

EVALUATION VON SUCHSYSTEMEN FÜR EINE AUF NEOS CMS BASIEREN- DE VERANSTALTUNGSVERWALTUNG



Technische Hochschule
Ingolstadt



BACHELORARBEIT

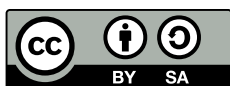
TECHNISCHE HOCHSCHULE INGOLSTADT

Fakultät Elektrotechnik und Informatik
Studiengang Informatik

Anmeldung 22.12.2015
Abgabe 21.01.2016

von
Stefan Braun
stefan.braun@respon.se

Erstprüfer Prof. Dr. Franz Regensburger
Zweitprüfer Prof. Dr. Bernd Hafenrichter
Betreuer Günter Huber



Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung – Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/>.

ABSTRACT

Im Zuge einer neuentwickelten Webanwendung für die Verwaltung öffentlicher Veranstaltungen der Stadt Ingolstadt wird eine Suchlösung benötigt. Hierfür stellt der Markt von datenbankintegrierten Systemen über lokalen Anwendungen bis hin zu cloudbasierten Produkten verschiedene Ansätze zur Verfügung. Um aus dieser Menge von Search Engines eine Auswahl treffen zu können und um eine konkrete Empfehlung abzugeben, werden systematisch Anforderungen ermittelt. Diese basieren teils auf einer vorangehenden Vorstellung der technischen Aspekte der thematisierten Veranstaltungsverwaltung. Es folgt die abschließende Evaluation ausgewählter Systeme auf Grundlage eines Kriterienkatalogs.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS	vii
QUELLCODEVERZEICHNIS	vii
TABELLENVERZEICHNIS	viii
VORWORT	ix
1 EINLEITUNG	1
1.1 Über das Veranstaltungsverwaltungssystem INsite	1
1.2 Aufgabenstellung	1
1.3 Abgrenzung der Eigenleistung	3
2 DAS NACHFOLGESYSTEM REEVENT	4
2.1 Eine Veranstaltungsverwaltung	4
2.2 Auswahl der zugrundeliegenden Plattform	5
2.2.1 Adobe ColdFusion	5
2.2.2 Flow und Neos CMS	6
2.2.3 Alternativen	7
2.2.4 Fazit	9
2.3 Wahl der Datenbanklösung	9
2.4 Ansätze zur Modellierung der Mehrsprachigkeit	10
2.4.1 Verwendung des Content Dimension-Konzepts von Neos CMS . . .	11
2.4.2 Ersetzung einzelner Attribute durch übersetzbare Objekte	12
2.4.3 Trennung von übersetzbarem und statischen Inhalt im Domänenmodell	13
2.5 Zeitpunktabhängige Daten	15
2.6 Workspaces	17
3 ANFORDERUNGSANALYSE	19
3.1 Motivation	19
3.2 Klassifizierung von Anforderungen	19
3.3 Anforderungsquellen	20
3.3.1 Identifikation der Stakeholder	20
3.3.2 Bestimmung des Systemkontextes	21
3.3.3 Bestehende Systeme	24
3.3.4 Dokumente	28
4 SUCHSYSTEME – EINE MARKTÜBERSICHT	29
4.1 SQL-basierte Suche	29
4.1.1 Filterung	29
4.1.2 Suche mit LIKE-Operator	30
4.1.3 Eingebaute Volltextsuche	31

4.1.4	Levenshtein-Distanz	33
4.1.5	N-Gramme	33
4.2	Suchsysteme als lokale Anwendung	34
4.3	Google Seach Appliance	36
4.4	Google Site Search	36
4.5	Clouddienste mit Suchfunktionalität	37
5	POLYGLOTTER ANSATZ MIT NEO4J	39
5.1	Empfehlung auf Grundlage vorheriger Seitenbesucher	39
5.2	Identifizierung des Nutzers	40
5.3	Graphdatenbank neo4j zur Speicherung	40
5.4	Abfrage der Empfehlungen durch Cypher	41
5.5	Erweiterung und Alternativen	42
5.6	Mögliche Nachteile des vorgestellten Ansatzes	43
6	VORAUSWAHL DER ZU EVALUIERENDEN SYSTEME	44
6.1	Auflistung der zu evaluierenden Suchlösungen	45
7	EVALUIERUNG	46
7.1	Aufstellung eines Bewertungsschemas	46
7.2	Ein Überblick über die Testumgebung	46
7.3	Herkunft der Testdaten	47
7.4	Definition des Kriterienkataloges	47
7.5	Über die Qualität einer Volltextsuche	48
7.5.1	Effektivität	48
7.5.2	Effizienz	49
7.6	Anmerkungen zu einzelnen Tests	49
7.6.1	Indizierung bei SQL-basierter Suche	49
7.6.2	Kostenkalkulation für Amazon CloudSearch	51
7.7	Quellen für Testfehler	51
8	FAZIT	53
8.1	Auswertung der Testergebnisse	53
8.2	Empfehlung	54
8.3	Ausblick	54
	ANHANG	56
A	KRITERIENKATALOG	57
B	TESTERGEBNISSE	61
C	PESSIMISTISCHES UND OPTIMISTISCHES SPERREN	62
D	USERTRACKING FÜR VERHALTENSBASIERTE EMPFEHLUNGEN	64
D.1	HTTP Cookie	64
D.2	Frontend-User	65
D.3	Browser Fingerprinting	65

E	STATISTIKEN	67
F	TABELLEN	68
G	LISTINGS	69
	LITERATUR	71
	ERKLÄRUNG	76

ABBILDUNGSVERZEICHNIS

Abbildung 1	Screenshot von INsite	2
Abbildung 2	Screenshot des bestehenden Suchformulars	2
Abbildung 3	Neos CMS Backend	7
Abbildung 4	Abstrakte Darstellung des Content Dimensions Konzepts aus Neos CMS	11
Abbildung 5	Auslagerung des übersetzbaren Teils einer Klasse in einen Neos Node am Beispiel der Klasse <code>GeoLocation</code>	12
Abbildung 6	Translatable-Objekte am Beispiel der Klasse <code>GeoLocation</code>	13
Abbildung 7	Auslagerung des übersetzbaren Teils einer Klasse in einen Neos Node am Beispiel der Klasse <code>GeoLocation</code>	14
Abbildung 8	Realisierung von <i>HistorySteps</i> an der Klasse <code>AbstractEntityWithContent</code>	16
Abbildung 9	Scope, Systemkontext und irrelevante Umgebung	22
Abbildung 10	Historische Schritte am Beispiel einer Veranstaltung	23
Abbildung 11	Häufige Suchbegriffe	25
Abbildung 12	Screenshot der Suche des Outlook-Kalenders	27
Abbildung 13	Marktanteile von Websuchmaschinen im Jahr 2015	28
Abbildung 14	Beispiel eines Graphen in neo4j, Benutzer besuchen eine Veranstaltung	41
Abbildung 15	Verteilung von Webprogrammiersprachen	67
Abbildung 16	Veranstaltungen pro Jahr in INsite	67

QUELLCODEVERZEICHNIS

Listing 2.1	Annotation der Property <code>\$eventName</code> mit einem Validator für Textlänge	13
Listing 4.1	Filterung nach Region	29
Listing 4.2	Suche nach Veranstaltungen mittels SQL LIKE	31
Listing 4.3	FULLTEXT Index unter MySQL	32
Listing 4.4	Volltextsuche unter MySQL	32
Listing 4.5	JSON-Dokument für die Indizierung in Solr	35
Listing 4.6	Google-Websuche mit site-Parameter	37
Listing 5.1	Einfügen von Benutzer, Veranstaltung und VISITED-Beziehung	41
Listing 5.2	Abfrage von Vorschlägen für einen User mit der ID 1, die Anzahl der Vorschläge wird auf fünf begrenzt	42

Listing 7.1	EXPLAIN bei Trigramm-basierter Suche	50
Listing 7.2	Aktivierung von Trigrammen unter PostgreSQL für eine Datenbank	50
Listing A.1	T-SQL Abfrage <code>sp_spaceused</code>	60
Listing D.1	User-Agent von Google Chrome 47 (64-bit) unter Windows 8.1 Pro	65
Listing G.1	SQL Statement zur Auflistung der jährlichen Neueintragen in INsite	69
Listing G.2	Volltextsuche mit PostgreSQL	69
Listing G.3	N-Gramm basierte Suche mit PostgreSQL	70
Listing G.4	LIKE basierte Suche mit PostgreSQL	70
Listing G.5	Suche in Amazon CloudSearch mit <i>dismax</i> Parser	70
Listing G.6	Suche in Apache Solr mit <i>edismax</i> Parser	70

TABELLENVERZEICHNIS

Tabelle 1	Zeitleiste der TH Ingolstadt von der Gründung bis zum Jahr 2015	15
Tabelle 2	Beispieldatensätze von Veranstaltungen	30
Tabelle 3	Beispieldokumente für einen Inverted Index	34
Tabelle 4	Beispiel eines Inverted Index	35
Tabelle 5	Testresultate	61
Tabelle 6	Messwerte bei einzelnen Tests	61
Tabelle 7	ColdFusion Frameworks	68

ABKÜRZUNGEN

API	Application Programming Interface	JSON	JavaScript Object Notation
CSS	Cascading Stylesheets	OCR	Optical Character Recognition
CFML	ColdFusion Markup Language	ORM	Object-Relational Mapping
CMS	Content Management System	REST	Representational State Transfer
GIN	Generalized Inverted Index	SOAP	Simple Object Access Protocol
GiST	Generalized Search Tree	SQL	Structured Query Language
GPS	Global Positioning System	URL	Uniform Resource Locator
GSA	Google Search Appliance	XML	Extensible Markup Language
HTML	Hypertext Markup Language		
ID	Identifikator		

VORWORT

*Im Jahr 2014 wurde eine Milliarde registrierter Internetseiten verzeichnet.*¹

– internet live stats [Int14]

Seit der Entstehung des Internets ist die Menge der Webseiten und der darüber abrufbaren Daten ins nahezu Unermessliche gestiegen. Lediglich mit Hilfe von Websuchmaschinen wie Google, Yahoo oder Bing ist es Nutzern noch möglich, aus dieser Informationsflut die für sie relevanten Inhalte aufzuspüren.

Doch nicht nur im Web nimmt die Größe von Datensätzen stetig zu: In Zeiten von Big-Data und umfangreichen lokalen Applikationen gestaltet es sich auch im Unternehmensumfeld immer schwieriger, den Überblick zu behalten. Wie für das Internet die Websuchmaschinen, so erlauben lokale *Search Engines* die Durchsuchung interner Dokumente und Datenbanken. Und wie im Internet bei der Auswahl der Websuchmaschine steht auch hier der Anwender vor der Wahl: Welche *Search Engine* kann meine Anforderungen am besten erfüllen?

DANKSAGUNG

allen Freunden, Kommilitonen und meiner Familie für die Unterstützung während des gesamten Studiums und der Korrekturlesung,

Prof. Dr. Regensburger von der TH Ingolstadt für die Betreuung dieser Arbeit,

Günter Huber und Gerhard Rupp von response für die Vergabe des Bachelorarbeitsthemas,

meinen **Arbeitskollegen** bei response für die Hilfe rund um technische Aspekte von ReEvent, Neos CMS und Flow,

Dr. Paul Spannaus und David Dionis von der TH Ingolstadt für ihre Einführung zu L^AT_EX aus dem Kurs „Professionelle Textsatzsysteme“,

der **tex.stackexchange.com** Community für ihre Hilfe in vielen kleineren und auch größeren Problemen rund um das Thema Textsatz,

sowie **allen anderen**, die der Meinung sind, vergessen worden zu sein – Danke!

¹ Gezählt wurden registrierte Domains, welche auch zu IP-Adressen auflösen [Int14]

1.1 ÜBER DAS VERANSTALTUNGSVERWALTUNGSSYSTEM INSITE

Seit dem Jahr 1999 pflegen Mitarbeiter der Stadt Ingolstadt und der *Ingolstadt Tourismus und Kongress GmbH*¹ jährlich circa 4 000 Veranstaltungen für den Internetauftritt der Stadt Ingolstadt.² Dies erfolgt bis dato mit dem webbasierten Veranstaltungsverwaltungssystem INsite, welches von der *response informationsdesign gmbh & co. kg*³ entwickelt wurde. Technische Grundlage für INsite stellt der Applicationserver Adobe ColdFusion sowie die zugehörige Programmiersprache CFML dar.

Veranstaltungen für INsite werden größtenteils händisch eingetragen. Zudem besteht für Bürger der Stadt die Möglichkeit, per Webformular⁴ Vorschläge für neue Veranstaltungen zu melden. Diese Eintragungen durchlaufen anschließend einen mehrstufigen Genehmigungsprozess – welcher jedoch nicht in INsite abgebildet ist – und werden dementsprechend entweder abgelehnt oder über INsite in den Kalender eingetragen.

Des Weiteren werden Sitzungen des Stadtrates automatisiert aus dem Ratsinformationssystem *SessionNet* zu INsite synchronisiert.⁵ Auch nach außen hin bietet INsite Schnittstellen an: Über einen SOAP-Webservice werden aktuelle Veranstaltungen beispielsweise an den Webauftritt der ITK GmbH weitergegeben.

Allerdings fehlt es INsite an einigen, für Veranstaltungsverwaltungen typischen, Funktionen. So ist es beispielsweise nicht möglich, Serientermine zu erfassen. Auch der bereits erwähnte Genehmigungsprozess für bestimmte Veranstaltungen ist nicht in INsite integriert. Einen weiteren Mangel stellt das Design der Applikation dar, zu sehen in Abbildung 1, welches als nicht mehr zeitgemäß betrachtet werden kann.

Aus diesen Gründen entschied sich die Stadtverwaltung der Stadt Ingolstadt für eine Neuentwicklung der Veranstaltungsverwaltung, welche bei *response* in Auftrag gegeben wurde. Auf dem *ReEvent* genannten Nachfolgesystem basiert wiederum die vorliegende Arbeit. Insbesondere die relevanten technischen Aspekte von *ReEvent* werden in Kapitel 2 dargestellt.

1.2 AUFGABENSTELLUNG

Die von INsite bereitgestellten Veranstaltungsdaten heutigen Datums werden in erster Linie auf der Startseite der städtischen Homepage angezeigt. Zudem existiert ein Formular zur gefilterten Anzeige aller vorliegenden Veranstaltungen, beispielsweise durch Eingren-

¹ im Folgenden als *ITK GmbH* bezeichnet

² siehe <http://www.ingolstadt.de>

³ Schreibweise im Corporate Design, im Weiteren als *response* aufgeführt

⁴ siehe http://www2.ingolstadt.de/Aktuelles/Veranstaltungen/Veranstaltung_eintragen/

⁵ vergleiche <http://www.ingolstadt.de/sessionnet/si0040.php>, SessionNet Veranstaltungskalender der Stadt Ingolstadt

INsite - Veranstaltungsverwaltung

Veranstaltung

[Hinzufügen](#)
[Anzeigen/Bearbeiten](#)
[Suchen](#)

[Veranstaltung](#)
[Veranstalter](#)
[Kategorie](#)
[Ort](#)
[Vorverkauf](#)

5 . 8 . 2015 . [Anzeigen](#) [Akt. Monat](#)

Anzeigekriterien

Zeitraum: 5.8.2015-0:00 bis 5.8.2015-23:59

Veranstaltungen

Datum von	Datum bis	von	bis	Name der Veranstaltung	Veranstaltungsort	Veranstalter
15.06.2014	30.09.2015	11:00	18:00	Rost auf Stahl - Bleistift auf Papier	Lechner Museum	Lechner Museum
17.09.2014	27.09.2015			Die Alpen im Krieg – Krieg in den Alpen. Die Anfänge der deutschen Gebirgstruppe 1915	Reduit Tilly	Bayerisches Armeemuseum

Abbildung 1. Screenshot der aktuellen⁶ Version von INsite, angezeigt wird eine Liste von Veranstaltungen

zung des Zeitraums oder Festlegen einer Kategorie.⁷ Ein optionales Suchfeld ergänzt die Filterfunktion. Wird ein Suchbegriff angegeben, so werden nur Veranstaltungen angezeigt, deren Namen oder Beschreibungstext die Zeichenfolge enthalten. Ein ähnliches Suchformular existiert im Backend von INsite. Serverseitig wird die Suche als parametrisierte SQL-Abfrage realisiert.

Veranstaltung suchen

Von

Bis

Region

Kategorie

Suchwort

Abbildung 2. Screenshot des bestehenden Formulars zur Filterung und Suche von Veranstaltungen im Frontend von INsite⁸

Auch für ReEvent wird eine Suchlösung benötigt. Neben der Minimallösung einer einfachen SQL-Abfrage existieren jedoch verschiedene Ansätze zur Suche, wie beispielsweise eine dedizierte SearchEngine wie *Apache Solr* [GP14, S. 4]. Gegenstand dieser Arbeit soll es sein, mögliche Suchsysteme zu recherchieren und in Hinblick auf die Tauglichkeit für

⁷ siehe <http://www2.ingolstadt.de/Aktuelles/Veranstaltungen/>

die neue Veranstaltungsverwaltung ReEvent zu evaluieren. Die Ergebnisse der Evaluation sollen es erlauben, eine ausgewählte Suchlösung für die Suche in ReEvent zu empfehlen.

1.3 ABGRENZUNG DER EIGENLEISTUNG

Die Konzepte aus [2.4.1](#), [2.4.3](#), [2.5](#) und [2.6](#) wurden von Mitarbeitern der response informationsdesign gmbh & co. kg ausgearbeitet und werden an dieser Stelle lediglich als Hintergrundinformation wiedergegeben.

2 | DAS NACHFOLGESYSTEM REEVENT

Nach [Rup14, S. 85] ist es für den Prozess des Requirements Engineerings notwendig, den Systemkontext zu betrachten. Aus Sicht der Veranstaltungssuche stellt das Gesamtprojekt ReEvent einen unmittelbaren Nachbarn dar. Da die technischen Aspekte ReEvents Einfluss auf die Wahl der Suchlösung nehmen können, soll im Folgenden ein Überblick über grundlegende Eigenschaften des Systems vermittelt werden.

2.1 EINE VERANSTALTUNGSVERWALTUNG

Wie bereits eingangs dargestellt, soll ReEvent das Altsystem INsite ablösen und die Funktion einer öffentlichen, kommunalen Veranstaltungsverwaltung übernehmen. Hierzu gehört die Eingabe neuer Veranstaltungsdaten aus diversen Quellen – Vorschläge, die an die Redakteure per Email oder Webformular eingehen, ebenso wie automatisiert per Webservice übertragene Veranstaltungen des Ratsinformationssystems der Stadt.¹

Veranstaltungen erhalten einen Namen, einen kurzen Beschreibungstext für eine Vorschau und einen optionalen, längeren Beschreibungstext. Des Weiteren können beliebige Dateien zu Veranstaltungen abgespeichert werden. Möglich sind beispielsweise Veranstaltungseinladungen im PDF- oder Microsoft Word-Format, Videodateien oder Bilddateien.

Eingegebene Veranstaltungen können schließlich auf einer entsprechenden Website – fortführend wird der Begriff des *Frontends* verwendet – veröffentlicht werden. Hier erhalten Benutzer einen Überblick über tagesaktuelle Veranstaltungen und können weiterführende Informationen über diese abrufen. Sowohl für das Frontend als auch das Backend soll eine Suchfunktion entstehen – was den Kern dieser Arbeit darstellt.

Es sei des Weiteren erwähnt, dass sich ReEvent zum Zeitpunkt der Erstellung dieser Arbeit in Entwicklung befindet. Getroffene Aussagen über diese Software spiegeln demnach nur den aktuellen Stand wieder und müssen nicht auf die endgültige Version des Produktes zutreffen.

BACKEND einer Webanwendung ist der zugangsbeschränkte Teil, in dem Konfiguration oder redaktionelle Aufgaben durchgeführt werden. In ReEvent werden im Backend Veranstaltungen eingepflegt und bearbeitet.

FRONTEND wiederum ist eine Menge von Seiten, welche prinzipiell öffentlich zugänglich sind. In ReEvent sollen Veranstaltungen in einem frei zugänglichen Kalender über das Internet aufgerufen werden können. Wird für manche oder alle Teile des Frontends ebenfalls ein Benutzerkonto benötigt – beispielsweise für ein Kundenkonto in einem Onlineshop-System – so wird dies als *Frontend-Benutzer* bezeichnet.

¹ siehe <http://www4.ingolstadt.de/sessionnet/infobi.php>, das Ratsinformationssystem *SessionNet* der Firma Somacos GmbH & Co. KG.

2.2 AUSWAHL DER ZUGRUNDELIEGENDEN PLATTFORM

Zu Beginn der Entwicklung von ReEvent steht die Frage nach der zu verwendenden Programmiersprache, beziehungsweise nach einem darauf aufbauenden Framework. Hierfür bieten sich mehrere Möglichkeiten an, welche teils schon in vorangegangenen Projekten bei response eingesetzt wurden.

2.2.1 Adobe ColdFusion

Bereits die bestehende Anwendung INsite wurde auf Basis des Application Servers *Adobe ColdFusion* entwickelt. Die ColdFusion-Plattform bietet eine Implementierung der Webprogrammiersprache *ColdFusion Markup Language* – kurz CFML – an [FAC10, S. 8].²

Neben INsite wurden auch weitere Teilbereiche des Webauftritts der Stadt Ingolstadt mit ColdFusion erstellt: Beispiele hierfür stellen der Newsletterversand³ und die Seiten des Stadttheaters⁴ dar. Somit kann response auf einen aufgebauten Erfahrungsschatz zurückgreifen, was durchaus für die Verwendung von ColdFusion für die Entwicklung von ReEvent spricht.

Im Gegensatz zu vergleichbaren Programmiersprachen ist Adobe ColdFusion jedoch nicht frei verfügbar und muss von Adobe kostenpflichtig lizenziert werden. Die *Standard-Edition* von ColdFusion 11 wird für 1 783,81 € angeboten. Auch wenn dies prinzipiell nur ein einmaliger Posten ist, müssen die Upgradegebühren auf die nächste Versionsnummer ebenfalls mit eingeplant werden.⁵ Ein Upgrade kann nach Auslaufen des Supportzeitrahmens einer Version zwingend notwendig werden, da keine weiteren Sicherheitsupdates ausgeliefert werden. Alternativ zu Adobe ColdFusion existieren auch freie Implementierungen der CFML, wie beispielsweise Railo [Dre11, Abschnitt „Inexpensive and free“]. Für die Stadt Ingolstadt spielt dieser Punkt nur eine untergeordnete Rolle, da aufgrund bestehender Applikationen in jedem Fall eine Lizenz für Adobe ColdFusion vorliegt. Allerdings würde sich mit ReEvent als ColdFusion-Applikation eine weitere Abhängigkeit im Falle eines Systemwechsels aufbauen.

Ein weiteres Argument gegen die Verwendung von ColdFusion stellt die stetig sinkende Verbreitung der Programmiersprache dar. Beispielsweise ist die CFML nicht länger im TIOBE Index enthalten [TIO15]. Der nicht unumstrittene Index ermittelt die Trefferzahlen für die Suchanfrage + "<language> programming" in 25 verschiedenen Suchmaschinen und leitet daraus ein Popularitätsranking ab [Men15]. Aufgeführt werden bis zu 100 Sprachen – derzeit ohne ColdFusion.⁷ Auch die Statistik in Abbildung 15 von W3Techs weist für ColdFusion einen Marktanteil von lediglich 0.7% Prozent vor. Während die Verbreitung einer Programmiersprache selbst nichts über ihre Qualität oder Eignung für ein spezifisches Projekt aussagt, so sind die daraus entstehenden Folgen durchaus legitime Entscheidungskriterien. So ist über die vergangenen Jahre ein Sterben einiger großer auf ColdFusion aufbauender Frameworks zu beobachten. Tabelle 7 listet frei verfügbare Frameworks

² ColdFusion und CFML werden in der Literatur häufig synonym verwendet

³ siehe <http://www2.ingolstadt.de/newsletter>

⁴ siehe <http://theater.ingolstadt.de/>

⁵ Preisangabe inklusive Mehrwertsteuer, Stand 06.09.2015, vergleiche <http://www.adobe.com/de/products/coldfusion-family.html>

⁶ das Upgrade von ColdFusion 10 Standard auf die Version 11 schlägt derzeit mit 891,31 € zu Buche

⁷ Stand 22.10.2015, TIOBE Index Oktober

für ColdFusion auf. Lediglich zwei der Einträge werden derzeit aktiv weiterentwickelt.⁸ Entsprechend ist die Kompatibilität dieser Frameworks mit kommenden Versionen von Adobe ColdFusion oder Railo nicht sichergestellt. ORM-Bestandteile der Frameworks werden ebenfalls nicht länger an neue Versionen der gängigen Datenbanksysteme angepasst. Somit ist fraglich, ob die Verwendung der ColdFusion-Plattform als zukunftssicher bezeichnet werden kann.

2.2.2 Flow und Neos CMS

Neben Adobe ColdFusion nutzt response auch das Content Management System TYPO3 sowohl für Kundenprojekte als auch für die firmeneigene Homepage [res]. Zusätzlich zur Verwaltung von einfachen Webseiteninhalten wie Text und Bild bietet das PHP-basierte TYPO3 auch die Möglichkeit, das System durch selbstgeschriebene Erweiterungen zu ergänzen [RKH14, S. IX]. Diese *Extensions* können zudem Funktionen des Grundsystems nutzen – wie beispielsweise die Benutzerverwaltung. Aufgrund dessen stellt TYPO3 eine mögliche Entwicklungsplattform für ReEvent dar.

Hierbei ist allerdings eine grundsätzliche Entscheidung zu treffen: Zum aktuellen Zeitpunkt existieren zwei Entwicklungsstränge des CMS. Zur Entwicklung der Version 5 von TYPO3 wurde das System von Grund auf neugeschrieben. Die neue Version sollte auf einem eigens hierfür programmierten PHP-Framework namens FLOW3 basieren. Im weiteren Verlauf wurde jedoch entschieden, das neue System als eigenständiges Produkt weiterzuentwickeln [RKH14, S. X]. Die beiden Projekte divergierten weiter, bis sie sich schließlich voneinander trennten. Seit Herbst 2015 ist das inzwischen umbenannte *Neos CMS* von der TYPO3 Association losgelöst, auch der Name des Frameworks wurde zu *Flow* geändert [Neo15b]. Das ursprüngliche TYPO3 wird nun als TYPO3 CMS fortgeführt. Um Verwechslungen mit Neos zu vermeiden hat man die folgende Version von TYPO3 CMS auf die Versionsnummer 6 angehoben [RKH14, S. XIII].

Flow stellt ein nach dem *Model-View-Controller* Pattern funktionierendes Webframework dar, welches mit ORM, AOP, Dependency Injection und Templating viele moderne Features mit sich bringt. Als objektrelationaler Mapper wird *Doctrine* eingesetzt, die Views werden von der Templating Engine *Fluid* gerendert.

Wie bereits eingangs erläutert, kann TYPO3 durch Extensions um benötigte Funktionen erweitert werden. Für TYPO3 CMS gibt es frei verfügbare Erweiterungen, wie das *powermail*-Plugin zum Versand von Newsletter-Mails. Diese können vom offiziellen Extension Repository⁹ bezogen werden, welches derzeit¹⁰ 1316 Plugins bereitstellt. Wenn auch geplant, so gibt es derzeit kein vergleichbares Angebot für Neos CMS. Die offizielle Webseite von Neos geht von „ein paar hundert“ verfügbaren Extensions aus und verweist auf die Seite <https://packagist.org/> [Neo15a]. Die geringere und schlechter organisierte Verfügbarkeit von Erweiterungen schränkt Neos im Vergleich zu TYPO3 CMS ein, insbesondere beim Aufbau komplexerer Webseiten könnte sich ein reiches Angebot an Plugins als hilfreich erweisen.

Das ursprüngliche TYPO3 CMS hat den Ruf eines „komplexe[n] und schwer zu erlernende[n] Programm[es]“ [BKH11, S. X]. Dies ist zum Teil auch dem schieren Umfang der

⁸ Stand 25.10.2015

⁹ siehe <https://typo3.org/extensions/repository/>

¹⁰ Stand 25.12.2015

als Enterprise-CMS titulierten Anwendung geschuldet: Mehrsprachigkeit, Content-Review und Benutzerverwaltung sind Funktionen, die eher in größeren Webauftritten notwendig sind. Einen Schritt in die richtige Richtung machte Neos mit einer starken Überarbeitung des Benutzerinterfaces: Als auffälligstes Merkmal wird die Frontendseite direkt in das Backend eingebettet, Inhalte können *inline* bearbeitet werden. Dadurch erhält der Benutzer bereits bei der Eingabe der Änderung ein Feedback über die Auswirkung auf die ausgegebene Seite.

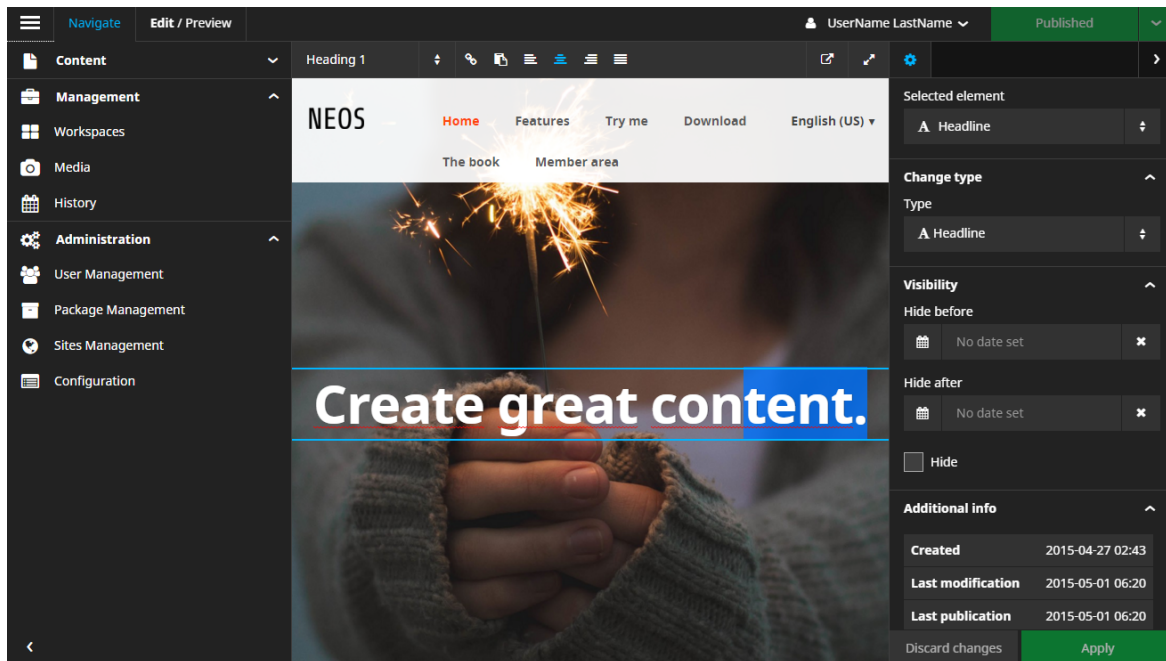


Abbildung 3. Backend von Neos CMS, gezeigt wird der Inline-Texteditor. Das Backend ist responsive aufgebaut, über das Menü links oben kann auf installierte Packages zugegriffen werden – wie auch auf ReEvent.

Die Auftrennung in zwei Entwicklungsstränge bringt nicht nur Positives mit sich: Finanzielle Mittel verbleiben bei der TYPO3 Association, Neos CMS ist daher teils auf Spenden angewiesen [Neo15b]. Neben der monetären Aufteilung bedeutet die gleichzeitige Pflege zweier Produkte auch eine verkleinerte Zahl an Entwicklern für ein einzelnes Projekt. Auch die Anwender werden durch die Konkurrenzsituation zwischen den beiden Projekten in zwei Lager getrennt, was einer Schwächung der Community gleichkommen dürfte.

2.2.3 Alternativen

Adobe ColdFusion und Neos CMS sind nicht die einzigen Lösungen, die für ein Projekt wie ReEvent in Frage kommen.

symfony

symfony ist wie auch Flow ein Web-Framework auf Basis von PHP. Mit einem ersten Release im Dezember 2005 ist es vergleichsweise länger am Markt als das 2011 veröffentlichte Flow [Pot, S. 21]. Wenn auch nicht auf den ersten Blick ersichtlich, so bestehen dennoch Zusammenhänge zwischen den beiden Frameworks: So verwenden beide Doctrine als ORM.

Auch nutzt Flow einige Packages von symfony, wie beispielsweise zum Parsen von Konfigurationsdateien.¹¹ Anders als Flow unterstützt symfony jedoch keine aspektorientierte Programmierung.

Ruby on Rails

Bereits seit 2004 gibt es mit dem auf der Skriptsprache Ruby basierenden *Ruby on Rails* einen weiteren Vertreter der Webframeworks [OKo7, S. 1]. Anders als Doctrine unter Flow setzt Rails auf das *ActiveRecord*-Pattern zur Speicherung und Bearbeitung von Objekten in der Datenbank. Während Doctrine mit einfachen PHP-Objekten arbeitet, welche über Repositories in die Datenbank übertragen werden, verfügt unter Rails jede Model-Klasse über Funktionen wie `save` oder `delete`.

Der Einfluss von Ruby on Rails auf jüngere Frameworks ist durchaus gegeben: Das ColdFusion-Framework CFWheels – eines der beiden, die noch aktiv weiterentwickelt werden – ist der Struktur nach Ruby on Rails nachempfunden.

Bisher konnte das Entwicklerteam bei response weder mit der Skriptsprache noch mit dem Framework Erfahrungen sammeln, was vermutlich in einer hohen Einarbeitungsdauer resultieren würde.

Andere erweiterbare CMS

Neben TYPO3 CMS und Neos CMS existieren auch weitere Content Management Systeme, deren Funktionsumfang durch Erweiterungen ergänzt werden kann. Als Beispiele hierfür sind WordPress oder Joomla zu nennen – beide basieren ebenfalls auf PHP.

Laut einer weiteren Studie von W3Techs nutzen 25,7 % aller Webseiten die Blogsoftware WordPress [W3T16] – was möglicherweise auch der einfachen Bedienung geschuldet ist. Prinzipiell ist WordPress eher darauf ausgelegt, auch von Anwendern mit geringerem technischem Hintergrund eingerichtet und verwendet zu werden. So wirbt WordPress auf der offiziellen HomePage¹² mit einer „5-minute installation“. Auf der technischen Seite fällt allerdings ein Unterschied zu beispielsweise Neos CMS auf: Als Datenbank wird offiziell nur MySQL unterstützt, Ports zu PostgreSQL wurden nicht weitergeführt.¹³

Es sei des Weiteren angemerkt, dass CMS-Funktionen nicht den entscheidenden Faktor für die Systementwicklung von ReEvent darstellen. Ein CMS stellt üblicherweise Komponenten wie eine Benutzer- oder Mediendatenverwaltung bereit. Auch wenn diese in ReEvent möglicherweise wiederverwendet werden könnten, so sollte dennoch ein zugrundeliegendes Framework die Entwicklung von ReEvent maximal unterstützen. Ausgehend von den Dokumentationen von WordPress und Joomla geht nicht hervor, dass die zur Verfügung gestellte Plattform in ihrem Funktionsumfang mit Flow mithalten kann.¹⁴

¹¹ siehe `composer.json` von Flow, <https://github.com/neos/flow-development-collection/blob/893358e66fc3e3e0fe4439db18564568298ab6a6/composer.json>

¹² siehe <https://wordpress.org/>, Stand 11.01.2016

¹³ siehe https://codex.wordpress.org/Using_Alternative_Databases

¹⁴ siehe http://codex.wordpress.org/Writing_a_Plugin und https://docs.joomla.org/J3.x:Developing_a_n_MVC_Component/Developing_a_Basic_Component

2.2.4 Fazit

Selbstverständlich ist die hier vorgestellte Auswahl an Möglichkeiten nicht erschöpfend. Mit diversen Java-Webframeworks, ASP .Net, Drupal oder node.js gibt es noch eine Vielzahl weiterer aktueller Systeme, die prinzipiell für die Entwicklung von ReEvent in Frage kommen. Die präsentierten Varianten bilden jedoch einen groben Querschnitt über die am Markt befindlichen Konkurrenten zu Flow und ColdFusion stehen auch stellvertretend für nicht miteinbezogene Alternativen.

Basierend auf der Annahme, dass jene Programmiersprache am geeignetsten ist, mit welcher das Entwicklerteam am besten umgehen kann, ist PHP durchaus eine schlüssige Wahl. Sowohl Flow als auch andere PHP-Frameworks wurden jedoch in keinen früheren Projekten bei response eingesetzt.

Derzeit¹⁵ wird seit beinahe einem Jahr unter der Kombination von Flow und Neos CMS an ReEvent gearbeitet. Die Wahl erfolgte unter anderem unter dem Gesichtspunkt der Vorreiterrolle als Erster eine Veranstaltungsverwaltung für Neos CMS anbieten zu können. In diesem Zusammenhang offenbarten sich auch einige Kritikpunkte an Flow – beispielsweise die nach wie vor unvollständige Dokumentation¹⁶ oder Caching-Probleme und lange Ladezeiten nach Änderungen von PHP-Quelltext. Somit wäre es möglicherweise vorteilhaft gewesen, auf ein etablierteres Framework wie symfony zu setzen.

Insbesondere da mit der Entwicklung von ReEvent auf Basis von Neos CMS und Flow bereits begonnen wurde, wird diese Auswahl für die kommenden Inhalte dieser Arbeit als gesetzt angenommen.

2.3 WAHL DER DATENBANKLÖSUNG

Die von ReEvent verwalteten Daten – prinzipiell PHP-Objekte des Flow-Frameworks – werden in einer Datenbank abgespeichert. Das Anlegen, Lesen, Aktualisieren und Löschen¹⁷ dieser Objekte wird vom ORM-System *Doctrine* in SQL-Anfragen umgewandelt. Da verschiedene Datenbanksysteme die SQL-Standards nur teilweise oder leicht unterschiedlich implementieren, greift Doctrine auf die zugrundeliegende Datenbank über eine *Database Abstraction Layer* zu, welche – wie der Name sagt – den Zugriff von der verwendeten Datenbank abstrahiert. Doctrine kann derzeit folgende Datenbanksysteme ansprechen [The15a].

- MySQL
- PostgreSQL
- Microsoft SQL Server (MSSQL)
- Oracle
- Sybase SQL Anywhere
- SQLite
- Drizzle

response hatte zuvor in anderen Projekten die Stadt Ingolstadt betreffend sowohl MySQL als auch das Pendant von Microsoft eingesetzt. Einem LAMP-Stack¹⁸ entsprechend

¹⁵ Stand 10.01.2016

¹⁶ vergleiche <http://flowframework.readthedocs.org/en/stable/TheDefinitiveGuide/index.html>

¹⁷ CRUD - create, read, update und delete sind die gängigen Operationen einer Anwendung

¹⁸ Ein LAMP-Stack bezeichnet eine in der Webentwicklung häufig eingesetzte Kombination aus Linux, Apache Webserver, MySQL Datenbank und PHP

wird ReEvent derzeit mit MySQL entwickelt. Ein Test mit dem aktuellen Entwicklungsstand von ReEvent ergab, dass die Datenbank problemlos durch PostgreSQL ersetzt werden könnte.

Neben den hier aufgelisteten relationalen Datenbanken gab es bereits Ansätze, eine dokumentenbasierte Datenbank wie *CouchDB* in das Framework Flow zu integrieren. Diese wurde jedoch scheinbar nicht fortgeführt.¹⁹ Derzeit verwaltet INsite circa 45 000 Veranstaltungen und damit eine ähnliche Anzahl von Zeilen in der zugehörigen relationalen Datenbank.²⁰ Probleme hinsichtlich Auslastung der Datenbank sind zum aktuellen Zeitpunkt nicht aufgetreten. Da die Zahl der Datensätze in Flow vermutlich ähnlich dimensioniert sein wird, ist die Einführung einer NoSQL-Datenbank hinsichtlich besserer Verteilbarkeit und Verfügbarkeit nicht notwendig.²¹ Dennoch sollte man die Möglichkeit eines Umstiegs auf eine NoSQL-Lösung nicht völlig außer Acht lassen. Aufgrund der starken Abhängigkeit des Flow-Frameworks und des geschriebenen Quellcodes zu Doctrine würde ein solcher Umstieg zu einem späteren Zeitpunkt jedoch großen Aufwand erfordern.

Was Doctrine schließlich noch im Vergleich zu ORM-Lösungen anderer Programmiersprachen²² abhebt, ist die Fähigkeit aus den geschriebenen PHP-Klassen heraus automatisch die Tabellenstruktur der Datenbank anzulegen. Werden Änderungen am Klassenmodell vorgenommen, so kann SQL-Code generiert werden²³, der die Datenbanktabellen wieder entsprechend anpasst.

2.4 ANSÄTZE ZUR MODELLIERUNG DER MEHRSPRACHIGKEIT

Anders als INsite soll ReEvent grundsätzlich auf Mehrsprachigkeit ausgelegt sein. Dies betrifft zum einen die Benutzeroberfläche, welche spezifisch für den User lokalisiert werden soll. Da jedoch städtische Veranstaltungen durchaus für ein internationales und multikulturelles Publikum von Interesse sind, sollte auch eine Übersetzung einzelner Veranstaltungen in ReEvent möglich sein. Ein relevantes Beispiel hierfür stellen Ereignisse im Rahmen der Ingolstädter Städtepartnerschaften dar.

Während die Internationalisierung statischer Oberflächen ein gängiges Feature aktueller Web-Frameworks ist – Flow setzt an dieser Stelle auf Übersetzungskataloge im xliiff-Format auf welche über die eingebaute Template-Engine Fluid zugegriffen werden kann [The15b, S. 278, 280f] – fehlt meist die Möglichkeit, eingepflegte Inhalte zu übersetzen. Dies ist auch beim PHP-Framework Flow der Fall.

Dementsprechend stellt sich die Frage der Umsetzung der Mehrsprachigkeit, welche im Folgenden thematisiert wird.

¹⁹ siehe <https://github.com/radmiraal/Radmiraal.CouchDB>

²⁰ siehe Statistik 16

²¹ Die im Folgenden vorgestellten Konzepte verlangen es teilweise, dass von einer Veranstaltung mehrere Versionen persistiert werden. Abhängig davon, wie stark diese Funktionen genutzt werden, könnte auch die Zahl der Datensätze ansteigen.

²² wie beispielsweise Hibernate unter Java

²³ sogenannte Migration-Skripte

2.4.1 Verwendung des Content Dimension-Konzepts von Neos CMS

Die in diesem Abschnitt 2.4.1 vorgestellten Konzepte wurden von Mitarbeitern der Firma response ausgearbeitet und werden an dieser Stelle lediglich als Hintergrundinformation mit angegeben.

Das Content Management System Neos ermöglicht Übersetzung von Inhalten auf Basis von *Nodes* genannten Strukturen. Jeder Inhalt einer durch Neos verwalteten Seite – beispielsweise Textblöcke oder Grafiken – wird durch einen Node repräsentiert [The15c, S. 15f]. Welche Attribute ein Node beinhaltet, ist abhängig vom *Node Type* dieses Nodes. *Node Types* stellen „Baupläne“ für Nodes dar, sie verhalten sich somit ähnlich wie Klassen zu Objekten [The15c, S. 15f].

Zur Übersetzung lässt sich in Neos das generische Konzept von *Content Dimensions* nutzen. Eine Content Dimension entspricht einer Möglichkeit, für einen Node Inhaltsvarianten anzulegen. Wurde beispielsweise eine Content Dimension *Sprache* definiert, so können für einzelne Sprachen {en_GB, de} übersetzte Varianten des Nodes hinterlegt werden.

Content Dimensions lassen sich zudem miteinander verbinden: Angenommen, es wird neben *Sprache* auch *Altersgruppe* und *Ausgabe-Gerätetyp* erzeugt, so können für alle möglichen Kombinationen personalisierte und angepasste Inhalte eines Nodes definiert werden [The15c, S. 60].

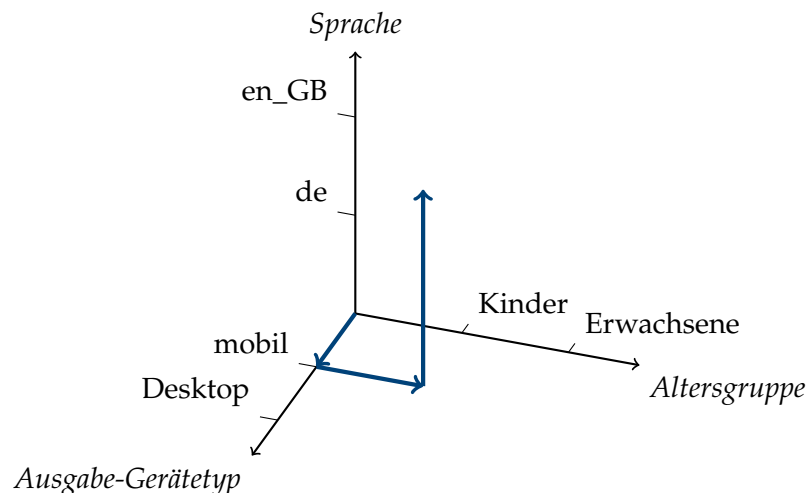


Abbildung 4. Content Dimensions können in Neos miteinander kombiniert werden – die Grafik zeigt eine mögliche Auswahl nach Sprache, Altersgruppe und Gerätetyp für die Ausgabe

Da ReEvent als Backend-Modul für Neos konzipiert wurde, wirft dies die Frage auf, ob sich Content Dimensions für die Mehrsprachigkeit in ReEvent verwenden lassen.

Dies verlangt es, den übersetzbaren Teil einer Klasse vom sprachinvarianten Part zu trennen. Beispielsweise verbleiben Relationen und Zahlenwerte wie GPS-Koordinaten in der ursprünglichen Klasse, während Benennungen oder Beschreibungen in einen Node verschoben werden.

Über Getter- und Setter-Funktionen kann anschließend transparent auf die im Node gespeicherten Daten zugegriffen werden.²⁴

²⁴ *information hiding* oder Geheimnisprinzip, vgl. [Bal99, S. 26]

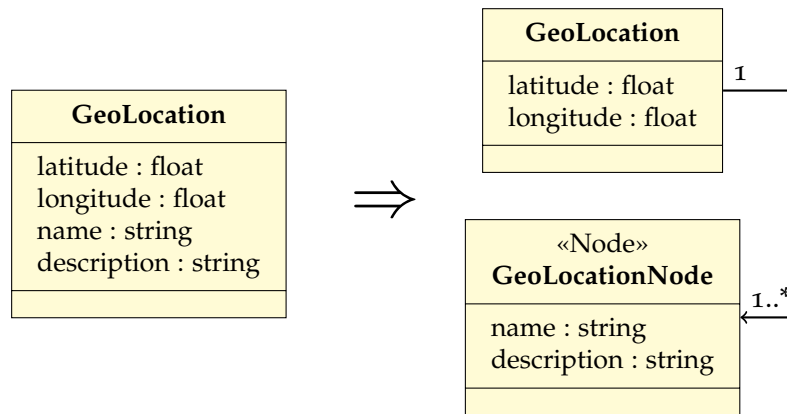


Abbildung 5. Auslagerung des übersetzbaren Teils einer Klasse in einen Neos Node am Beispiel der Klasse GeoLocation

Nachteilig an diesem Vorgehen ist die Art, in welcher Nodes in der Datenbank abgespeichert werden. Da Nodes vom Prinzip her generisch angelegt sind, werden die Attribute eines Nodes nicht in einer Klasse, sondern in externen Konfigurationsdateien definiert [The15c, S. 16].²⁵ Dies führt allerdings dazu, dass dem ORM-Framework die Attribute eines speziellen Nodes nicht bekannt sind, sie werden daher als JSON kodiert und in der Datenbank abgespeichert.²⁶ Im Umkehrschluss bedeutet dies, dass das ORM auch beim Zugriff auf die Inhalte des Nodes eingeschränkt ist: Funktionen zur Filterung oder Sortierung, welche vom ORM für normale PHP-Objekte angeboten werden, können im Falle von Nodes nicht auf die als JSON gespeicherten Daten zugreifen. Entsprechend müssten diese Funktionen außerhalb der Datenbank nachimplementiert werden.²⁷

2.4.2 Ersetzung einzelner Attribute durch übersetzbare Objekte

Alternativ kann jedes einzelne Attribut, welches übersetzbar sein soll, in ein eigenes Objekt ausgelagert werden. Dieses Objekt stellt einen Container dar, welcher für einzelne Sprachen eine Variante des Attributs beinhaltet. Eine mögliche Modellierung hiervon zeigt die Grafik 6.

Da alle Daten innerhalb von PHP-Objekten gespeichert werden, kann im Gegensatz zum Ansatz aus Abschnitt 2.4.1 das ORM-Framework zur Filterung und Sortierung nach Attributen in der jeweiligen Sprachvariante genutzt werden.

Allerdings führt dieses Vorgehen dazu, dass die automatische Validierung von Flow nicht länger möglich ist. Durch Annotationen am entsprechenden Attribut können Validierungsregeln wie Minimal- und Maximallänge für Strings definiert werden [The15b, S. 99]. Wird ein Objekt einer derart annotierten Klasse als Argument an einen Controller des Flow Frameworks weitergegeben, so wird die Validierung automatisch angestoßen [The15b, S. 194]. Ein Beispiel hierfür zeigt Listing 2.1.

²⁵ eine entsprechende Definition in einer Konfigurationsdatei heißt *NodeType*

²⁶ vgl. <https://github.com/neos/typo3cr/blob/d4dd8bcab165b7bd7248093e444f89ce49a751a0/Classes/TYP03/TYP03CR/Domain/Model/AbstractNodeData.php#L35>, Zeile 35

²⁷ Es sei darauf hingewiesen, dass gängige Datenbanksysteme wie MySQL, PostgreSQL und MSSQL einen Datentyp für JSON Dokumente anbieten. Dieser wird allerdings lediglich im Falle von PostgreSQL durch Doctrine verwendet [The, vgl. 8.2 Mapping Types].

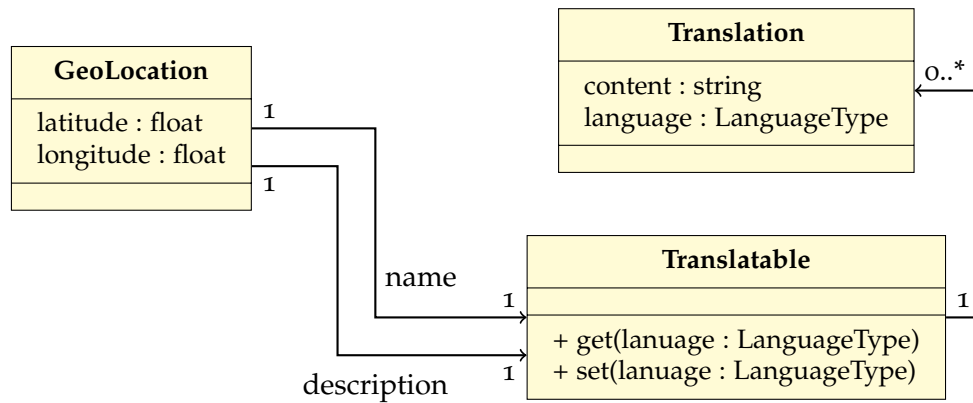


Abbildung 6. Transferierung von übersetzbaren Attributen in eine Translatable Klasse, welche in Abhängigkeit von der übergeben Sprache auf eine Translation zugreift

```

1  /**
2  * @Flow\Validate(type="StringLength", options={"maximum"=500 })
3  * @var string
4  */
5  protected $eventName;
  
```

Listing 2.1. Annotation der Property `$eventName` mit einem Validator für Textlänge

Die StringLength-Validierung kann nicht länger am Attribut angebracht werden, da es statt vom Typ `string` nun vom Typ `Translatable` ist. Auch in der `Translation`-Klasse ist die Annotation fehl am Platz, da nicht zwangsweise für jede übersetzbare Eigenschaft die gleichen Einschränkungen gelten müssen.²⁸

Um die automatische Validierung wiederherzustellen, wird eine Subklasse des `\TYPO3\Flow\Validation\Validator\AbstractValidator` benötigt, welcher in der Lage ist, den Inhalt der `Translation`-Objekte auszulesen und diesen an die originale Flow-Validierung weiterzureichen [The15b, S. 196].

Zudem stellt das Modell aus Abbildung 6 keine Konsistenz der Übersetzungen sicher: Das Modell erlaubt es beispielsweise, dass eine französische Variante des Namens vorliegt, während eine entsprechende Übersetzung der Beschreibung fehlen könnte.²⁹ Diese Einschränkung müsste somit in der Ebene der *business logic* sichergestellt werden.

2.4.3 Trennung von übersetzbarem und statischen Inhalt im Domänenmodell

Die in diesem Abschnitt 2.4.3 vorgestellten Konzepte wurden von Mitarbeitern der Firma response ausgearbeitet und werden an dieser Stelle lediglich als Hintergrundinformation mit angegeben.

Ein dritter Ansatz greift das Node-Konzept von Neos auf: Jede Klasse, welche übersetzbaren Content enthält, wird wie in Abschnitt 2.4.1 zweigeteilt. Eine Klasse beinhaltet Re-

²⁸ Beispielsweise erhält der Name einer `GeoLocation` eine Maximallänge von 500, während die Beschreibung bis zu 4000 Zeichen umfassen kann

²⁹ „Name“ und „Beschreibung“ beziehen sich in diesem Fall auf die Attribute `$name` und `$description` der Klasse `GeoLocation` aus Abbildung 6

lationen zu anderen Klassen und nicht übersetzbare Attribute, eine zweite *Content*-Klasse enthält alle übersetzbaren Eigenschaften.

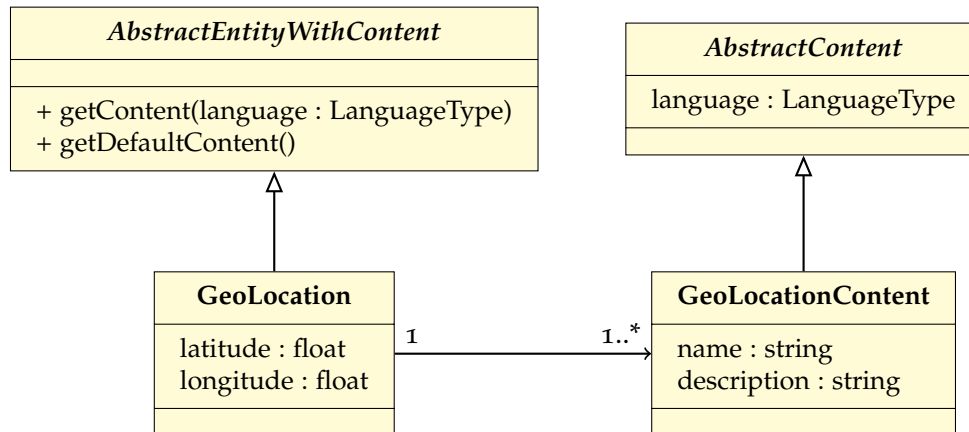


Abbildung 7. Auslagerung des übersetzbaren Teils einer Klasse in einen Neos Node am Beispiel der Klasse *GeoLocation*

Dies kommt einer Nachentwicklung des Node-Konzepts aus Abschnitt 2.4.1 gleich – da allerdings alle übersetzbaren Attribute in PHP-Klassen enthalten sind, kann die annotationsbasierte Validierung von Flow verwendet werden.

Um häufig verwendete Funktionen wie etwa den Zugriff auf das *Content*-Objekt für eine spezifische Sprache nicht mehrfach implementieren zu müssen, erben alle Entities von einer *AbstractEntityWithContent*, welche wiederum auf Kindklassen der *AbstractContent* operiert.

Allerdings führt dies zu einer doppelten Anzahl an zu pflegenden Klassen, da jeweils sowohl die „normale“ als auch die Content-Klasse erstellt werden muss.

Problematisch an dieser Konstruktion ist zudem die starke Abhängigkeit aller Entity-Klassen von der Superklasse *AbstractEntityWithContent*. Sollten an der Vaterklasse Änderungen notwendig sein, könnte dies auch unerwünschte Auswirkungen auf die Kindklassen haben. Da die Vererbung in erster Linie zur Vermeidung von Coderedundanz verwendet wird, könnte sie durch Objektkomposition ersetzt werden [BL11, S. 506f].

Des Weiteren kann eine derart allgemeine Vaterklasse auch die Vererbungshierarchie stören. Beispielsweise tritt ein Problem auf, wenn die *GeoLocation* von einer *Location*-Klasse einer Fremdbibliothek erben soll. Da die Fremdbibliotheksklasse nicht von der *AbstractEntityWithContent* erbt und PHP keine Mehrfachvererbung unterstützt, ist eine solche Struktur nicht möglich.

Seit PHP 5.4 existiert mit *Traits* eine Möglichkeit, sich wiederholenden Code ohne Vererbung in beliebige Klassen zu injizieren. Somit ließen sich die zuletzt genannten Kritikpunkte umgehen oder abschwächen. Traits stellen eine Summe von Methoden und Attributen bereit, welche mittels `use` Statement in eine Klasse eingebunden werden können [Loc15, S. 17-19].³⁰ Traits werden derzeit in ReEvent nicht verwendet.

Nach aktuellem Stand wird in ReEvent das in diesem Abschnitt 2.4.3 vorgestellte Konzept genutzt, insbesondere da die automatische Validierung von Flow auf diesem Wege verfügbar ist.

³⁰ DRY-Prinzip oder *Don't repeat yourself* stellt ein Prinzip der Softwareentwicklung dar, nach welchem Codeduplizierung zu vermeiden ist [MCo9, S. 289f]

2.5 ZEITPUNKTABHÄNGIGE DATEN

Die einzige Konstante im Universum ist die Veränderung.

– Heraklit von Ephesos

Dinge – und somit auch Daten – sind ständiger Veränderung unterworfen. Firmen wechseln ihren Standort, Kontakte erhalten eine neue Telefonnummer oder Zuständigkeiten ändern sich. Beispielsweise wurde die Technische Hochschule Ingolstadt seit ihrer Gründung im Jahr 1994 bis 2015 dreimal umbenannt.

1994	Fachhochschule Ingolstadt
2008	Hochschule für angewandte Wissenschaften FH Ingolstadt
2012	Hochschule für angewandte Wissenschaften Ingolstadt
2013	Technische Hochschule Ingolstadt

Tabelle 1. Zeitleiste der TH Ingolstadt von der Gründung bis zum Jahr 2015

Angenommen, in ReEvent wurde im Jahr 2012 die Veranstaltung A an der „HAW Ingolstadt“ eingetragen. Ein Jahr später findet eine weitere Veranstaltung B statt, inzwischen wurde der Titel „Technische Hochschule“ verliehen. Für den ReEvent-Nutzer, welcher die Veranstaltung B einpflegt, bleiben folgende Möglichkeiten.³¹

- Umbenennung der Hochschule in „Technische Hochschule Ingolstadt“
- Anlegen eines neuen Veranstaltungsortes „Technische Hochschule Ingolstadt“

Im Falle der Umbenennung ändert sich auch der Veranstaltungsort für die Veranstaltung A. Das Anlegen eines neuen Ortes führt dazu, dass zwei – eigentlich identische – Entitäten in der Datenbank gespeichert werden, welche dasselbe repräsentieren.

Die folgenden, in diesem Abschnitt 2.5 vorgestellten Konzepte wurden von Mitarbeitern der Firma response ausgearbeitet und werden an dieser Stelle lediglich als Hintergrundinformation mit angegeben.

ReEvent erlaubt eine weitere Lösung für dieses Problem durch Ausnutzung der *AbstractEntityWithContent* aus Abschnitt 2.4.3. Da alle Entitäten von dieser Klasse erben, wird folgende Konstruktion aus Abbildung 8 ermöglicht.

Ergibt sich an einer Entität eine Änderung zu einem gewissen Zeitpunkt, so wird aus Benutzersicht ein neuer *historischer Schritt*³² angelegt. Auf Modellebene wird vom bestehenden Objekt X ein Duplikat Y erzeugt. X erhält Y als successor und wird selbst predecessor von Y. Als successor von Y wird der ehemalige successor von X eingefügt. Zuletzt erhält Y ein Datum für `$validFrom`, welcher den Zeitpunkt der Änderung markiert. Dies entspricht einer doppelt verketteten Liste [OW12, S. 39].

³¹ Hinweis: Da ReEvent zum aktuellen Zeitpunkt noch nicht fertiggestellt ist, ist auch das angegebene Beispiel lediglich exemplarisch.

³² Eigenbezeichnung innerhalb des Entwicklerteams

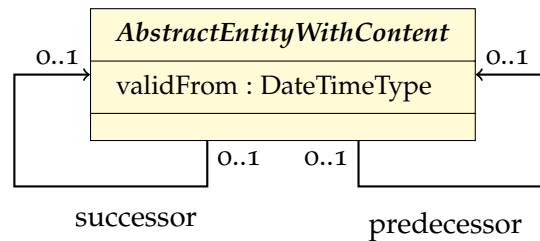


Abbildung 8. Realisierung von *HistorySteps* an der Klasse *AbstractEntityWithContent*

Im Beispiel aus Tabelle 1 liegen zwei Veranstaltungen vor, welche zu unterschiedlichen Zeitpunkten am selben Ort stattfinden. Zwischen beiden Veranstaltungen wird für den Veranstaltungsort – die TH Ingolstadt – ein neuer Name vergeben. Entsprechend muss für die Hochschule an dieser Stelle ein historischer Schritt angelegt werden. Beide Veranstaltungen referenzieren den Kopf der dadurch entstehenden Liste. Somit ist es anhand des Termins der Veranstaltung und der *validFrom*-Zeitstempel möglich, den zur Zeit der Veranstaltung korrekten Namen der Hochschule anzuzeigen.

Dieses Konstrukt ist explizit nicht für die Behebung von Fehleingaben gedacht, sondern nur für reale Änderungen. Wurde beispielsweise für eine Veranstaltung ein falscher Veranstaltungsort korrigiert, ist dies kein gewollter Anwendungsfall für *historische Schritte*. Hat sich der Veranstaltungsort nach der erstmaligen Eintragung jedoch geändert – womöglich fällt der ursprüngliche Ort aufgrund eines technischen Defekts oder einer Doppelbuchung aus – so kann diese Änderung über dieses Konzept abgebildet werden.

Mit dem neuen SQL Standard SQL:2011 wurden sogenannte *temporale Tabellen* eingeführt – welche wiederum exakt die hier beschriebene Problematik lösen soll. Hierzu erhält die betroffene Datenbanktabelle zwei Spalten vom Typ *date* oder *timestamp*, welche Beginn und Ende der Gültigkeit kennzeichnen. Mit **PERIOD FOR a_period (start, end)** wird eine virtuelle Spalte für die temporale Tabelle angelegt. Dies ermöglicht es, mit

FOR PORTION OF a_period FROM DATE '01.01.2014' TO DATE '01.01.2015'

beispielsweise **UPDATE**-Anweisungen zu schreiben, die bei überlappenden Zeiträumen automatisch neue Zeilen anlegen und umgebende Zeiträume anpassen [KM12, S. 35f].

Im Endeffekt erlaubt es dieser Standard, zeitpunktabhängige Daten abzuspeichern – was mit ähnlicher Tabellenstruktur auch schon zuvor möglich gewesen wäre, jedoch mehrere und komplexere Anfragen benötigt hätte.

Wie in Abschnitt 2.3 beschrieben, verwendet ReEvent Doctrine als ORM-Lösung. Doctrine generiert sowohl die Tabellenstruktur als auch die Anfragen an die Datenbank zum Einfügen oder Auslesen von Daten. Da Doctrine diesen Standard derzeit nicht unterstützt, müsste man somit manuell sowohl in die Migrationsskripte als auch in die Datenbankanfragen eingreifen.

Eine Suche nach den Begriffen „temporal“ oder „SQL 2011“ im Issue-Tracker von Doctrine^{33,34} ergab keine Treffer, scheinbar ist eine Implementierung dieses Features derzeit nicht geplant.

Umsetzung
auf Daten-
bankebene:
Der SQL
Standard
SQL:2011

³³ Stand 29.12.2015

³⁴ Doctrine verwendet den Issue-Tracker von github, <https://github.com/doctrine>

Außerdem unterstützen derzeit nur sehr wenige RDBMS diesen Standard: Neben Oracle, und DB2 liegt für PostgreSQL eine Extension vor, durch welche die Funktionalität integriert werden kann.³⁵

2.6 WORKSPACES

Die in diesem Abschnitt 2.6 vorgestellten Konzepte wurden von Mitarbeitern der Firma response ausgearbeitet und werden an dieser Stelle lediglich als Hintergrundinformation mit angegeben.

Eine weitere generische Funktion von ReEvent – welche in ihren Grundzügen aus dem Content Management System Neos stammt – stellen die *Workspaces* dar.

Änderungen oder die Beschreibung von Veranstaltungen können durchaus umfangreich sein. Möglicherweise liegen dem ReEvent-Benutzer zum Zeitpunkt, an dem mit dem Eintragen einer bestimmten Veranstaltung begonnen wurde noch alle benötigten Informationen vor. Der Benutzer kann die unvollständige Arbeit nicht abspeichern, da sie sonst veröffentlicht werden würde. Daher müsste der Benutzer die begonnene Arbeit verwerfen und zu einem späteren Zeitpunkt wiederholen.

In ReEvent wird von zu bearbeitenden Veranstaltungen – oder auch anderen Objekten – eine Kopie angelegt und in einem *privaten Workspace* gespeichert. Änderungen an diesem Objekt sind fortan nur für diesen Benutzer sichtbar. Er kann also Änderungen beliebig unterbrechen und fortsetzen, ohne dass andere ReEvent-Benutzer oder Frontend-Benutzer beeinflusst werden, da seine Änderungen immer nur die Kopie betreffen. Sobald die Arbeiten abgeschlossen sind, können die gemachten Änderungen in den *globalen Workspace* überführt werden: Sie sind nun auch wieder für Andere sichtbar.

Bis zu diesem Punkt stellt die Implementierung der Workspaces keinen größeren Aufwand dar. Probleme treten jedoch auf, wenn mehrere Benutzer gleichzeitig Arbeiten am selben Objekt durchführen.

- Objekt X liegt im globalen Workspace
- Benutzer A bearbeitet Objekt X, X wird zum privaten Workspace von Benutzer A kopiert
- Benutzer B bearbeitet Objekt X, X wird zum privaten Workspace von Benutzer B kopiert
- Benutzer A schreibt geändertes Objekt X zurück in den globalen Workspace
- Benutzer B schreibt geändertes Objekt X zurück in den globalen Workspace

In diesem Beispiel würden die Änderungen von Benutzer A verloren gehen, ohne dass Benutzer B die Möglichkeit hat, sie mit seinen Änderungen zu vergleichen und gegebenenfalls zu übernehmen. Dieses Concurrency-Problem wird als *Lost Update* bezeichnet [BG81, S. 186f]. Um solche Fälle zu behandeln, existieren unterschiedliche Ansätze, welche in Anhang C diskutiert werden.

ReEvent setzt hierbei auf ein optimistisches Sperrverfahren.

³⁵ vgl. http://pgxn.org/dist/temporal_tables/ oder https://github.com/arkhipov/temporal_tables

Seit Neos 2.1 ist es möglich, verschiedene Versionen einer Seite über Workspaces hinweg über eine externe Diff-Library³⁶ zu vergleichen. Hierbei werden Änderungen – ähnlich wie in einem lokal installierten Diff-Tool eines Versionskontrollsystems – in einer „Side-By-Side“ Ansicht angezeigt. Derzeit wird überprüft, ob sich diese auf dem Python-Package *difflib* basierende Bibliothek auch für Objekte in ReEvent verwenden lässt.

Konfliktbehebung wie in Versionskontrollsystemen

³⁶ vergleiche <https://github.com/neos/neos-development-collection/tree/master/Neos.Diff>

3

ANFORDERUNGSANALYSE

Anforderungsanalyse wird von [Bal09, S. 598] als „systematische Ermittlung, Beschreibung, Modellierung und Analyse von Anforderungen“ definiert – dieses Kapitel soll aufzeigen, welche Anforderungen an ein Suchsystem gestellt werden. Neben der Motivation für die Sammlung der Requirements wird primär das Vorgehen hierzu erläutert, beispielsweise welche verschiedenen Anforderungsquellen beachtet werden.

3.1 MOTIVATION

Zu Beginn der Suche nach Anforderungen steht die Frage, welchen Mehrwert *Requirements Engineering* für das Projekt bedeutet.

[Poh08, S.7] zählt Requirements Engineering zu den zwingenden Prämissen für erfolgreiche Softwareprojekte.

Eine Studie zum Thema „Erfolg und Scheitern im Projektmanagement“ der GPM¹ untersuchte, welche Faktoren den größten Einfluss auf den Erfolg eines Projektes haben. Hierzu wurden 79 Unternehmen aus unterschiedlichen Branchen befragt. „Unklare Anforderungen und Ziele“ wurden in den Studienergebnissen als zweithäufigste Ursache für gescheiterte Projekte genannt [GPo8, S. 3, 8].

Schlussendlich ist klar: Ohne zu wissen, was eine Search Engine für ReEvent leisten soll, ist es auch nicht möglich eine Evaluation durchzuführen oder eine Empfehlung abzugeben.

Weshalb ist Requirements Engineering von Bedeutung für die Auswahl der Suchlösung?

3.2 KLASSIFIZIERUNG VON ANFORDERUNGEN

Anforderungen werden gängigerweise in *funktionale* und *nicht funktionale*² unterschieden. Funktional ist eine Anforderung dann, wenn sie sich auf eine Funktion oder ein Verhalten bezieht, welche das System leisten soll. Nicht-Funktionale Anforderungen stellen Qualitätsmerkmale dar – übliche Punkte sind hier Ausfallsicherheit oder Performanz. Zusätzlich können auch *Randbedingungen* vorliegen, die zwar vom das System eingehalten werden müssen, aber keine Funktion oder Qualitätsfaktoren darstellen [PR15, S. 8f]. Beispiele hierfür sind die Kostenplanung oder gesetzliche Einschränkungen.

Eine weitere Klassifizierung, welche die Kunden- oder Benutzerzufriedenheit im Blick hat, stellt das Kano-Modell dar. *Basismerkmale* sind Funktionen, welche vom Produkt erwartet werden, etwa weil diese Funktion marktüblicher Standard sind. Systeme im Betrieb wie in Abschnitt 3.3.3 können als Quelle für diese Anforderung dienen. Was der Kunde

¹ GPM Deutsche Gesellschaft für Projektmanagement e. V.

² [Poh08, S. 16] nennt nicht funktionale Anforderungen auch Qualitätsanforderungen, beziehungsweise unterteilt nicht funktionale Anforderungen in funktionale und Qualitätsanforderungen

explizit fordert – möglicherweise in der Form eines Lastenheftes – wird unter *Zusatzmerkmale* zusammengefasst. Systemeigenschaften, die über das hinausgehen, was der Kunde erwartet stellen *Begeisterungsmerkmale* dar. Während Basismerkmale für das Produkt zwingend umgesetzt werden müssen, steigert die Umsetzung von Begeisterungsmerkmalen die Kundenzufriedenheit noch stärker als die der Zusatzmerkmale [Poho8, S. 534].

Die grundlegende Suchfunktion – Abbildung einer Suchanfrage auf eine Treffermenge, die die Suchanfrage enthalten – kann als Basismerkmal einer Suchmaschine aufgefasst werden. Unter Beachtung von Abschnitt 3.3.4 und 3.3.3 ist die Filterung von Veranstaltungen nach Datum ein Zusatzmerkmal. *Natural Language Searching*³ – die semantische Analyse eines Suchbegriffes und dem Versuch die Intention des Benutzers herauszulesen – könnte in diesem Zusammenhang als Begeisterungsmerkmal bezeichnet werden [Aks11, S. 6]. So könnten aus der Anfrage „Sportveranstaltung in der Saturn-Arena“ automatisch Filterkriterien für die Veranstaltungskategorie(Sport) und den Ort(Saturn-Arena) gewonnen werden.

3.3 ANFORDERUNGSQUELLEN

Um Anforderungen für ein spezifisches Projekt zu ermitteln, müssen verschiedene Aspekte des Projektumfeldes beachtet werden, welche das geplante System beeinflussen könnten.

3.3.1 Identifikation der Stakeholder

Einzelpersonen oder Gruppen, welche vom Projekt betroffen sind oder auf dieses Einfluss ausüben, werden *Stakeholder* genannt [PR15, S. 21].

Veranstaltungsbesucher

Veranstaltungsbesucher – ob potentiell oder nicht – suchen nach Veranstaltungen, für die sie ein Interesse hegen. Eine Recherche starten sie, weil sie eine bestimmte, kommende Veranstaltung besuchen möchten, Informationen für eine vergangene Veranstaltung einholen möchten – beispielsweise Kontaktdaten der Organisatoren – oder einfach ins Blaue hinein nach Events für die zugehörige Zielgruppe suchen.

Es wird angenommen, dass diese Gruppe verhältnismäßig den größten Anteil der Suchanfragen durchführt. In Abhängigkeit des Recherchebedürfnisses und des Interesses an Veranstaltungen steigt oder fällt dadurch auch die Menge der Suchanfragen – somit auch die Auslastung des Suchservers.

Ein mögliches Feature könnte die Umkreissuche sein: Veranstaltungen wird in ReEvent ein Veranstaltungsort zugewiesen, der üblicherweise über GPS-Koordinaten verfügen kann⁴. Anhand des ermittelten, aktuellen Standortes des Benutzers oder durch einer vom Benutzer gewählten GPS-Position und eines Maximalabstandes können Veranstaltungen

„Welche Veranstaltungen finden in meiner Nähe statt?“

³ ein Teilbereich der Computerlinguistik

⁴ Eine Ausnahme hiervon stellen beispielsweise sogenannte *Webinare* dar. Das Kofferwort aus Web und Seminar beschreibt nach Duden ein „online stattfindendes Seminar“, der Veranstaltungsort wird durch eine URL definiert.

gefiltert werden, die nicht in Reichweite des Benutzers liegen. Der Fachbegriff hierfür lautet *geospatial search* [GP14, S. 521].

Schließlich ist es grundlegendes Ziel jedes Veranstaltungsbesuchers, bei der Suche Veranstaltungen zu finden, die seinen Interessen – und damit auch seiner Suchanfrage – entsprechen. Außerdem liegt es nahe, den Präferenzen des Nutzers entsprechend einzelne Veranstaltungen vorzuschlagen, die der Benutzer andernfalls verpassen könnte. Um die User Experience nicht zu beeinträchtigen, sollten Anfragen an die Suchmaschine nicht um ein Vielfaches länger dauern als andere Seitenaufrufe auf diesem Server.

Veranstalter

Auch Organisatoren von Veranstaltungen können ReEvent durchsuchen. In erster Linie werden die eigenen Veranstaltungen Ziel der Suche sein, da der öffentliche Eintrag in ReEvent eine Informationsquelle für Besucher darstellt. Falls Korrekturen an diesem Eintrag notwendig werden und der Veranstalter überprüfen möchte, ob diese bereits durchgeführt wurden, könnte er diese Veranstaltung über die Suche aufrufen.

Ebenfalls im Blickpunkt eines Veranstalters sind *ähnliche* Veranstaltungen zu den Eigenen. Um das Publikum für die Veranstaltung nicht aufzuteilen, könnte es im Sinne des Veranstalters sein, eigene Veranstaltungstermine nicht gleichzeitig zu ähnlichen Konkurrenzveranstaltungen zu legen. Führen beispielsweise zwei Zirkusunternehmen gleichzeitig Vorstellungen am selben Ort auf, so würde sich der durchschnittliche Besucher für eines der beiden entscheiden.

Einen weiteren Berührungspunkt mit dem Suchsystem haben Veranstalter in Bezug auf die Anzahl der Veranstaltungen. Je nachdem wie viele Veranstaltungen durchgeführt werden, muss das Suchsystem auch eine andere Menge an Veranstaltungen handhaben.

Editoren

Ähnlich wie auch Veranstalter könnten auch Backend-Nutzer von ReEvent Events im Fokus haben, welche sie persönlich erstellt oder editiert haben. Insbesondere um die Auswirkungen einer Änderung im Frontend zu überprüfen, wäre eine Filterung nach „zuletzt bearbeitet von“ sinnvoll.⁵

Sowohl Änderungen als auch Neueintragungen sollten daher sofort oder sobald wie möglich von der Suche erfasst werden.

3.3.2 Bestimmung des Systemkontextes

[Rup14, S. 85-87] teilt die Umgebung eines Projektes in drei Bereiche ein: den *Scope* des zu erstellenden Systems, den *Systemkontext* und den *irrelevanten* Teil der Umgebung. Der Systemkontext entspricht denjenigen Teil des Umfeldes, welcher für die Erstellung und Interpretation des Projektes von Bedeutung ist [Poho8, S. 55]. Währenddessen wird die irrelevante Umgebung – wie der Name bereits vermuten lässt – im weiteren Verlauf des Requirements Engineering ausgeblendet [Bal09, S. 462].

Der Systemkontext soll gemäß [Bal09, S. 461] festgelegt werden, da die in ihm enthaltenen Objekte Einfluss auf Anforderungen an das eigentliche System nehmen können.

⁵ alternativ sollte im Backend für jede Veranstaltung eine Verlinkung für die entsprechende Seite im Frontend enthalten sein

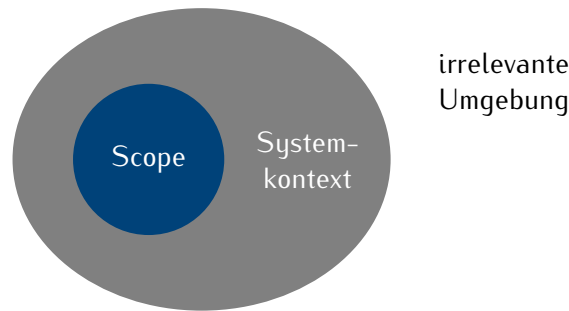


Abbildung 9. Scope, Systemkontext und irrelevante Umgebung nach [Rup14, S. 86]

ReEvent

Mit ReEvent als Quelle für die zu durchsuchenden Daten steht bereits ein Teil des Systemkontextes fest. Entsprechend können die Inhalte des Kapitels 2 wieder aufgegriffen und hinsichtlich möglichen Anforderungen untersucht werden. Die eingetragenen Veranstaltungen enthalten primär Textfelder, wie den Namen der Veranstaltung sowie Beschreibungstexte. Hierbei ist es möglich, dass insbesondere Beschreibungsfelder eine Auszeichnungssprache wie HTML oder Markdown enthalten. Die primäre Aufgabe des Suchsystems soll eine Volltextsuche über diese Daten sein. Des Weiteren können zusätzliche Dateien angefügt werden.⁶ Diese können unterteilt werden in textbasierte Formate und solche ohne Textinformationen. Zur ersten Kategorie kann man entsprechende PDF- oder Microsoft Word-Dokumente zählen, aber auch andere Formate wie HTML oder XML sind prinzipiell maschinenlesbar. Möglicherweise müssen diese Dateien jedoch aufbereitet werden. Ebenfalls möglich ist die Abspeicherung weiterer Dateitypen, die nicht in erster Linie textbasiert sind. Hier kommen in erster Linie Medien wie Bilder, Sound oder Videos in Betracht. Diese verfügen aber meist über Metadaten – so können MP3-Dateien über sogenannte *ID3-Tags* mit Informationen zu Künstler, Album und Liedtitel angereichert werden.

Durchsuchung von Anhängen im PDF- oder Word-Format

Insbesondere bei Bilddateien kann man noch einen Schritt weitergehen: durch Verfahren der *Optical Character Recognition* können aus diesen wiederum Textinformationen gewonnen werden. Enthält das Bild beispielsweise ein Namensschild oder stellt es eine Aufnahme einer Speisekarte dar, so besteht mittels OCR die Möglichkeit, dass diese Informationen aus der Grafik ausgelesen und anschließend für die Suche bereitgestellt werden können.

Die Oberfläche von ReEvent – sowohl im Backend als auch im Frontend – kann lokalisiert werden. Das Suchformular sowie die Auflistung der Suchergebnisse sollten ebenfalls entsprechend angepasst werden können.

Ebenfalls ein Kriterium stellen die mehrsprachig eingetragenen Texte dar: Wie in Abschnitt 2.4 gezeigt, können Veranstaltungen übersetzt werden. Das Suchsystem muss somit in der Lage sein, Varianten ein und derselben Veranstaltung, aber in einer anderen Sprache zu indizieren und zu durchsuchen. In erster Linie werden Veranstaltungen der Stadt Ingolstadt in deutscher Sprache abgefasst werden, mit einem gewissen Anteil ins Englische übertragener Events ist zu rechnen. Da die Stadt Ingolstadt Städtepartnerschaften unter anderem zu Moskau(Russland), Legmoin(Burkina Faso) und Foshan(China) pflegt, kommen auch die zugehörigen Landessprachen in Betracht [Sta14a]. Insbesondere das Chinesische als

Mehrsprachige Inhalte

⁶ vergleiche Abschnitt 2.1

CJK Sprache könnte eine potentielle Schwierigkeit für ein Suchsystem darstellen [BCC10, S. 95f].

Auch das Konzept der Workspaces stellt eine Herausforderung für ein Suchsystem dar. Da theoretisch für jeden Editor eine Instanz eines jeden Objektes existieren kann, beläuft sich die Gesamtsumme auf $n + 1$ Varianten bei n Editoren, auch der globale, beziehungsweise öffentliche Workspace muss dazu gezählt werden. Für diese Situation existieren zwei unterschiedliche Ansätze.

Mehrere Varianten durch Workspaces und zeitpunktabhängige Daten

- Suche für *alle* Workspaces
- Reduktion auf den globalen Workspace

Wenn alle Workspaces durchsucht werden sollen, kann der Aufwand hierfür linear mit der Anzahl der Editoren ansteigen. Alternativ kann man den Scope der Suche auch einschränken: so können lediglich veröffentlichte Objekte durchsucht werden – was aber den Vorteil hat, dass sich der Aufwand auf nur eine Variante dieses Objektes reduziert. Diese Entscheidung hat lediglich Auswirkungen auf das Backend, das Frontend hat schließlich nur Zugriff auf den öffentlichen Workspace. Im Backend könnten für die erste Variante sogar zwei Suchen angeboten werden: Eine für den privaten Workspace des angemeldeten Benutzers, sowie für den öffentlichen Workspace. Bei der zweiten Möglichkeit sind dem Editor die eigenen Änderungen über die Suche nicht sichtbar. Es stellt sich somit die Frage, ob der Mehrwert für den einzelnen Editor den zusätzlichen Aufwand rechtfertigt.

Schlussendlich sollen auch die historischen Schritte aus Abschnitt 2.5 für die Suche verfügbar sein. Hier liegt die grundlegende Problematik darin, dass jeder historischer Schritt eine kleine Abwandlung des Vorgängers darstellt. Würde man in der Suche jeden historischen Schritt getrennt voneinander betrachten, könnte folgendes Szenario auftreten.⁷

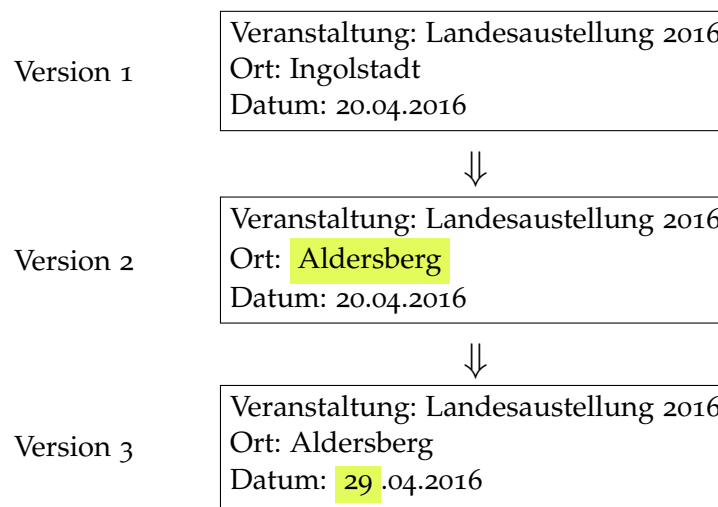


Abbildung 10. Historische Schritte am Beispiel einer Veranstaltung. Die jeweiligen Änderungen wurden gelb hinterlegt. Das Beispiel basiert lose auf der Vergabe der Landesaustellung 2016 mit dem Thema „Bier in Bayern“.

Die Landesaustellung 2016 stellt eine – mehr oder weniger – einzigartige Veranstaltung dar, zumindest alle drei historischen Schritte aus Abbildung 10 beziehen sich im Grunde

⁷ in Anlehnung an <http://www.augsburger-allgemeine.de/neuburg/So-haelt-Ingolstadt-dagegen-1d33881902.html>

auf ein und dieselbe Veranstaltung. Würde man nun bei einer Suche nach „Landesaustellung 2016“ diese historischen Schritte als eigenständige Objekte handhaben, so könnte die Ergebnisliste exakt diese drei Objekte als Treffer enthalten – obwohl sie dieselbe Veranstaltung darstellen. Auch kann man die Suche nicht auf die aktuellste Version beschränken: Würde ein Benutzer, der noch nichts vom neuen Veranstaltungsort erfahren hat nach einer Landesaustellung in Ingolstadt suchen, würde er keinen Treffer erzielen. Somit wäre der Vorteil der historischen Schritte – Nachvollziehbarkeit von realen Änderungen – hinfällig.

ReEvent selbst basiert auf *Neos CMS* und *Flow*. Zur primären Datenhaltung wird das RDBMS *MySQL* genutzt.⁸ Das Suchsystem muss die Daten aus diesem Kontext erhalten, möglicherweise muss hierfür ein entsprechendes Importskript für die Kopplung der Systeme entwickelt werden.

Sowohl ReEvent als auch die zugehörige Datenbank werden auf virtualisierten Maschinen im *Rechenzentrum* der Stadt Ingolstadt gehostet. Unter anderem werden hier die Betriebssysteme Windows Server 2008 und Debian verwendet. Sollte die Suchlösung auf einer eigenen Maschine innerhalb des Rechenzentrums installiert werden, müssen eventuell vorhandene Richtlinien des Rechenzentrums beachtet werden. Auch notwendige Netzwerkfreigaben⁹ sind in Abstimmung mit dem Rechenzentrum einzurichten.

Da eine Search Engine möglicherweise auch für andere Seiten und Applikationen der Stadt nutzbar ist, könnten hierdurch Synergieeffekte entstehen. Ein Beispiel hierfür stellt die Hauptseite der Stadt Ingolstadt¹⁰ dar, welche auf dem CMS IKISS der Firma Advantic basiert [Adv15].¹¹

3.3.3 Bestehende Systeme

Auch Vorgängersysteme oder Produkte der Konkurrenz können Quellen für Anforderungen darstellen [PR15, S. 21]. Benutzer sind an Funktionen dieser Software gewöhnt, sie erwarten somit ähnliche Funktionen und Funktionsweisen gegenüber dem neuen System. Zudem erleichtert es den Einstieg in die Arbeit mit dem neuen Produkt, wenn es sich ähnlich verhält wie vergleichbare Systeme.

INsite

Eine bedeutende Rolle für mögliche Anforderungen an das Suchsystem spielt auch das Altsystem INsite. Hierzu besteht die Möglichkeit, vorliegende Webserverlogs auszuwerten. Suchanfragen über das Frontend von INsite werden hierbei wie folgt geloggt:

```
1 0.0.0.0 - [13/Nov/2015:00:13:55 +0100] GET
   ↪ /relaunch/veranstaltungen.cfm?regionString=&[...]&searchPhrase=&submit=Senden
   ↪ HTTP/1.0 200 249401
```

Alle Filter – wie Beispielsweise `regionString` – werden als GET-Parameter an die URL angehängt. Im obigen beispielhaften Logeintrag wurden weitere Filter, welche üblicher-

⁸ Das Datenbanksystem ist nicht zwingend gesetzt, vergleiche Abschnitt 2.3

⁹ beispielsweise Portfreigaben

¹⁰ siehe <http://www2.ingolstadt.de/>

¹¹ IKISS bietet bereits eine Volltextsuche, möglicherweise basierend auf Apache Solr

Suchanfragen pro Tag.

Die Abbildung 11 zeigt die aus den Logs extrahierten Suchbegriffe. Die Schriftgröße stellt dabei ein Maß für die Häufigkeit einer Anfrage dar. Fehler bei der Groß- und Kleinschreibung oder weitergehende Rechtschreibfehler wurden nicht korrigiert. Dies wird an den prominentesten Begriffen *Flohmarkt* und *flohmarkt* sichtbar, welche jeweils 79 beziehungsweise 64 Suchanfragen repräsentieren.

Lediglich 15 Suchanfragen enthalten Sonderzeichen wie +, #, % oder *, welche in anderen Suchsystemen erweiterte Funktionen bereitstellen können – % und * werden beispielsweise als Wildcard-Operatoren verwendet [Wie09, S. 111]. Gängige boolesche Operatoren wie AND oder OR sind nicht enthalten. Da die INsite-Suche auf dem SQL-Operator **LIKE** basiert, ist hiervon lediglich das %-Zeichen von Relevanz. Die Verwendung von derartigen Sonderzeichen lässt auf ein entsprechendes Vorwissen um Volltextsuche schließen. Da der Anteil dieser Anfragen jedoch verschwindend gering ist, kann auch von einem geringen Prozentsatz vergleichsweise fortgeschrittener Benutzer ausgegangen werden.

Werden Operatoren genutzt?

Eine Untersuchung mit der Rechtschreibüberprüfung *hunspell* ergab einen Anteil von 69 % fehlerhaften Suchanfragen. Hierbei ist jedoch zu beachten, dass hiervon Groß- und Kleinschreibfehler sowie von *hunspell* nicht erkannte Eigennamen einen großen Teil beitragen. Dennoch sind auch zahlreiche Rechtschreibfehler enthalten. Dies lässt die Schlussfolgerung zu, dass eine automatische Korrektur von Rechtschreibfehlern einen Mehrwert für die Suche darstellt.

Weitergehende Untersuchung der Logeinträge zeigt, dass innerhalb eines Zeitintervalls von fünf Sekunden maximal vier Suchanfragen getätigt wurden.¹³ Für das Intervall wurden fünf Sekunden gewählt, da Anfragen an den Server – beziehungsweise die Suchmaschine – nicht länger dauern sollten. Somit stellt die Zahl von vier Suchanfragen die maximale, gleichzeitige Last an den Server dar. Angesichts dieser Werte – maximal vier Anfragen in kurzer Zeit und durchschnittlich 10,36 pro Tag – ist davon auszugehen, dass das neue Suchsystem keine hohen Lastspitzen abfangen muss. Deswegen stellt auch Skalierungsfähigkeit bezüglich der Suchanfragen kein zwingendes Kriterium dar. Für die Suche im Backend von INsite liegen keine Logeinträge vor.

Eine weitere Kennzahl, welche das Altsystem bereitstellt, ist die Anzahl an Veranstaltungen, welche Jahr für Jahr in INsite eingepflegt wurden. Einen Verlauf der jährlichen Neueintragungen zeigt das Diagramm in Abbildung 16, durchschnittlich wurden pro Jahr 2 995 Veranstaltungen eingepflegt, insgesamt enthält die Datenbank derzeit knapp 45 000 Einträge. Dies erlaubt eine Prognose für die Anzahl der Veranstaltungen in ReEvent. Für die neue Veranstaltungsverwaltung ist ein Webservice für die automatisierte Eingabe von Veranstaltungen geplant¹⁴, wodurch die Menge an Veranstaltungen steigen könnte. Außerdem verfügt ReEvent über die Möglichkeit, Dateianhänge¹⁵ zu Veranstaltungen hinzuzufügen, was das Volumen an durchsuchbaren Text pro Veranstaltung erhöhen könnte.

¹³ Diese Untersuchung wurde über ein Java-Programm durchgeführt, welches für jeden Zeitstempel aus den Logdateien überprüft, von wie vielen weiteren Zeitstempeln er innerhalb eines Zeitraums von 5 000 Millisekunden umgeben ist.

¹⁴ auch in INsite werden bereits automatisiert Veranstaltungen aus dem Ratsinformationssystem der Stadt Ingolstadt übernommen, jedoch durch einen aktiven Abruf eines Webservices des Fremdsystems

¹⁵ beispielsweise im PDF-Format, vergleiche 3.3.2

Outlook

Eine insbesondere im Büroalltag häufig genutzte Veranstaltungsverwaltung stellt die Kalenderfunktion von Microsoft Outlook dar.

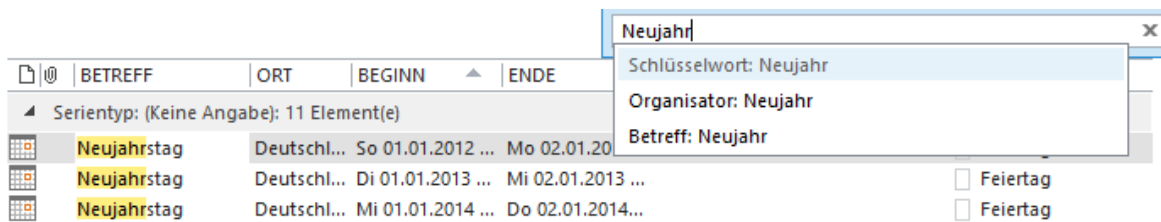



Abbildung 12. Screenshot der Suche des Outlook-Kalenders von Microsoft Outlook 2013, Desktopversion

Eine Funktion, welche beim Test der Suchfunktion auffällt, ist die automatische Aktualisierung der Trefferliste bei Veränderung des Suchbegriffs. Ohne die Suche aktiv starten zu müssen – zum Beispiel über einen entsprechenden Button – werden somit stets passende Veranstaltungen angezeigt. Dies könnte die notwendigen Tastaturanschläge bis zum gewünschten Suchergebnis minimieren.

Drückt man nach Eingabe eines Suchbegriffes die -Taste, so wird ein Dropdownmenü angezeigt. Die Auswahl eines der unteren Einträge führt zu einer dynamischen Erweiterung des Suchbegriffes: „Organisator: Neujahr“ ersetzt den Suchbegriff Neujahr durch organisator: (Neujahr). Während standardmäßig alle Eigenschaften von Veranstaltungen durchsucht werden, kann die Suche hierdurch auf eine Spalte – in diesem Beispiel den Organisator der Veranstaltung – eingegrenzt werden. Zudem ist es ebenfalls möglich, selbst entsprechende Anfragen zu erstellen und durch Eingabe von ort: (...) die Suche auf den Veranstaltungsort zu beschränken. Das Menü stellt somit eine Möglichkeit dar, ohne Vorwissen komplexere Suchanfragen zu erzeugen.

Weitere Tests ergeben, dass die Suche auch boolesche Operatoren wie AND und OR unterstützt, die allerdings Case-sensitiv sind. Während auf die Suchanfrage betreff: (Neujahr) AND ort: (Land) eine ähnliche Trefferliste wie in Abbildung 12 folgt, findet betreff: (Neujahr) and ort: (Land) keine einzige Veranstaltung.

Ebenfalls ins Auge sticht die Hervorhebung von Suchtreffern in der Auflistung der Veranstaltungen. Betrachtet man hierzu erneut den Screenshot 12, so fällt die entsprechende gelbe Hinterlegung auf.

Google

Wenn auch keine direkte Veranstaltungssuche, so ist die Google Websuche doch eine verständliche Assoziation zum Begriff der Suchmaschine im Allgemeinen. Weshalb in diesem Abschnitt Bezug auf Google und nicht auf eine andere Suchmaschine genommen wird, veranschaulicht die Abbildung 13.

In Anbetracht des Marktanteils von über 90 % kommt Google eine gewisse Vormachtstellung zu. Dies sollte es legitimieren, konkurrierende Suchmaschinen für den Zweck dieses Abschnittes – der Analyse von für den Benutzer gewohnten Referenzsystemen – zu vernachlässigen.

Auffällig bei der Websuche über Google ist zum einen die Autovervollständigung: Nach Eingabe der Zeichenkette „Ingols“ werden automatisch passende Vorschläge angezeigt,

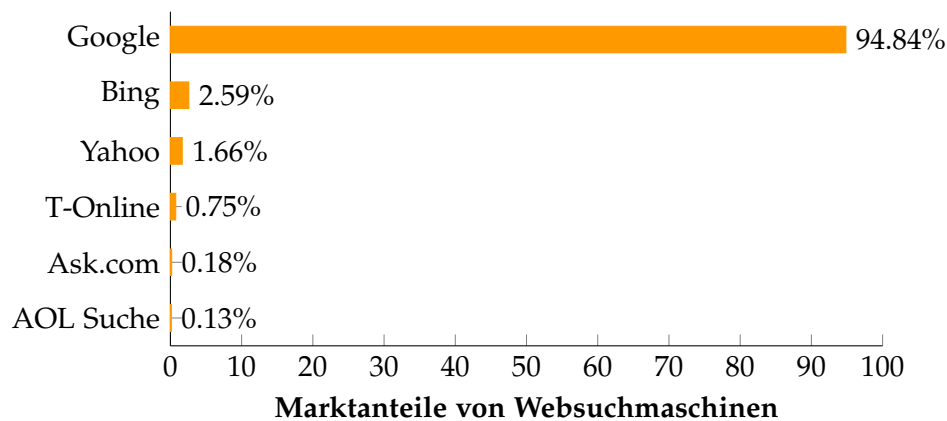


Abbildung 13. Marktanteile von Websuchmaschinen im Jahr 2015 nach [Sta14b, S. 31]

welche die Eingabe ergänzen – in diesem Beispiel unter anderem zu „ingolstadt“ und „ingolstadt today“.

Auch die automatische Rechtschreibkorrektur von Suchbegriffen zeigt eine mögliche Lösung der in Abschnitt 3.3.3 beschriebenen Probleme. Ein fehlerhafter Suchbegriff wie „Inglstadt“ führt zuerst zu einer Suche nach „ingolstadt“, Google weist auf den Fehler hin und bietet auch die Suche nach dem originalen – fehlerhaften – Suchbegriff an.

Automatische
Korrektur von
Eingabefehl-
ern

3.3.4 Dokumente

Auch vorliegende Dokumente können Basis für neue Anforderungen sein. [Rup14, S. 77] nennt hierfür Normen oder Gesetzestexte. Lastenheft und Pflichtenheft listen ebenfalls Anforderungen an das zu erstellende Produkt auf [Gra14, S. 48f].

Für ReEvent liegt derzeit kein Lasten- oder Pflichtenheft vor.¹⁶ Im Angebot¹⁷ wird spezifiziert: „Abbildung der bestehenden Funktionalität (INsite-Veranstaltungen)“. INsite wiederum bietet sowohl im Backend als auch im Frontend lediglich eine einfache SQL-basierte Suche an, welche um Filterkriterien ergänzt werden kann.¹⁸ Somit würde eine äquivalente SQL-basierte Suche bereits den vertraglichen Anforderungen genügen.

¹⁶ Stand 05.11.2015

¹⁷ Stand 14.07.2009

¹⁸ Vergleiche Abbildung 2 aus dem Abschnitt 1.2

4

SUCHSYSTEME – EINE MARKTÜBERSICHT

4.1 SQL-BASIERTE SUCHE

Bereits das Vorgängersystem INsite verfügt über die Möglichkeit, nach Veranstaltungen zu suchen und nach auszuwählenden Kriterien zu filtern.¹ Dies wird durch eine konfigurierbare und parametrisierbare SQL-Abfrage realisiert.

4.1.1 Filterung

Beispielsweise wird der Filter *Region* clientseitig durch eine Selectbox mit den Auswahlmöglichkeiten {*alle*, *IN*, *EI*, *ND*, ...} umgesetzt. Hierbei verursacht *alle* keine Filterung. Die weiteren Elemente stellen Abkürzungen für Regionen rund um den Raum Ingolstadt dar, nach welchen die Veranstaltungen entsprechend eingeschränkt werden können.²

```
1 <cfquery>
2 SELECT
3     Veranstaltungen.*
4 FROM
5     Veranstaltungen
6 <cfif FORM.region is not 'alle'>
7 WHERE
8     Veranstaltungen.region = <cfqueryparam value='#FORM.region#'>
9 </cfif>
10 </cfquery>
```

Listing 4.1. Die Filterung nach Region wird nur durchgeführt, falls nicht *alle* als Parameter übergeben wurde.

FORM.region stellt in Listing 4.1 eine ColdFusion-Variable dar, deren Wert aus dem entsprechenden HTML-Suchformular stammt. Lediglich wenn an dieser Stelle nicht der String *alle* übergeben wurde, wird die Filterung durchgeführt. In diesem Fall wird der SQL String um eine Einschränkung erweitert. `<cfqueryparam>` wird an dieser Stelle verwendet, um SQL-Injections zu verhindern [FAC10, S. 171f] – ähnlich wie beispielsweise *Prepared Statements* in Java.³

¹ Vergleiche Abbildung 2 aus dem Abschnitt 1.2

² IN: Ingolstadt, EI: Eichstätt, ND: Neuburg an der Donau

³ zur Erinnerung: INsite wurde unter Adobe ColdFusion entwickelt

4.1.2 Suche mit LIKE-Operator

Ebenso wird mit dem optionalen Parameter *Suchwort* verfahren. Im Vergleich zu obiger Filterung fallen jedoch zwei Unterschiede auf: Zum einen betrifft die Suche nicht notwendigerweise genau eine Spalte einer Datenbanktabelle. Wie Listing 4.2 zeigt, soll nicht nur der Name einer Veranstaltung durchsucht werden, auch die Spalten *Ort* und *Beschreibung* sind Texte, welche den Suchbegriff enthalten können. Da eine Veranstaltung bereits als Suchtreffer gelten soll, wenn lediglich eine dieser Spalten zum Suchbegriff passt, werden die einzelnen Bedingungen per logischem Oder verknüpft. Der andere wesentliche Unterschied betrifft die Art der Einschränkung. Bei der Filterung nach einer Region wurde durch den „=“ Operator überprüft, ob der Wert der Spalte exakt mit dem übergebenem Wert übereinstimmt. Dies ist bei einer Suche im Allgemeinen nicht erwünscht.

Stattdessen soll die Bedingung *Spalte enthält Suchwort* realisiert werden. Dies geschieht in SQL mittels **LIKE** Operator, welcher Wildcards unterstützt. Durch Einbettung des Suchbegriffes in %-Zeichen wird eine *enthält*-Bedingung erzeugt [Wie09, S. 110].

ID	Name	Ort
1	Stadtführung	Rathausplatz Ingolstadt
2	Mathematik-Prüfung	TH Ingolstadt
3	Stadtradeln	Ingolstadt
4	Dia-Vortrag von Prof. Dr. Mueller	Festsaal
5	Wanderung von Rom nach Florenz	Petersplatz, Rom

Tabelle 2. Beispieldatensätze von Veranstaltungen

Wird eine Abfrage `Veranstaltungen.ort = 'Ingolstadt'` auf die Tabelle 2 angewandt, so resultiert dies lediglich im Datensatz 3. Währenddessen liefert die Query `Veranstaltungen.ort LIKE '%Ingolstadt%'` die Veranstaltungen 1, 2 und 3.

Eine weitere Einschränkung liegt in der möglichen Case-Sensitivität von Datenbanken. Möglich deshalb, da dies von der verwendeten *Collation*⁴ der Datenbank abhängig ist. Diese gibt für eine Datenbank die Sortierreihenfolge von Inhalten vor und kann sich auch auf Einschränkungen wie den **LIKE**-Operator auswirken. Ist die Collation von der Groß- und Kleinschreibung abhängig, so wird dies üblicherweise durch ein CS im Namen gekennzeichnet. Andernfalls enthält der Name der Collation meist ein CI [Mic15a] [TW06, S. 185f]. Für eine case-sensitive Collation würde die Suchanfrage `Veranstaltungen.ort LIKE '%ingolstadt%'` keinen Treffer erzielen. Dies lässt sich durch Verwendung der SQL-Funktion **LOWER()** umgehen: Indem sowohl Suchbegriff als auch der Wert in der Spalte in Kleinbuchstaben umgewandelt werden, spielt die Case-Sensitivität der Datenbank an dieser Stelle keine Rolle mehr.

Die INsite Datenbank verwendet derzeit mit `SQL_Latin1_General_CP1_CI_AS` eine case-insensitive Collation, somit musste die INsite Suchquery nicht entsprechend angepasst werden. In diesem Zusammenhang ist auch der Namensbestandteil `_AS` von Interesse: dieser kennzeichnet die Collation als unterscheidend zwischen Lettern mit und ohne Akzenten. Die Zeichen a und á sind somit für diese Collation nicht identisch. Das Gegenteil hierzu stellt in MSSQL `_AI` dar [Mic15a].

⁴ englisch für „Sortierreihenfolge“


```

1 <cfquery>
2 SELECT
3     Veranstaltungen.*
4 FROM
5     Veranstaltungen
6 WHERE
7     Veranstaltungen.freigabe = 1
8 <cfif Len(FORM.search)>
9 AND (
10     Veranstaltungen.name LIKE <cfqueryparam value='%#FORM.search#%'>
11 OR
12     Veranstaltungen.beschreibung LIKE <cfqueryparam value='%#FORM.search#%'>
13 OR
14     Veranstaltungen.ort LIKE <cfqueryparam value='%#FORM.search#%'>
15 )
16 </cfif>
17 </cfquery>

```

Listing 4.2. Veranstaltungen werden im Vorgängersystem INsite mittels **LIKE** gefiltert.

Auch wenn die Suche nun nicht mehr von der Groß- und Kleinschreibung abhängig ist, werden dennoch nur solche Datensätze zur Treffermenge hinzugefügt, welche die einzelnen Zeichen des Suchbegriffes exakt in der angegebenen Reihenfolge enthalten. Möchte ein Benutzer beispielsweise die Tabelle 2 nach Veranstaltungen an der „Technische[n] Hochschule Ingolstadt“ durchsuchen, so wurde er mit dem bisherigen Verfahren keinen einzigen Treffer erzielen, da die Technische Hochschule zu TH abgekürzt wurde. Um trotzdem Suchtreffer zu erhalten, könnte der Suchbegriff vor der Konstruktion der SQL-Abfrage nach Leerzeichen aufgetrennt werden. „Technische Hochschule Ingolstadt“ wird hierdurch zu drei einzelnen Suchbegriffen {*Technische, Hochschule, Ingolstadt*}. Anschließend kann eine Suche nach jedem einzelnen Wort durchgeführt werden, beispielsweise durch eine **OR**-Verknüpfung oder eine Mengenvereinigung mittels **UNION**. Somit würde der Suchbegriff wieder zur Ergebnismenge {1,2,3} führen, da diese drei Datensätze „Ingolstadt“ enthalten. Auch der in erster Linie für die Suche relevante Datensatz mit der ID 2 wird als Suchtreffer aufgeführt [GP14, S. 52].

Eine solche Ausweitung der Suche bringt jedoch nicht nur Vorteile mit sich: Startet man mit diesem Verfahren eine Suchanfrage nach „Wilhelm von Humboldt“, so resultiert dies in den Ergebnissen {5,6}. Obwohl die Suchanfrage thematisch zu keiner der Veranstaltungen passt, enthalten diese beiden Datensätze ebenfalls das Wort „von“. Solche *Stopwort* genannten Satzteile sind häufig auftretende Wörter einer Sprache, wie beispielsweise Artikel. Professionelle Suchsysteme verfügen meist über Listen dieser Stopwörter, welche bei einer Suchanfrage ignoriert werden [BCC10, S. 89].

4.1.3 Eingebaute Volltextsuche

Diverse moderne relationale Datenbanksysteme verfügen über eine integrierte Volltextsuche – so auch das derzeit in ReEvent verwendete RDBMS MySQL. Diese ist allerdings

üblicherweise inaktiv und kann für eine oder mehrere Spalten einer Tabelle Volltextsuche aktiviert werden. Dies geschieht per FULLTEXT-Anweisung [Nix14, S. 191].

```
1 ALTER TABLE event ADD FULLTEXT(name,description);
```

Listing 4.3. Anlegen eines FULLTEXT Indexes unter MySQL

Anschließend können für die so indizierten Spalten Suchanfragen gestartet werden. Listing 4.4 zeigt die zugehörige Syntax: per **MATCH** werden die zu durchsuchenden Spalten angegeben, die zuvor in Listing 4.3 hierfür vorbereitet wurden. **AGAINST** enthält schließlich den Suchbegriff. Das optionale **IN BOOLEAN MODE** ermöglicht es, die Suchumfrage um Operatoren wie +, – oder " zu erweitern. Ein Pluszeichen vor einem Begriff macht diesen erforderlich, ein Minuszeichen markiert einen Begriff als Ausschlusskriterium. Zusammengehörende Begriffe, welche von Anführungszeichen umgeben sind, werden in exakt dieser Reihenfolge als Phrase gesucht [Nix14, S. 197f]. Neben der Unterstützung derartiger Operatoren nutzt die Volltextsuche auch Stopwortlisten [Nix14, S. 191].

```
1 SELECT * FROM event WHERE
2 MATCH(name,description) AGAINST('suchbegriff' IN BOOLEAN MODE);
```

Listing 4.4. Durchführung einer Volltextsuche unter MySQL

Die **MATCH()** **AGAINST()** Anweisung berechnet für den Suchbegriff eine Punktzahl auf Basis der Inhalte der angegebenen Spalten. Ist der Wert ungleich Null, so liegt ein Treffer vor. Ein höherer Wert impliziert zudem einen Treffer, der besser zum Suchbegriff passt. Somit lassen sich die Ergebnisse hiernach absteigend sortieren.

Für die integrierte Suche spricht – ebenso wie für die einfachere Suche mittels **LIKE** – dass kein zusätzliches System installiert und konfiguriert werden muss.

Ein Nachteil an der integrierten Volltextsuche ist die Tatsache, dass sie derzeit kein standardisiertes Verfahren darstellen – andere RDBMS wie PostgreSQL oder MSSQL bieten ebenfalls Funktionen zur Volltextsuche an, nutzen aber nicht die identische Syntax wie MySQL [RWC12, S. 39-41] [Mic15b]. Entsprechend müsste für jedes Datenbanksystem, welches von der grundlegenden Applikation unterstützt werden sollte, eine angepasste Version der SQL-Anfragen entwickelt werden.

Ebenfalls erwähnenswert ist die Abhängigkeit von der verwendeten *Storage Engine*. Die Storage Engine stellt einen Bestandteil von MySQL dar. Von ihr hängt ab, wie die Daten abgespeichert werden und ob bestimmte Funktionen der Datenbank unterstützt werden – wie auch Transaktionen [Ora15b]. Unter anderem unterstützt nicht jede MySQL Storage Engine Volltextsuche. Bis zur Version 5.5 von MySQL konnte die InnoDB Engine – im Gegensatz zur damals als Standard verwendeten MYISAM – keine Volltextsuche durchführen. InnoDB stellt allerdings die Engine der Wahl für den vom Flow-Framework genutzten objektrelationalen Mapper Doctrine dar. Seit MySQL 5.6 wurde InnoDB zum Standard und unterstützt nun auch Volltextsuche [Nix14, S.191].

4.1.4 Levenshtein-Distanz

Die Levenshtein-Distanz ist eine Metrik, die eine Aussage über die Ähnlichkeit zweier Zeichenketten zueinander trifft. Sie gibt an, wie viele einzelne Buchstaben des einen Strings ersetzt, ergänzt oder entfernt werden müssen um den zweiten String zu erhalten. Jede derartige Änderung bedeutet eine um Eins erhöhte Levenshtein-Distanz. Eine geringere Levenshtein-Distanz weist darauf auf einen höheren Grad der Ähnlichkeit hin.

Fels → Feld → Geld → Gold

Da genau drei Änderungen notwendig sind, gilt $\text{levenshtein}('Fels', 'Gold') = 3$.

Aufgrund des Vorgehens dürfte die Levenshtein-Distanz bei leichten Rechtschreibfehlern und insbesondere Tippfehlern sehr verzeihend wirken.

$\text{levenshtein}('Fischerfest', 'Ficherfest') = 1$ weist für diesen Fall nur eine minimale Distanz auf, während der **LIKE**-Operator aus Abschnitt 4.1.2 hierbei scheitern würde.

Andererseits verhält sich Levenshtein bei *Text enthält Suchwort* weniger optimal.

$$\begin{aligned}\text{levenshtein}('Hund', 'Buch') &= 3 \\ \text{levenshtein}('Hund', 'Hundehalsband') &= 9\end{aligned}\tag{2}$$

„Buch“ wird in diesem Fall eine höhere Ähnlichkeit zugesprochen als „Hundehalsband“, hier würde ein Vergleich mittels **LIKE** nur das „Hundehalsband“ finden.

Während PostgreSQL die Levenshtein-Distanz bereits eingebaut hat [RWC12, S. 38], könnte sie in anderen RDBMS als *User Defined Function* nachträglich integriert werden. Es bleibt jedoch die Frage, ob die Performanz einer UDF mit einer eingebauten Funktion mithalten kann.

Gegebenenfalls müssen die Zeichenketten wie bereits in Abschnitt 4.1.2 mittels **LOWER()** auf Kleinbuchstaben reduziert werden, falls gewünscht ist, dass $\text{levenshtein}('T', 't') = 0$ gilt [RWC12, S. 38].

4.1.5 N-Gramme

Ein N-Gramm einer Zeichenkette ist die Menge aller möglichen Teilzeichenketten der Länge n . Ein Trigramm von „Ingolstadt“ wäre

$$\text{trigram}('Ingolstadt') = \{_in, ing, ngo, gol, ols, lst, sta, tad, adt, dt_ \}\tag{3}$$

Hierbei entspricht $_$ dem Beginn oder Ende eines Wortes [BCC10, S. 92f]. Bei einer Suche wird nun sowohl vom Suchbegriff als auch vom zu testenden Text ein Trigramm erstellt. Anschließend wird überprüft, welche Mächtigkeit die Schnittmenge der beiden Trigramme hat – je größer der Wert umso ähnlicher sind sich die beiden Zeichenketten.

$$\begin{aligned}\text{trigram}('Ingolstadt Fest') &= \\ \{_in, ing, ngl, gls, lst, sta, tad, adt, dt_ _Fes, est, st_ \}\end{aligned}\tag{4}$$

Die Zeichenkette „Ingolstadt Fest“ weist sowohl einen Tippfehler als auch ein zusätzliches Wort auf.

$$|\text{trigram}('Ingolstadt Fest') \cap \text{trigram}('Ingolstadt')| = 7\tag{5}$$

PostgreSQL bietet hierfür die Funktion `similarity(text, text)` an, welche auf Basis von Trigrammen einen Wert zwischen 0 und 1 berechnet, wobei 1 bedeutet, dass die beiden Mengen identisch sind. Auch MySQL bietet N-Gramme an [Ora15a].

Während in diesem Abschnitt nur N-Gramme von der Länge drei behandelt wurden (Trigramme), funktioniert das Prinzip auch mit anderen Substringlängen. Laut [BCC10, S. 95] sind manche Werte für N für bestimmte Sprachen besser geeignet als andere. Für die deutsche Sprache würden so Pentagramme⁵ das Optimum darstellen.

4.2 SUCHSYSTEME ALS LOKALE ANWENDUNG

Neben der einfachen Textsuche, welche auf Basis von SQL im Abschnitt 4.1.2 vorgestellt wurde, existieren auch weitere Softwareprodukte, die es im Gegensatz zu relationalen Datenbanken zur primären Aufgabe haben, eine Volltextsuche zu ermöglichen. Diese *Search Engines* lassen sich in verschiedene Gruppen unterteilen, beispielsweise in solche, die als zusätzliche Anwendung lokal eingerichtet werden. Vertreter hiervon sind Apache Solr und Elasticsearch, beides Search Engines auf Basis der Java-Bibliothek Apache Lucene.

Lucene wird von diesen Systemen unter anderem dazu genutzt, einen *Inverted Index* für die Textsuche aufzubauen [GP14, S. 11]. Dieser Index basiert grundlegend auf den einzelnen *Dokumenten*, welche durchsucht werden sollen. Ein Dokument enthält einen oder mehrere zusammengehörige Texte, welche sich wiederum aus einzelnen Wörtern zusammensetzen. Im Kontext von ReEvent könnte die Summe aus Name, Ort, Beschreibung und Kategorie einer Veranstaltung ein Dokument darstellen.

ID	Inhalt
1	Neuburger Schlossfest
2	Ein Sommernachtstraum
3	Freyunger Schlossfest
4	Rost auf Stahl - Bleistift auf Papier
5	Büchereizeit: Ein Weihnachtsengelbesuch
6	Technikgeschichte auf Papier
7	Büchereizeit: Unterwegs auf großer Fahrt

Tabelle 3. Beispieldokumente für einen Inverted Index

Tabelle 3 zeigt eine Reihe von Dokumenten, welche mit einer eindeutigen, numerischen ID versehen sind. Einen Inverted Index hierzu zeigt Tabelle 4. Zu jedem Wort aus den Dokumenten wird aufgelistet, in welchen Dokumenten es enthalten ist. Zudem werden die einzelnen Wörter alphabetisch aufsteigend sortiert [GP14, S. 54]. Dadurch wird beispielsweise die Suche nach einem einzelnen Wort stark vereinfacht. Um alle Dokumente zum Suchbegriff „Schlossfest“ zu finden, muss die Wort-Spalte von Tabelle 4 überprüft werden. Enthält diese den Begriff, so kann über die Dokument-ID-Spalte auf entsprechende Dokumente geschlossen werden. Die Erstellung dieses Indexes für eine Menge an Dokumenten – und die Durchführung weiterer zugehöriger Aufgaben – wird *Indizierung* genannt.

⁵ N-Gramme der Länge fünf

Wort	Dokument-ID	Wort	Dokument-ID
auf	4;6;7
Bleistift	4	Papier	4;6
Büchereizeit	5;7	Rost	4
ein	2;5	Schlossfest	1;3
Fahrt	7	Sommernachtstraum	2
Freyunger	3	Stahl	4
großer	7	Technikgeschichte	6
Neuburger	1	Weihnachtsengelbesuch	5

Tabelle 4. Beispiel eines Inverted Index auf Basis der Dokumente aus Tabelle 3 nach [GP14, S. 54]. Jedes Wort zeigt auf eine Liste von Dokumenten-IDs, jedes der darüber referenzierten Dokumente enthält dieses Wort.

Auch die Volltextsuche in MySQL, beziehungsweise der FULLTEXT-Index basieren auf einem Inverted Index [Ora15c].

Die Umsetzung von booleschen Verknüpfungen kann über den Index realisiert werden: Für eine Anfrage „Papier AND Rost“ wird zuerst der Index nach den beiden Einzelbegriffen durchsucht. Die beiden Ergebnismengen {4;6} und {4} können anschließend über Mengenoperatoren verknüpft werden, für AND ist dies die Schnittmenge [GP14, S. 58].

$$\{4;6\} \cap \{4\} = \{4\} \quad (6)$$

Indizierung von Dokumenten und Suche können unter Apache Solr über HTTP-Anfragen durchgeführt werden.

```

1 [
2 {
3   "id" : "123456",
4   "title" : "Sitzung des Stadtrates",
5   "description" : "zum Thema Solarbeleuchtung für Ingolstadt",
6   "venue" : "Neues Rathaus"
7 }
8 ]

```

Listing 4.5. JSON-Dokument für die Indizierung in Solr

Führt man einen HTTP POST an die URL `http://<solr_server>/solr/<core>/update` mit den Daten aus Listing 4.5 durch, so wird dieses Dokument zur angegebenen <core> hinzugefügt [GP14, S. 142]. Ein Core entspricht einer Menge von gleichartigen Dokumenten – beispielsweise Veranstaltungen. Welche Eigenschaften Dokumente eines Cores haben – etwa Datentypen – wird in einer Datei namens *schema.xml* definiert. Dies ist notwendig, da Solr für unterschiedliche Datentypen andere Analysevorgänge durchführen muss [GP14, S. 125]. `id` kommt hierbei eine Sonderrolle zu: Das Feld stellt einen eindeutigen Schlüssel dar, welchem dieses Dokument zugeordnet ist. Weitere Anfragen oder Änderungen werden über diesen Identifier aufgelöst [GP14, S. 122f].

Eine Suche kann per GET-Request http://<solr_server>/solr/<core>/select gestartet werden. Suchparameter werden als GET-Parameter angefügt. Neben dem eigentlichen Suchbegriff können etwa Filterung, Sortierreihenfolge und eine Submenge der Ergebnisliste⁶ angegeben werden [GP14, S. 36f]. Als Antwort erhält man erneut ein JSON-Dokument⁷, in welchem unter anderem die Treffer aufgelistet sind.

4.3 GOOGLE SEARCH APPLIANCE

Wie bereits in Abschnitt 3.3.3 dargestellt, ist Google ein Marktführer im Bereich der Websuche. In Form der *Google Search Appliance* ist es möglich, die grundlegende Suchfunktionalität der Google-Suche für eigene Applikationen zu nutzen. Die Appliance wird als physischer Server geliefert und im lokalen Netzwerk installiert [Goo15b].

Anders als beispielsweise bei Apache Solr kommen auf den Betreiber der GSA zusätzliche Gebühren zu: In Abhängigkeit von der Anzahl der indizierten Dokumente verlangt Google einen Mietpreis. Derzeit⁸ bietet Google zwei verschiedene Versionen der Appliance an, wovon eine für bis zu 20 Millionen und die andere für bis zu 100 Million Dokumente ausgelegt ist [Goo15b].

Einen weitere Unterschied zu Apache Solr stellt der Ablauf der Indizierung dar: In einer Administrationsoberfläche der Google Search Appliance werden hierbei verschiedene Quellen eingetragen, welche die GSA anschließend regelmäßig *aktiv* neu indiziert (Crawling). URLs oder Datenbank-Queries stellen hierbei mögliche Arten von Quellen dar [Goo12, S. 17, 26].

Mehrfache Anfragen an den GSA-Kundensupport bezüglich Kosten, Vertragsmodalitäten und Testgeräte verliefen im Sande oder sind zum aktuellen Zeitpunkt unbeantwortet. Dementsprechend kann hierzu keine Aussage getroffen werden. In einem von Google bereitgestellten Marketing-Dokument wird die GSA in einer Kostenkalkulation mit 45 000 \$ pro Jahr angesetzt [Goo13]. Neben den – im Vergleich zu den zuvor vorgestellten Lösungen – hohen Kosten spricht somit auch der Eindruck eines begrenzt hilfreichen Kundensupports gegen den Einsatz der GSA.

4.4 GOOGLE SITE SEARCH

Neben der professionellen Search Appliance bietet Google auch noch eine andere Möglichkeit, eine Suche über Dokumente einer bestimmten Domain zu realisieren. Bereits die Google-Websuche erlaubt es, Suchergebnisse entsprechend einzugrenzen. Hierzu wird zusätzlich zu weiteren Suchbegriffen die gewünschte Domain mit dem Parameter `site:` angegeben.

Google ermöglicht es, eine für eine Domain spezifische Suche in die eigene Webseite einzubinden. Im entsprechenden Online-Portal der *Google Site Search*⁹ kann eine solche „Suchmaschine“ konfiguriert werden. Im Anschluss daran generiert das Portal

⁶ für das Paging

⁷ falls als GET-Parameter angegeben, kann die Trefferliste auch in anderen Formaten wie XML angefordert werden

⁸ Stand 13.12.2015

⁹ siehe <https://cse.google.com/>

1 `site:www.thi.de VPN`

Listing 4.6. Google-Websuche mit site-Parameter, der die Suche auf die Website `www.thi.de` einschränkt

ein Javascript-Snippet, welches beim Laden einer Webseite automatisch ausgeführt wird und ein Suchformular einbindet. Suchergebnisse werden hierbei in einem Popup-artigen Overlay-Kontainer angezeigt.

In der Grundversion ist dieses Angebot kostenlos, jedoch werden zu Suchtreffern auch stets Werbeanzeigen mit angezeigt. Gegen Entrichtung einer jährlichen Gebühr – welche abhängig von der Anzahl der Suchanfragen ist – entfällt die Werbeeinblendung. Für 20 000 Anfragen innerhalb eines Jahres setzt Google einen Betrag von 100 \$ an [Goo15a].

Der Vorteil liegt hierbei darin, dass das Suchsystem nicht selbst gehostet werden muss, da sowohl Indizierung als auch alle Suchanfragen von Google-Servern durchgeführt werden. Viele gleichzeitige Suchanfragen – welche üblicherweise den eigenen Server auslasten würden – können somit von der Infrastruktur des Suchmaschinenanbieters verarbeitet werden.

Allerdings kann die Site Search nur Dokumente durchsuchen, welche aus Sicht des Google-Servers auch öffentlich erreichbar sind. Sind Veranstaltungen beispielsweise noch nicht veröffentlicht worden¹⁰ oder werden bestimmte Veranstaltungen nur per zugriffsgeschützten Webservice ausgeliefert, so können diese nicht von der Suche erfasst werden. Eine weitere Einschränkung könnte die robots.txt Datei darstellen, welche das Indizieren für einzelne Pfade des Webserverns verbieten könnte.

Während das Suchformular und die Trefferliste größtenteils durch CSS anpassbar sind, verpflichtet man sich bei Akzeptierung der Nutzungsbedingungen dazu, ein Google-Logo neben dem Suchfeld einzubinden und dieses auf eine von Google vorgegebene URL zu verlinken [Goo11, S. 2.1].

4.5 CLOUDDIENSTE MIT SUCHFUNKTIONALITÄT

Die letzte der hier vorgestellten Gruppen von Suchsystemen ist die der Cloudbasierten, bei denen der Suchserver in einem externen Rechenzentrum gehostet wird. Zwei Vertreter dieser Gruppe sind *Amazon CloudSearch* und *Microsoft Azure Search*.

Wie auch bereits bei der Google Site Search befindet sich somit das Suchsystem nicht im lokalen Netz, entsprechend müssen Server und Software nicht selbst gewartet werden. Von einem lokalen Suchsystem unterscheidet sich ein Cloud-Suchdienst auch dahingehend, dass die Daten *aktiv* einem Dritten übergeben werden – in Zeiten von Überwachungs-skandalen und politischer Einflussnahme auf Hosting-Betreiber ein nicht unbedeutender Punkt. Hierbei muss jedoch bedacht werden, dass ReEvent im Betrieb der Stadt Ingolstadt vermutlich nur einen geringen Teil sensibler Daten aufnehmen wird.

Ein weiterer, gravierender Unterschied zu Apache Solr und ähnlichen Suchsystem liegt darin, dass Clouddienste üblicherweise gemietet werden. Hierbei wird allerdings kein

¹⁰ Workspaces

jährlicher, fester Betrag fällig: Die Gebühren berechnen sich anhand der Nutzung. Beim cloudbasierten Suchdienst von Amazon setzen sich die Kosten wie folgt zusammen:¹¹

- 0,112 \$ pro angefangene Stunde für eine *search.m3.medium* Instanz
- 0,1 \$ pro 1 000 Indizierungsanforderungen von je max. 5 MB
- 0,09 \$ pro TB ausgehender Daten¹²

Vorteilhaft an Cloud-Diensten ist vor allem die Möglichkeit, die gemieteten Recheneinheiten je nach Bedarf vertikal skalieren zu können. So können insbesondere Lastspitzen – möglicherweise bei Veranstaltungen mit großem Publikum – abgefangen werden, ohne den Rest des Jahres teure und leistungsstarke Hardware nahezu ungenutzt zu lassen.

Unter Amazon Cloudsearch erfolgt die Suche und Indizierung ähnlich wie bei Apache Solr über eine Webservice-Schnittstelle [Ama15, S. 12f, 223].

¹¹ vergleiche <https://aws.amazon.com/de/cloudsearch/pricing/>, Stand 01.01.2016, alle Preisangaben zuzüglich Steuern und in Bezug auf die Region „EU (Frankfurt)“

¹² für die ersten 10 TB

Seit einiger Zeit konkurrieren neue Arten von Datenhaltungssystemen mit den althergebrachten relationalen Datenbanken. Die sogenannten *NoSQL*-Datenbanken – was je nach Quelle als „No SQL“¹ oder „Not only SQL“² interpretiert wird – speichern Daten üblicherweise nicht in der von relationalen Datenbanken gewohnten zeilenorientierten Tabellenstruktur ab. Stattdessen kristallisierten sich verschiedene Typen von NoSQL-Datenbanken heraus. So gibt es Key-Value-Datenbanken, Spaltenorientierte, Dokumentenorientierte und auch Graphdatenbanken [RWC12, S. 307-310]. Auch ein Suchserver wie Apache Solr wird als NoSQL bezeichnet [GP14, S. 4].

Eine jede dieser Gruppen hat ihr Anwendungsgebiet. Andererseits gibt es Fälle, in welchen ein bestimmter Typ von NoSQL-Lösung falsch am Platz ist. So eignet sich etwa ein Key-Value Speicher weniger für stark mit einander verknüpfte Datensätze [RWC12, S. 308], auch ein Suchserver dürfte als primäre Datenhaltung für eine Lagerverwaltung die falsche Wahl sein. Daher gilt es beim Einsatz von NoSQL-Produkten die zu den Anforderungen passende Lösung zu bestimmen.

Bei der Entwicklung einer *polyglotten* Persistenz wird diese Eigenschaft von NoSQL-Datenbanken beachtet: Anstatt nur eine einzige Datenbank für jedwede Speicherung von Daten zu verwenden, wird für unterschiedliche Teilbereiche des entstehenden Systems eine *passende* Datenbanklösung gewählt [RWC12, S. 292]. Genau genommen stellt die Kombination aus RDBMS und Search Engine in ReEvent bereits ein polyglottes System dar. Denn für Volltextsuche ist eine relationale Datenbank nicht in erster Linie ausgelegt, eine eigens für diesen Zweck entwickelte Search Engine jedoch sehr wohl. Dieses Kapitel soll Ansätze zeigen, wie die Graphdatenbank neo4j für die Suche nach Veranstaltungen verwendet werden kann.

Im Zusammenhang der Evaluation erfolgt dies gewissermaßen „außer Konkurrenz“, die in diesem Kapitel beschriebenen Inhalte können eine herkömmliche Suchlösung zwar ergänzen aber nicht ersetzen.

5.1 EMPFEHLUNG AUF GRUNDLAGE VORHERIGER SEITENBESUCHER

Die übliche Volltextsuche lässt sich mathematisch als Funktion beschreiben, welche einem Suchbegriff – welcher vom Benutzer eingegeben wird – eine bestimmte Treffermenge zuordnet. Jedoch ist eine Suche nicht der einzig denkbare Weg, wie ein Nutzer zu einer möglicherweise interessanten Veranstaltung gelangen kann: Auch eine Empfehlungs- oder Vorschlagsfunktion wäre denkbar. Von der grundlegenden Logik unterscheiden sich Suchtreffer und Empfehlung nicht: Während beim einen der Suchbegriff entscheidender

¹ englisch: kein SQL

² englisch: nicht nur SQL

Parameter ist, wird die Auswahl beim zweiten anhand der Identität des Benutzers oder dessen Verhalten gegenüber einzelnen Veranstaltungen getroffen [GP14, S. 571].

Es bleibt jedoch die Frage, auf welcher Grundlage Empfehlungen für den Nutzer ausgesprochen werden können. Hier hilft ein Blick zum Internethändler *amazon.de* weiter: Ruft man die Detailansicht eines Artikels auf, so erscheint eine Auswahl weiterer Produkte unter der Überschrift „Kunden, die diesen Artikel gekauft haben, kauften auch“. Ähnlich könnte auch ReEvent Vorschläge auswählen. Angenommen ein Nutzer *X* hat die Detailseiten zu den Veranstaltungen $\{A, B, C\}$ aufgerufen, und zuvor interessierten sich eine große Menge anderer Nutzer für die Veranstaltungen $\{A, B, C, D, E\}$, so könnten die Veranstaltungen $\{D, E\}$ auch für diesen Nutzer *X* relevant sein.

Folgende Abschnitte skizzieren eine Umsetzung eines Systems, welches derartige Vorschläge ermöglichen soll.

5.2 IDENTIFIZIERUNG DES NUTZERS

ReEvents Veranstaltungen werden von Nutzern über das Web-Frontend abgerufen. Da das hierfür genutzte HTTP-Protokoll jedoch statuslos ist, fehlt dem Server der Zusammenhang zwischen mehreren, aufeinanderfolgenden Anfragen desselben Nutzers.

Damit eine sinnvolle Empfehlung abgegeben werden kann, müssen mehrere Anfragen desselben Benutzers miteinander in Verbindung gebracht werden. Dies geschieht durch eine serverseitige Identifizierung des Benutzers. Eine Übersicht gängiger Lösungsansätze hierzu bietet Anhang D.

5.3 GRAPHDATENBANK NEO4J ZUR SPEICHERUNG

Immer wenn ein Benutzer eine Veranstaltung aufruft, wird diese Aktion abgespeichert. Hierfür sollte sowohl der Benutzer als auch die Veranstaltung möglichst eindeutig identifiziert werden. Auf Basis einer der Methoden aus Abschnitt 5.2 wird ein eindeutiger Schlüssel für den Nutzer generiert, beispielsweise ein Hash von durch Browser-Fingerprinting gefundene Merkmale. Da die Veranstaltungen in ReEvent eine eindeutige ID erhalten, kann diese hier wiederverwendet werden.

Für die Speicherung in der NoSQL-Datenbank gelten folgende Anforderungen:

- ist noch kein Benutzer mit der ID_user vorhanden, so wird er angelegt
- ist die besuchte Veranstaltung mit der ID_event nicht vorhanden, so wird diese angelegt
- anschließend werden beide Objekte durch eine Beziehung VISITED miteinander verknüpft

In einer relationalen Datenbank könnte man dies in drei Tabellen modellieren: Eine nimmt die Benutzer auf, eine zweite die Veranstaltungen und eine dritte die Seitenbesuche, welche Fremdschlüsselbeziehungen zu den ersten beiden enthält.

Im Rahmen dieses Kapitels wird für die Persistenz die Graphdatenbank *neo4j* ausgewählt. Laut [RWC12, S. 258f, S. 310] eignen sich Graphdatenbanken insbesondere für stark

verknüpfte Daten und Abfragen über mehrere Beziehungen hinweg. In einer relationalen Datenbank würde die Umsetzung der hier vorgestellten Anfragen viele verknüpfte **JOIN**-Anweisungen über dieselben Tabellen erfordern.

Eine neo4j-Datenbank entspricht einem Graphen: Daten werden in *nodes* abgespeichert, ein *node* kann wiederum Schlüssel-Wert-Paare enthalten. Anschließend können zwei *nodes* durch eine Beziehung verknüpft werden. Ein einfaches Beispiel für einen Graphen, welcher den hier beschriebenen Sachverhalt speichert, zeigt die Grafik 14.

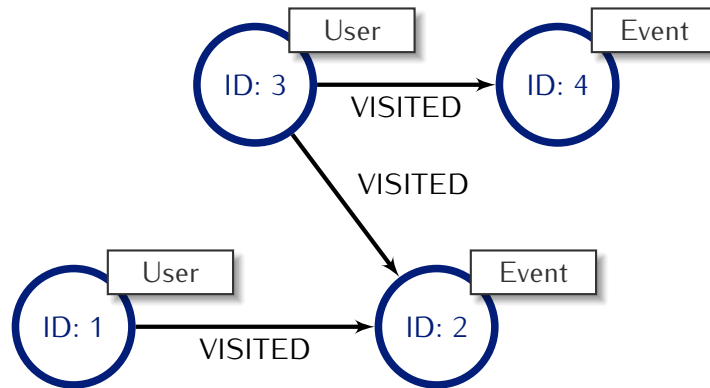


Abbildung 14. Beispiel eines Graphen in neo4j, Benutzer besuchen eine Veranstaltung

Für die Interaktion mit neo4j wird die Abfragesprache Cypher genutzt. Alternativ hierzu bietet neo4j eine Java- oder REST-Schnittstelle an, auch die auf Groovy basierende Abfragesprache Gremlin ist kompatibel [RWC12, S. 223]. Listing 5.1 zeigt die Eintragung von Benutzer, Veranstaltung und der VISITED-Beziehung zwischen beiden.

```
1 MERGE (u:User {ID : 3})-[:VISITED]->(e:Event {ID : 4});
```

Listing 5.1. Einfügen von Benutzer, Veranstaltung und VISITED-Beziehung

5.4 ABFRAGE DER EMPFEHLUNGEN DURCH CYPHER

Nachdem im vorhergehenden Abschnitt die Speicherung der Seitenbesuche dargestellt wurde, soll im Folgenden gezeigt werden wie für einen einzelnen Benutzer Empfehlungen abgegeben werden können. Wie bereits in Abschnitt 5.1 dargestellt, besucht jeder Benutzer eine bestimmte Menge an Veranstaltungen. Jede dieser Veranstaltungen hat auch weitere Interessenten. Jeder einzelne dieser Interessenten hat – wie auch schon der Benutzer, für den Empfehlungen ausgewählt werden sollen – Veranstaltungen über das Frontend aufgerufen. Die folgende neo4j-Abfrage basiert auf der Idee, dass Veranstaltungen, die sehr häufig von Mitinteressenten aufgerufen worden sind auch für den primären Nutzer relevant sein könnten. Bezugnehmend auf die Abbildung 14 hat der User mit ID:1 die Veranstaltung ID:2 besucht. Diese wurde ebenfalls von Benutzer mit ID:3 frequentiert, welcher sich auch für die Veranstaltung ID:4 interessiert hat. Dementsprechend wird User ID:1 die Veranstaltung ID:4 empfohlen.

```

1  MATCH
2      (me:User) - [:VISITED] -> (e:Event) <- [:VISITED] - (somebody:User),
3      (somebody) - [:VISITED] -> (other_event:Event)
4  WHERE
5      me.ID = 1
6  AND NOT
7      (me) - [:VISITED] -> (other_event)
8  RETURN
9      other_event.ID,
10     COUNT(other_event) AS weight
11 ORDER BY
12     weight DESC
13 LIMIT 5

```

Listing 5.2. Abfrage von Vorschlägen für einen User mit der ID 1, die Anzahl der Vorschläge wird auf fünf begrenzt

Der **MATCH**-Clause zu Beginn filtert die Veranstaltungen wie eben beschrieben. Nach **WHERE** wird der Startpunkt der Suche festgelegt, dies geschieht durch Eingrenzung über die Benutzer-ID. Zeile sieben verhindert, dass Veranstaltungen empfohlen werden, die der Benutzer bereits betrachtet hat. Anschließend wird die Veranstaltungs-ID zurückgegeben, aus welcher beispielsweise ein Link zur zugehörigen Seite im Frontend generiert werden kann. Entscheidender Faktor ist schließlich die Aggregats-Funktion `COUNT()`, welche die Anzahl von an Mitinteressenten für eine einzelne Veranstaltung zählt. Diese Summe wird als Entscheidungskriterium für die Auswahl der Empfehlungen verwendet: Per **ORDER BY** werden die Veranstaltungen nach diesem Gewicht absteigend sortiert, **LIMIT** begrenzt die maximale Anzahl an Vorschlägen auf fünf.

5.5 ERWEITERUNG UND ALTERNATIVEN

Neben dem einfachen Ansatz, der soeben vorgestellt wurde, wären auch diverse Anpassungen denkbar, durch welche sich möglicherweise die Qualität der Empfehlungen verbessern ließe.

Die **MERGE**-Anweisung aus Listing 5.1 verhindert, dass zwischen einem Benutzer und einer Veranstaltung mehr als eine **VISITED**-Beziehung eingetragen werden kann. Ein Benutzer könnte eine Veranstaltung auch mehrfach aufrufen, was auf besonderes Interesse in Bezug auf diese Veranstaltung hindeutet. Würde man nun jeden einzelnen Aufruf dieser Veranstaltung abspeichern³, könnte man zu jedem Seitenabruf den aktuellen Zeitstempel hinzufügen. Liegen zwischen einem Benutzer und einer Veranstaltung mehrere Beziehungen vor, könnte man dies stärker gewichten. Anhand des Alters der **VISITED**-Beziehung ließe sich zudem eine Straffunktion implementieren, sodass ältere Seitenaufrufe weniger stark ins Gewicht fallen.

Die Suche nach Veranstaltungen von Mitinteressenten ähnelt der Breitensuche in Graphen: Ausgehend von einem Startknoten (der Nutzer) werden zuerst Veranstaltungen ge-

³ hierzu müsste die Cypher-Anweisung in Listing 5.1 entsprechend angepasst werden

sucht, zu diesen wiederum Nutzer und ausgehend davon erneut Veranstaltungen. Theoretisch könnte man dies auch ausweiten. Jeder Veranstaltungsknoten, der als Vorschlag in Frage kommt, kann ebenfalls von weiteren Nutzern besucht worden sein – welche selbst erneut mit Veranstaltungen verknüpft sind. Da diese jedoch weiter vom Startknoten entfernt sind, müssten diese ebenfalls geringer gewichtet werden.

Laut [GP14, S. 586-590] lassen sich entsprechende Empfehlungen auch mit Apache Solr implementieren. Hierzu wird jedes Dokument mit einem Array aus Benutzer-IDs versehen, welche diese Veranstaltung bereits besucht haben. Im *Inverted Index* wird diese Struktur wie in Abschnitt 4.2 beschrieben zu einer Zuordnung von Benutzern zu Veranstaltungen umgekehrt. Darauf basierend lassen sich ähnliche Anfragen wie die hier vorgestellten effizient umsetzen.

5.6 MÖGLICHE NACHTEILE DES VORGESTELLTEN ANSATZES

Es sei angemerkt, dass in diesem Kapitel datenschutzrechtliche Aspekte teilweise ausgeklammert wurden. Sowohl Methoden des Browser-Fingerprinting als auch die Abspeicherung mehrerer aufeinanderfolgender Seitenzugriffe inklusive der Zuordnung zu einer – wenn auch anonymen Person – könnten die Persistierung personenbezogener Daten implizieren.

Außerdem entsteht durch ein zusätzliches System – in diesem Fall neo4j – auch weiterer Wartungsaufwand, insbesondere bezüglich Updates und Konfiguration. Je mehr Systeme miteinander kommunizieren, desto fehleranfälliger könnte auch das Gesamtsystem werden: Die relative Wahrscheinlichkeit, dass ein notwendiges Teilsystem ausfällt, steigt an.

6

VORAUSWAHL DER ZU EVALUIERENDEN SYSTEME

Im Kapitel 4 wurden insgesamt zehn verschiedene Ansätze zur Suche aufgezeigt. Ein Vergleich all dieser Systeme würde den Rahmen dieser Arbeit sprengen. Aus diesem Grund wird eine begründete Vorauswahl von möglichen Lösungen getroffen, welche im späteren Verlauf der Arbeit evaluiert werden sollen.

Die Tests in Kapitel 7 sollen aber prinzipiell so ausgelegt sein, dass – falls notwendig – weitere Durchläufe mit den hier ausgegrenzten Suchsysteme möglich sind und die entstanden Ergebnisse untereinander vergleichbar sind.

APACHE SOLR UND ELASTICSEARCH Beide sind unter der Apache License 2.0 stehende freie Search Engines – und beide basieren auf der Java-Bibliothek Apache Lucene. Davon ausgehend wird angenommen, dass hinsichtlich der Anforderungen aus Kapitel 3 beide ähnliche Resultate erzielen würden. Die Tatsache, dass die Anwendung bereits von response verwendet wurde gibt schließlich den Ausschlag für Apache Solr.

LEVENSHTEIN UND TRIGRAMME Zusammen mit dem **LIKE**-Operator stellen Levenshtein und Trigramme vergleichsweise simple Ansätze zur Suche dar. In Hinblick auf die in Kapitel 4 dargestellten Einschränkungen von **LIKE** und Levenshtein wird der Trigramm-basierte Vergleich als Vertreter dieser Gruppe ausgewählt. Um jedoch den Wert einer Volltextsuche ermessen zu können, werden die hier dargestellten Suchsysteme mit der **LIKE**-basierten Suche aus INsite verglichen.

EINGEBAUTE VOLLTEXTSUCHE VON RDBMS Ähnlich wie bereits bei Apache Solr und Elasticsearch liegen verschiedene Datenbanksysteme vor, die über eine eingebaute Volltextsuche verfügen. Kandidaten sind hierbei PostgreSQL, MSSQL und das vorgestellte MySQL. Um Synergieeffekte bezüglich dem vorherigen Absatz zu nutzen und da die MySQL-Volltextsuche laut [Bel15] vergleichsweise limitiert in ihrem Funktionsumfang sei, wird die Volltextsuche von PostgreSQL genutzt.

GOOGLE SEARCH APPLIANCE Wenn auch ein interessantes Konzept, so kann die GSA nicht die Evaluation mit einbezogen werden, da zum momentanen Zeitpunkt kein Testgerät vorliegt. Zudem dürften die Kosten – die wie in Abschnitt 4.3 dargestellt im fünf- bis sechststelligen Bereich liegen könnten – den Kostenrahmen des Projektes übersteigen. Lediglich für den Fall, dass die Stadt Ingolstadt in Zukunft eine zentrale Suchlösung benötigen sollte, könnte man die Google Search Appliance erneut als mögliche Lösung ins Auge fassen – inklusive der Einbindung von ReEvent.

GOOGLE SITE SEARCH Gegen die Google Site Search spricht, dass nur öffentlich zugängliche Daten durchsucht werden können. Insbesondere die Workspaces aus Abschnitt 2.6 oder der angedachte Review-Prozess für die Veröffentlichung externer Veranstaltungen werden somit konstruktionsbedingt nicht von der Suche erfasst.

CLOUDBASIERTE SUCHSYSTEME Im Abschnitt 4.5 wurde die cloudbasierte Suche von *Amazon CloudSearch* und *Microsoft Azure Search* vorgestellt. Ähnlich wie bei Apache Solr fällt die Entscheidung für das Cloud-Angebot von Amazon, da hierfür bei response bereits ein registriertes Konto vorliegt.

6.1 AUFLISTUNG DER ZU EVALUIERENDEN SUCHLÖSUNGEN

Folgende Suchlösungen werden daher in die Evaluation mit einbezogen.

- Apache Solr
- PostgreSQL – **LIKE**
- PostgreSQL – Trigramme
- PostgreSQL – Volltextsuche
- Amazon CloudSearch

7 | EVALUIERUNG

Ziel dieser Arbeit ist die Auswahl einer Suchmaschine, welche die Anforderungen aus Kapitel 3 erfüllt. Hierzu werden im Anhang A Tests und Kriterien definiert, auf welche die zuvor in Kapitel 6 ausgewählten Suchmaschinen überprüft werden.

7.1 AUFSTELLUNG EINES BEWERTUNGSSCHEMAS

Um einen Vergleich mehrerer Suchmaschinen zu ermöglichen, muss ein wie auch immer geartetes Maß für die jeweilige Eignung der Suchmaschine gefunden werden. Dieses Maß setzt sich für eine Suchmaschine s aus verschiedenen Auswahlkriterien zusammen. Ein Kriterium a kann zu einem bestimmten Anteil $\alpha_{s,a} \in [0;1]$ den Anforderungen genügen. Da nicht jedes Kriterium für die Auswahl von gleicher Wichtigkeit sein muss, wird dieser Anteil mit einem multiplikativen Gewicht $\beta_a \in \mathbb{R}_0^+$ versehen. Die Gesamtbewertung r_s einer Suchmaschine s über alle Kriterien A ergibt sich somit aus

$$r_s = \sum_{a \in A} \alpha_{s,a} \cdot \beta_a \quad (7)$$

Somit reduziert sich das Problem der Evaluation auf folgende Punkte:

- Finden aller relevanten Kriterien A , welche in die Evaluation einbezogen werden sollen
- Ermitteln eines Gewichtes β_a für alle $a \in A$
- Aufstellung eines Tests zur Ermittlung einer Bewertung $\alpha_{s,a}$ für alle $a \in A$
- Durchführung aller Tests und Bestimmung des Resultates

Es ergibt sich eine maximale Bewertung von

$$\sum_{a \in A} \beta_a \quad (8)$$

7.2 EIN ÜBERBLICK ÜBER DIE TESTUMGEBUNG

Die Tests finden auf einem Rechner mit folgenden Eckdaten statt:

- Windows 8.1 64bit
- 8 GB Arbeitsspeicher
- Intel Core i5-3317U 1,70 GHz CPU

- SanDisk SSD U100 128 GB
- Internetanbindung per DSL¹ mit 6 769 kbit/s Download und 714 kbit/s Upload
- Lucee 4.5 zur Ausführung der Tests (freie ColdFusion Implementierung)²
- PostgreSQL 9.4.5
- Apache Solr 5.4.0
- Amazon CloudSearch API 2013-01-01

Wird nichts Gegenteiliges verlautet, so werden die Standardeinstellung nach der Installation oder Ersteinrichtung des Suchdienstes verwendet.

7.3 HERKUNFT DER TESTDATEN

Zum aktuellen Zeitpunkt³ befindet sich ReEvent noch in der Entwicklung. Insbesondere der Teil, der Veranstaltungen aufnehmen und verwalten kann, ist derzeit noch nicht fertiggestellt. An ein noch nicht fertiggestelltes System kann allerdings auch keine direkte Anbindung zur Indizierung oder Abfrage von Suchergebnissen entwickelt werden. Dementsprechend werden im Folgenden Testdaten verwendet, die per Hand angelegt wurden oder aus der INsite-Datenbank stammen und dem geplanten Modell von ReEvent nahe kommen sollen.

7.4 DEFINITION DES KRITERIENKATALOGES

Jede einzelne Suchmaschine, die im Kapitel 6 ausgewählt wurde, wird nach mehreren Kriterien auf ihre Tauglichkeit hin überprüft. Hierzu soll ein *Kriterienkatalog* aufgestellt werden. Wie bereits eingangs erwähnt, entstammen die Kriterien dem Requirements Engineering in Kapitel 3.

Für jedes einzelne Kriterium a wird eine Beschreibung hinterlegt und ein Testablauf definiert. Das Testergebnis muss anschließend in eine Note $\alpha_{s,a} \in [0;1]$ überführt werden. Letzter festzulegender Parameter ist das Gewicht $\beta_a \in \mathbb{R}_0^+$, welches sowohl relativ zu anderen als auch in Bezug auf die Einordnung der Anforderung hinsichtlich dem Abschnitt 3.2 gewählt wird. Die Gewichtung der Einzelnoten β_a hat einen starken Einfluss auf den Ausgang der Gesamtbewertung. Entsprechend minutiös muss der Wert hierfür gewählt werden.

Bei einigen Anforderungen stellte sich heraus, dass sie nicht direkt vom Suchsystem selbst sondern von der Anbindung dessen an ReEvent abhängig sind. Hierunter fallen unter anderem die lokalisierbare Suchmaske oder die automatische Aktualisierung der Trefferliste bei Änderung des Suchbegriffes beispielsweise durch AJAX.

¹ Median aus drei aufeinanderfolgenden Tests mit <http://www.wieistmeineip.de/speedtest/>

² Lucee stellt einen Fork des zu Beginn erwähnten Railo dar

³ Stand 29.12.2015

Die Bewertung mancher filterbasierter Tests gestaltet sich in Hinblick auf SQL-basierte Suchen schwierig: Mit entsprechendem Aufwand könnte möglicherweise jede dieser Aufgaben über SQL abgebildet werden. Daher wird festgelegt, dass ein Test bezüglich einer Filterung erfolgreich ist, wenn er innerhalb einer Abfrage unter vertretbarem Aufwand mit Standardfunktionen von SQL durchgeführt werden kann. Beispielsweise könnte die Umkreissuche theoretisch umgesetzt werden⁴, eine Filterung nach Datum ist mit gängigen Vergleichen wie **BETWEEN** verhältnismäßig einfach.

7.5 ÜBER DIE QUALITÄT EINER VOLLTEXTSUCHE

Neben der Vielfalt an Filtern und sonstigen Funktionen, die im Kriterienkatalog definiert werden, ist die bloße Volltextsuche wichtigste Aufgabe des Suchsystems – was sich auch in der Gewichtung der zugehörigen Tests widerspiegeln soll. Es verbleibt jedoch die Frage, wie sich die Qualität einer Suchmaschine messen lässt. Hierzu sollen die folgenden zwei Aspekte betrachtet werden.

7.5.1 Effektivität

Primärer Zweck einer Volltextsuche ist die Auffindung gewünschter Dokumente, welche zur Suchanfrage passen sollen. Zur Bewertung dieser wird hier das Modell von *Recall* und *Precision* verwendet.

Entscheidend für die Bewertung sind zwei Mengen von Dokumenten: die Trefferliste *T* für eine Suchanfrage sowie die für diese Suchanfrage relevanten Dokumente *R*.

$$recall = \frac{|T \cap R|}{|R|} \quad precision = \frac{|T \cap R|}{|T|} \quad (9)$$

Precision entspricht dem Anteil relevanter Dokumente innerhalb der Trefferliste, Recall berechnet den Prozentsatz der gefunden, relevanten Dokumente insgesamt. Eine qualitativ hochwertige Volltextsuche zeichnen sowohl hohe Werte für Recall als auch für Precision aus – beide Werte sind nur im Zusammenhang aussagekräftig. Beispielsweise könnte eine Suchmaschine unabhängig von der Suchanfrage alle Dokumente zurückliefern: Recall hätte somit einen Wert von 1⁵, die Precision würde allerdings maßgeblich abfallen [BCC10, S. 407]. Beide werden im sogenannten *F-Maß* kombiniert, welches das harmonische Mittel von Recall und Precision berechnet [Bel15, S. 68].

$$F = \frac{2 \cdot recall \cdot precision}{recall + precision} \quad (10)$$

Das F-Maß wird im zugehörigen Test im Anhang A als Kriterium für die Effektivität festgelegt.

Das Problem dieser Metriken ist, dass die vorherige Einordnung der Dokumente in Bezug auf Relevanz zu einer Suchanfrage nur manuell durchgeführt werden kann und

⁴ PostgreSQL bietet hierfür die Erweiterung *earthdistance* an, <http://www.postgresql.org/docs/9.4/static/earthdistance.html> oder auch PostGIS <http://postgis.net/>

⁵ Recall wäre somit maximal

diese mit einem hohen zeitlichen Aufwand verbunden ist. Entsprechend muss die Anzahl der durchsuchten Dokumente reduziert werden.

7.5.2 Effizienz

Während die Effektivität die Qualität der Suchergebnisse misst, erfasst die Effizienz den *Durchsatz* und die *Latenz* von Anfragen. Durchsatz wird hierbei als Menge von Suchanfragen definiert, die ein Suchsystem pro Zeiteinheit abarbeiten kann. Die Latenz gibt die Dauer einer einzelnen Suchanfrage an und kann – in Abhängigkeit von der Definition – auch die Netzwerklatenz beinhalten [BCC10, S. 470].

Auf den ersten Blick könnte es als müßig erscheinen, beide zu bestimmen. Der Durchsatz ist allerdings nicht notwendigerweise der Kehrwert der Latenz: Aufgrund von Queuing von Anfragen kann sich die Latenz unter Last stark erhöhen. Andererseits steigt der Durchsatz, falls das Suchsystem in der Lage ist, mehrere Suchanfragen parallel abzuarbeiten [BCC10, S. 471]. Während der Durchsatz in erster Linie aus Serversicht interessant ist – Stichwort Skalierung – ist die Latenz entscheidend für die Benutzerzufriedenheit. Insbesondere verhältnismäßig hohe Latenzwerte können sich störend auf die User Experience auswirken [BCC10, S. 472].

7.6 ANMERKUNGEN ZU EINZELNEN TESTS

Die Tests, die lediglich überprüfen, ob eine bestimmte Funktionalität vorhanden ist, beziehen sich unter anderem auf folgende Dokumentationen.

- <http://www.postgresql.org/docs/9.4/static/textsearch.html>
- <http://mirror2.shellbot.com/apache/lucene/solr/ref-guide>
- <http://docs.aws.amazon.com/cloudsearch/latest/developerguide/cloudsearch-dg.pdf>

Suchanfragen über Apache Solr oder Amazon CloudSearch beziehen sich immer auf den *edismax*- oder *dismax*-Modus. Der gewählte Suchmodus entscheidet unter anderem darüber, wie die angegebene Suchanfrage abgearbeitet wird. *dismax* soll es erlauben, Benutzereingaben direkt entgegenzunehmen, ohne Syntaxfehler befürchten zu müssen. Dadurch ist dieser Suchmodus prädestiniert für eine Sucheingabe über ein einfaches, clientseitiges Textfeld [GP14, S. 222f].

Eine Ausnahme hiervon stellt der Test bezüglich Rechtschreibfehler bei Amazon CloudSearch dar: Er erfolgte über die „suggest“-Schnittstelle des Webservices. Diese berechnet letztendlich eine Levenshtein-Distanz und überprüft, ob diese kleiner zwei ist. Auch Apache Solr wurde entsprechend konfiguriert und nutzte bei diesem Test Levenshtein [GP14, S. 315]. Welche maximale Distanz Solr an dieser Stelle verwendet hat, ist nicht bekannt.

7.6.1 Indizierung bei SQL-basierter Suche

Auch für die SQL-basierte Suche kann es sinnvoll sein, einen Datenbankindex anzulegen. Ziel dessen ist es, eine Beschleunigung der Suchvorgänge zu erreichen. Für die Volltext-

suche bietet PostgreSQL hier die Varianten GIN und GiST an. GIN-Indizes erlauben im Vergleich zu GiST eine schnellere Suche, allerdings einhergehend mit einer langsameren Indizierungsvorgang und höheren Speicheranforderungen [The16a].

In den Tests wurde zuerst sowohl für die Trigramm-Suche als auch für die PostgreSQL-Volltextsuche ein Index vom Typ GIN⁶ angelegt. Unter PostgreSQL kann durch Voranstellung des Schlüsselwortes **EXPLAIN** überprüft werden, ob ein erstellter Index in einer Anfrage genutzt wird. Die Query returniert so eine Spalte namens `QUERY PLAN`, deren Einträge einzelne Schritte zur Ausführung der Abfrage enthalten.

```
1 EXPLAIN SELECT * FROM event WHERE venue % 'Suchbegriff'
```

Listing 7.1. **EXPLAIN** bei Trigramm-basierter Suche

Obige Anfrage an eine mit einem GIN-Index versehene Tabelle gibt unter anderem Bitmap Index Scan on idx_trgm_v[...] aus und deutet somit darauf hin, dass der entsprechende Index idx_trgm_v genutzt wurde. Der **EXPLAIN** Befehl weist bei den für die Tests verwendeten Anfragen aus den Listings G.2 und G.3 darauf hin, dass keine Indizes genutzt werden. Dies kann zwei mögliche Ursachen haben: Entweder ist die Anfrage selbst fehlerhaft und verhindert die Nutzung eines Index oder aber PostgreSQL entscheidet, dass Indizes für diese Anfrage nicht genutzt werden sollen, da die Query ohne Index schneller abgearbeitet werden könnte. Möglicherweise wird ein Index erst bei größeren Datenmengen relevant als sie hier verwendet wurden. Entsprechend wurde für Trigramm-Vergleich und PostgreSQL-basierter Volltextsuche kein Index eingerichtet.

Bei **LIKE** wäre ein Datenbankindex üblicherweise nur sinnvoll, wenn der Vergleich nicht mit einem Wildcard-Operator beginnen würde. Was PostgreSQL in diesem Punkt von anderen Datenbanken unterscheidet, ist die Möglichkeit Trigramm-basierte Indizes für Suchanfragen wie **LIKE** '%Suchbegriff%' zu nutzen. Hierbei werden Trigramme vom Suchbegriff erstellt und im Index nachgeschlagen [The16b]. Tests ergaben, dass sich so der Durchsatz nach Test #2 nahezu verdoppelt. Interessanterweise gilt dies jedoch auch für die gemessene Latenz aus Testkriterium #3. Zusammen mit der geringeren Bewertung für die Indizierungsgeschwindigkeit wäre die Gesamtnote insgesamt schlechter ausgefallen, daher wurde der entsprechende Testlauf ohne Trigramm-Index durchgeführt.

Für die Nutzung von Trigrammen muss ein PostgreSQL-Modul wie folgt aktiviert werden.

```
1 CREATE EXTENSION pg_trgm;
```

Listing 7.2. Aktivierung von Trigrammen unter PostgreSQL für eine Datenbank

Ähnlich wie die vergleichbaren Batch-Anweisungen bei Apache Solr und Amazon CloudSearch wurden bei der Geschwindigkeitsmessung der Indizierung alle Datensätze in einer **INSERT**-Anweisung übertragen.⁷

⁶ General Inverted Index

⁷ sogenanntes *multi-row insert*

7.6.2 Kostenkalkulation für Amazon CloudSearch

Der Test #11 erfordert die Schätzung jährlicher Kosten bei der Nutzung eines Suchsystems. Lediglich für das cloudbasierte Angebot von Amazon ist kostenpflichtig und muss daher einer Kostenkalkulation unterzogen werden.

Die jährlichen Kosten von Amazon CloudSearch sind von folgenden Faktoren abhängig: Nutzungsdauer des Dienstes gemessen in begonnenen Stunden, Menge der Indizierungsanforderungen und der ausgehende Datenverkehr. Die Testdefinition in Anhang A gibt hierfür 4 540 Suchanfragen und über 3 600 Veranstaltungen zu je 1,13 KB an. Zuerst wird abgeschätzt, wie viele Stunden der Dienst genutzt werden würde. Hierfür wird einmal mehr das Webserverlog aus Abschnitt 3.3.3 herangezogen. Eine weitere Auswertung dessen ergab, dass die 2 632 Suchanfragen innerhalb von 1 136 Stunden getätigt wurden. Dies erlaubt eine Hochrechnung für die Testkriterien.

$$\frac{4540}{2632} \cdot 1163h \approx 2006h \quad (11)$$

Im Folgenden wird angenommen, dass hierfür keine Skalierung erforderlich ist und zu jedem Zeitpunkt die kleinste Instanz⁸ ausreichend ist. Ebenfalls würde jede Suchanfrage im Schnitt v Veranstaltungen zurückliefern und jede einzelne Veranstaltung wird i mal indiziert, so ergibt sich die Summe zu:

$$2006h \cdot 0,112\$ + 0,10\$ \cdot i \cdot \frac{3600}{1000} + 4540 \cdot v \cdot 1,13KB \frac{0,09\$}{10^5KB} \approx \quad (12)$$

$$224,67\$ + 0,36\$ \cdot i + 0,04\$ \cdot v \quad (13)$$

Anhand der Teilbeträge für Datenübertragung und Indizierung wird ersichtlich, dass diese unabhängig von den Faktoren i und v kaum ins Gewicht fallen. Hauptaugenmerk gilt der Gebühr für die Nutzungsdauer. Eine Umrechnung mit aktuellen Wechselkurs⁹ ergibt einen Wert von 206,86 €.

Es sei jedoch angemerkt, dass die Nutzung von Autovervollständigung mit einer Vielzahl zusätzlicher Suchanfragen einhergehen würde. Entsprechend müsste der Faktor v deutlich nach oben korrigiert werden, was wiederum zu einem Anstieg der Gesamtsumme führen würde.

7.7 QUELLEN FÜR TESTFEHLER

Die in Anhang A aufgeführten Tests bergen jedoch auch Fehlerpotential, wodurch das Testergebnis möglicherweise verfälscht werden könnte. Ein kritischer Punkt hierbei sind die Testdaten, welche einerseits nicht aus dem Zielsystem ReEvent stammen und andererseits in ihrem Umfang beschränkt sind. [BCC10, S. 443] weist auf Testdatenbanken für Messungen der Effektivität mit einem Umfang von 500 000 Dokumenten hin – um ein Vielfaches mehr als die hier verwendeten 30. Ein weiterer Einschnitt hinsichtlich ReEvent: Sobald eine externe Suchmaschine wie Solr verwendet wird, muss für dieses ein Import-

⁸ „search.m3.medium“

⁹ 1,0861\$ pro €, Stand 03.01.2016

skript entwickelt werden, um die Daten aus der ReEvent-Datenbank in die Suchmaschine zu überführen. Auch bei der Suche wird nicht direkt die Schnittstelle der jeweiligen Suchmaschine verwendet, es wird zudem PHP-Code seitens ReEvent ausgeführt. In den Tests wurde allerdings meist direkt eine (REST-)Schnittstelle des jeweiligen Suchsystems aufgerufen. Je nach Implementierung der Skripte für Import und Suche könnte beispielsweise die Effizienz variieren.

Insbesondere bei Tests, die die Effizienz¹⁰ betreffen, muss bedacht werden, dass die Tests nicht auf einem Produktivserver der Stadt Ingolstadt durchgeführt wurden.

Kritik muss auch daran geübt werden, dass sowohl Testskript als auch Suchmaschine in den meisten Fällen am selben Rechner installiert waren. Dies kann unter Last dazu führen, dass beide Anwendungen um Systemressourcen konkurrieren und somit das Testergebnis verfälscht wird.

Schlussendlich können für einzelne Suchmaschinen auch Erweiterungen oder Konfigurationen existieren, durch welche die hier geschilderten Anforderungen besser bewältigt werden. Wie bereits in Abschnitt 7.2 erwähnt, wird stets eine unveränderte Installation eines Suchsystems getestet, falls nichts anderes angegeben wurde.

¹⁰ vergleiche Abschnitt 7.5.2

8.1 AUSWERTUNG DER TESTERGEBNISSE

Die Ergebnisse der in Anhang A beschriebenen Tests werden in Tabelle 5 aufgeführt. Als Testsieger lässt sich Apache Solr ausmachen, welches sich mit knappen Vorsprung vor der in PostgreSQL integrierten Volltextsuche platziert hat. Wenn auch mit einem Hauch von Willkürlichkeit behaftet und dadurch in ihrer Aussagekraft eingeschränkt, so erlauben die Tests dennoch einen groben Einblick in die Stärken und Schwächen der einzelnen Systeme.

TRIGRAMME Überrascht hat so beispielsweise der Trigramm-basierte Ansatz: Obwohl überragend bei der Erkennung von Rechtschreib- und Tippfehlern sinkt die *Performanz* mit steigender Anzahl der hinterlegten Datensätze stark. Tests ergaben, dass insbesondere das vergleichsweise lange Freitextfeld für die Veranstaltungsbeschreibung hierfür verantwortlich war: Lässt man dieses beim Trigramm-Vergleich außen vor, so reduziert sich die durchschnittliche Abfragedauer auf ähnliche Werte wie die der anderen Systeme.

LIKE-SUCHE Ebenfalls erwähnenswert ist das hohe Resultat der **LIKE**-basierten Suche für die *Effektivität*. Dieses Ergebnis ist möglicherweise dem geringen Umfang und der niedrigen Qualität der Testdaten und -Anfragen geschuldet. Wie auch für die Trigramme gilt für die grundsätzlich SQL-basierte Suche, dass komplexere Filter-Funktionen nur mit Hilfe von Erweiterungen möglich sind, die in diesen Test nicht mit einbezogen wurden.

AMAZON CLOUDSEARCH Auch für eine Überraschung gut war das kostenpflichtige Produkt von Amazon: Erst gegen Ende der Evaluation stellte sich so heraus, dass Amazon CloudSearch selbst wiederum auf Apache Solr basiert.¹ Insbesondere die API für Suchabfragen ähnelt sich stark. Auch wenn sich die Kosten – wie in Abschnitt 7.6.2 kalkuliert – in Grenzen halten, so rechnet sich eine cloudbasierte Lösung vermutlich erst bei höheren Lastanforderungen als für die Veranstaltungsverwaltung derzeit in Betracht kommen. In diesem Fall könnte es hilfreich sein, mit Apache Solr schon zuvor auf eine Lösung mit ähnlicher Schnittstelle zu setzen, um die Migration zu vereinfachen. Einen Nachteil von Amazon CloudSearch, der von den Tests nicht erfasst wurde, stellen die hohen Wartezeiten nach Konfigurationsänderungen dar, welche die Entwicklung einer Anbindung an diese Suche erschweren könnten. So ist das System nach einer Änderung des Dokumentschemas erst nach zehn Minuten bis einer halben Stunde wieder voll einsatzbereit.

POSTGRESQL – VOLLTEXTSUCHE In puncto *Latenz und Durchsatz* punktete stattdessen die in PostgreSQL integrierte Volltextsuche. Lediglich bei der Erkennung von Rechtschreibfehlern schwächelte die Lösung – was sich aber möglicherweise durch eine Kombination mit Trigrammen oder Levenshtein ausgleichen lässt.

¹ siehe <http://aws.amazon.com/de/cloudsearch/faqs/>

APACHE SOLR erweist sich schließlich der Punktzahl nach als Testsieger. Die bei Evaluation gewonnen Erfahrungen aus dem Umgang mit dieser Search Engine sowie in diesem Zusammenhang durchgeführte Recherchen lassen jedoch darauf schließen, dass Solr die bei Weitem umfangreichste der hier thematisierten Anwendungen ist. Ähnlich wie PostgreSQL kann auch Solr erweitert werden: So kann die Suchmaschine mit *Apache Tika*² kombiniert werden, um auch beliebige Dateien wie PDF, Word oder MP3 nach Textinhalten oder Metadaten zu durchsuchen. In Verbindung mit der freien OCR-Software *tesseract* ermöglicht es die Indizierung von Bildinhalten.

8.2 EMPFEHLUNG

In Hinblick auf die Integration in ReEvent würde sich die in PostgreSQL eingebaute Volltextsuche als einfach umzusetzende Lösung empfehlen, welche bereits viele Leistungsmerkmale einer vollwertigen Suchmaschine mit sich bringt. Da der von ReEvent verwendete OR-Mapper Doctrine wie bereits in Abschnitt 2.3 dargestellt auch PostgreSQL unterstützt, wäre in diesem Fall nur eine einzelne Datenhaltung notwendig – die Synchronisation einer zusätzlichen Datenbank wie beim Einsatz von Apache Solr würde entfallen. Auch für andere, einfache Anwendungen würde sich der Einsatz dieser Lösung aus Kostengründen empfehlen. Beispielsweise könnte man die Suche im Altsystem INsite für die restliche Dauer der Entwicklung von ReEvent hierauf umstellen, um bereits jetzt eine hochwertige Volltextsuche zu erhalten. Vor Einrichtung der PostgreSQL-basierten Volltextsuche würde es sich empfehlen, die Indexnutzung nochmals mit einer Kopie von Live-Datenbanken zu testen und zu überprüfen, ob nicht doch ein Datenbankindex eingerichtet werden sollte.³

Soll die Suche allerdings umfangreicher implementiert werden – insbesondere in Hinblick auf externe Dateien oder Geodaten – so wäre Apache Solr das Mittel der Wahl. Den höheren Funktionsumfang erkaufte man sich allerdings mit erhöhten Konfigurations- und Entwicklungsaufwand. Vor allem die Konfiguration des Dokumentenschemas erfordert einiges an Zeit für die Einarbeitung, wie während den Testläufen aufgefallen ist. Dafür erhält man ein Suchsystem, welches nahezu alle Anforderungen aus Kapitel 3 erfüllen kann. Sollte die Menge der Anfragen oder die Anzahl der zu durchsuchenden Elemente stärker zunehmen als für die hier verwendeten Tests prognostiziert, so kann Apache Solr auch als verteilte Search Engine eingerichtet werden um die Last auf mehrere Knoten zu verteilen.

8.3 AUSBLICK

Unabhängig von den Testergebnissen lässt sich feststellen, dass auch in Zeiten von NoSQL-Datenbanken die relationalen Vertreter durchaus eine Existenzberechtigung haben. Vor allem PostgreSQL konnte mit der integrierten Volltextsuche die eigene Funktionsvielfalt unterstreichen.

Auch wenn die Suchfunktion einer Anwendung meist eher in den Hintergrund tritt, so zeigt diese Arbeit dennoch, dass das Feld der Volltextsuche ein komplexes und umfangrei-

² vergleiche <https://tika.apache.org/1.11/formats.html>

³ vergleiche hierzu die Diskussion zu Datenbankindizes in Abschnitt 7.6.1, wobei angemerkt wurde, dass PostgreSQL eingerichtete Indizes scheinbar nicht nutzt

ches ist. Die hier abgegebene Empfehlung erlaubt es nun eine Anbindung der gewählten Suchlösung an ReEvent zu implementieren. Ein möglicher Ansatz hierbei könnte die in Klassendefinitionen verwendeten Annotationen für Datentypen auswerten und Suchanfragen und Indizierungsanforderungen daraus generisch erstellen. So könnten möglicherweise die bereits vorhandenen Annotationen für Doctrine wiederverwendet werden, um eine *schema.xml*-Datei für Apache Solr zu erzeugen. Eine derart entwickelte Suchanbindung wäre somit nicht notwendigerweise spezifisch für ReEvent und würde daher auch eine Wiederverwendung in andere Flow-Applikationen erlauben. Möglicherweise könnte ein solches Modul als Open Source-Veröffentlichung einen Grundstein für die weitere Verbreitung des Flow-Frameworks darstellen.

ANHANG

Folgende Tabellen definieren die einzelnen Testabläufe für die Evaluation in Kapitel 7. Jeder Test erhält eine eindeutige Nummer, einen Titel, eine Beschreibung des Testablaufs, ein Benotungsschema und ein Gewicht β .

#1 Volltextsuche – Effektivität ($\beta_1 = 5$)

TESTABLAUF Hierzu wird eine Datenbank mit 30 veranstaltungsbezogenen Datensätzen aufgebaut, je zehn Datensätze sind einem Suchbegriff zugeordnet. Anschließend wird mit jedem der drei Suchbegriffe eine Suche durchgeführt und über alle Suchanfragen hinweg Precision und Recall berechnet. Die Suchbegriffe lauten „Bus“, „Stadtrat Sitzung“ sowie „Viehmarkt“.

BENOTUNG Für α wird das F-Maß berechnet, welches in Abschnitt 7.5.1 dargestellt wurde.

#2 Volltextsuche – Durchsatz ($\beta_2 = 2$)

TESTABLAUF 10 000 Veranstaltungsdatensätze werden im Suchsystem hinterlegt. Die Daten stammen aus INsite und bestehen aus Veranstaltungsname, Beschreibung und Ort. Die Anfragen basieren auf den ausgewerteten Webserverlogs von INsite. Anschließend werden 500 Suchanfragen gestartet. Es wird die Zeit vom Start der ersten Anfrage bis zum Erhalt des letzten Ergebnisses gemessen. Das Testskript startet acht Threads, welche die 500 Anfragen parallel abarbeiten sollen.

BENOTUNG Die Note durch eine Normierung auf das beste Testergebnis ermittelt. Sei d_{max} der maximale Durchsatz, so ergibt sich $\alpha = \frac{d}{d_{max}}$.

#3 Volltextsuche – Latenz ($\beta_3 = 3$)

TESTABLAUF 10 000 Veranstaltungsdatensätze werden im Suchsystem hinterlegt. Die Daten stammen aus INsite und bestehen aus Veranstaltungsname, Beschreibung und Ort. Anschließend werden 500 Suchanfragen gestartet. Es wird die Dauer jeder einzelnen Anfrage gemessen. Wie auch schon beim Test zuvor werden acht Threads gestartet, welche die 500 Anfragen parallel abarbeiten sollen.

BENOTUNG Bei der Analyse von Latenzen sind insbesondere hohe Werte kritisch. Entsprechend wird ein Durchschnitt über die längsten 5 % der Anfragen gebildet. Alle 0,2 Sekunden dieses Wertes wird α um 0,1 reduziert.

#4 Filterung nach Datum ($\beta_4 = e$)

TESTABLAUF Ein Dokument wird mit Beginn- (01.01.2014 12:30 Uhr) und Endezeitpunkt (02.01.2014 09:00 Uhr) versehen. Getestet wird ob das Dokument durch Angabe eines Zeitraums vor Beginn oder nach Ende aus der Ergebnismenge entfernt wird, sowie ob es bei einem überlappenden Zeitraum in der Ergebnismenge bleibt.¹

BENOTUNG Neben der offensichtlichen direkten Filterung von Veranstaltungen würde dies auch die Abfrage zeitpunktabhängiger Daten erlauben. Bei korrekter Filterung $\alpha = 1$. Bei fehlerhafter Filterung, oder falls die Filterung systembedingt nicht durchgeführt werden kann $\alpha = 0$.²

#5 Filterung nach numerischer ID ($\beta_5 = 2$)

TESTABLAUF Ist das Suchsystem in der Lage, nach numerischen Identifikatoren zu filtern? Neben der Einsatzmöglichkeit zur Filterung nach direkten Eigenschaften der Veranstaltung² könnte somit auch das Konzept der Workspaces abgebildet werden, indem jedem Workspace eine eindeutige ID zugeordnet wird.

BENOTUNG $\alpha = 0$ falls die Filterung nicht möglich ist, $\alpha = 1$ andernfalls.

#6 Umkreissuche ($\beta_6 = 1$)

TESTABLAUF Getestet wird eine Filterung auf Basis von GPS-Koordinaten. Hierbei werden zwei Dokumente jeweils mit den folgenden GPS-Koordinaten versehen: Rathausplatz Ingolstadt (48.762951, 11.425548) und Eiffelturm Paris (48.858363, 2.294438). Es wird überprüft, ob das System in der Lage ist, Veranstaltungen im Radius $r = 10km$ um den Punkt (48.760335, 11.438925)³ zu bestimmen. Lediglich die erste der beiden GPS-Koordinaten liegt innerhalb des Radius.

BENOTUNG Bei korrekter Filterung $\alpha = 1$. Bei fehlerhafter Filterung, oder falls die Filterung systembedingt nicht durchgeführt werden kann $\alpha = 0$.

#7 Geschwindigkeit der Indizierung ($\beta_7 = 1$)

TESTABLAUF Es wird überprüft, wie lange es dauert, 1 000 Veranstaltungsdatensätze zu indizieren oder abzuspeichern. Die Daten stammen aus INsite und bestehen aus Veranstaltungsname, Beschreibung, Ort, Beginn und Ende. Es wird jeweils der Median aus drei Testdurchläufen verwendet. Es wird die clientseitige Dauer zwischen Beginn und Ende der Indizierungsanforderung gemessen. Dementsprechend kann es sein, dass das Suchsystem asynchron weiterarbeitet und der eigentliche Vorgang länger dauert als gemessen.

¹ in Anlehnung an das Suchformular von INsite

² beispielsweise der Veranstaltungsort, der per Datenbank-ID identifiziert wird

³ Saturnarena Ingolstadt, GPS-Koordinaten wurden mit Hilfe des Onlinekartendienstes Google Maps bestimmt

BENOTUNG Für jede ganze Sekunde, die der Indizierungsvorgang benötigt, wird 0,1 von α abgezogen, minimaler Wert für α ist 0. Bei SQL-basierten Suchsystemen wird die Dauer aus dem Unterschied zwischen normaler Eintragung der Daten und Eintragung bei einer mit Index versehenen Tabelle berechnet.

#8 Mehrsprachige Inhalte ($\beta_8 = 1$)

TESTABLAUF Es wird überprüft, ob das Suchsystem nach eigener Aussage (beispielsweise Dokumentation) diverse für ReEvent relevante Sprachen unterstützt.

BENOTUNG α setzt sich aus den Einzelbewertungen für Deutsch(0,6), Englisch(0,2), Chinesisch(0,05)⁴, Russisch(0,05), Französisch(0,05) und Italienisch(0,05)⁵ zusammen. Wird davon ausgegangen, dass die Suche unabhängig von der Sprache funktioniert, erhält das Suchsystem volle Punktzahl.

#9 Rechtschreibfehlererkennung ($\beta_9 = 2$)

TESTABLAUF Die deutschsprachige Wikipedia stellt unter https://de.wikipedia.org/w/index.php?title=Wikipedia:Liste_von_Tippfehlern/Für_Maschinen&oldid=105950498⁶ eine Liste von 1 963 häufig auftretenden Rechtschreibfehlern bereit.

Hiervon wird je der korrekte Teil in die Suchmaschine übertragen und anschließend durch eine Abfrage, welche die fehlerhafte Variante enthält überprüft, ob das Suchsystem den Rechtschreibfehler erkennen kann.

BENOTUNG α entspricht dem auf zwei Nachkommastellen gerundeten Anteil der korrekt erkannten Begriffe.

#10 Durchsuchung von externen Dateien ($\beta_{10} = 1$)

TESTABLAUF Es wird überprüft, ob das Suchsystem in der Lage ist, HTML, Microsoft Word docx, und PDF-Dateien ohne vorherige Aufbereitung zu indizieren. Weitere Tests beinhalten OCR und die Durchsuchung von MP3-Metadaten.

BENOTUNG α setzt sich aus den Einzelbewertungen für HTML(0,3), PDF(0,3), Word(0,3), OCR(0,05) und MP3(0,05) zusammen.

#11 Jährliche Kosten ($\beta_{11} = 2$)

TESTABLAUF Fallen durch das Suchsystem jährliche Kosten an, so werden diese durch dieses Kriterium erfasst. Eine Hochrechnung der Kosten geschieht anhand der Nutzungsstatistiken von INsite mit einem Aufschlag von 20 %. Es ergeben sich somit 4 540 Anfragen,

⁴ traditionell und vereinfacht

⁵ In Anlehnung an die Städtepartnerschaften der Stadt Ingolstadt, weitere Sprachen wie Polnisch oder Türkisch wurden ausgeklammert, vergleiche <http://www.ingolstadt.de/partnerstaedte>

⁶ Permalink, Stand 02.01.2016, bearbeitet von den Wikipedia-Nutzern „Dachaz dewiki“ und „Das Robert“ und bereitgestellt unter der Creative Commons Attribution-ShareAlike 3.0 Unported Lizenz

3 600 Veranstaltungen mit durchschnittlich 1,13 KB Speicherbedarf. Quellen hierzu stellen das Diagramm 16 und der Abschnitt 3.3.3 dar. Die folgende T-SQL⁷ Anfrage A.1 gibt den gesamten benötigten Speicherplatz der angegebenen Tabelle von INsite sowie die Anzahl der eingetragenen Zeilen zurück. Hieraus lässt sich der Speicherbedarf pro INsite-Veranstaltung abschätzen.

Unter anderem aufgrund der Konzepte aus dem Kapitel 2 (Workspaces, Mehrsprachigkeit, zeitpunktabhängige Daten) können aus ein und derselben Veranstaltung mehrere Dokumente entstehen – in Abhängigkeit von der Nutzung dieser Aspekte. Somit kann selbst bei einer gleichbleibenden Anzahl von Veranstaltungen mit einer gesteigerten Anzahl an Dokumenten gerechnet werden. Indirekte Kosten wie für Wartung, Strom und Internetanbindung werden nicht in die Kalkulation mit einbezogen.

BENOTUNG

$$\alpha = \begin{cases} 1 & \text{für } \textit{Kosten} = 0 \text{ €} \\ \max(0, 1 - \frac{1}{10} \lceil \frac{\textit{Kosten}}{100 \text{ €}} \rceil) & \end{cases} \quad (14)$$

#12 Hervorhebung von Suchbegriffen ($\beta_{12} = 1$)

TESTABLAUF Verfügt das Suchsystem über eine Highlighting-Funktion, um Teile des Suchbegriffs in der Trefferliste optisch hervorheben zu können?

BENOTUNG $\alpha = 1$ falls das Suchsystem hierzu in der Lage ist, $\alpha = 0$ andernfalls.

1 **EXECUTE** sp_spaceused 'VV_Veranstaltungen'

Listing A.1. T-SQL Abfrage sp_spaceused

#13 Wildcards in Benutzereingaben ($\beta_{13} = 0,2$)

TESTABLAUF Ist es möglich, den Suchbegriff um Wildcards zu ergänzen?

BENOTUNG $\alpha = 1$ falls Wildcards unterstützt werden oder systembedingt nicht notwendig sind, $\alpha = 0$ andernfalls.

#14 Boolesche Operatoren ($\beta_{14} = 0,2$)

TESTABLAUF Kann der Benutzer einzelne Suchbegriffe mit booleschen Operatoren wie AND, OR und NOT versehen oder verknüpfen?

BENOTUNG α erhöht sich um $\frac{1}{3}$ für jeden unterstützten Operator.

⁷ SQL-Dialekt von Microsoft SQL Server

B | TESTERGEBNISSE

Test	β	Solr	Volltext	CloudSearch	LIKE	Trigramme
# 1 Volltextsuche – Effektivität	5	0,38	0,4	0,4	0,59	0,46
# 2 Volltextsuche – Durchsatz	2	0,33	1	0,11	0,26	0,01
# 3 Volltextsuche – Latenz	3	0,9	0,9	0,7	0,8	0
# 4 Filterung nach Datum	2	1	1	1	1	1
# 5 Filterung nach numerischer ID	2	1	1	1	1	1
# 6 Umkreissuche	1	1	0	1	0	0
# 7 Indizierungsgeschwindigkeit	1	1	1	0	1	1
# 8 Mehrsprachige Inhalte	1	1	0,95	1	1	1
# 9 Rechtschreibkorrektur	2	0,78	0,17	0,9	0	0,97
# 10 Durchsuchung von Dateien	1	0	0	0	0	0
# 11 Jährliche Kosten	2	1	1	0,7	1	1
# 12 Suchbegriff-Highlighting	1	1	1	1	0	0
# 13 Wildcards-Unterstützung	0,2	1	0	0	1	0
# 14 Boolesche Operatoren	0,2	1	1	1	0	0
Summe	23,4	17,42	15,99	14,72	14,07	12,26
Platzierung		1	2	3	4	5

Tabelle 5. Resultate der Tests aus Anhang A. Volltext, LIKE und Trigramme beziehen sich auf die jeweiligen Suchfunktionen in PostgreSQL.

	recall	precision		Latenz in s	Durchsatz in $\frac{\text{Anfragen}}{\text{s}}$
LIKE	0,5	0,73	LIKE	0,50	55,4
Trigramme	0,375	0,6	Trigramme	12,74	1,97
Volltext	0,25	1	Volltext	0,32	212,3
Solr	0,25	0,8	Solr	0,31	69,3
CloudSearch	0,25	1	CloudSearch	0,61	23,4

(a) Recall und Precision

(b) Latenz und Durchsatz

Tabelle 6. Messwerte bei einzelnen Tests



PESSIMISTISCHES UND OPTIMISTISCHES SPERREN

Im Abschnitt 2.6 wurde das Workspace-Konzept von ReEvent vorgestellt. Hierbei können Concurrency-Probleme auftreten. Dieses Kapitel diskutiert Ansätze zur Auflösung oder Umgehung solcher Probleme in Bezug auf ReEvent.

PESSIMISTISCHES SPERREN entspricht dem, was klassischerweise als *Locking* bezeichnet wird. Zu Beginn der Änderung erhält der Prozess¹, der eine Änderung durchführen möchte, einen *write lock* für das zu ändernde Objekt. Dieser Lock entspricht dem Privileg, das Objekt ändern zu dürfen. Möchte ein anderer Prozess eine Änderung an diesem Objekt durchführen, so muss er warten bis der erste Prozess diesen Lock wieder freigegeben hat. Wurden alle gewünschten Änderungen durchgeführt, kann das Objekt gespeichert werden – ein Lost Update ist nicht möglich, da kein anderer Prozess in der Zwischenzeit eine Änderung durchführen konnte [Harog, S. 311].

Aus der Verwendung dieses Verfahrens entstehen jedoch weitere Probleme. Zu aller erst können keine zwei Editoren gleichzeitig eine Änderung an beispielsweise derselben Veranstaltung durchführen – was eine Einschränkung in der User Experience darstellt. Wird eine Veranstaltung bereits editiert, soll heißen das für diese Veranstaltung der Lock belegt ist, muss an einem weiteren Editor eine entsprechende Meldung ausgegeben werden.

Einen weiteren Nachteil liegt in der Art, in der der Lock wieder freigegeben wird. Dies ist an eine Aktion des Benutzers gekoppelt, nämlich dem Abspeichern der Änderungen. Versäumt der Benutzer dies durchzuführen, oder möchte er die Änderungen zu einem deutlich späteren Zeitpunkt fortsetzen, so sind die geänderten Objekte dauerhaft für fremde Änderungen gesperrt. Um eine solche Situation wieder aufzulösen, muss der Lock an ein Timeout gekoppelt werden. Hält der Editor den exklusiven Schreibzugriff länger als eine festgelegte Dauer, werden seine Änderungen verworfen und der Lock wieder freigegeben. Für diesen Timeout muss ein Mittelweg zwischen Bearbeitungsdauer und maximal erträglicher Wartezeit für weiteres Editieren des Objekten gewählt werden.

Als letztendlich klassisches Problem von nebenläufigen Prozessen kann man *Deadlocks* bezeichnen. Wird bei der Bearbeitung einer Veranstaltung auch der Veranstaltungsort editiert, so wird für jedes der beiden Objekte der exklusive Schreibzugriff benötigt. Auch die historischen Schritte aus 2.5 und die Mehrsprachigkeit in 2.4 werden über mehrere zusammenhängende Objekte realisiert, für welche gegebenenfalls einzeln Locks verteilt werden müssten. Angenommen die beiden Benutzer A und B aus obigen Beispiel möchten Anpassungen sowohl an einer Veranstaltung als auch am zugehörigen Ort durchführen. A erhält den Lock für die Veranstaltung und B für den Ort. Nun möchten beide jeweils das zweite Objekt editieren – können dies jedoch nicht, da der andere den Lock für das Objekt hält. Wenn nun beide Benutzer darauf warten, dass der andere seinen exklusiven Schreibzugriff abgibt, liegt ein Deadlock vor. [Harog, S. 314] zählt zwei verschiedene Möglichkeiten auf, Deadlocks zu beheben. *Detection* bedeutet Situationen zu erkennen, in welchen ein Dead-

¹ Prozess ist im folgenden gleichbedeutend mit Benutzer, in der Literatur wird Concurrency häufig an einem abstrakten Prozessbegriff dargestellt

lock vorliegt. In diesem Fall muss sich das System für einen der Benutzer entscheiden, welcher alle benötigten Locks erhält – der Benutzer, der den exklusiven Schreibzugriff zuvor hielt, verliert diesen. Nachdem der erste Benutzer seine Arbeiten abgeschlossen hat, kann der zweite seine Arbeit fortsetzen. Alternativ dazu kann ein Benutzer vor Beginn der Änderung alle zu ändernden Objekte auswählen und darf mit der Editierung nur beginnen, wenn er auch alle hierzu notwendigen Locks erhält. Dies wird üblicherweise dadurch realisiert, dass die Locks in einer feststehenden Reihenfolge angefragt werden.² Setzt man diese Anforderung jedoch in den Kontext, dass ein Benutzer von ReEvent üblicherweise kein Wissen über das Domänenmodell des Systems hat – welches zur Auswahl der Objekte notwendig wäre – wird klar, dass dies nur mit starken Einschränkungen der Usability oder großem Mehraufwand bei der Implementierung umsetzbar wäre.³

OPTIMISTISCHES SPERREN stellt eine Alternative zum pessimistischen Ansatz dar, welcher eingesetzt werden kann, wenn die Wahrscheinlichkeit für auftretende Probleme – wie Lost Update – als gering erachtet wird. Dies ist der Fall, wenn Lesevorgänge im Vergleich zu Schreibvorgängen sehr viel häufiger auftreten. Das Vorgehen setzt voraus, dass von den zu bearbeitenden Daten eine Arbeitskopie erstellt wird. Lesezugriffe finden auf dem Original statt, Bearbeitung an einer der Kopien. Vor dem Abspeichern der Änderung wird überprüft, ob an dem Original zwischenzeitlich fremde Änderungen durchgeführt wurden. Wird kein solcher Konflikt entdeckt, können die eigenen Änderungen gespeichert werden. Tritt ein Konflikt auf – wie im obigen Beispiel – wird der Speichervorgang abgebrochen [Harog, S. 318]. Der Benutzer kann nun über die geänderten Daten informiert werden und anschließend erneut versuchen, seine Änderungen abzuspeichern.

² Die Reihenfolge kann beispielsweise anhand von globalen, eindeutigen Identifier erstellt werden. Objekte in Flow erhalten eine `Persistence_Object_Identifier` genannte UUID. Vergleiche hierzu <https://github.com/neos/flow/blob/41d1034a96e93daa057684c7860d7b49ed6d06fc/Classes/TYP03/Flow/Persistence/Aspect/PersistenceMagicAspect.php#L92>

³ Denkbar wäre beispielsweise ein Assistent, der den Benutzer Schritt für Schritt durch eine Auswahl der benötigten Objekte führt.

D

USERTRACKING FÜR VERHALTENSBASIERTE EMPFEHLUNGEN

Für die Empfehlung von Veranstaltungen auf Basis vorheriger Seitenaufrufe, welche in Kapitel 5 dargestellt wird, müssen mehrere Seitenaufrufe eines Nutzers miteinander in Verbindung gebracht werden können. Gründe hierfür zeigt Abschnitt 5.2 auf. Dieses Kapitel zeigt mögliche Ansätze hierfür.

D.1 HTTP COOKIE

HTTP-Cookies sind Schlüssel-Wert-Paare, welche durch HTTP Response Header oder Javascript gesetzt werden können. Bei fort folgenden Aufrufen der gleichen Domain wird der gesetzte Cookie als Parameter mit übertragen [Int11, S. 7, 25].

Durch Vergabe einer über alle Benutzer einer Webseite eindeutigen ID als Cookie kann dieser Benutzer über mehrere Anfragen hinweg identifiziert werden.

Cookies funktionieren jedoch nur, wenn der Browser diese auch abspeichert. RFC 6265 legt fest, dass der *Set-Cookie* auch ignoriert werden kann [Int11, S. 16]. Moderne Browser verfügen über Einstellungsmöglichkeiten, um das Setzen von Cookies einzuschränken oder gänzlich zu deaktivieren. Beispielsweise kann im aktuellen Google Chrome zwischen den folgenden Einstellungsmöglichkeiten gewählt werden (1) Cookies erlauben (2) Cookies bis zum Beenden des Browsers speichern (3) Cookies verbieten.

Des Weiteren sind Cookies üblicherweise an einen Timeout gebunden – welches allerdings vom Server durch das *Expires* Attribut im Response Header gesetzt wird. Auch können Cookies jederzeit vom Benutzer gelöscht werden.¹

Außerdem ist die Anwesenheit eines Cookies kein eindeutiges Identifikationsmerkmal: Einerseits könnte es sein, dass sich zwei Personen den selben Arbeitsplatz und den selben Browser teilen. Andererseits kann eine Person auch an zwei getrennten Browsern arbeiten, an beiden würde er einen anderen Cookie und somit eine andere Identität erhalten, wenn er die entsprechende Seite besucht.

Eine gesetzliche Unwägbarkeit betreffend der Verteilung von Cookies ist schließlich die EU-Richtlinie 2009/136/EG. Sie legt fest, dass „die Speicherung von Informationen [...] nur gestattet ist, wenn der betreffende [...] Nutzer [...] seine Einwilligung gegeben hat“ [EUR09, Artikel 2 Absatz 5]. Somit müsste ein Nutzer erst ausdrücklich seine Genehmigung erteilen, ehe ein Webserver Cookies überträgt. Im Gegensatz zu Verordnungen stellt eine EU-Richtlinie kein direkt geltendes Gesetz dar. Jedoch ist jedes EU-Land dazu verpflichtet, diese Richtlinie in das nationale Recht einzugliedern. Dies ist in Deutschland bis

¹ vergleiche vorherigen Absatz

dato noch nicht geschehen [Stö14].² Diverse Webseiten weisen Besucher daher per Einblendung oder Pop-Up darauf hin, dass diese Seite Cookies verwendet.³

All diese Faktoren schränken die Eignung für die längerfristige Identifikation des Benutzers ein.

D.2 FRONTEND-USER

Eine weitere Möglichkeit zur Identifikation stellt eine Mitglieder-Funktion für das Frontend dar. Hierbei könnte sich ein Besucher der Seite registrieren und anschließend wiederholt am Webserver anmelden. Üblicherweise geschieht die Authentifizierung des Benutzers über die Eingabe eines Benutzernamens und eines Passworts. Nach der Anmeldung wird wieder ein Cookie gesetzt, worüber der Benutzer über mehrere Anfragen hinweg dem selben Benutzerkonto zugewiesen werden kann.

Somit gelten wieder die selben Einschränkungen wie in Abschnitt D.1. Allerdings ist es möglich, die selbe Person geräteübergreifend zu identifizieren, da dieser wiederholt seine Identität per Benutzername und Passwort bereitstellt. Erneut kann es allerdings passieren, dass sich mehrere Benutzer ein Konto teilen.

In Bezug auf ReEvent könnten über einen Frontend-Bereich eventuell weitere Einstellungen für die Ausgabe getroffen werden, auch könnten einzelne Veranstaltungen vorgemerkt werden. Allein die Empfehlungsfunktion rechtfertigt den zusätzlichen Aufwand – sowohl in der Entwicklung als auch für jeden einzelnen Benutzer – nicht. Derzeit ist keine Umsetzung von Frontend-Usern in ReEvent geplant.

D.3 BROWSER FINGERPRINTING

Selbst wenn ein Nutzer seinen Browser derart konfiguriert hat, dass dieser keine Cookies annimmt, bestehen dennoch Ansätze diesen Nutzer zu *tracken*. Basierend auf Einstellungen oder Merkmale des Browsers oder des zugrundeliegenden Betriebssystems lässt sich ein Browser mittels *Browser Fingerprinting* mit einer gewissen Wahrscheinlichkeit wiedererkennen [Boo15, S. 185].

Ein Beispiel ist der User-Agent, der vom Browser bei jeder Anfrage mit übertragen wird.

```
Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like
→ Gecko) Chrome/47.0.2526.106 Safari/537.36
```

Listing D.1. User-Agent von Google Chrome 47 (64-bit) unter Windows 8.1 Pro

Auch per Javascript auslesbare Informationen können hierzu verwendet werden. Die Eigenschaften `outerHeight` und `outerWidth` des globalen `window`-Objektes liefern unter Google Chrome die Außenmaße des Browserfensters, das Array `navigator.plugins[]` enthält

² Wobei die zitierte Quelle in *heise online* darauf hinweist, dass Brüssel scheinbar davon ausgeht, dass bestehende deutsche Gesetze diese Grundlage bereits erfüllen.

³ aktuelle Beispiele hierfür sind <http://www.chip.de> oder <http://www.hp.com>

Einträge über installierte Browser-Plugins und deren Versionsnummern. Eigenheiten von CPU und GPU lassen sich per Benchmarking bestimmen [JZo].

Schließlich kann auch auf lokal installierte Schriftarten per Bruteforce getestet werden. Ein Ansatz hierzu weist einem HTML-Element einen vorgegebenen Text und eine Schriftart zu, beispielsweise Comic Sans.⁴ Anschließend überprüft man die Breite des umgebenden HTML-Elements, setzt die Schriftart per `font-family: 'Arial', 'Comic Sans MS'` auf die gesuchte Schriftart (Arial) mit Fallback zur vorherigen und überprüft erneut die Breite des Elements. Hat sich diese geändert, kann davon ausgegangen werden, dass die Schriftart installiert ist [Shao8]. Das Vorhandensein mancher Schriftarten deutet auch auf die Installation bestimmter Programme hin, welche diese Fontdateien automatisch einrichten. *Myriad Pro* und *Minion Pro* werden beispielsweise durch den Acrobat Reader installiert.

Generell gilt, je stärker ein Benutzer seinen Browser angepasst hat, umso leichter lässt sich dieser per Browser Fingerprinting eindeutig zuordnen [Boo15, S. 185].

Fingerprinting, welches auf Javascript basiert, funktioniert allerdings nur, solange Javascript auch im Browser aktiviert ist.

⁴ Laut [Shao8] eigne sich Comic Sans hierfür besonders gut, da sich die Schriftart vergleichsweise stark von anderen unterscheide.

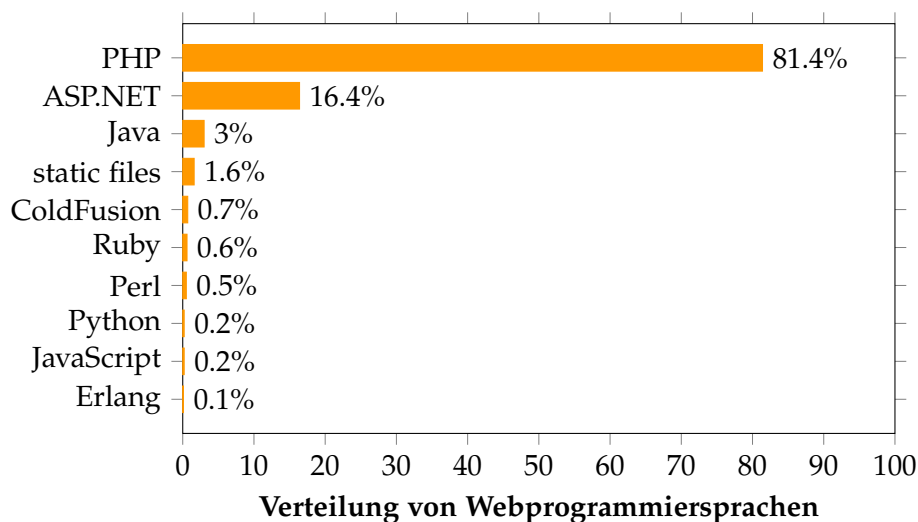


Abbildung 15. Verteilung von Webprogrammiersprachen, ermittelt von W3Techs/Q-Success DI Gelbmann GmbH. Die Grafik zeigt den Anteil der jeweiligen Sprache an den Top 10 Millionen des Alexa Rankings. Da Webseiten gleichzeitig mehrere Technologien verwenden können, kann die Summe 100% übersteigen. Sprachen mit einem Anteil unter 0,1% werden nicht aufgeführt. [QSu15]

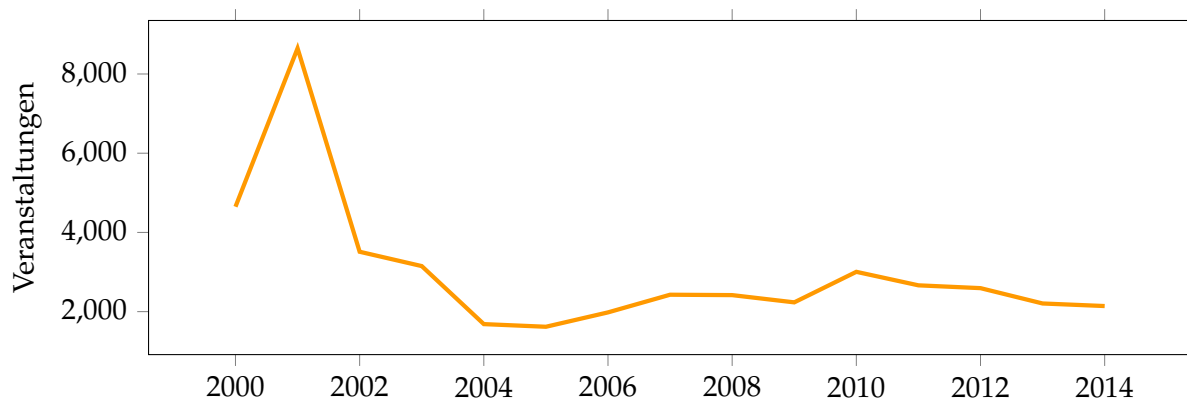


Abbildung 16. Veranstaltungen pro Jahr in INsite. Der Zeitbereich reicht vom Jahr 2000 bis 2014. Ausschlaggebend für die Zuordnung zu einem Jahr ist das Beginn-Datum der Veranstaltung, nicht das Ende-Datum. Durchschnittlich werden pro Jahr 2 995 Veranstaltungen hinzugefügt, insgesamt verwaltet die INsite-Datenbank 44 938 Veranstaltungen. Die Daten wurden aus der INsite-Datenbank auf Basis der SQL-Abfrage aus Listing G.1 gewonnen.

F

TABELLEN

Framework	letzte Änderung	Anmerkungen
Coldbox	14.04.2015	
CFWheels	20.10.2015	
Fusebox	11.05.2012	
Mach II	15.01.2014	Weiterentwicklung wurde offiziell für beendet erklärt [Tea13]
Model Glue	14.10.2014	

Tabelle 7. Auflistung diverser Webframeworks für ColdFusion. „letzte Änderung“ bezieht sich auf den master-Branch der folgenden github-Repositories. <https://github.com/coldbox/coldbox-platform>
<https://github.com/cfwheels/cfwheels>
<https://github.com/fusebox-framework/Fusebox-ColdFusion>
<https://github.com/Mach-II/Mach-II-Framework>
<https://github.com/modelglue/modelglue-framework>
Stand: 25.10.2015

G | LISTINGS

```
1 SELECT
2     COUNT(VV_Veranstaltungen.Veranst_ID) AS events,
3     YEAR(VV_Veranstaltungen.Datum_von) AS _year
4 FROM
5     VV_Veranstaltungen
6 GROUP BY
7     YEAR(VV_Veranstaltungen.Datum_von)
8 ORDER BY
9     _year DESC
```

Listing G.1. SQL Statement zur Auflistung der jährlichen Neueintragungen in INsite

```
1 SELECT
2     *
3 FROM
4     event
5 WHERE
6     to_tsvector(
7         coalesce(name, '') || ' ' ||
8         coalesce(description, '') || ' ' ||
9         coalesce(venue, '')
10    ) @@ plainto_tsquery('Suchbegriff')
11 LIMIT 10
```

Listing G.2. Volltextsuche mit PostgreSQL

```

1 SELECT
2     *
3 FROM
4     event
5 WHERE
6     name % 'Suchbegriff'
7 OR
8     description % 'Suchbegriff'
9 OR
10    venue % 'Suchbegriff'
11 LIMIT 10

```

Listing G.3. N-Gramm basierte Suche mit PostgreSQL

```

1 SELECT
2     *
3 FROM
4     event
5 WHERE
6     name LIKE '%Suchbegriff%'
7 OR
8     description LIKE '%Suchbegriff%'
9 OR
10    venue LIKE '%Suchbegriff%'
11 LIMIT 10

```

Listing G.4. LIKE basierte Suche mit PostgreSQL

```

1 http://<AWS Server ID>.us-east-1.cloudsearch.amazonaws.com/
   ↪ 2013-01-01/search?q=Suchbegriff&q.parser=dismax

```

Listing G.5. Suche in Amazon CloudSearch mit *dismax* Parser

```

1 http://localhost:8983/solr/<core>/select?q=Suchbegriff
   ↪ &defType=edismax&qf=name_s+venue_s+description_t&stopwords=true

```

Listing G.6. Suche in Apache Solr mit *edismax* Parser

LITERATUR

- [Adv15] Advantic Systemhaus GmbH. *iKISS - Anmeldung*. Hrsg. von Advantic Systemhaus GmbH. 2015. (Besucht am 05. 11. 2015) (siehe S. 24).
- [Aks11] Andrew Aksyonoff. *Introduction to Search with Sphinx. From installation to relevance tuning*. eng. Sebastopol: O'Reilly Media Inc, 2011. 147 S. (siehe S. 20).
- [Ama15] Amazon Web Services, Hrsg. *Amazon CloudSearch. Developer Guide*. 2015. URL: <http://docs.aws.amazon.com/cloudsearch/latest/developerguide/cloudsearch-dg.pdf> (besucht am 27. 12. 2015) (siehe S. 38).
- [Bal09] Helmut Balzert. *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. ger. 3. Auflage. Lehrbücher der Informatik. Heidelberg: Spektrum Akademischer Verlag, 2009 (siehe S. 19, 21).
- [Bal99] Heide Balzert. *Lehrbuch der Objektmodellierung. Analyse und Entwurf ; mit CD-ROM*. ger. Lehrbücher der Informatik. Heidelberg: Spektrum Akad. Verl., 1999. 575 S. (siehe S. 11).
- [BCC10] Stefan Büttcher, Charles L. A. Clarke und Gordon V. Cormack. *Information retrieval. Implementing and evaluating search engines*. eng. Cambridge, Mass.: MIT Press, 2010. 606 S. (siehe S. 23, 31, 33, 34, 48, 49, 51).
- [Bel15] Rachid Belaid. *Postgres full-text search is Good Enough!* 2015. URL: <http://rachbelaid.com/postgres-full-text-search-is-good-enough/> (besucht am 01. 01. 2016) (siehe S. 44, 48).
- [BG81] Philip A. Bernstein und Nathan Goodman. „Concurrency Control in Distributed Database Systems“. en. In: *ACM Computing Surveys* 13 (2 1981), S. 185–221 (siehe S. 17).
- [BKH11] Michael Bielitz, Christoph Klümpel und Pascal Hinz. *TYPO3-Handbuch für Redakteure. [Web-Inhalte optimal aufbereiten]*. ger. 3. Aufl. Beijing: O'Reilly, 2011. 574 S. (siehe S. 6).
- [BL11] Helmut Balzert und Peter Liggesmeyer. *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb*. ger. 3. Aufl. Lehrbücher der Informatik. Heidelberg: Spektrum Akademischer Verlag, 2011 (siehe S. 14).
- [Boo15] Carina Boos. *Verbraucher- und Datenschutz bei Online-Versanddiensten. Automatisierte Einschätzung der Vertrauenswürdigkeit durch ein Browser-Add-on*. Univ., Diss.–Kassel, 2015. ger. Bd. 1. ITeG - Interdisciplinary Research on Information System Design. Kassel: Kassel Univ. Press, 2015. 511 S. (siehe S. 65, 66).
- [Dre11] Mark Drew. *Railo 3 beginners guide. Easily develop and deploy complex applications online using the powerful Railo server*. eng. Learn by doing. Birmingham, U.K: Packt Pub, 2011. 339 S. (siehe S. 5).
- [EUR09] EUROPEAN PARLIAMENT AND OF THE COUNCIL. 2009/136/EG. 2009/136/EG. 2009 (siehe S. 64).

- [FAC10] Ben Forta, Charlie Arehart und Raymond Camden. *Adobe ColdFusion 9. Web application construction kit : Volume 1: Getting started*. eng. Berkeley, Calif: Peachpit, 2010. 581 S. (siehe S. 5, 29).
- [Goo11] Google, Hrsg. *Nutzungsbedingungen. Getting started with Custom Search*. 2011. URL: <https://support.google.com/customsearch/answer/1714300> (besucht am 13. 12. 2015) (siehe S. 37).
- [Goo12] Google, Hrsg. *Google Search Appliance. Getting the Most from Your Google Search Appliance*. Version 7.0. 2012. URL: http://static.googleusercontent.com/media/www.google.com/de//support/enterprise/static/gsa/docs/admin/70/gsa_doc_set/quick_start/quick_start.pdf (besucht am 27. 12. 2015) (siehe S. 36).
- [Goo13] Google, Hrsg. *Return on Information: Improving your ROI with Google Enterprise Search. How Google search solutions can boost your bottom line*. 2013. URL: https://static.googleusercontent.com/media/www.google.com/en/us/enterprise/search/files/Internal_Search_ROI.pdf (besucht am 26. 12. 2015) (siehe S. 36).
- [Goo15a] Google, Hrsg. *Google Search for Work. Google Site Search*. 2015. URL: https://www.google.de/intx/de/work/search/products/gss.html#pricing_content (besucht am 13. 12. 2015) (siehe S. 37).
- [Goo15b] Google, Hrsg. *Planning for Search Appliance Installation. How Does the Search Appliance Work?* Version 7.4. 2015. URL: http://www.google.com/support/enterprise/static/gsa/docs/admin/74/gsa_doc_set/planning/planning.html#1073674 (besucht am 13. 12. 2015) (siehe S. 36).
- [GPo8] GPM Deutsche Gesellschaft für Projektmanagement e.V. und PA Consulting Group. *Ergebnisse der Projektmanagement Studie 2008. - Erfolg und Scheitern im Projektmanagement* -. de. 2008. URL: http://www.gpm-ipma.de/fileadmin/user_upload/Know-How/Ergebnisse_Erfolg_und_Scheitern-Studie_2008.pdf (besucht am 07. 11. 2015) (siehe S. 19).
- [GP14] Trey Grainger und Timothy Potter. *Solr in action*. eng. Online-Ausg. Shelter Island, NY: Manning, 2014. 1 S. (siehe S. 2, 21, 31, 34–36, 39, 40, 43, 49).
- [Gra14] Marcus Grande. *100 Minuten für Anforderungsmanagement. Kompaktes Wissen nicht nur für Projektleiter und Entwickler*. ger. 2., aktual. Aufl. Wiesbaden: Springer Vieweg, 2014. 146 S. (siehe S. 28).
- [Har09] Jan L. Harrington. *Relational database design. Clearly explained*. eng. 3rd ed. Burlington, MA: Morgan Kaufmann/Elsevier, 2009. 1 S. (siehe S. 62, 63).
- [Int11] Internet Engineering Task Force, Hrsg. *HTTP State Management Mechanism*. U.C. Berkeley, 1. Apr. 2011 (siehe S. 64).
- [Int14] Internet Live Stats. *Anzahl der Webseiten weltweit in den Jahren von 1992 bis 2014*. Hrsg. von Internet Live Stats. 2014. URL: <http://www.internetlivestats.com/total-number-of-websites/> (besucht am 06. 01. 2016) (siehe S. ix).
- [JZo] Arthur Janc und Michal Zalewski. *Technical analysis of client identification mechanisms*. o.J. URL: <https://www.chromium.org/Home/chromium-security/client-identification-mechanisms> (besucht am 17. 12. 2015) (siehe S. 66).

- [KM12] Krishna Kulkarni und Jan-Eike Michels. „Temporal features in SQL. 2011“. In: *ACM SIGMOD Record* 41 (3 2012), S. 34 (siehe S. 16).
- [Loc15] Josh Lockhart. *Modern PHP. New features and good practices*. en. Sebastopol, CA: O'Reilly Media, 2015. 1 online resource (1 volume) (siehe S. 14).
- [MC09] Robert C. Martin und James O. Coplien. *Clean code. A handbook of agile software craftsmanship*. eng. Robert C. Martin series. Upper Saddle River, NJ: Prentice-Hall, 2009. 431 S. (siehe S. 14).
- [Men15] Rainald Menge-Sonnentag. *Programmiersprachen-Top 10: Objective-C raus, Ruby wieder rein*. de. Hrsg. von Heise Medien GmbH & Co. KG. 2015. URL: <http://www.heise.de/developer/meldung/Programmiersprachen-Top-10-Objective-C-raus-Ruby-wieder-rein-2837323.html> (besucht am 22. 10. 2015) (siehe S. 5).
- [Mic15a] Microsoft, Hrsg. *Sortierung und Unicode-Unterstützung*. SQL Server 2014. 2015. URL: [https://msdn.microsoft.com/de-de/library/ms143726\(v=sql.120\).aspx](https://msdn.microsoft.com/de-de/library/ms143726(v=sql.120).aspx) (besucht am 19. 11. 2015) (siehe S. 30).
- [Mic15b] Microsoft, Hrsg. *Volltextsuche*. SQL Server 2014. 2015. URL: [https://msdn.microsoft.com/de-de/library/ms142571\(v=sql.120\).aspx](https://msdn.microsoft.com/de-de/library/ms142571(v=sql.120).aspx) (besucht am 24. 11. 2015) (siehe S. 32).
- [Neo15a] Neos CMS, Hrsg. *Frequently Asked Questions*. e. 2015. URL: <https://www.neos.io/learn/faq.html> (besucht am 26. 12. 2015) (siehe S. 6).
- [Neo15b] Neos CMS, Hrsg. *Neos Split FAQ*. 2015. URL: <https://www.neos.io/learn/neos-split-faq.html> (besucht am 18. 11. 2015) (siehe S. 6, 7).
- [Nix14] Robin Nixon. *Learning PHP, MySQL, and JavaScript. With jQuery, CSS & HTML5*. eng. Fourth edition. Sebastopol, CA: O'Reilly, 2014. 1 S. (siehe S. 32).
- [OK07] Rob Orsini und Peter Klicman. *Rails Kochbuch. [rapid Web development mit Rails 1.2]*. ger. Dt. Ausg., 1. Aufl. Beijing: O'Reilly, 2007. 545 S. (siehe S. 8).
- [Ora15a] Oracle Corporation and/or its affiliates, Hrsg. *Changes in MySQL 5.7.6 (2015-03-09, Milestone 16)*. 2015. URL: <http://dev.mysql.com/doc/relnotes/mysql/5.7/en/news-5-7-6.html> (besucht am 29. 12. 2015) (siehe S. 34).
- [Ora15b] Oracle Corporation and/or its affiliates, Hrsg. *MySQL 5.5 Reference Manual. Chapter 15 Alternative Storage Engines*. 2015. URL: <https://dev.mysql.com/doc/refman/5.5/en/storage-engines.html> (besucht am 24. 11. 2015) (siehe S. 32).
- [Ora15c] Oracle Corporation and/or its affiliates, Hrsg. *MySQL 5.6 Reference Manual. 14.2.6.3 InnoDB FULLTEXT Indexes*. Version 5.6. 2015. URL: <https://dev.mysql.com/doc/refman/5.6/en/innodb-fulltext-index.html> (besucht am 06. 12. 2015) (siehe S. 35).
- [OW12] Thomas Ottmann und Peter Widmayer. *Algorithmen und Datenstrukturen*. ger. 5. Aufl. Heidelberg: Spektrum Akademischer Verlag, 2012 (siehe S. 15).
- [Poho8] Klaus Pohl. *Requirements engineering. Grundlagen, Prinzipien, Techniken*. ger. 2., korrigierte Aufl. Heidelberg: dpunkt-Verl., 2008. 748 S. (siehe S. 19–21).
- [Pot] F. Potencier. *Practical symfony - Doctrine edition*. Lulu.com (siehe S. 7).

- [PR15] Klaus Pohl und Chris Rupp. *Basiswissen Requirements Engineering. Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*. ger. 4. Aufl. s.l.: dpunkt, 2015. 195 S. (siehe S. 19, 20, 24).
- [QSu15] Q-Success. *Usage of server-side programming languages for websites*. en. Hrsg. von Q-Success. W3Techs. 2015. URL: http://w3techs.com/technologies/overview/programming_language/all (besucht am 22. 10. 2015) (siehe S. 67).
- [res] response informationsdesign gmbh & co. kg. *TYPO3 – Webdesign aus Ingolstadt*. de. URL: <http://respon.se/creative/typo3.html> (besucht am 28. 10. 2015) (siehe S. 6).
- [RKH14] Jochen Rau, Sebastian Kurfürst und Martin Helmich. *Zukunftssichere TYPO3-Extensions mit Extbase & Fluid. [der Einstieg in die Extension-Entwicklung]*. ger. 2. Aufl. Beijing: O'Reilly, 2014. 331 S. (siehe S. 6).
- [Rup14] Chris Rupp. *Requirements-Engineering und -Management. Aus der Praxis von klassisch bis agil*. ger. 6., aktualisierte u. erw. Aufl. München: Hanser, 2014. 570 S. (siehe S. 4, 21, 22, 28).
- [RWC12] Eric Redmond, Jim R. Wilson und Jacquelyn Carter, Hrsg. *Seven databases in seven weeks. A guide to modern databases and the NoSQL movement*. eng. Book version: P2.o. The pragmatic programmers. Dallas, Tex.: Pragmatic Bookshelf, 2012. 333 S. (siehe S. 32, 33, 39–41).
- [Shao8] Remy Sharp. *How to detect if a font is installed (only using JavaScript)*. 2008. URL: <https://remysharp.com/2008/07/08/how-to-detect-if-a-font-is-installed-only-using-javascript> (besucht am 17. 12. 2015) (siehe S. 66).
- [Sta14a] Stadt Ingolstadt, Hrsg. *Ingolstadt und seine Partnerstädte*. Stadt Ingolstadt Presse- und Informationsamt. 2014. URL: <http://www.ingolstadt.de/partnerstaedte/index.php?id=home> (besucht am 28. 11. 2015) (siehe S. 22).
- [Sta14b] Statista GmbH, Hrsg. *Google - Statista-Dossier. Statista-Dossier*. Hamburg: Statista GmbH, September 2014 (siehe S. 28).
- [Stö14] Marc Störing. *EU Kommission zur Cookie-Richtlinie: Vorgaben für Cookies gelten in Deutschland*. Hrsg. von Heise Medien GmbH & Co. KG. 2014. URL: <http://www.heise.de/newsticker/meldung/EU-Kommission-zur-Cookie-Richtlinie-Vorgaben-fuer-Cookies-gelten-in-Deutschland-2107770.html> (besucht am 16. 12. 2015) (siehe S. 65).
- [Tea13] Team Mach-II. *Sunsetting Mach-II*. 2013. URL: <http://mach-ii.com/> (besucht am 25. 10. 2015) (siehe S. 68).
- [The] The Doctrine Project, Hrsg. *Doctrine DBAL 2 documentation. 8. Types*. URL: <http://doctrine-dbal.readthedocs.org/en/latest/reference/types.html> (besucht am 13. 10. 2015) (siehe S. 12).
- [The15a] The Doctrine Project, Hrsg. *Doctrine DBAL 2 documentation. Introduction*. Version latest. 2015. URL: <http://docs.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/introduction.html> (besucht am 26. 12. 2015) (siehe S. 9).

- [The15b] The Neos Team. *Flow Framework Documentation*. Version Release 3.0.0. 2015. URL: <https://media.readthedocs.org/pdf/flowframework/3.0/flowframework.pdf> (besucht am 07. 10. 2015) (siehe S. 10, 12, 13).
- [The15c] The Neos Team. *Neos CMS Documentation*. Version Release 2.0.0-beta6. 2015. URL: <https://media.readthedocs.org/pdf/neos/2.0/neos.pdf> (besucht am 07. 10. 2015) (siehe S. 11, 12).
- [The16a] The PostgreSQL Global Development Group, Hrsg. *GiST and GIN Index Types*. 2016. URL: <http://www.postgresql.org/docs/9.1/static/textsearch-indexes.html> (besucht am 04. 01. 2016) (siehe S. 50).
- [The16b] The PostgreSQL Global Development Group, Hrsg. *pgtrgm*. 2016. URL: <http://www.postgresql.org/docs/9.1/static/pgtrgm.html> (besucht am 19. 01. 2016) (siehe S. 50).
- [TIO15] TIOBE Software BV. *TIOBE Index for October 2015*. en. TIOBE Software BV. 2015. URL: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html> (besucht am 21. 10. 2015) (siehe S. 5).
- [TWo6] Seyed M. M. Tahaghoghi und Hugh E. Williams. *Learning MySQL*. eng. Safari Books Online. Sebastopol, Calif.: O'Reilly, 2006. 618 S. (siehe S. 30).
- [W3T16] W3Techs. *Usage of content management systems for websites*. 2016. URL: http://w3techs.com/technologies/overview/content_management/all (besucht am 11. 01. 2016) (siehe S. 8).
- [Wie09] John-Harry Wieken. *SQL. Einstieg für Anspruchsvolle. - Title from resource description page (viewed May 7, 2009). - Includes index*. ger. 1. Aufl. Safari Books Online. München: Addison-Wesley Verlag, 2009. 429 S. (siehe S. 26, 30).

ERKLÄRUNG

Ich erkläre hiermit, dass ich die Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.¹

Ingolstadt, den 19. Januar 2016

Stefan Braun

¹ nach „Muster für die Erklärung nach § 18 Abs. 4 Nr. 7 APO HI“