfernt.

```
@extends('layouts.app')
@section('content')
  @foreach($questions as $question)
    <a href="{{ url('question', $question->id) }}">
      <h3>{{ $question->subject }}</h3>
      <p>
        <strong>{{ $question->user->name }}</strong>
        am {{ $question->created_at->format('d.m.Y') }} -
        {{ $question->answers->count() }}
        Antwort{{ ($question->answers->count() != 1) ? 'en' : '' }}
      </p>
    </a>
  @endforeach
  <!-- Formular aus Platzgründen entfernt -->
@endsection
```

Programm 5.6: Auszug aus der Entity-Klasse *Answer*. Die Annotationen mit @ORM definieren Eigenschaften für die Datenbank. $user referenziert den Verfasser als User-Objekt, welches wiederum die Referenz zu allen verfassten Antworten enthält. Die Art der Beziehung wird in diesem Fall mit `ManyToOne` definiert.

```
  /**
  * @var string
  *
  * @ORM\Column(name="text", type="text")
  * @Assert\NotBlank()
  */
  private $text;
  /**
  * @ORM\ManyToOne(targetEntity="User", inversedBy="answers")
  * @ORM\JoinColumn(name="user_id", referencedColumnName="id")
  */
  protected $user;
```

wieder per Annotation festgelegt (siehe Programm 5.6). Um die Schreibarbeit beim Erstellen dieser Klassen zu vermeiden, kann neben anderen nützlichen Befehlen mit `php bin/console generate:entity` ein interaktiver Generator verwendet werden. Mit einfachen Befehlen wird außerdem automatisch die entsprechende Datenbankstruktur erzeugt.

Im Controller können dann Instanzen dieser Klasse erzeugt oder aus der Datenbank geladen werden. Für die Interaktion mit der Datenbank wird ein

Programm 5.7: Dieser Auszug aus dem QuestionController in Symfony zeigt ausschließlich die Zeilen, die in Verbindung mit ORM stehen. Zunächst wird eine neue Antwort erstellt und die aktuelle Frage aus der Datenbank geladen. Dann wird das Antwort-Objekt bearbeitet und gespeichert. Zuletzt werden alle zur geladenen Frage gehörenden Antworten geladen, um sie anzuzeigen.

```
1 $answer = new Answer(); // neues Antwort-Objekt
2 $em = $this->getDoctrine()->getManager(); // Manager erzeugen
3 $question = $em->getRepository('AppBundle:Question')
4   ->findOneBy(['id' => $id]); // Frage mit übergebener ID laden
5 $answer
6   ->setTimestamp(new \DateTime())
7   ->setUser($this->getUser()) // Beziehung zum Verfasser erstellen
8   ->setQuestion($question); // Beziehung zur Frage erstellen
9 $em->persist($answer);  // zur Datenbank hinzufügen
10 $em->flush(); // Änderungen anwenden
11 $answers = $question->getAnswers(); // alle Antworten für Ausgabe
```

Manager-Objekt erzeugt, das Objekte laden und speichern kann. Die Entity-Instanzen werden zunächst unabhängig von der Datenbank bearbeitet. Mit der Methode `flush()` des Managers werden letztendlich die Änderungen mit entsprechenden Statements in die Datenbank übertragen (siehe Programm 5.7).

5.5.2 Laravel und Eloquent

Bei Laravel wird das Model etwas anders definiert. Dabei wird generell zwischen Datenbankmodell und ORM-Modell unterschieden. Für ein automatisches Erzeugen der Datenbankstruktur und zum einfacheren Austausch von späteren Änderungen können *Migrations* verwendet werden (siehe Programm 5.8). Sie enthalten jeweils eine `up()`-Methode, um Änderungen anzuwenden, und eine `down()`-Methode, um diese rückgängig zu machen.

Im ORM-Model werden lediglich Besonderheiten für die Verwendung der Objekte definiert. Im Fall des Models für die Antworten wird festgelegt, welche Felder mit Benutzereingaben initialisiert werden dürfen, und Methoden für die Beziehungen zu Frage und Verfasser definiert (siehe Programm 5.9). Diese sind jedoch nur zur einfacheren Handhabung der Instanzen im Controller nötig. Die üblicherweise fast immer benötigten Felder `id`, `created_at` und `updated_at` werden von Laravel standardmäßig vorgesehen und automatisch verwaltet, was einen großen Teil der Implementierungen vorwegnimmt.

Im Gegensatz zu Symfony ist die Syntax bei der Verwendung im Controller sowohl kompakter als auch sprechender. Den gesamten Vorgang zum Hinzufügen einer Antwort zeigt Programm 5.10.

Programm 5.8: Inhalt der `up()`-Methode in einer Migration, um die Tabelle für Antworten einzurichten. Dabei werden zunächst die Felder definiert und zusätzlich Fremdschlüssel erstellt.

```
1 Schema::create('answers', function (Blueprint $table) {
2   $table->increments('id');
3   $table->integer('user_id')->unsigned();
4   $table->integer('question_id')->unsigned();
5   $table->text('text');
6   $table->timestamps();
7   $table->foreign('question_id')
8     ->references('id')->on('questions');
9   $table->foreign('user_id')
10    ->references('id')->on('users');
11 });
```

Programm 5.9: Die Model-Klasse für Antworten in Laravel. Mit `question()` kann nun das Frage-Objekt bezogen werden, zu dem die Antwort-Instanz gehört. Analog dazu funktioniert `user()`.

```
1 class Answer extends Model
2 {
3   public function question() {
4     return $this->belongsTo('App\Question');
5   }
6   public function user() {
7     return $this->belongsTo('App\User');
8   }
9 }
```

Laravel bzw. Eloquent ist zwar anfangs nicht so intuitiv zu verstehen wie Symfony bzw. Doctrine, da einige Vorgänge automatisch passieren, von denen man wissen muss. Jedoch ist die nötige Menge an Code dadurch weitaus geringer und die Syntax gut lesbar. Durch die Trennung in Migrationen und ORM-Modell wird die Implementierung zwar etwas anspruchsvoller, aber gleichzeitig flexibler und reduziert sich auf die absolut nötigen Bereiche.

5.6 Formulare und Validierung

Wie fast alle Webanwendungen beruht das Forum grundlegend auf Benutzereingaben. Diese werden durch HTML-Formulare ermöglicht, die zwar immer auf ähnliche Weise, aber mit einem gewissen Aufwand erstellt werden müssen. Es ist also sinnvoll, wenn ein möglichst großer Teil dieser Arbeit durch das Framework übernommen wird.

Ein weiterer wichtiger Punkt ist die Validierung der am Server eintreffen-

Programm 5.10: Auszug aus der Controller-Methode zum Hinzufügen einer Antwort in Laravel. Zuerst wird eine neue Antwort mit den Benutzereingaben initialisiert. Dann wird die Beziehung zur Frage hergestellt und schließlich das Objekt als eine Antwort des Verfassers (aktueller Nutzer aus dem Request) gespeichert. Die Variable $question ist ein Argument der Controller-Methode und wird durch Erkennung des Datentyps *Question* automatisch anhand der ID im Routing-Pfad aus der Datenbank geladen.

```
1 $answer = new Answer($request->all());
2 $answer->question()->associate($question);
3 $request->user()->answers()->save($answer);
```

den Daten, bevor sie verarbeitet und gespeichert werden (siehe 2.3.3). Hier hilft in beiden Fällen ein Abgleich mit Regeln, die für das ORM-Modell definiert werden.

Als Beispiel dient nun das Erstellen einer neuen Frage auf der Übersichtsseite. Das Formular enthält die Felder für Betreff und Text, was den durch den Nutzer manipulierbaren Variablen der Model-Klasse entspricht. Bei Eintreffen eines POST-Requests, also wenn das Formular abgeschickt wurde, muss sichergestellt werden, dass nur ein String für den Betreff und einer für den Text aus den Eingaben verarbeitet werden.

5.6.1 Formularverarbeitung in Symfony

In Symfony gibt es einen Formular-Generator, mit dem Formulare anhand eines Models im Controller erstellt werden können. Das geschieht hier durch den Aufruf von `$this->createForm(QuestionType::class, $question)`, dem mit $question ein Model-Objekt übergeben wird. Anhand dessen werden Formularfelder vorausgefüllt und genau diese Instanz wird bei erfolgreichem Absenden in der Datenbank gespeichert. Die genaue Beschaffenheit des Formulars wird in einer entsprechenden *FormType-Klasse* definiert, die als erster Parameter übergeben wird (siehe Programm 5.11). Die Formular-Variable wird letztendlich an den View übergeben, wo anhand verschiedener Methoden der HTML-Code erzeugt werden kann, am einfachsten mit `{{ form(form) }}`. Da hier im Template zuvor ein Formular-Layout im Bootstrap-Stil aktiviert wird, entsteht dabei die für Bootstrap nötige HTML-Struktur.

Zur Verarbeitung abgesendeter Formulare muss keine eigene Route für POST-Requests definiert, sondern lediglich `$form->isSubmitted()` im Controller überprüft werden. Im Zusammenspiel mit einem erfolgreichen Aufruf von `$form->isValid()` kann dann das zuvor übergebene Question-Objekt verarbeitet werden.

Die Kriterien zur Validierung werden im Model per Annotation angege-

Programm 5.11: Definieren der Formularfelder in der Klasse *QuestionType*. Das erste Argument der `add()`-Methode ist der Name des Felds. Standardmäßig wird der Feldtyp und das Label automatisch festgelegt. Abweichende Werte können in weiteren Argumenten übergeben werden.

```
1 $builder
2   ->add('subject', null, array('label' => 'Betreff'))
3   ->add('text');
4 $builder
5   ->add('send', SubmitType::class, array('label' => 'Absenden'));
```

Programm 5.12: Annotationen für den Betreff einer Frage im Symfony-Model.

```
1 * @Assert\Length(
2 *      min = 10,
3 *      max = 255,
4 *      minMessage = "Betreff muss min. {{ limit }} Zeichen lang sein.",
5 *      maxMessage = "Betreff darf max. {{ limit }} Zeichen lang sein."
6 * )
```

ben. Dabei stellt *Assert* eine Vielzahl von Funktionen zur Verfügung, wie in Programm 5.12 ersichtlich.

5.6.2 Validierung in Laravel

Während Symfony ein Werkzeug für den gesamten Umgang mit Formularen bietet, gibt es in Laravel unabhängige Lösungen für die einzelnen Bereiche. Formulare werden hier nicht im Controller erzeugt. Dafür können nach Aktivierung einer Formularkomponente des *Illuminate*-Pakets Formularfelder im Template mit Hilfsfunktionen erzeugt werden.

Um abgesendete Formulare zu behandeln, wird eine neue Route mitsamt Controller für einen POST-Request erstellt (siehe Programm 5.13). Die Methode `$this->validate()` bewirkt einen Redirect, wenn die Validierung fehlschlägt. Die Kriterien dafür können wie in diesem Fall direkt als Array übergeben werden. Bei komplexeren Definitionen empfiehlt sich aber das Auslagern in eine *FormRequest*-Klasse. Diese wird dann als Datentyp des übergebenen Request-Objekts angegeben und erwirkt damit eine Validierung, bevor der Controller ausgeführt wird.

Laravel bietet keine so konsequente Verbindung von Model, Formularen und Validierung. Stattdessen werden unabhängige Lösungen für die einzelnen Probleme angeboten. Das bedeutet etwas geringere Übersichtlichkeit, aber auch größere Flexibilität.

Programm 5.13: Gesamte Controller-Methode zur Behandlung eines abgesendeten Frageformulars in Laravel. Bei erfolgreichem Speichern wird direkt zur Frage weitergeleitet.

```
1 public function newQuestion(Request $request) {
2     $this->validate($request, [ // oder auslagern in FormRequest
3       'subject' => 'required|min:10',
4       'text' => 'required'
5     ]);
6     $question = new Question($request->all());
7     $request->user()->questions()->save($question);
8     return redirect()->route('question', ['question' => $question->id])
9       ->with(['notice' => 'Deine Frage wurde erstellt.']);
10 }
```

5.7 Authentifizierung

Die Fragen und Antworten sind zwar für jeden sichtbar, aber um neue Fragen stellen zu können oder auf andere zu antworten, sollen sich die Benutzer zuvor anmelden. Dazu sind Formulare für Registrierung und Login, die entsprechenden Controller und eine Klasse *User* im Model nötig.

Da das Thema Authentifizierung bei vielen Anwendungen in Erscheinung tritt, gibt es für beide Frameworks Komplettlösungen, die diese Funktionalität auf typische Weise implementieren. Sie müssen lediglich konfiguriert und an manchen Stellen angepasst werden.

5.7.1 Das FOSUserBundle in Symfony

Aufgrund der Bundle-Struktur von Symfony kann theoretisch jedes Authentifizierungssystem, das unabhängig in einem eigenen Bundle implementiert wurde, sehr einfach wiederverwendet werden. Von offizieller Seite wird das *FOSUserBundle* empfohlen, das auch in diese Applikation eingebunden wird. Dadurch wird gleichzeitig das Verwenden eines Bundles von einem Drittanbieter demonstriert.

Zunächst wird das Bundle mit Composer geladen und durch einen Eintrag in *AppKernel.php* aktiviert. Es enthält bereits einige Model-Klassen, wobei die für den Benutzer in einer eigenen abgeleiteten Klasse *User* ergänzt werden kann. In diesem Fall werden die Beziehungen zu verfassten Fragen und Antworten hinzugefügt. Zusätzlich müssen Konfigurationen wie Benutzergruppen und Zugriffsrechte in der Datei *security.yml* festgelegt und der Eintrag für das neue Bundle in *config.yml* hinzugefügt werden. Außerdem müssen die Routing-Informationen des Bundles in *routing.yml* importiert werden. Nachdem die Datenbank per Konsolen-Befehl aktualisiert wurde, steht die volle Funktionalität zur Verfügung.

Programm 5.14: Ausschnitt aus dem Twig-Template für die Übersichtsseite. Für nicht angemeldete Nutzer werden anstelle des Formulars Links zu Login und Registrierung mit den importierten Routes des Bundles angezeigt.

```
1 {% if app.user %}
2   {{ form(form) }}
3 {% else %}
4   <p><a href="{{ path('fos_user_security_login') }}">Melde dich an</a>
      oder <a href="{{ path('fos_user_registration_register') }}">
      registriere dich</a>, um selbst eine Frage stellen zu können.</p>
5 {% endif %}
```

Programm 5.15: Ausschnitt aus dem Controller zum Erstellen einer neuen Frage in Symfony. Wenn der Nutzer angemeldet ist, wird unter anderem das User-Objekt in der neuen Frage referenziert.

```
1 // ...
2 if ($form->isValid() && $this->getUser()) {
3   $question
4     ->setTimestamp(new \DateTime())
5     ->setUser($this->getUser());
6   // ...
```

Damit sich die Seiten für Registrierung und Login gut in das Layout einfügen, werden die entsprechenden Templates überschrieben. Das geschieht durch das Erstellen einer gleichnamigen Datei mit gleichem Pfad unter `app/Resources/FOSUserBundle/`.

Die Formulare für neue Fragen und Antworten sollen nur sichtbar sein, wenn der Benutzer angemeldet ist. Dazu wird im Template der Status der globalen Variable *app.user* in einer If-Bedingung abgefragt (siehe Programm 5.14). Im Controller kann ebenso einfach auf das *User*-Objekt zugegriffen werden (siehe Programm 5.15). Um den Zugriff auf die Profil-Seite generell nur angemeldeten Nutzern zu gewähren, wird in *security.yml* der Eintrag - `{ path: ^/profile, role: ROLE_USER }` hinzugefügt.

5.7.2 Authentifizierung generieren in Laravel

Da Authentifizierung so häufig benötigt wird, gibt es in Laravel einen eigenen Generator dafür. Mit dem Befehl `php artisan make:auth` werden automatisch die nötigen Routes, Controller und Templates erstellt. Das Model und die Datenbankmigration für den Benutzer existieren bereits. Nachdem die Datenbank per Migration aktualisiert wurde, ist das System bereits einsatzbereit.

In der Model-Klasse *User* werden nun die Methoden für die Beziehungen

zu eigenen Fragen und Antworten hinzugefügt. Außerdem werden die Views angepasst, indem direkt die generierten Templates bearbeitet werden.

Um im Template zu überprüfen, ob der Nutzer angemeldet ist, wird wie bei Symfony eine einzige Variable abgefragt, in diesem Fall `Auth::user()`. Im Controller kann mit `$request->user()` auf das *User*-Objekt zugegriffen werden (Programm 5.13).

Für ein so häufiges Problem wie Authentifizierung ist es sinnvoll, dass Laravel die Erstellung der nötigen Dateien automatisiert. Das spart Arbeit und ist gleichzeitig noch einfacher nachzuvollziehen als die Konfiguration eines fremden Bundles in Symfony.

Kapitel 6

Fazit

Grundsätzlich bieten Symfony und Laravel einen ähnlichen Umfang an Funktionalität und verfolgen bei einigen Lösungen ähnliche Strategien. Sie sind zudem beide für ein breites Spektrum von Einsatzszenarien ausgelegt. Auch das praktische Beispiel eines Forums hat gezeigt, wie umfassend die Frameworks den Entwicklungsprozess unterstützen.

Laravel ist das modernere Framework, das zudem auf Symfony aufbaut. Es macht sich bewährte Symfony-Komponenten zunutze und ersetzt andere Teile durch zeitgemäßere Lösungen. So zielt beispielsweise der gesamte Arbeitsprozess auf maximale Effizienz bei der Entwicklung von Applikationen ab, wie sie in der heutigen Zeit am weitesten verbreitet sind.

Anhand der Erkenntnisse aus dieser Arbeit soll nun abschließend rekapituliert werden, was Laravel als PHP-Framework auszeichnet. Dazu werden die Auffälligkeiten zusammengefasst, die sich im praktischen Testprojekt gezeigt haben. Außerdem werden mögliche Einsatzgebiete festgestellt, die sich aus den gewonnenen Erkenntnissen ergeben. Zuletzt wird resümiert, welche Faktoren möglicherweise in Summe dazu geführt haben, dass Laravel so große Popularität genießt.

6.1 Unterschiede bei der Implementierung des Testprojekts

Der Hauptunterschied zu Symfony ist, dass Laravel durchgehend wie aus einem Guss erscheint. Das hat verschiedene Ursachen. Wie schon im theoretischen Teil der Arbeit festgestellt wurde, besteht Symfony aus einer Kombination unabhängiger Komponenten, die bereits über viele Jahre weiterentwickelt werden. Laravel hingegen wurde als Ganzes entworfen. In der Praxis zeigt sich das in einer geradlinigen Arbeitsweise, die dazu führt, dass insgesamt weniger Zeilen selbst geschrieben werden müssen. Außerdem sorgt die konsequente Verwendung von PHP in allen Bereichen, von Routes über

Controller bis hin zur Template-Syntax, für bessere Übersicht und größere Flexibilität.

Das Hauptversprechen von Laravel ist eine besonders natürliche und nachvollziehbare Syntax, wie bereits in 3.2 beschrieben. Das hat sich im Praxistest generell bewahrheitet, besonders im Umgang mit Eloquent-ORM. In fast allen Bereichen kann extrem kompakt programmiert werden, wobei sich die Zeilen trotzdem fast wie Prosatext lesen. Ein Beispiel dafür wäre die Zeile `redirect()->route('question', [...])->with([...]);`, die einen Redirect zur Route namens *question* mit einer bestimmten Flash-Nachricht bewirkt.

Anfangs kann die Syntax etwas verwirren, da man zuerst über die Funktionsweise im Hintergrund Bescheid wissen muss. Das liegt besonders daran, dass die Kompaktheit durch Automatisierung bestimmter Prozesse erreicht wird. Ein Beispiel dafür ist die Übergabe eines selbst definierten *FormRequest*-Objekts an eine Controller-Methode, was eine automatische Validierung noch vor dem Ausführen des Controllers bewirkt. Sobald man sich jedoch eingearbeitet hat, lässt sich dafür umso effizienter entwickeln.

Eine Spezialität von Symfony, die in Laravel so nicht vorhanden ist, bildet das nahtlose Zusammenspiel verschiedener Bereiche im Model. So wird in der Model-Klasse von Symfony zentral definiert, welche Variablen das Model besitzt, wie diese in der Datenbank gespeichert werden, welche Beziehungen zu anderen Models bestehen und welche Kriterien bei der Validierung anzuwenden sind. Außerdem können Formulare anhand des Models automatisch erstellt werden. Wer diese Art der logischen Bündelung des Models schätzt, könnte mit Laravel unzufrieden sein.

6.2 Sinnvolle Anwendungsbereiche von Laravel

Grundsätzlich ist Laravel für verschiedenste Anwendungen ausgelegt, von kleinen Projekten über komplexe Shop-Systeme und Netzwerke bis hin zu reinen REST-Schnittstellen. Um das Einsatzgebiet im Vergleich zu Symfony zu differenzieren, fällt besonders die Struktur der Frameworks ins Gewicht. Laravel ist ein einheitliches System mit auf Effizienz optimierten Prozessen, während sich Symfony durch die Flexibilität der Konfiguration und die Wiederverwendung einzelner Bundles auszeichnet.

Laravel eignet sich demnach besonders, wenn ein einzelnes Projekt, egal welcher Größenordnung, schnell und unkompliziert umgesetzt werden soll. Auch für Einsteiger ist es gegenüber Symfony vorzuziehen, da sich der Arbeitsprozess nah an den tatsächlichen Abläufen orientiert und die Syntax sprechende Bezeichnungen verwendet. Außerdem sind alle Bereiche extrem detailliert dokumentiert, sodass Konfigurationsdateien größtenteils selbsterklärend sind.

Die Schwächen liegen in der flexiblen Zusammenstellung des Systems

und der Wiederverwendung von Implementierungen über mehrere Projekte hinweg. Gerade wenn in großen Gruppen gearbeitet wird, oder aufgrund von Spezialisierung eigene Produkte mehrfach zum Einsatz kommen, bietet sich die Struktur von Symfony an. Für erfahrene Entwickler, die sich ein optimales System aus eigenen und fremden Komponenten zusammenstellen wollen, kann es daher gute Gründe geben, sich für Symfony zu entscheiden.

6.3 Mögliche Gründe für die Beliebtheit

Die bisher genannten Punkte legen nahe, dass Laravel für einen Großteil der PHP-Entwickler ein empfehlenswertes Framework ist. Das bedeutet jedoch nicht, dass es in jedem Fall die absolut beste Wahl ist.

Ein wichtiger Faktor für die Popularität ist sicherlich die erfolgreiche Öffentlichkeitsarbeit. Laravel tritt mit klaren Versprechungen und einer wiedererkennbaren Identität auf. Wenn damit Aufmerksamkeit geweckt wird und sich die Versprechungen zudem bewahrheiten, wird positiv darüber gesprochen. Das steigert wiederum den Bekanntheitsgrad.

Hilfreich könnte ebenso die Verwandtschaft mit Symfony sein, das bereits bei der Entstehung des Laravel-Projekts über eine große Community verfügte.

Ein weiterer wichtiger Punkt ist letztendlich auch die niedrige Hürde für Neueinsteiger. Dazu trägt von Anfang an die umfangreiche Dokumentation bei. Außerdem existiert mit *Laracasts* (siehe 3.3) eine einzigartige Möglichkeit, das Framework in seiner gesamten Funktionalität schnell, bequem und anhand von Best-Practices kennenzulernen.

6.4 Schlussbemerkungen

Die große Popularität Laravels ist in Bezug auf die Ergebnisse dieser Arbeit absolut nachvollziehbar. Allerdings sollte man daraus nicht den Schluss ziehen, dass es generell anderen Frameworks vorzuziehen ist. In Abschnitt 2 sind einige Punkte beschrieben, die bei der Wahl eines geeigneten Frameworks eine Rolle spielen. Diese wurden an passenden Stellen im Vergleich aufgegriffen, wobei die jeweiligen Unterschiede zwischen Laravel und Symfony erläutert wurden. Für die Wahl des optimalen Frameworks empfiehlt es sich, die einzelnen Unterschiede zu berücksichtigen und mit den Anforderungen des Projekts oder auch persönlichen Präferenzen abzugleichen.

Quellenverzeichnis

Literatur

[1] Trygve Reenskaug. *The model-view-controller (mvc) its past and present*. Techn. Ber. Oslo, Norway: University of Oslo, Aug. 2003. URL: http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf (siehe S. 5).

[2] Petar Tahchiev u. a. *JUnit in Action*. Greenwich, CT, USA: Manning Publications Co., 2010 (siehe S. 3).

Online-Quellen

[3] Irfan Fauzi. *15 Best Free PHP Frameworks of 2015*. 2015. URL: http://beebom.com/2015/02/best-free-php-frameworks (besucht am 31. 12. 2015) (siehe S. 12).

[4] Kristian Köhntopp. *Prüfe importierte Parameter. Traue niemandem*. 2008. URL: http://www.php-faq.de/q/q-sicherheit-parameter.html (besucht am 12. 12. 2015) (siehe S. 10).

[5] *Laravel (Projects using Symfony)*. 2015. URL: https://symfony.com/projects/laravel (besucht am 01. 01. 2016) (siehe S. 14, 18).

[6] *Laravel - The PHP Framework For Web Artisans*. 2015. URL: https://laravel.com/ (besucht am 31. 12. 2015) (siehe S. 13).

[7] Hartmut Schlosser. *Innovationsschub für Laravel: Das PHP Framework für Web-Künstler*. 2014. URL: https://entwickler.de/online/innovationsschub-fuer-laravel-das-php-framework-fuer-web-kuenstler-139455.html (besucht am 31. 12. 2015) (siehe S. 13).

[8] Bruno Skvorc. *The Best PHP Framework for 2015: SitePoint Survey Results*. 2015. URL: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/ (besucht am 31. 12. 2015) (siehe S. 12).

[9] *The Symfony Book*. 2015. URL: http://symfony.com/doc/current/book/index.html (besucht am 01. 24. 2016) (siehe S. 22).

[10] *Usage of server-side programming languages for websites*. 2015. URL: http://w3techs.com/technologies/overview/programming_language/all (besucht am 11. 12. 2015) (siehe S. 7).

[11] *Usage statistics and market share of PHP for websites*. 2015. URL: http://w3techs.com/technologies/details/pl-php/all/all (besucht am 11. 12. 2015) (siehe S. 7, 8).

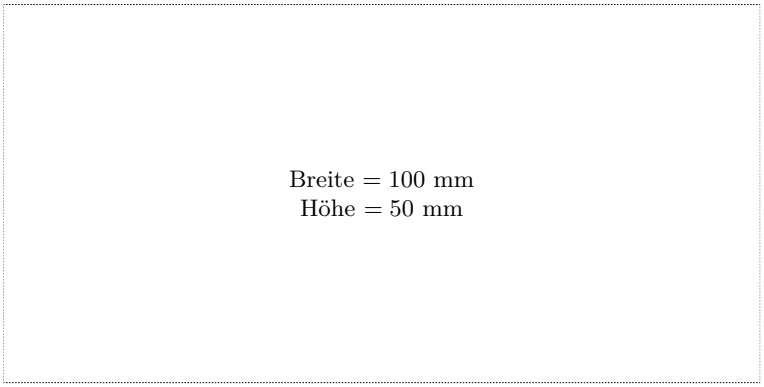
[12] Jeffrey Way. *Laracasts*. 2015. URL: https://laracasts.com/ (besucht am 31. 12. 2015) (siehe S. 14).

[13] Jeffrey Way. *Laravel 5 Fundamentals*. 2015. URL: https://laracasts.com/series/laravel-5-fundamentals (besucht am 01. 24. 2016) (siehe S. 22).

[14] *What is Symfony*. 2015. URL: http://symfony.com/what-is-symfony (besucht am 01. 01. 2016) (siehe S. 14).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —