Entwicklung erfolgreicher Webanwendungen

Leitfaden Webentwicklung 2015

www.bitkom.org

bitkom

Herausgeber

Bitkom e. V.
Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V.
Albrechtstraße 10 | 10117 Berlin

Ansprechpartner

Frank Termer | Projektleiter Software, Technologien und Märkte
T 030 27576-232 | f.termer@bitkom.org

Autoren

- Manuel Fischer | Bitkom e. V.
- Miriam Hänel |]init[AG für digitale Kommunikation
- Georg Klassen | Rohde & Schwarz GmbH & Co. KG
- Bastian Kolmsee | Trusted Shops GmbH
- Stephan Krause | CSC Deutschland GmbH
- Dr. Matthias Kurz | QUA-LiS NRW
- Stefan Polster | Rohde & Schwarz GmbH & Co. KG
- Alexander Rühl | SyroCon Consulting GmbH
- Rolf vom Stein | Jester Secure iT GmbH
- Markus Steinhauser| Testbirds GmbH
- Frank Termer | Bitkom e. V.

Gestaltung

Astrid Scheibe

Entwicklung erfolgreicher Webanwendungen

Leitfaden Webentwicklung 2015

Inhaltsverzeichnis

Verzeichnis der Abbildungen

Verzeichnis der Tabellen

1 Zielsetzung des Leitfadens

Immer häufiger entschließen sich Softwarehersteller, Anwendungen nicht mehr als Desktopanwendungen (auch: Rich-Client-Anwendungen), sondern als Webanwendung umzusetzen. Neben den klassischen Websites wie z. B. Firmenpräsenzen, Newsportale etc. werden nunmehr auch komplexe Office- oder Fachanwendungen in einer Web-Frontend-Backend-Architektur umgesetzt. Gründe dafür sind die stetig steigende Verfügbarkeit und Qualität von Internetverbindungen zum einen, als auch die vergleichsweise hohe Leistungsfähigkeit der Browser.

Mit diesem Leitfaden des Arbeitskreises Webentwicklung sollen IT-Verantwortliche und technische Entscheider einen ersten Einblick in ausgewählte Bereiche der Webentwicklung erhalten, wobei der Schwerpunkt vor allem auf aktuelle Anforderungen an zeitgemäße Webanwendungen gelegt wird. In der ersten Version des Leitfadens werden die Entwicklung einer modernen, barrierefreien Oberfläche, verschiedene Sicherheitsaspekte sowie Methoden zur Analyse und Erfolgsmessung näher beleuchtet. Zudem gibt das Papier einen Überblick über den groben Aufbau von Webanwendungen.

Zu den einzelnen Themenbereichen werden – sofern vorhanden und sinnvoll – vertiefende Informationen durch referenzierte Leitfäden des Bitkom zur Verfügung gestellt. Darüber hinaus werden Empfehlungen ausgesprochen, die jedoch eine Beratung durch einen Experten nicht ersetzen können.

Der Bitkom Arbeitskreis Webentwicklung möchte mit dem vorliegenden Leitfaden eine Ausgangsbasis zur Verfügung stellen, welche permanent weiterentwickelt und um weitere Bausteine sowie Leitfäden aus anderen Arbeitskreisen des Bitkom ergänzt werden soll. Ferner unterliegt die Entwicklung von Webanwendungen fortwährenden Veränderungen, die durch neue Technologien, Methoden und Anwendungsgebiete ausgelöst werden, diese sollen zukünftig ergänzt und konkretisiert werden.

2 Die Komponenten einer Webanwendung

Grundsätzlich zeichnen sich Webanwendungen dadurch aus, dass mindestens ihre Benutzeroberfläche im Webbrowser dargestellt wird, sowie eine Kommunikation mit einem Webserver stattfindet. Mit dem Aufkommen mobiler Anwendungen wird in diesem Zusammenhang häufig auch von Web-Apps gesprochen.

Im Gegensatz dazu stehen sogenannte native Anwendungen, die aufsetzend auf dem Betriebssystem bzw. innerhalb spezieller Laufzeitumgebungen operieren und somit entsprechende Funktionalitäten ausnutzen können. In aller Regel bedürfen native Anwendungen eines Installationsprozesses und werden speziell für ein Betriebssystem bzw. für eine Laufzeitumgebung entwickelt.

Gerade im mobilen Umfeld verschwimmt die Grenze zwischen Web-App und nativer Anwendung zunehmend. Für den Nutzer ist oftmals nicht ersichtlich, ob die aktuelle Ansicht in einem Browser ausgeführt wird oder in einer Art Container (App), welcher ebenfalls auf die Rendering Engine eines Browsers zurückgreift. Die dargestellten Inhalte können für beide Arten von Anwendungen aus derselben Quelle stammen. Aus diesem Grund ist die Unterscheidung zwischen Web-App und nativer Anwendung weniger aus Benutzersicht, wohl aber aus technischer Sicht notwendig.

2.1 Aufbau einer Webanwendung

Webanwendungen bestehen grundsätzlich aus einem Frontend und einem Backend. Die Art und Weise, wie diese Elemente miteinander verknüpft werden, wird über die so genannte Geschäftslogik der Anwendung gesteuert. Vereinfacht ausgedrückt ist dabei das Frontend das, was der Nutzer der Webanwendung angezeigt bekommt. Demgegenüber steht das Backend, das der Verwaltung, der Administration sowie der Dateneingabe und -pflege dient. Die Daten selbst können in einer Datenbank, einem Dateisystem oder auf einem externen Server abgelegt sein. Dies wird auch als Persistenzschicht bezeichnet.

2.1.1 Komponenten

Neben der Aufteilung in Front- und Backend kann die Architektur von Webanwendungen auch anhand ihrer Komponenten beschrieben werden:

- Die Benutzerschnittstelle (User Interface bzw. UI) umfasst alle grafischen Elemente, welche dem Nutzer Informationen oder Eingabemöglichkeiten zur Verfügung stellt. Beispielsweise zählen dazu Texte, Grafiken, Hyperlinks und Formulare.

- Die Prozesslogik steuert den Interaktionsablauf der Anwendung. Beispielsweise wird die Prozesslogik eines Online-Shops sicherstellen, dass der Anwender seine Zahlungsinformationen eingibt, bevor eine Bestellung abgeschlossen wird.

- Die Geschäftslogik verarbeitet alle Eingaben des Anwenders. Sie stellt somit das »Herz« der Anwendung dar und verwaltet alle Zugriffe auf die Datenbank. In einem Online-Shop wird die Geschäftslogik beispielsweise prüfen, welche Produkte überhaupt lieferbar sind.

- Die Datenhaltung beschreibt die Art der Speicherung aller nichtflüchtigen Daten der Anwendung. In einem Online-Shop werden hier beispielsweise die Bestellungen in einer Datenbank abgelegt.

2.1.2 Architekturansätze

Architekturansätze beschreiben auf einer abstrakten Ebene, wie Anwendungen aufgebaut sind, welche der genannten Komponenten mit welcher Technologie umgesetzt sind und wie sie miteinander verknüpft werden. In der Praxis unterstützen Software- und IT-Architekten bei der Konzeption von Webanwendungen, indem sie Hinweise auf erprobte Lösungen geben.

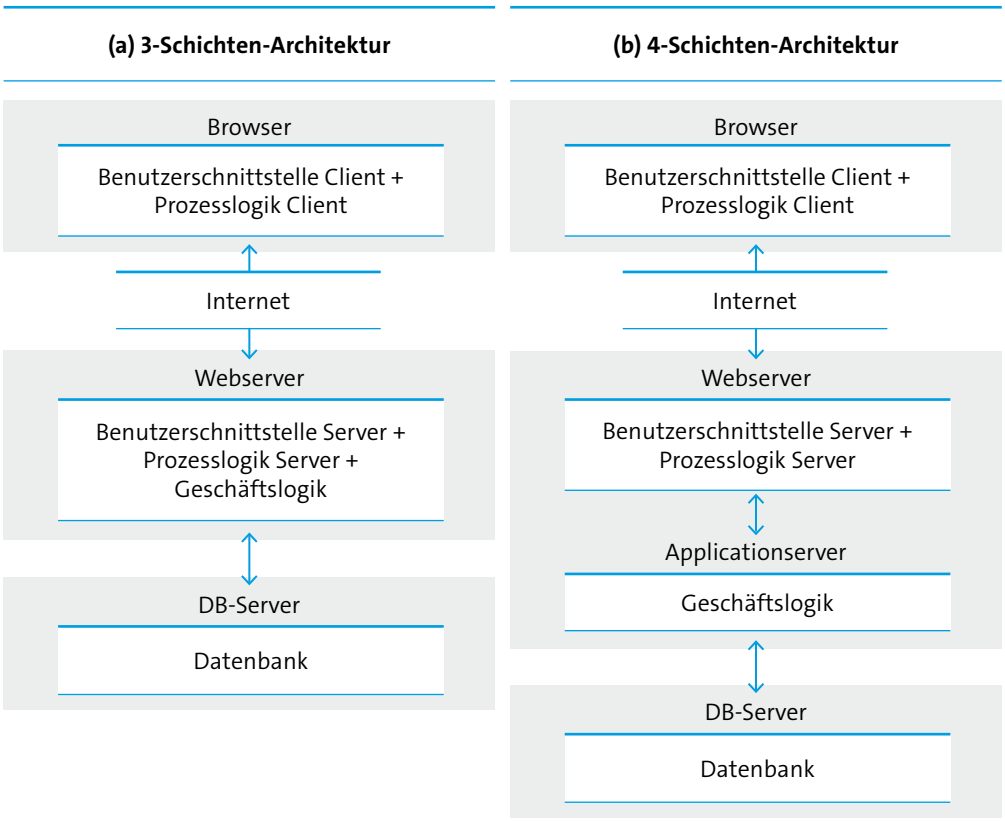

Abbildung 1: Architekturansätze für Webanwendungen

Quelle: Eigene Darstellung

Grundsätzlich zeichnen sich zwei verbreitete Architekturansätze ab (vgl. Abbildung 1):

- Bei der 3-Schichten-Architektur (vgl. Abbildung 1a) wird die Prozesslogik direkt mit der Geschäftslogik verwoben. Diese Architektur besteht damit aus der Benutzerschnittstelle, der Geschäftslogik und der Datenhaltung.

- Bei komplexen Webanwendungen wird meist die 4-Schichten-Architektur (vgl. Abbildung 1b) bevorzugt. Hierbei wird die Prozesslogik strikt von der Geschäftslogik getrennt. Dies ermöglicht

es, dass Geschäftslogik und Prozesslogik weitgehend unabhängig voneinander weiterentwickelt werden können. Häufig werden Prozess- und Geschäftslogik auch von unterschiedlichen Entwicklungsteams mithilfe unterschiedlicher Technologien entwickelt. So wird die Geschäftslogik meist in Form einzelner Services bereitgestellt[1]. Die für die Geschäftslogik verwendeten Frameworks[2] stellen Funktionen zur effizienten Kommunikation mit der Datenbank oder mit anderen Services zur Verfügung. Für die Prozesslogik sowie die Benutzerschnittstelle werden hingegen Frameworks eingesetzt, die bereits umfangreiche Komponentenbibliotheken für die Prozesslogik bereitstellen.

Die Benutzerschnittstelle sowie die Prozesslogik können in beiden Referenzarchitekturen sowohl Client- als auch Serverseitig realisiert werden, so dass zwischen Client- und Server-zentrierten Architekturansätzen unterschieden wird. Oftmals werden Client- und Server-seitige Prozesslogik kombiniert eingesetzt. Beispielsweise wird der Wechsel zwischen Formularen oft Server-seitig verwaltet, während einzelne Benutzereingaben mithilfe von JavaScript-Fragmenten Client-seitig geprüft werden.

Beim Server-zentrierten Ansatz führt in der Regel jede Aktion des Nutzers (z. B. der Klick auf einen Button) zu einer Anfrage beim Webserver, dessen Antwort zu einem vollständigen oder teilweisen Austausch der dargestellten Bildschirmansicht führt. Die Geschäftslogik befindet sich dabei fast ausschließlich auf dem Server. Hierbei zeigt der Browser (Client) nur an, was der Server ihm liefert. Nach jeder Eingabe wird der komplette Bildschirminhalt neu übertragen. Dieser Ansatz eignet sich gut für seitenbasierte Webanwendungen (z. B. für einen Bestellprozess aus mehreren Schritten).

Im Gegensatz zum Server-zentrierten Ansatz ähneln sogenannte Rich Internet Applications oder auch Client-zentrierte Webanwendungen in ihren Interaktionsmöglichkeiten und Reaktionszeiten den Desktop-Anwendungen. Hier übernimmt der Client einen großen Teil der Anwendungslogik. Die Kommunikation mit dem Webserver findet für den Nutzer unbemerkt mittels JavaScript-Technologien statt. Statt den kompletten Bildschirminhalt in Form eines HTML-Dokuments zu übertragen, werden nur wenige Informationen übertragen, die von der Client-seitigen Prozesslogik ausgewertet werden. Letztere wird dann die Client-seitige Benutzerschnittstelle instruieren, die Informationen entsprechend aufbereitet anzuzeigen. Obgleich hier nur bestehende Technologien wie JavaScript und HTTP genutzt werden, hat sich für dieses Prinzip der Begriff AJAX[3] etabliert. Mit HTML5 gibt es zudem die Möglichkeit, Daten auf dem Client-Rechner zu speichern und diese auch zu einem späteren Zeitpunkt wieder zu verwenden. Client-zentrierte Webanwendungen werden häufig eingesetzt, um entweder klassische Desktop-Anwendungen zu ersetzen oder komplexe Fachanwendungen zu realisieren. Auch aufwendige Online-Spiele verwenden diese technologische Basis.

In der Praxis werden der Server-zentrierte Ansatz und der Client-zentrierte Ansatz auch oft gemeinsam eingesetzt. So setzen beispielsweise Server-zentrierte Webanwendungen häufig auch Elemente von AJAX ein, um einzelne Funktionen mit einem besseren Nutzererlebnis zu versehen. So ist der Bestellprozess bei vielen Online-Shops häufig noch Server-zentriert realisiert.

1 Diese Services werden meist als so genannte Web-Services bereitgestellt und sind in der Regel über SOAP oder REST zugänglich.

2 Bei Frameworks handelt es sich um Hilfsmittel für Entwickler, auf deren Basis eigene Anwendungen entwickelt werden. Frameworks stellen meist wiederverwendbare Bibliotheken und Best Practices für die Lösung häufig auftretender Aufgabenstellungen zur Verfügung. Im Java-Umfeld ist beispielsweise das JavaEE-Framework für die Entwicklung Server-seitiger Geschäftslogik weit verbreitet.

3 AJAX steht für Asynchronous JavaScript and XML und dient der asynchronen Datenübertragung zwischen Client und Server (vgl. http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29).

Bei der Suche hingegen werden bereits – ganz im Sinne Client-zentrierter Webanwendungen – erste Suchergebnisse zurückgeliefert, während der Anwender noch die Suchbegriffe eingibt.

Empfehlung

Mit Verbreitung moderner und leistungsfähiger Browser sowie schneller Internetverbindungen lohnt sich der Einsatz einer Client-zentrierten Webanwendung. Durch ihre performante und komfortable Interaktion mit den Anwendern lösen sie immer häufiger auch leistungsfähige lokal installierte Anwendungen ab, die aufwändig installiert und nur mittels Updates aktuell gehalten werden können. Durch ihre geringen technischen Voraussetzungen sind Client-zentrierte Webanwendungen ferner häufig das Mittel der Wahl, um plattformunabhängige Cloud-Anwendungen zur Verfügung zu stellen. Zudem reduziert die Verlagerung der Prozesslogik, weg vom Server hin zum Client, die Anforderungen an einen Server. Dies ermöglicht es, bei gleichem Funktionsumfang Anwendungen kostengünstiger zu betreiben bzw. besser auf steigende Nutzerzahlen zu reagieren.

2.2 Das Web-Frontend

Der Begriff Web-Frontend beschreibt typischerweise die Darstellung von Texten, Grafiken und Interaktionselementen im Webbrowser. In diesem Zusammenhang wird häufig vom GUI (graphical user interface) oder kurz UI (user interface) gesprochen. Die Implementierung der einzelnen GUI-Elemente, wie beispielsweise Buttons, Textboxen oder Auswahlfelder, erfolgt in den meisten Fällen mittels der Seitenbeschreibungssprache HTML, der Formatierungssprache CSS sowie der Scriptsprache JavaScript. Weitere, auf zusätzliche Browser Plugins angewiesene Techniken, wären z. B. Adobe Flash oder Microsoft Silverlight.

Das Frontend einer Webanwendung trägt oft entscheidend zur Akzeptanz und zur positiven Beurteilung der Qualität der Software durch den Nutzer bei. Eine angemessene Bedienung, ein klares übersichtliches Layout sowie eine stabile, fehlertolerante Steuerbarkeit ermöglichen eine flüssige Interaktion und ein einfaches Auffinden der gewünschten Informationen.

2.2.1 HTML5/CSS3

Nach jahrelangem Stillstand in der Entwicklung neuer Web-Standards ist derzeit HTML5 vor allem durch die umfangreiche Unterstützung von Multimediaformaten die erste Wahl zur Neuerstellung eines Web-Frontends. Davon unabhängig unterstützen insbesondere Browser auf mobilen Geräten aber auch moderne Desktop-Browser HTML5 bereits in hohem Maße. Im Gegensatz zu den Vorgängerversionen umfasst HTML5 nicht nur Elemente zur Seitenbeschreibung, sondern auch viele weitere Technologien, beispielsweise zur Kontrolle von Multimediaelementen, Manipulation des Browser-Verlaufs, Drag & Drop, Offlineanwendungen und vieles mehr.

Mit HTML5 setzt sich auch der seit dem Jahr 2000 in Entwicklung befindliche CSS Level 3 Standard durch. Insbesondere die Unterstützung von Media-Queries und Grafikfiltern hat sich in den letzten Jahren deutlich verbessert und wird damit praxisrelevanter.

2.2.2 JavaScript

HTML und CSS sorgen für die Darstellung von Inhalten und Elementen im Browser, verfügen jedoch über eine verhältnismäßig eingeschränkte Dynamik. Deswegen kommt mit der zunehmenden Erwartung an eine benutzerfreundliche, interaktive Oberfläche, welche in der

Reaktionsgeschwindigkeit nativen GUIs gleicht, vermehrt JavaScript zum Einsatz. Die wachsende Komplexität browserbasierter Anwendungen erfordert jedoch einen deutlichen Mehraufwand in der Entwicklung und Pflege des Frontend und hier insbesondere des JavaScripts.

JavaScript wird beispielsweise verwendet, um Client-seitig HTML-Elemente (DOM) zu manipulieren, Plausibilitätsprüfungen von Nutzereingaben vorzunehmen oder auch Desktop-Funktionen wie etwa Drag&Drop nachzubilden. Zudem wird JavaScript verwendet, um eine Kommunikation über AJAX mit dem Webserver herzustellen und Daten zwischen Client und Server auszutauschen, ohne dass der Nutzer diese Interaktion direkt zu spüren bekommt. Die Verwendung von JavaScript ermöglicht eine hohe Dynamik der Webanwendung, welche es erlaubt, das »Look&Feel« von nativen Anwendungen nachzubilden und für den Nutzer eine sehr flüssige Interaktion zu gewährleisten.

Mit der Verbreitung des HTML5-Standards wird auch JavaScript in Zukunft weiter an Bedeutung gewinnen. Beide Technologien, HTML5 und JavaScript, werden nahtlos verschmelzen. Bereits seit einiger Zeit stehen stabile JavaScript-Frameworks zur Verfügung, mit denen interaktive, qualitativ hochwertige JavaScript-Lösungen entwickelt werden können.

2.2.3 Responsive Webdesign

Der Begriff Responsive Webdesign wurde von einem der führenden internationalen Fachmagazine »A List Apart« geprägt. Der Begriff umfasst verschiedene konzeptionelle und technische Ansätze, welche eine Anpassung von Weboberflächen an das jeweilige Browserfenster (Viewport) ermöglichen. Mit diesem Ansatz kann eine Weboberfläche jeweils optimiert für Smartphones, Tablets oder Desktoprechner angezeigt werden. Technische Voraussetzung auf Seiten des Webbrowsers ist die Unterstützung von Media Queries, sowie weiteren HTML5- und CSS3-Neuerungen.

Weiter gefasst bedeutet Responsive Webdesign die Bereitstellung einer Oberfläche optimiert für verschiedene Ausgabegeräte und Browser auf Basis einer Datenquelle und einer gemeinsamen Geschäftslogik. Der Begriff umfasst also u. U. nicht nur die Anpassung des Seitenrasters, sondern ebenfalls Optimierungen einzelner Bestandteile der Oberfläche für das jeweilige Ausgabemedium wie z. B. größere Bedienelemente für Touch-fähige Geräte oder auch die Auslieferung von hochauflösenden Grafiken und Bildern für Retina Displays.

Bereits im Planungsprozess sollte kritisch hinterfragt werden, wie und auf welchen Gerätetypen die spätere Webanwendung genutzt werden soll. Eine gute Planung bereits in der Konzeptionsphase spart Aufwände im späteren Entwicklungsprozess. Insbesondere folgende Fragestellungen sollten beantwortet werden:

- Wird das Frontend zukünftig auf unterschiedlichen Endgerättypen (Devices) genutzt werden?
- Können Aussagen zur jeweiligen Netzanbindung und Nutzungsszenarien getroffen werden?
- Welche Rolle spielt die Performance im Webprojekt?
- Gibt es diesbezüglich gesonderte Anforderungen an die mobile Nutzung?
- Sollen unterschiedliche Inhalte auf die verschiedenen Devices ausgeliefert werden?
- Sollen unterschiedliche Funktionalitäten für die Endgeräte bereitgestellt werden?
- Sollen bestimmte Fähigkeiten oder auch Software der Geräte, wie beispielsweise Lokalisierung per GPS, genutzt werden?

Anhand dieser Fragestellungen können Entscheidungen hinsichtlich des Umfangs der notwendigen Optimierungen und dem konzeptionellen Vorgehen getroffen werden. Sollen die gleichen Inhalte sowohl auf Desktop, als auch auf mobile Geräte ausgeliefert werden, empfiehlt sich unter Umständen eine Reduzierung der anzuzeigenden Elemente.
Die Konzeption sollte idealerweise für ein Tablet-Gerät erfolgen, da dieses den Kompromiss zwischen Smartphone mit sehr kleinem Viewport und Desktoprechner mit großem Viewport darstellt (»Tablet First«-Ansatz). Sollen unterschiedliche Inhalte für Desktop und mobile Geräte ausgeliefert werden oder Services auf bestimmten Geräten in den Vordergrund gestellt werden, sollten spezielle Techniken in Betracht gezogen werden, welche in Teilen unterschiedliche Inhalte auf die verschiedenen Geräteklassen ausliefern können oder sogar eine separate mobile Version in Betracht gezogen werden.

Empfehlung

Soll ein neues Projekt geplant werden, empfiehlt sich die Verwendung von HTML5 und CSS3. Eine Umstellung von älteren HTML Versionen lohnt sich insbesondere dann, wenn eine Vielzahl neuer mobiler Geräte unterstützt bzw. multimediale Inhalte auf diesen Plattformen genutzt werden sollen. Eine Umstellung auf HTML5 lohnt sich weniger, wenn kein Relaunch geplant ist und mobile Geräte nicht unterstützt werden sollen, wie das z. B. bei Intranet Anwendungen oder Fachanwendungen der Fall sein könnte.
Bei der Planung und Erstellung nutzerfreundlicher, interaktiver Webanwendungen wird in Zukunft kein Weg an einer JavaScript-Nutzung vorbei führen. Der eingeschränkte, uneinheitliche Sprachstandard wird hierbei durch mächtige Frameworks wie etwa AngularJS, JQuery, Prototype etc. ergänzt.
Eine flexible Oberfläche, welche es dem Nutzer ermöglicht, in verschiedenen Situationen und auf verschiedenen Endgeräten Inhalte und Services schnell und übersichtlich aufzufinden, wird immer selbstverständlicher. Statt für jede Geräteklasse unterschiedliche Websites zu entwickeln, kann auf Basis des gleichen Datenbestandes das Web Frontend Client-seitig mit Hilfe flexibler Raster und Media Queries anpassbar gestaltet werden. Zur Unterstützung verschiedener Funktionen oder einer performanten Auslieferung der Inhalte können Geräte, falls nötig, über Device Templates unterschieden werden. Bereits zu Beginn des Projekts sollte entschieden werden, wie mobile Geräte zukünftig unterstützt werden sollen. Fragen zum inhaltlichen Umfang der späteren Website, der gewünschten Performance, der möglichen Nutzungszenarien und zur Nutzung von gerätetypischen Funktionen und Erfordernissen sind essentiell für den Erfolg des Projekts.

2.2.4 Barrierefreie Webanwendungen

Moderne browserbasierte Fachanwendungen und Portale sind zunehmend im öffentlichen Interesse, bspw. bei eGovernment-Lösungen, Bürgerportalen etc. Hinzu kommen diverse EU-Richtlinien (bspw. EU-Mandat 376 etc.), die die Barrierefreiheit derartiger Webanwendungen explizit fordern werden.

2.2.4.1 Normen und Gesetze

International ist die Web Content Accessability Guideline (WCAG) die einschlägige und anerkannte Norm, die Barrierefreiheit für Websites definiert. Innerhalb der WCAG gibt es drei verschiedene Stufen: Stufe A sieht nur eine elementare Unterstützung für Menschen mit Behinderungen vor. Stufe AA, die weitestgehend mit der Deutschen Barrierefreie-Informationstechnik-Verordnung (BITV) 2.0 identisch ist, fordert deutlich weitergehende Barrierearmut von Webanwendungen. In der höchsten Stufe AAA sind alle Bedingungen der WCAG zu erfüllen. In den USA ist Barrierefreiheit in öffentlich zugänglichen Systemen über die sog. »Section 508« geregelt. Da das Thema Barrierefreiheit eine Schnittmenge mit der

Benutzbarkeit (Usability) von Webanwendungen hat, ist in diesem Zusammenhang auch die Norm EN ISO 9241 zu beachten.

2.2.4.2 Umsetzung

Einige Punkte der WCAG sind nicht technisch, sondern fachlich, inhaltlich bzw. semantisch umzusetzen und oft nicht objektiv messbar (bspw. »Beschreibt der alternative Text eine Grafik ausreichend?« oder »Strukturieren die Überschriften die Webseite sinnvoll?«). Diese Herausforderungen müssen bereits bei der Konzeption bzw. der späteren redaktionellen Pflege von Webanwendungen berücksichtigt werden.

Andere Forderungen wie bspw. die Skalierbarkeit/Linearisierbarkeit von Texten bei der Nutzung der Vergrößerungsfunktion des Browsers, die Tastaturbedienbarkeit oder die generelle Nutzung von deskriptiven Texten in Tabellen müssen in der Realisierung beachtet werden. Dabei helfen kommerzielle oder Open-Source-basierte GUI-Frameworks wie z. B. ICEfaces oder MyFaces Trinidad, die auf JSF und Java basieren. Die GUI-Komponenten dieser Frameworks berücksichtigen z.T. schon die Vorgaben der WCAG, so dass die Verwendung bereits eine Unterstützung für Menschen mit Behinderungen mit sich bringt.

Barrierearme Seiten müssen nicht mehr zwingend ohne JavaScript auskommen: um assistive Technologie wie z. B. Screenreader zu unterstützen stellt das World Wide Web Consortium (W3C) mit WAI-ARIA ein Hilfsmittel zur Verfügung.

2.2.4.3 Testen

Viele in der WCAG 2 geforderten Eigenschaften lassen sich Tool-gestützt prüfen: Die HTML-Validität, die Hinterlegung von alternativen Texten für Grafiken sowie die Kontraste von Schrift- und Hintergrundfarben. Einschlägige Werkzeuge hierfür sind bspw. AChecker[4] oder checkmycolors.com[5].

Mit dem Plugin WAT 2.0[6] für den Internet Explorer lassen sich bspw. CSS- oder TABLE-Anweisungen in HTML-Seiten ausblenden, so dass auch sehende Menschen einen Einblick in die linearisierte Form der Webseite bekommen, so wie sie sich auch für Nutzer eines Braille-Lesers darstellt. In Deutschland haben sich zwei Testverfahren etabliert. Zum einen der durch das vom BMAS geförderte BIK-Projekt entwickelte BITV-Test[7], zum anderen der vom Bundesverwaltungsamt herausgegebene BANU Test[8].

4 http://achecker.ca/checker/index.php

5 http://www.checkmycolours.com

6 http://www.paciellogroup.com

7 http://www.bitvtest.de/bitvtest.html

8 http://www.banu.bund.de/DE/Home/home_node.html

Empfehlung

In Lastenheften für Webanwendungen sollte nicht nur die WCAG referenziert werden, sondern die jeweils geforderte Stufe (A bis AAA). I. d. R. ist Stufe AA ausreichend. Soll ein deutschsprachiger Test durchgeführt werden, kann der BITV-Test entweder in Form eines Selbsttests oder durch Beauftragung eines Experten umgesetzt werden.
Grundsätzlich sollte auf spezielle Alternativ- oder Textversionen einer Website verzichtet werden.
Da die WCAG 2 für alle Arten von Webanwendungen Vorgaben macht, die oft aber für das aktuelle Projekt keine Bedeutung haben, empfiehlt es sich für das jeweilige Projekt, eine reduzierte Version der WCAG 2 zu erstellen, die diese nicht-relevanten Vorgaben (bspw. Lautstärken für Video-Sequenzen oder Flimmer-Raten) nicht enthält. Damit wird das Anwenden und Testen der WCAG für alle Beteiligten deutlich vereinfacht.
Aufgrund der Komplexität, Subjektivität und Vielfalt des Themas kann es sinnvoll sein entsprechendes Know-how von externen Personen zu Beginn in das Projekt zu integrieren. Oft sind hier wenige Beratertage ausreichend.

2.3 Das Web-Backend

Webanwendungen sind in besonderem Maße einer permanenten technologischen Weiterentwicklung unterworfen, da sich viele der eingesetzten Basistechnologien selbst noch in einer frühen Phase der Entwicklung befinden – beispielsweise wird HTML5 explizit als »Work in Progress« bezeichnet. Neue Möglichkeiten wie 3D-Rendering im Browser (WebGL) sind gerade erst im Entstehen. Und mit Web-Sockets wird die Dominanz von HTTP als Trägermedium zunehmend in Frage gestellt. Die explosionsartige Verbreitung von mobilen Geräten stellt völlig neue Anforderungen an Webanwendungen.

Unternehmen sind daher mit der Herausforderung konfrontiert, einerseits ihre bereits getätigten Investitionen in eine Webanwendung zu monetarisieren, wodurch das Interesse, regelmäßig neue Technologien in eine bestehende Anwendung zu integrieren, eher begrenzt ist. Andererseits müssen im hochdynamischen Internet-Segment neue Technologien proaktiv in Anwendungen integriert werden, um wettbewerbsfähig zu bleiben.

Um diesen Gegensatz aufzulösen, empfiehlt es sich, bei den Komponenten einer Webanwendung zwei Typen zu unterscheiden. Der erste Typ sind Komponenten, welche sich durch den technologischen Fortschritt kontinuierlich und teilweise auch radikal ändern werden. Dazu zählt insbesondere die Benutzerschnittstelle, welche die Interaktion mit dem Anwender steuert. Hier sind auch in Zukunft durch die rasante Entwicklung sowohl im Bereich der mobilen Geräte als auch der Desktop Rechner weitere Technologiesprünge zu erwarten. Aber auch Teile der Prozesslogik sind aufgrund neuer Benutzerschnittstellen, neuer Eingabeformen wie beispielsweise Spracherkennung oder neuer Kundenanforderungen Veränderungen unterworfen.

Der zweite Typ sind hingegen Komponenten, bei denen mehr Stabilität zu erwarten ist. Das betrifft vor allem die Geschäftslogik einer Anwendung, welche fachliche Logik in Form von Web-Services bereitstellt. Diese Services sind weitgehend unabhängig von den eingesetzten Technologien für die Benutzerschnittstelle sowie die Prozesslogik und sind daher in der Regel weniger von entsprechenden Technologieschüben oder Innovationen in der Benutzerführung betroffen.

Lose Kopplung ist hierbei kein Garant für eine geringe Änderungshäufigkeit. Denn bei den zunehmend angewendeten agilen Vorgehensweisen treten auch vermehrt schichtenübergreifende Inkremente auf. Dennoch ändern sich die in der Geschäftslogik realisierten elementaren fachliche Funktionen meist seltener als die Benutzerführung (d. h. Benutzerschnittstelle und

Prozesslogik). Dies gilt insbesondere für verteilte Anwendungen, die über Firmengrenzen hinweg eingesetzt werden und – ganz im Sinne Service-orientierter Architekturen (SOA) – Teile der Geschäftslogik in Form von Services für andere Anwendungen und Unternehmen anbieten.

Die weiter oben vorgestellte 4-Schichten-Architektur trägt dazu bei, diesen technologischen Wandel zu bewältigen, da so die Komponenten Benutzerschnittstelle, Prozesslogik, Geschäftslogik und Datenbank stärker voneinander getrennt sind. Durch klare und möglichst kompakte Schnittstellen zwischen diesen Komponenten können die Auswirkungen von Änderungen an einzelnen Komponenten stark begrenzt werden. Auf diese Weise können beispielsweise Apps und Websites für neue Gerätekategorien ohne größere zeitraubende Änderungen an der Geschäftslogik bereitgestellt werden.

Ferner ist es so auch möglich, mehrere Varianten einer Anwendung bereitzustellen, ohne die Geschäftslogik redundant zu entwickeln. Beispielsweise können die mobile Version sowie die Desktop-Version einer Webanwendung auf die gleiche Geschäftslogik zugreifen. Auch sind Szenarien denkbar, in der eine browserbasierte Webanwendung und eine mobile App die gleiche Geschäftslogik nutzen.

In Abbildung 2 ist eine solche beispielhafte Architektur dargestellt. Da mobile Anwendungen (Apps) meist die komplette Nutzerinteraktion (d.h. Benutzerschnittstelle und Prozesslogik) selbst verwalten, greifen sie in der Regel direkt auf die Geschäftslogik zu. Ferner zeigt die Erfahrung, dass es oft günstig ist, auch die Benutzerschnittstelle und Prozesslogik für Desktop-Browser möglichst komplett auf die Clientseite zu verlagern, da so insgesamt weniger Komponenten zu pflegen sind und eine größere Einheitlichkeit der Architektur herstellt wird.

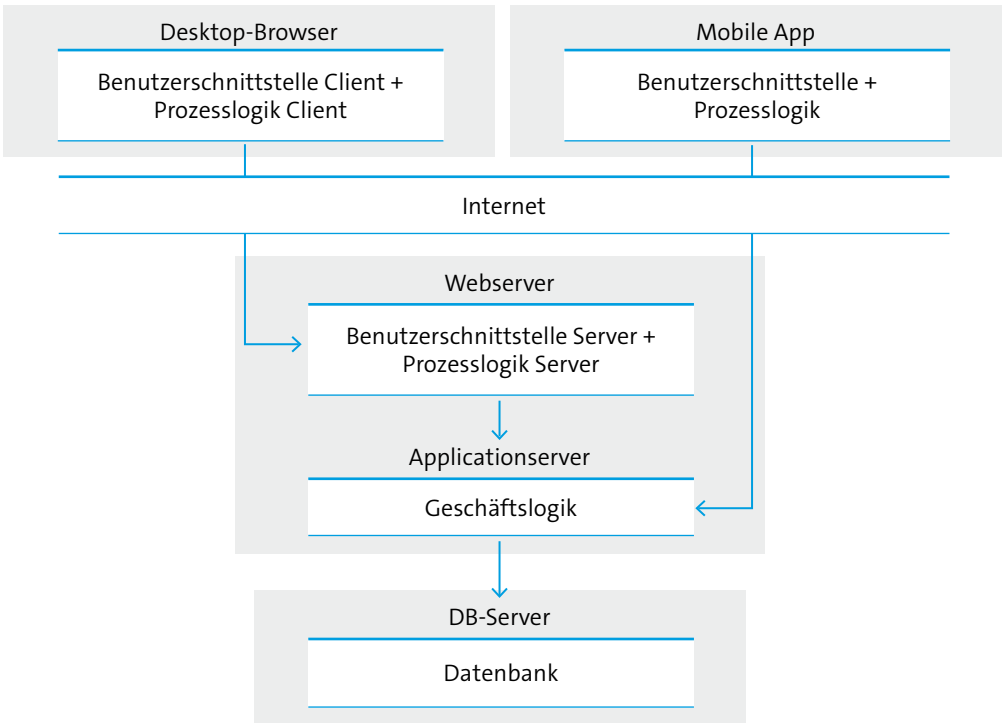

Abbildung 2: Client-Unabhängigkeit durch lose Kopplung der Geschäftslogik

Quelle: Eigene Darstellung

Wie bei der Frontend-Entwicklung gibt es auch für die Entwicklung des Backends leistungsfähige Frameworks. Diese stellen Entwicklern häufig benötigte Funktionen wie etwa Authentisierung zur Verfügung. Im Java-Umfeld sind beispielsweise Spring sowie JavaServer Faces bekannte Vertreter von Web-Frameworks. Viele dieser Frameworks umfassen auch die entsprechenden Frontend-Module. Auf diese Weise kann die Architektur von Front- und Backend stärker vereinheitlicht werden. Neben fertigen Komponenten erleichtern Frameworks die Entwicklung von wartbaren sowie sukzessive erweiterbaren Webanwendungen auch durch erprobte Best Practices, Entwurfsmuster und Architekturen.

Die lose Kopplung der Komponenten von Webanwendungen ermöglicht gerade in größeren Projekten die stärkere Spezialisierung im Entwicklungsteam. So werden Benutzerschnittstelle, Prozesslogik, Geschäftslogik und Datenbanken häufig von Entwicklern mit völlig unterschiedlichen Kompetenzprofilen umgesetzt (vgl. Tabelle 1). Die lose Kopplung der technischen Komponenten vereinfacht die Zusammenarbeit der verschiedenen Entwicklerrollen, da die technischen Abhängigkeiten klarer und weniger komplex sind.

Tabelle 1: Komponenten und Rollen im Entwicklerteam

Komponente	Maßgebliche Rollen
Benutzerschnittstelle	User Interface Designer
Prozesslogik	User Experience Designer Software-Entwickler Frontend
Geschäftslogik	Software-Entwickler Backend
Datenbank	Datenbank-Entwickler

Empfehlung

In neuen Projekten mit umfangreicher Geschäftslogik, bei denen absehbar oder zumindest wahrscheinlich ist, dass die Anwendung über einen längeren Zeitraum bestehen wird, sollte die Geschäftslogik ganz im Sinne der 4-Schichten-Architektur in Form von Services als eigene Schicht bereit gestellt werden, die von Prozesslogik und Benutzerschnittstelle unabhängig ist. Die Services der Geschäftslogik enthalten somit nur noch fachliche Operationen und keine Operationen, welche der Geschäftslogik – also der Steuerung von Abläufen der Benutzerschnittstelle – dienen.
Auf diese Weise können Benutzerschnittstelle und Prozesslogik rasch ausgetauscht werden, ohne die Geschäftslogik anpassen zu müssen. Dies ermöglicht es, neue Gerätetypen oder neue Browser-Technologien mit möglichst geringem Aufwand zu unterstützen. Das ist unverzichtbar, da insbesondere bei den Technologien für die Benutzerschnittstellen auch zukünftig weitere Veränderungen und Fortschritte zu erwarten sind.
Bei Webanwendungen, die nur eine begrenzte Lebensdauer aufweisen, da sie beispielsweise nur für einzelne Marketingaktionen entwickelt wurden, kann hingegen auch die schneller umzusetzende 3-Schichten-Architektur eine ökonomische Alternative darstellen. In Fällen, in denen die Geschäftslogik im Vergleich zur Prozesslogik und der Benutzerschnittstelle nur eine geringe Komplexität aufweist, wird eine Webanwendung ebenfalls nicht von einer 4-Schichten-Architektur profitieren können.
Grundsätzlich empfiehlt es sich, Frameworks für die Entwicklung von Webanwendungen einzusetzen. Auf diese Weise fließen viele Jahre von Erfahrungen in der Entwicklung von Webanwendungen in das konkrete Projekt ein. Angesichts des sich stetig verändernden Angebots von Webframeworks ist eine generelle Empfehlung nicht möglich. Eine entsprechende Entscheidung sollte stets in Zusammenarbeit mit einem erfahrenen Partner oder Berater gefällt werden. Grundsätzlich ist jedoch insbesondere bei Webanwendungen mit einem absehbar längeren Lebenszyklus anzuraten, auf erprobte Frameworks zu setzen, die eine gewisse Stabilität erreicht haben und für die auch auf Jahre hinweg Weiterentwicklung und Unterstützung durch den Hersteller gesichert sind. Anderenfalls besteht die Gefahr, nach kurzer Zeit in einer technologischen Sackgasse zu landen, die eine weitgehende Neuentwicklung erforderlich macht.

3 Testen von Webanwendungen

In diesem Kapitel werden zunächst einige Grundbegriffe, Verfahren und Bereiche im Thema Testen beleuchtet und dann speziell auf das Testen von Webanwendungen eingegangen.

Tabelle 2: Wichtige Begriffe aus dem Bereich Testen

Begriff	Bedeutung
Abweichung	Ereignis beim Testen, das genauer untersucht werden muss
Fehler	▪ Abweichung zwischen Sollverhalten (Anforderungen) und Istverhalten ▪ führt bei Auftreten zur Nichtbenutzbarkeit eines Features oder gar der Anwendung
Fehlerwirkung	(failure) sichtbare Auswirkung eines Fehlers
Fehlerzustand	(fault) Ursache des Fehlers
Mangel	▪ Anforderung/Erwartung nicht angemessen erfüllt ▪ führt bei Auftreten zur Einschränkung bei der Nutzung eines Features
Testen vs. Debuggen	▪ Testen heißt Aufdecken von Fehlern ▪ Debuggen heißt Auffinden der Ursache
Testmanagement	Planung, Aufwandsschätzung, Überwachung und Kontrolle von Testaktivitäten, die üblicherweise durch einen Testmanager erfolgen

3.1 Testprozess

Testen ermittelt die Qualität einer Software durch Vergleich von Soll und Ist und dient dem Auffinden von Fehlern. Ein vollständiger Test ist dabei praktisch unmöglich, daher ist eine sinnvolle Auswahl an Testfällen nötig. Der zu betreibende Aufwand für den Test ist projektabhängig – der Aufwand muss sich aber immer am Risiko von Fehlern ausrichten. Es ist also zu betrachten, wie wahrscheinlich der Eintritt eines Fehlers ist und wie hoch die Kosten in diesem Fall sind.

Der Test selbst sollte immer von einer anderen Person durchgeführt werden als der, die das zu testende Objekt (zum Beispiel ein Feature oder eine Softwarekomponente) erstellt hat, da auf diese Weise ein anderer Blickwinkel eingenommen wird (»Vier-Augen-Prinzip«). Hinzu kommt, dass erst durch das Testen die Qualität von Software messbar wird. Um strukturiert, effizient und nachvollziehbar zu testen, ist ein definierter Testprozess nötig. Zu einem vollständigen Testprozess gehören dabei Planung, Analyse/Design, Durchführung, Auswertung und Abschluss. Begleitend ist eine Steuerung und eine exakte Protokollierung des Tests nötig. Metriken können dabei helfen, den aktuellen Stand und die Entwicklung zu visualisieren – hier ist aber unbedingt ein pragmatischer Weg zu verfolgen, der die Metriken als Hilfsmittel nutzt, um mögliche Probleme aufzudecken, statt sich allein an Zahlenwerten zu orientieren.

3.2 Teststufen

Grundsätzlich sollte Testen so früh wie möglich und in jedem Fall begleitend zur Entwicklung stattfinden, da die Behebung von Fehlern oder gar grundsätzlichen Problemen umso teurer wird, je später diese entdeckt werden. So ist gerade im Zusammenspiel mit Hardware oft ein Zurückrufen ganzer Produktserien notwendig, wenn der Fehler erst nach der Auslieferung durch den Kunden entdeckt wird. Wird der selbe Fehler zum Entwicklungszeitpunkt bemerkt, so treten i. d. R. kaum Mehrkosten auf. Es gibt unterschiedliche Teststufen, die zu unterschiedlichen Zeitpunkten im Projekt und die sowohl punktuell als auch regelmäßig durchgeführt werden können.

3.2.1 Unittest

Bei einem Unittest wird eine Einheit des entwickelten Codes getestet. Dabei gilt als Grundsatz, dass dieser Test unabhängig von äußeren Einflüssen und ohne das Vorhandensein externer Systeme (z. B. einer Datenbank oder einer Messaging-Schnittstelle) laufen muss und nur die Logik innerhalb der Unit isoliert testen soll. Dazu ist es notwendig, sämtliche externe Referenzen durch sogenannte Mocks, d. h. simulierte Komponenten, zu ersetzen. Dabei zeigt sich, dass ein Entwickeln von Unittests parallel zum eigentlichen Code zu besser testbarem Code führt. Aber auch heute weit verbreitete Paradigmen wie Dependency Injection, also das Bereitstellen externer Referenzen von außen, erleichtern das Schreiben von Unittests, da sie dann leichter durch Mocks ersetzt werden können.

In diesem Zusammenhang ist auch das Thema Code-Coverage zu nennen, welches den Prozentwert der durch Unittest durchlaufenen Codezeilen angibt. Hier ist aber unbedingt ein pragmatischer Weg zu gehen, da allein eine hohe Code-Coverage keine Aussage über die Güte des Tests gibt. Das theoretische Ideal wäre im Test das Durchlaufen aller möglichen Wege durch den Code. Unter Aufwandsgesichtspunkten ist es aber sinnvoll, eine Auswahl zu treffen und sich an der zyklomatischen Komplexität zu orientieren, die besagt, wie viele verschiedene Wege es durch eine Unit gibt und hierüber komplexe Codestellen zu identifizieren, die einen erhöhten Testbedarf haben.

3.2.2 Integrationstest

Sollen mehrere Software-Komponenten im Verbund mit externen Systemen (z. B. einer Datenbank) getestet werden, geht dies über einen Unittest hinaus und wird als Integrationstest bezeichnet. Zudem werden hier im Gegensatz zu einem Unittest, der die Beschaffenheit des zu testenden Codes kennen darf (»white box«), das nach außen sichtbare Verhalten der betroffenen Komponenten getestet und die Interna nicht berücksichtigt (»black box«). Aber auch hier macht es Sinn, dass der Tester mit einem gewissen Know-how über die technische Realisierung an den Test geht, sprich in der Praxis ist auch der Integrations-Test in der Regel keine vollständige »black box«.

3.2.3 Systemtest

Im Systemtest wird ein komplettes System getestet. Im Projekt ist dies in der Regel eine eigene Phase im Anschluss an die Entwicklung und schon getätigte Integrationstests. Für einen Systemtest sind Testspezifikationen zu erstellen, die abgearbeitet und protokolliert werden.

3.2.4 Sicherheit

Die noch vorzustellende Bedrohungsmodellierung (vgl. Kapitel 4.3) liefert auch die Basis für die Sicherheitstests, die in den Entwicklungszyklus eingebettet werden sollten. Leider ist die gängige Praxis immer noch, dass nach Abschluss der Entwicklungs- und Implementierungsarbeiten, quasi unmittelbar vor (und oft auch nach) der Aufnahme des Wirkbetriebs ein mehr oder weniger wohldefinierter Sicherheitstest durchgeführt wird. Schwachstellen, die diese Tests identifizieren, müssen in der Regel dann sehr aufwendig und daher ressourcenintensiv am lebenden System behoben werden.

Prozessmodelle, wie OpenSAMM[9] oder BSIMM[10] (Building In Security Maturity Model), die die Best Practices der Entwicklung und des Betriebs von Software beinhalten, empfehlen unterschiedlich geartete Securitytests in den verschiedenen Phasen des Lebenszyklus der Software. So sind Code Reviews und Sicherheitstests auf Basis der Bedrohungsmodelle bereits während der Entwicklungsphase sinnvoll, d. h. auch beim noch nicht fertigen Produkt. Gerade die Sicherheitstests von Komponenten und APIs sollten parallel mit den QA-Tests durchgeführt und ausgewertet werden. Wird zu diesem Zeitpunkt getestet, dann können potentielle Schwachstellen oder Designfehler ressourcenschonender und schneller behoben werden, als später beim Betrieb.

Ist das Softwareprodukt produktionsreif sollte ein abschließender Penetrationstest durch erfahrene externe Kräfte, d.h. durch Tester, die nicht in den Entwicklungsprozess eingebunden waren, vor der eigentlichen Live-Stellung durchgeführt werden. Es empfiehlt sich jedoch, dass die Tester Zugriff auf die Ergebnisse vorangegangener Tests haben, da die Software und nicht die Kreativität der Tester getestet werden soll. Zudem sollte vorher geklärt werden, ob die Tester hinreichende Qualifikation und Erfahrung besitzen. Eine regelmäßige Wiederholung der Penetrationstests, insbesondere nach größeren Änderungen in der Software ist ebenfalls äußerst sinnvoll, um böse Überraschungen im Produktionsbetrieb zu vermeiden.

Eine zusätzliche Sicherheitszertifizierung durch einen geeigneten Zertifizierer kann zudem dokumentieren, dass die Anwendung ein hinreichendes Sicherheitsniveau hat und die begleitenden Prozesse dafür sorgen, dass dieses Niveau auch zukünftig Bestand hat. Jedoch ist bei Zertifizierungen der Anforderungskatalog[11] und die Prüfgrundlage für die Erteilung des Zertifikats das Maß aller Dinge und diese Grundlage sollte bei der Auswahl des Zertifikats im Vorfeld intensiv auf Tauglichkeit geprüft werden. Idealerweise beziehen sich die Anforderungen eines Zertifikats auf internationale Standards wie die ISO27000-Reihe und/oder erprobte Standards wie OWASP.

3.2.5 Abnahmetest

Der Abnahmetest dient der Bestätigung (oder Verweigerung) einer im Sinne des Vertrages korrekt erfüllten Leistung. Hier wird gegen einen zu Beginn mit dem Abnehmer der Software abgestimmten Kriterienkatalog verglichen.

9 siehe: http://www.opensamm.org/

10 siehe: https://www.bsimm.com/

11 Der Anforderungskatalog stellt die Mindestkriterien für die Erteilung des Zertifikats zusammen und sollte in der Regel veröffentlicht sein.

3.3 Testarten

3.3.1 Last- und Stresstest

Gerade bei neu entwickelten Webanwendungen ist es oft schwer vorauszusagen, welche Last auf das System wirken wird. Daher ist es sehr wichtig zu wissen, welche Lastgrenzen für ein System bestehen. Hierfür gibt es zwei Arten von Tests: einen Lasttest, der innerhalb definierter Grenzen das System unter Last setzt und testet, ob die Funktionalität noch gewährleistet ist, und einen Stresstest, der im Sinne eines »Break the System« die Last so weit erhöht, bis das System zusammenbricht, um die maximal mögliche Last zu ermitteln. Diese Art des Testens sollte bereits ab dem Moment regelmäßig geschehen, ab dem die grundsätzliche Architektur festgelegt ist und der Anwendungsrahmen vorhanden ist (»Durchstich«). Es geht dann nicht um das Scheitern einzelner Anwendungsfälle, sondern um das Gesamtverhalten des Systems. Messwerte können dabei zum Beispiel die durchschnittliche Antwortzeit einer Anfrage oder die maximale Anzahl gleichzeitiger Anfragen sein.

3.3.2 Securitytest/Penetrationstest

Das Testen im Bereich Security ist insbesondere für Anwendungen wichtig, die über das Internet erreichbar und damit potentiell einem großen Angriffsrisiko ausgesetzt sind. Dieses Thema wird näher im Kapitel IT-Sicherheit von Webanwendungen (Kapitel 4) beleuchtet.

3.3.3 Usabilitytest

Die Usability einer Anwendung ist schwerer zu messen, da sie in großen Teilen subjektiv ist. Daher sind in diesem Bereich die zukünftigen Nutzer des Systems frühzeitig einzubeziehen und von ihnen Feedback einzuholen. Wenn die Nutzergruppe nicht bekannt ist, müssen Annahmen über einen typischen Nutzer (»Persona«) getroffen werden. Hilfreich ist hier auch die Durchführung und Auswertung von systematischen Userbefragungen.

Ein wichtiges Thema bei Webanwendungen stellt die Browserkompatibilität dar – insbesondere ältere Browserversionen stellen häufig die Inhalte nicht wie gewünscht dar oder haben Einschränkungen in der Funktionalität. Hier ist es wichtig, die zu unterstützenden Browser festzulegen und den Test darauf zu fokussieren.

Die Barrierefreiheit ist ebenfalls dem Bereich Usability zuzurechnen und muss ebenfalls beim Testen berücksichtigt werden. Nähere Ausführungen werden im Kapitel Barrierefreie Webanwendungen (Kapitel 2.2.4) gemacht.

3.3.4 Interoperabilitätstest

Bezüglich Webanwendungen liegt hier die größte Herausforderung darin, ob die Anwendung in verschiedenen Browsern und auf verschiedenen Endgeräten korrekt dargestellt wird und bedient werden kann (»responsive«). Insbesondere bei Internetanwendungen ist die Zielplattform nicht voraussagbar, weshalb Annahmen über die typische Nutzung getroffen werden müssen, um den Test sinnvoll einschränken zu können.

3.4 Testautomation

Grundsätzlich sollten, wo immer möglich, automatisierte Tests genutzt werden, da diese beliebig oft und in kürzerer Zeit durchgeführt werden können und dabei weniger Fehler gemacht werden. Zudem ist der Test jederzeit reproduzierbar. In der konkreten Anwendung ist aber immer zu beurteilen, wie oft bestimmte Testoperationen durchgeführt werden und wie sehr das Produkt an der zu testenden Stelle Änderungen unterliegt, da unter Umständen der Aufwand für die Pflege von automatisierten Tests den Nutzen übersteigen kann.

Gerade im Bereich der GUI ist oft ein relativ hoher Aufwand zum automatisierten Test nötig. Dabei helfen aber spezielle Tools und eine möglichst große Abstraktion des Tests. Gegenüber einem simplen Capture & Replay beispielsweise hat das schlüsselwortgetriebene Abarbeiten von Replay-Szenarien eine größere Chance, bei moderaten Änderungen innerhalb der GUI noch verwendet bzw. ohne größeren Aufwand angepasst werden zu können.

Automatisierte Tests sollten aber in jedem Fall Bestandteil einer Continuous Integration sein, um bei jeder Änderung einer Software unmittelbar ausgeführt zu werden und damit direkte Rückmeldung geben zu können.

Empfehlung

Beim Testen von Webanwendungen sollten folgende Dinge berücksichtigt werden:

- Testen von Anfang an
- Laufende Zusammenarbeit von Entwicklern und Testern
- Weitestgehende Automatisierung von Tests
- Etablierung einer Continuous Integration

4 IT-Sicherheit von Webanwendungen

Nicht erst seit den spektakulären Hacker-Attacken auf Sony oder Sony Pictures[12], bei denen den Angreifern Millionen von Zugangsdaten und Zahlungsinformationen in die Hände gefallen sind bzw. ganze Infrastrukturen lahm gelegt wurden, ist den meisten klar, dass die Sicherheit von Webanwendungen oder Anwendungen im Allgemeinen eines der zentralen Merkmale einer Anwendung sein muss. Der Anwender muss und kann erwarten, dass das System, dem seine Daten anvertraut werden, einem hohen Sicherheitsmaß entspricht und auch nach der Inbetriebnahme weiterhin gepflegt wird und neuen Angriffsmechanismen widersteht.

4.1 Bedrohungsszenarien und Angriffsvektoren

Es gibt hunderte von bisher beobachteten Möglichkeiten, eine Anwendung dazu zu bringen, dass sie nicht nur die Funktionalität zeigt, für die sie entworfen wurde, sondern einem Angreifer Zugang zu Daten, Systemen und auch Geschäftsprozessen gibt, die dann manipuliert und kompromittiert werden können. Die Möglichkeiten reichen von Social Engineering Angriffen, wie Tailgating[13], über systembedingte Schwächen, wie mangelhaftes Patchmanagement der Komponenten, bis hin zu Software- oder Framework-spezifischen Schwächen, wie unzureichende Datenvalidierung. Jede dieser Möglichkeiten wird als Angriffsvektor bezeichnet und diese werden seit Jahren erfasst, katalogisiert und bzgl. ihres Bedrohungspotentials, ihrer Häufigkeit und der Ausnutzbarkeit bewertet. Eine umfängliche und ständig aktualisierte Liste von Angriffsvektoren ist das CWE-Verzeichnis (Common Weakness Enumeration Dictonary) mit über 750 erfassten Methoden.[14]

Des Weiteren existiert eine periodisch aktualisierte Liste der Top-Bedrohungen. Das sind diejenigen Bedrohungen, die einerseits am häufigsten durch die Unternehmen, Behörden oder Security-Experten beobachtet werden, und die andererseits hohen bis sehr hohen Schaden bewirken können. An dieser Stelle kann nur sehr oberflächlich auf diese Bedrohungen eingegangen werden. Als weitere Lektüre hierzu empfehlen sich die Seiten von OWASP[15] oder SANS[16].

4.2 OWASP Top 10 der Angriffsszenarien Version 2013

Eine detaillierte Darstellung aller Fehlerklassen der OWASP-TOP-10-Liste[17], welche zuletzt im Jahr 2013 erschienen ist, übersteigt den Rahmen dieses Dokuments. Im Folgenden werden daher lediglich die ersten drei Fehlerklassen näher betrachtet. Die Auswahl sollte jedoch nicht so verstanden werden, dass die anderen Klassen weniger bedeutend oder weniger häufig anzutreffen sind.

12 siehe hierzu auch: http://en.wikipedia.org/wiki/2011_PlayStation_Network_outage oder http://en.wikipedia.org/wiki/Sony_Pictures_Entertainment_hack

13 siehe: http://de.wikipedia.org/wiki/Social_Engineering_%28Sicherheit%29

14 http://cwe.mitre.org/, Stand: April 2015

15 http://www.owasp.org/

16 http://www.sans.org/

17 Aktuelle deutsche PDF-Version: https://www.owasp.org/images/4/42/OWASP_Top_10_2013_DE_Version_1_0.pdf

1. Injektionsangriffe
   In dieser Klasse werden die Angriffe subsummiert, bei denen fremde Skripte oder sonstiger Code in den Datenstrom oder in das System injiziert werden. Ein sehr prominentes und gefährliches Beispiel wäre die SQL-Injektion, bei dem der Angreifer bspw. über die Webanwendung direkt Datenbankkommandos absetzen kann. Liegt eine derartige Schwäche vor, dann kann der gesamte Geschäftsprozess betroffen sein. Ursache sind in der Regel mangelhafte Validierung der Benutzereingaben und die ungeprüfte Einbindung von externen Inhalten in Datenbankkommandos.

2. Fehler in Authentisierung und Session-Management
   Diese Klasse von Angriffsvektoren umfasst die Schwächen bei der Identifikation von Nutzern oder bereits existierenden Sessions. Ist bspw. eine Session-ID vorhersagbar oder gar von außen festsetzbar (Session Fixation), dann kann der Angreifer eine Session übernehmen und entsprechenden Schaden anrichten. Ursache hierfür ist oft mangelnde Absicherung von Session-Informationen bspw. durch fehlende Cookie-Parameter.

3. Cross Site Scripting
   Ca. 40–50 % aller beobachteten Schwächen von Webanwendungen fallen in diese Fehlerklasse. Bei Cross Site Scripting-Attacken aktiviert das Opfer Script-Code, der ihm z. B. via Link untergeschoben wird oder der innerhalb des Webportals gespeichert ist. Das Resultat kann dann der Verlust von Session-Informationen (z. B. Cookies) und die Kompromittierung der Sitzung sein. Auch hier ist im Wesentlichen die fehlerhafte Validierung von Eingabe und Ausgabe Quelle der Schwäche.

Der Vollständigkeit halber seien nachfolgend die weiteren Fehlerklassen aufgeführt:

4. Unsichere direkte Objektreferenzen: Schützenswerte Ressourcen werden bspw. durch URL-Parameter direkt adressiert.

5. Sicherheitsrelevante Fehlkonfiguration: Implementierte Sicherheitsmechanismen sind nicht oder nur unzureichend konfiguriert (z. B. Default-Passwörter).

6. Verlust der Vertraulichkeit sensibler Daten: Sensible Daten sind nicht gegen direkten Zugriff geschützt (bspw. Kennwörter [s2] sind hardcoded in der Webpage).

7. Fehlerhafte Autorisierung auf Anwendungsebene: Privilegierte Rollen sind bspw. durch »Erraten« der URL erreichbar.

8. Cross-Site Request Forgery (CSRF): Ein definierter Prozessablauf z. B. durch Formblätter kann unterwandert und kompromittiert werden.

9. Benutzen von Komponenten mit bekannten Schwachstellen: Subkomponenten oder das verwendete Framework haben bekannte Schwachstellen.

10. Ungeprüfte Um- und Weiterleitungen: Weiterleitungen oder Referenzen sind manipulierbar und fördern damit weitere Attacken wie bspw. Phishing.

4.3 Bedrohungsmodellierung

Ein wirksamer Ansatz, die oben aufgelisteten Schwächen bereits innerhalb des Entwicklungsprozesses zu vermeiden, ist die Technik der Bedrohungsmodellierung (Threat Risk Modeling). Ziel ist es, systematisch alle potentiellen Bedrohungen für ein IT-System, eine Anwendung oder einen Prozess zu erfassen und zu bewerten. Sie wird in der IT-Sicherheitstechnologie auch für die Modellierung von organisatorischen Risiken angewendet. Bei dieser Technik werden für die zu erstellende Anwendung alle beteiligten Prozesse, Komponenten und Interaktionen, wie Datenaustausch oder ähnliches, erfasst und mit den vorgegebenen Schutzzielen (z. B. Compliance mit PCI-DSS oder weiteren Anforderungen des Auftraggebers) in Beziehung gesetzt. Auch die bereits erwähnten Angriffsvektoren, seien es die OWASP Top 10 oder eine andere Auswahl von CWEs,[18] werden in die Betrachtung eingeschlossen. Ist beispielsweise eine Datenbank Teil der Infrastruktur der Anwendung und sind Datenbankoperationen abhängig von Benutzereingaben, dann ist die Anwendung potentiell verwundbar gegenüber SQL-Injektion-Angriffen. Diese Bedrohung muss dann in das Modell aufgenommen werden. Fordert in einem anderen Beispiel ein Anforderungskatalog, dass alle Kontoinformationen verschlüsselt sein müssen, dann sollte als potentielle Schwachstelle eine fehlerhafte Implementierung der Verschlüsselung oder die Verwendung von schwachen Kennwörtern als potentielle Schwachstellen in das Modell aufgenommen werden.

Nachdem, unter Umständen in mehreren Durchläufen, ein Modell der potentiellen Bedrohungen für die Anwendung erstellt wurde, können geeignete Gegenmaßnahmen definiert werden. Hierbei kann auf Best Practices zurückgegriffen werden, wie sie bei CWE[19] oder anderen einschlägigen Quellen (z. B. MSDN) beschrieben sind. In dem obigen Beispiel der potentiellen SQL-Injektion wären die Verwendung von prepared Statements, intensive Inputvalidierung durch White-Listing oder Parametrisierung geeignete Maßnahmen.

An dieser Stelle kann die Methodik der Bedrohungsmodellierung und ihre Einbettung in den Entwicklungsprozess nur rudimentär beschrieben werden. Für weitere Informationen wird auf die entsprechenden Seiten im Internet wie bspw. OWASP[20] oder Microsoft[21] mit dem Stichwort **Threat Risk Modeling** verwiesen. Die Methode der Bedrohungsmodellierung und die systematische Erfassung aller Komponenten und deren Verknüpfungen und Abhängigkeiten unter einander stellt eine gute und erprobte Möglichkeit dar, sicherheitsrelevante Programmier-, Design- oder Implementierungsfehler sehr früh in der Entwicklungsphase zu identifizieren bzw. zu vermeiden. Sie sollte ein zentraler Bestandteil des Softwareentwicklungsprozesses, insbesondere für Webanwendungen, sein.

18 siehe: http://www.owasp.org oder http://www.mitre.org

19 cwe.mitre.org

20 www.owasp.org

21 www.microsoft.com/security

5 Analyse & Optimierung von Webanwendungen

Dieses Kapitel behandelt die kennzahlengestützte Erfolgsmessung und Optimierung der Webanwendungen mithilfe der Webanalyse. Die folgende Definition des Web Controlling soll das Thema grob einordnen: »Web Controlling soll dazu dienen, die Ziele, welche mit einer Website verbunden sind, anhand von Key Performance Indicators (KPIs) zu überprüfen und gegebenfalls Maßnahmen einzuleiten.«[22]

5.1 Arten von Webprojekten

Die hier beschriebene Art der Analyse und Optimierung kann nahezu auf jedes beliebige Webprojekt angewendet werden. Es kann in den gängigen Webumfeldern wie Internet (z. B. Corporate Website), Extranet (z. B. Customer Portal), Intranet (z. B. Employee Portal) eingesetzt werden. Da die Entwicklung und Implementierung eines passenden Kennzahlensystems (fachlich) und des zugrunde liegenden Tracking (technisch) relativ aufwendig werden kann, soll die Umsetzung von folgenden Faktoren abhängig gemacht werden:

1. Wichtigkeit des Webprojekts bzw. der resultierenden Webanwendung und der übergeordneten strategischen Ziele für das Unternehmen
   Beispiel: Die Corporate Website ist die Visitenkarte des Unternehmens im Internet. Die Wichtigkeit der Anwendung ist hoch. Das Webprojekt »Relaunch der Corporate Website« beinhaltet einige grundlegende Änderungen der Navigationsstruktur, die eine bessere Auffindbarkeit der Produktinformationen ermöglichen soll. Somit ist die Analyse dieser Änderung ebenfalls wichtig.

2. Aufwand der Implementierung der Analyse, des Tracking und des Optimierungsprozesses (Zeit und Kosten)
   Beispiel: Beim Relaunch der Corporate Website werden auch einige kleinere Änderungen an Navigationselementen vorgenommen. Es werden beispielhaft neuartige Verlinkungen im Content-Bereich sowie Teaser in der rechten Marginalspalte eingeführt. Eine flächendeckende Tracking-Implementierung für diese Navigationselemente kann sich als zu aufwendig erweisen. So beschränkt man sich auf das Tracking und Analyse der Änderungen in der Struktur der Hauptnavigation.

3. Langlebigkeit der Webanwendung, die im Rahmen des Projekts entwickelt werden soll
   Beispiel: Die Corporate Website unterstützt die Unternehmenskommunikation hin zum Kunden über einen längeren Zeitraum. Sie soll kontinuierlich analysiert und optimiert werden. Im Gegensatz dazu soll eine kleinere Microsite, die nur für wenige Monate die Markteinführung eines neuen Produkts unterstützen soll, lediglich auf ihre Effizienz untersucht werden. Die Implementierung eines Optimierungsprozesses ergibt hingegen wenig Sinn, da die Umsetzung der Verbesserungsvorschläge u. U. in der kurzen Zeit nicht möglich ist.

22 Zumstein, D; Meier, A (2010) »Web-Controlling – Analyse und Optimierung der digitalen Wertschöpfungskette mit Web Analytics«

5.2 Bezug zu Projektphasen

Die Aufteilung der Projektentwicklung in die typischen Phasen Plan-Build-Run lässt sich ebenfalls auf die Analyse und Optimierung übertragen.

1. Plan – Bereits in der Konzeptionsphase sollen Anforderungen an die spätere Analyse und Optimierung festgelegt werden. Denn das hat sowohl fachliche Auswirkungen auf Prozesse, das gemeinsame Verständnis der Kennzahlen und Methoden, wie die Erkenntnisse aus den Analysen in umsetzbare Handlungsempfehlungen übersetzt werden.

2. Build – Für das Tracking werden meist Tools eines oder mehrerer Drittanbieter implementiert, da ein Tool nicht immer alle Anforderungen erfüllen kann. Somit wird die Tracking-Implementierung zu einem festen Bestandteil der Build-Phase.

3. Run – Im Betrieb werden die Tracking-Daten gesammelt, ausgewertet und analysiert. Im Anschluss werden Handlungsempfehlungen zur Optimierung der Webanwendung abgeleitet.

5.3 Optimierungskreislauf

Die Implementierung des Optimierungskreislaufs soll in Abstimmung mit den betroffenen Stakeholdern (z. B. Entwicklungsabteilung, Fachabteilungen) durchgeführt werden. Die Stakeholder bezeichnen die wesentlichen Personen (definiert über deren Funktion bzw. Verantwortlichkeiten), die regelmäßig Kennzahlen benötigen, um den Erfolg der von ihnen verantworteten Maßnahmen zu dokumentieren und zu steuern. Diese Rollen und deren Informationsbedarfe sind am besten zeitnah beim Start eines neuen Webprojekts zu identifizieren.

Abbildung 3 visualisiert den anzustrebenden Prozess der Optimierung einer Website bzw. eines Web-Services. Anschließend wird auf die einzelnen Schritte des Kreislaufs eingegangen, um ein transparentes Bild vom Prozess der Analyse und Optimierung zu vermitteln.

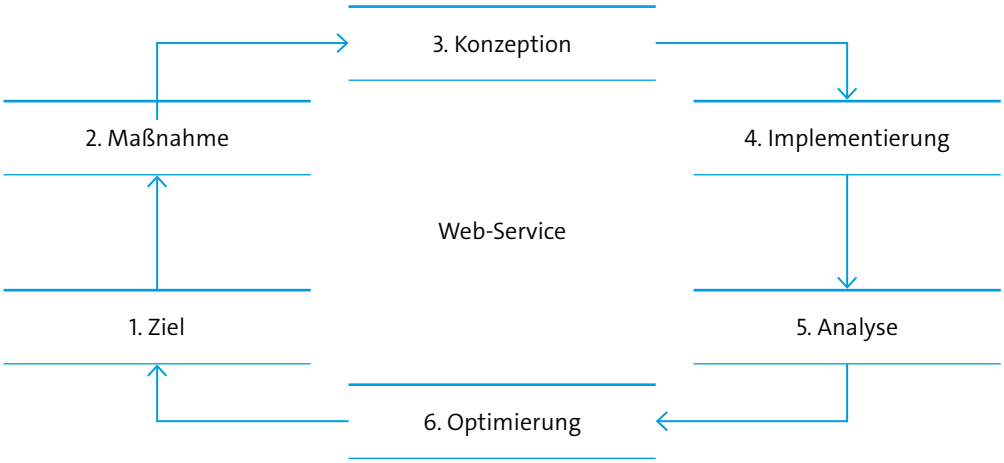

Abbildung 3: Optimierungskreislauf

In Anlehnung an die DMAIC-Methodik von Six Sigma: https://de.wikipedia.org/wiki/Six_Sigma
Quelle: Eigene Darstellung

Die Beschreibung der einzelnen Schritte enthält jeweils drei Abschnitte:

- **2Do** stellt die eigentliche Schrittdefinition dar
- **Key Player** benennt die wichtigsten Stakeholder
- **Tool** verweist auf relevante Tools

5.3.1 Ziel

Am Beginn einer jeden Analyse und Optimierung steht die Zielsetzung. Erst mit der Zielsetzung (vgl. Abbildung 4) wird eine kennzahlengestützte Optimierung möglich. Das Gegenteil dazu stellen die sog. Bauchentscheidungen dar, die zwar schneller getroffen werden, aber den Gesamtaufwand in die Höhe treiben können. Im Fall des Misserfolgs einer Maßnahme kann der Business Owner bei einer fehlenden Zielsetzung in Erklärungsnot geraten. Die Analyse der Ursachen für den Erfolg oder Misserfolg wird dadurch ebenfalls erschwert. Folgend ein praxisnahes Beispiel einer Zielsetzung:

Ein Unternehmen möchte in einem bestimmten Land expandieren. Seine entsprechende Länderwebsite soll zur Produktkommunikation (Information zu bestehenden und neuen Produkten in diesem Land) und Lead-Generierung (Möglichkeit einer Kontaktaufnahme über diverse Formulare) genutzt werden. Diese Website ist bei der wichtigsten lokalen Suchmaschine im Vergleich zu den Mitbewerbern schlechter eingestuft, was unter anderem eine geringere Reichweite der Website zur Folge hat. Das Unternehmen beschließt, das Suchmaschinen-Ranking für diese Website zu verbessern, um die Reichweite zu erhöhen und seinen potentiellen Kunden den Zugang zu den Informationen und Kontaktformularen zu erleichtern.

Daraufhin werden entsprechende SEO-Maßnahmen wie z. B. die komplette Übersetzung der Website in die Landessprache, die Anpassung der Keywords und Nutzung relevanter HTML Meta Tags, redaktionelle SEO-taugliche Erstellung der Seitentexte sowie Kennzahlen zur Messung der Zielerreichung festgelegt. Im Beispiel könnten dies folgende Kennzahlen sein:

- Ranking bestimmter Seiten auf bestimmte Suchbegriffe,
- Anteil des SEO-Traffics am Traffic gesamt,
- Conversion Rates innerhalb des SEO-Kanals

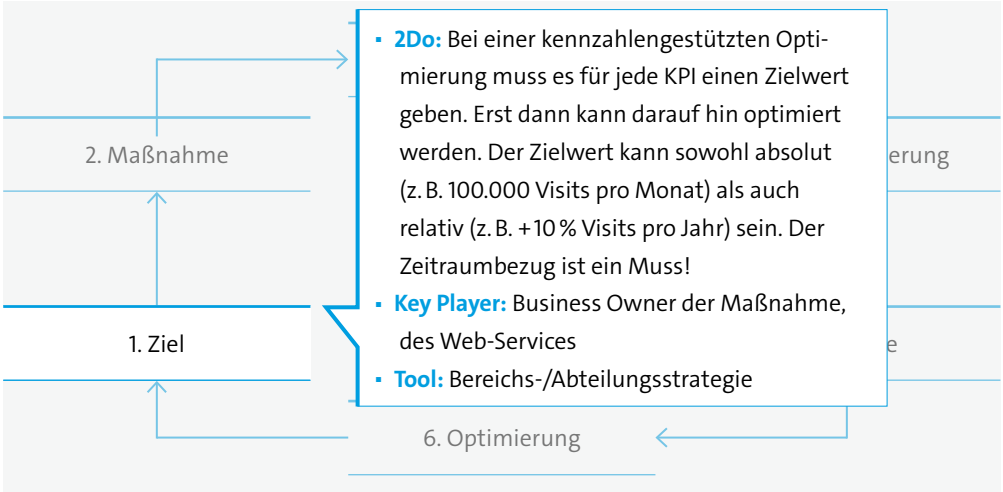

Abbildung 4: Schritt 1 – Ziel

Quelle: Eigene Darstellung

5.3.2 Maßnahme

Abhängig von der Zielsetzung werden die erforderlichen Maßnahmen festgelegt (vgl. Abbildung 5). Es wird empfohlen einen Maßnahmenkatalog aufzusetzen. In diesem Katalog sollten erprobte Maßnahmen samt ihrer Wirkung beschrieben und nach jeder Optimierung aktualisiert werden. Das steigert die Effizienz der Arbeit bei jedem weiteren Durchlauf.
Der Katalog kann als Diskussionsgrundlage im Vorfeld der Optimierungsprojekte dienen, was sowohl den Wissensaustausch innerhalb des Unternehmens fördern als auch eine bessere Transparenz über die Marketingmaßnahmen schaffen kann.

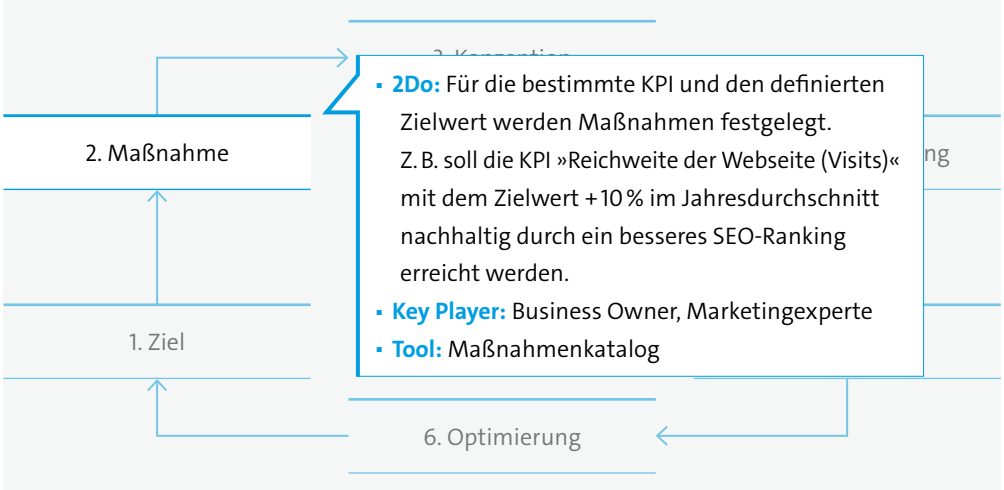

Abbildung 5: Schritt 2 – Maßnahme

Quelle: Eigene Darstellung

5.3.3 Konzeption

Ausgehend von den festgelegten Zielen und Maßnahmen und einer detaillierten Content- und Funktionsbeschreibung des Web-Services erstellt der Tracking-Experte das sog. Trackingkonzept sowie eine Anleitung für die Tracking-Implementierung (vgl. Abbildung 6). Das Trackingkonzept beinhaltet eine Definition der Kennzahlen und Auswertungen zur Dokumentation und Steuerung der Zielerreichung. Zudem legt es fest, wie die Messung mit Hilfe von der Webanalyse-Software umgesetzt werden soll. Die Anleitung zur Tracking-Implementierung beschreibt die technische Umsetzung des Trackings (also der Art und Weise, welche Daten wie im Web-Service erhoben werden). Das ist einerseits die Integration eines Tracking-Skripts auf allen relevanten Seiten der Website. Somit wird die Tracking-Funktionalität grundsätzlich zur Verfügung gestellt. Andererseits sind es Aufrufe bestimmter Tracking-Funktionen, die das Benutzerverhalten durch Ereignisse genauer erfassen sollen, wie z. B. Klicks auf Downloads oder innerhalb eines Videos.

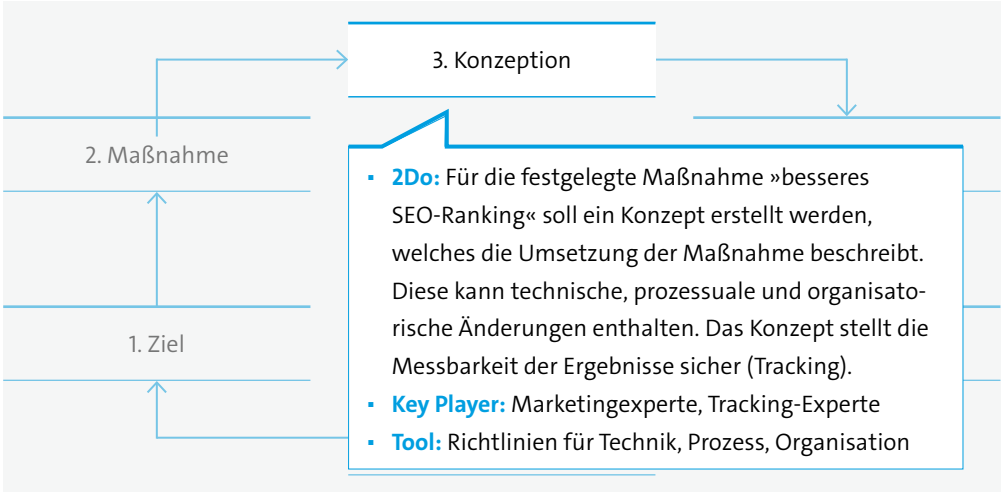

Abbildung 6: Schritt 3 – Konzeption

Quelle: Eigene Darstellung

5.3.4 Implementierung

Die Implementierung des Tracking (vgl. Abbildung 7) erfolgt durch den Webentwickler oder den Content-Produzenten, wenn das eingesetzte Content Management System die Tracking-Funktionalität »out of the box« anbietet. Parallel erfolgt eine Konfiguration der Analysesoftware.

Nach der Implementierung ist eine Qualitätssicherung durch den Tracking-Experten erforderlich. Dieser erstellt dann ggf. ein Rebriefing für die Überarbeitung der Implementierung. Auf Basis des Rebriefings muss die Implementierung angepasst werden. Dies erfordert ggf. eine erneute Taggingkontrolle/-analyse und ggf. zusätzliche Konfigurationen der Webanalyse-Software.

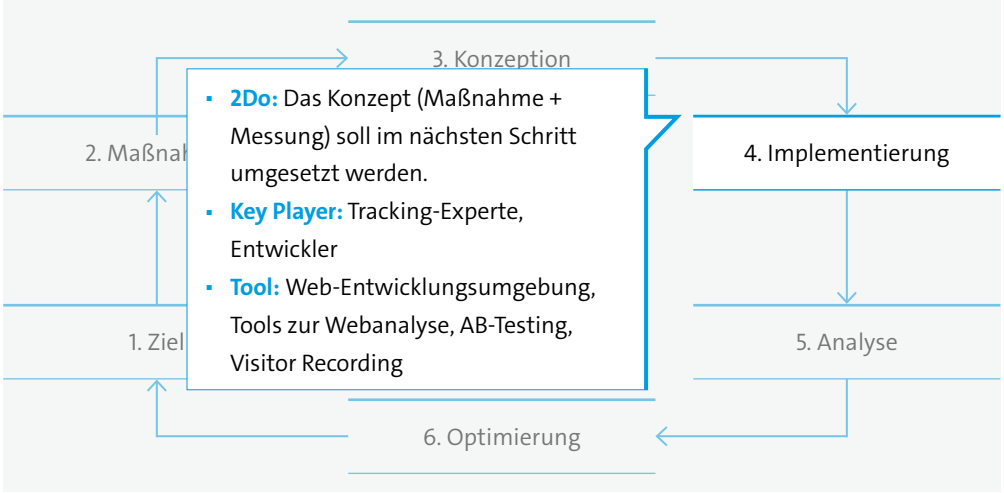

Abbildung 7: Schritt 4 – Implementierung

Quelle: Eigene Darstellung

5.3.5 Analyse

Die Analysephase (vgl. Abbildung 8) verfolgt das Ziel, Tracking-Daten zwecks der Erkenntnisgewinnung hinsichtlich der gesetzten Ziele auszuwerten. Hierzu dienen ein regelmäßiges Reporting der wichtigsten Kennzahlen sowie explorative Tiefenanalysen zur Identifikation von Gründen für beobachtete Kennzahlenentwicklungen. Die Erkenntnisse liefern Informationen über den Status quo (Trends, Tops und Flops) und schaffen eine Grundlage für den nächsten Schritt der Optimierung des Web-Services und der zugrunde liegenden Prozesse.

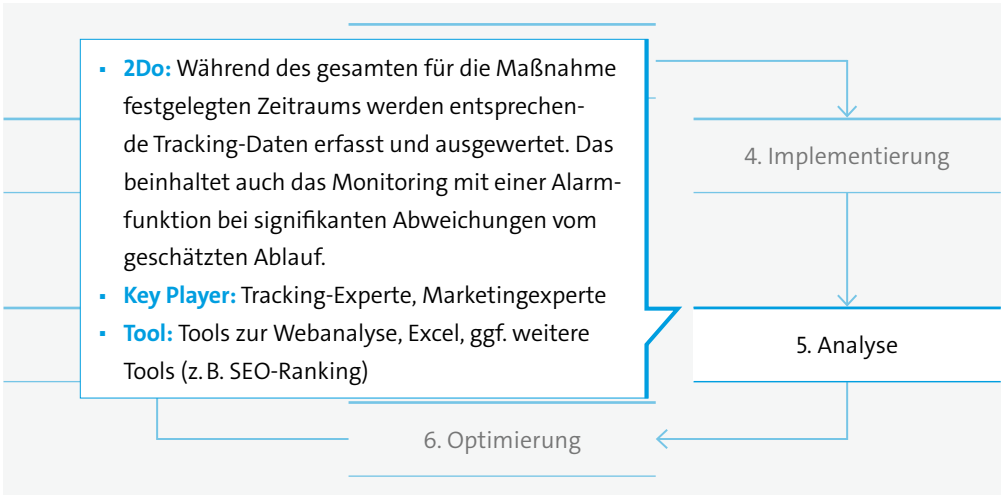

Abbildung 8: Schritt 5 – Analyse

Quelle: Eigene Darstellung

5.3.6 Optimierung

Auf Basis der im vorherigen Schritt der Analyse gewonnenen Erkenntnisse wird der Web-Service auf eventuelle Optimierungspotentiale hin untersucht (vgl. Abbildung 9). Es werden Handlungsempfehlungen zur Anpassung bzw. Ergänzung der Maßnahmen abgeleitet, welche zu einer besseren Zielerreichung beitragen sollen.

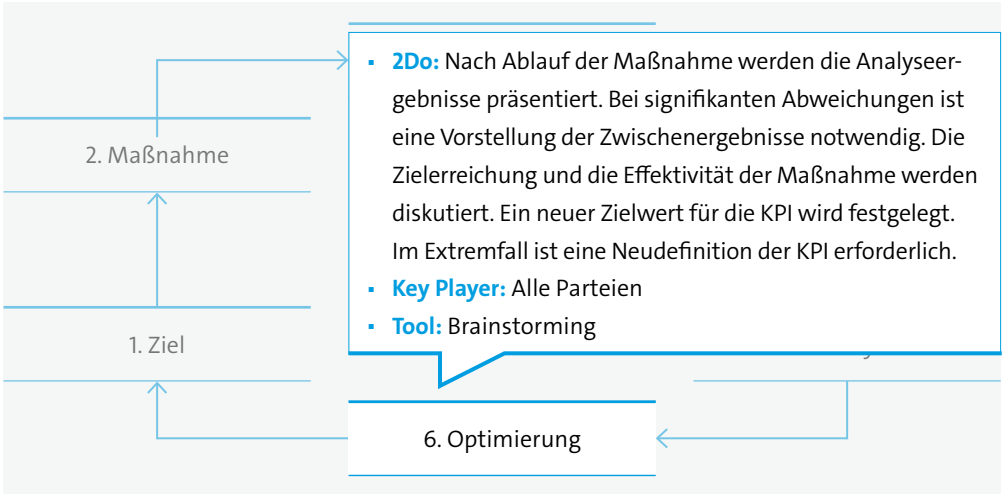

Abbildung 9: Schritt 6 – Optimierung

Quelle: Eigene Darstellung

5.4 Kennzahlenentwicklung

Es gibt viele Möglichkeiten, ein passendes Kennzahlensystem zu entwickeln. Die eine besteht darin, sich an ein Marketingmodell anzulehnen. Das unten erläuterte Beispiel ist eine Weiterentwicklung des sog. AIDA-Modells[23] (Awareness, Interest, Desire, Action) und stellt einen möglichen Customer Life Cycle dar (vgl. Abbildung 10):

- Reach – Kontakt des Benutzers mit allgemeinen Informationen auf der Website wie z. B. die Homepage oder Übersichtsseite eines Geschäftsbereichs.
- Brand – Kontakt mit technologie- und produktspezifischen Informationen auf der Website wie z. B. Produktdetailseite, Download eines Produktflyers, Produktkonfiguration.
- Lead – Kontaktaufnahme über ein Lead-Generierungsformular (Interesse an einem Produkt + Erlaubnis, kontaktiert zu werden).
- Sales – Der Verkauf kann z. B. in einem Webstore stattfinden oder auch offline. Im letzteren Fall ist eine saubere Pflege der CRM-Datenbank notwendig, um den Beitrag der Online-Maßnahme sicherzustellen.
- Aftersales – Download eines Firmware-Updates oder eines Treibers, Anmeldung im geschlossenen Kundenbereich bzw. eine Aktion, die den Benutzer als Kunde identifiziert.

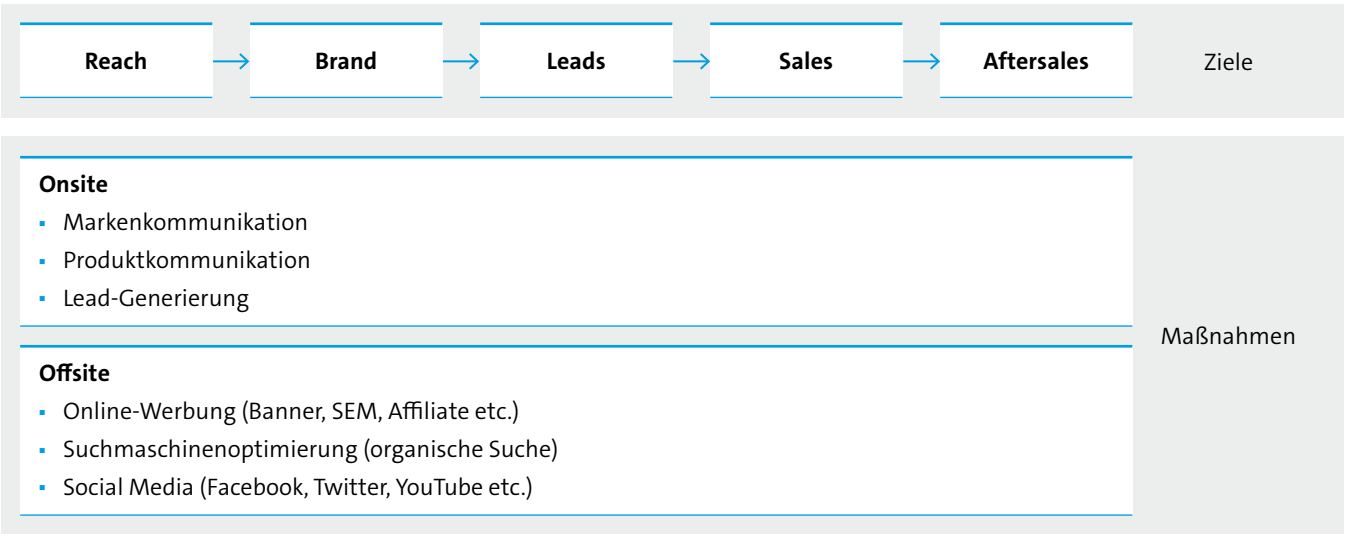

Abbildung 10: Ziele und Offsite-/Onsite-Maßnahmen

Quelle: Eigene Darstellung

Die aufgespannte Matrix soll nach und nach mit Key Performance Indikatoren und Metriken befüllt werden. Die KPIs für die Onsite- und Offsite-Aktivitäten sollen einen schnellen Einblick in die Performanceentwicklung der Online Aktivitäten der beiden Bereiche geben. Die um diese KPIs gruppierten Metriken sollen den Betrachter dann dabei unterstützen, die Aktivitäten detaillierter untersuchen zu können.

23 http://de.wikipedia.org/wiki/AIDA-Modell

Falls das Unternehmen mehrere Websites zum gleichen Zweck einsetzt (z. B. Länderwebsites zur Produktkommunikation mit lokalisierten Inhalten), soll das gleiche Kennzahlenset auf alle Websites angewandt werden. Das ermöglicht eine spätere Vergleichbarkeit der Websites miteinander.

Zu den ausgewählten Metriken sollten langfristig Ziel- und Schwellenwerte für einen festgelegten Zeitraum definiert werden. Diese Zielwerte bilden die Basis für die Performanceanalyse. Die Festlegung der Zielwerte kann frühestens nach den ersten validen Messungen auf den Websites stattfinden.

Zeitintervalle für die Reports werden mit dem Business Owner abgestimmt. Je nach Zielsetzung und Reportart können z. B. folgende Trendansichten relevant sein:

- Monatsansicht bzw. auf Tagesbasis (monats- bzw. tagesgenaue Analyse der Wirkungen von Online-Marketing Maßnahmen)

- Jahresansicht für die letzten 13 Monate (saisonale Schwankungen im Jahresverlauf, Monatsvergleich, Vergleich mit dem Monat aus dem Jahr zuvor)

Empfehlung

Analyse & Optimierung stellt einen begleitenden Prozess entlang der gesamten Lebensdauer einer Webanwendung dar. Prinzipiell kann es in zwei Teile unterteilt werden.

- Analyse (Schritte 1 bis 5) – reine Erfolgsmessung ohne Ableitung von Handlungsempfehlungen zur Verbesserung der Anwendung
- Optimierung – der anschließende Schritt 6, der auf der Analyse aufbaut und einen Input für die Weiterentwicklung der Anwendung liefert

Die Analyse bzw. die Erfolgsmessung soll in erster Linie bewerten, ob sich die Entwicklung und Nutzung einer Webanwendung gelohnt hat. Daher soll diese unabhängig von der Größe, der Lebensdauer und dem Umfeld immer stattfinden.
Die Optimierung empfiehlt sich eher bei langlebigen Webanwendungen, da sie einen zusätzlichen Entwicklungsaufwand erfordert und erst eine gewisse Zeit nach der Inbetriebnahme der Anwendung erfolgen kann. Dieser zeitliche Abstand ist meistens von der Nutzungsintensität abhängig. Wird die Webanwendung nur selten benutzt, können kaum Aussagen über das Benutzerverhalten getroffen werden. Demnach fehlt auch die statistische Basis für einen Verbesserungsvorschlag.
Eine kennzahlengestützte Analyse & Optimierung sollte allerdings nicht das alleinige Maß der Dinge sein. Denn oft spielen viele weitere Einflussfaktoren eine Rolle, die nur schwer quantifizierbar sind. Die Kennzahlen weisen nur auf Problemstellen hin. Die richtige Interpretation der Kennzahlen stellt oft eine Herausforderung dar und sollte in Zusammenarbeit mit allen Beteiligten und ggf. weiteren Experten durchgeführt werden.

Glossar

3-Schichten-Architektur
Architekturansatz, demzufolge eine Webanwendung aus drei Schichten besteht: Benutzeroberfläche (welche Benutzerschnittstelle und Prozesslogik umfasst), Geschäftslogik und Datenhaltung.

4-Schichten-Architektur
Architekturansatz, demzufolge eine Webanwendung aus vier Schichten besteht: Benutzerschnittstelle, Prozesslogik, Geschäftslogik und Datenbank.

Adobe Flash
siehe https://www.adobe.com/de/products/flash.html

AJAX (Asynchronous JavaScript and XML)
Bündel von auf JavaScript basierenden Webtechnologien, welche die Entwicklung von performanten Webanwendungen ermöglichen.

Client-zentrierte Webanwendungen
Auf AJAX basierende Webanwendungen, welche ähnliche Leistungsmerkmale und Reaktionszeiten ermöglichen wie lokal installierte Anwendungen oder Apps. Ein erheblicher Teil von Benutzerschnittstelle und Geschäftslogik läuft im Browser des Anwenders.

CSS (Cascading Style Sheets)
Mit Hilfe von CSS kann die Darstellung von HTML Elementen beeinflusst werden. CSS wird vom Browser interpretiert und je nach Anweisung werden HTML Elemente wie in den Style Sheets beschrieben zur Anzeige gebracht.
Weiterführende Links:
http://de.wikipedia.org/wiki/Cascading_Style_Sheets

HTML5
Standard für die HTML-basierende Benutzerschnittstelle von Web-Anwendungen.
Ermöglicht die Entwicklung leistungsfähiger Anwendungen, welche keine Browser-Plugins (bspw. Adobe Flash und Microsoft Silverlight) benötigen.

HTML (Hypertext Markup Language)
HTML ist eine Auszeichnungsprache mit der digitale Dokumente strukturiert werden können. So können beispielsweise Absätze, Bilder, Links, Multimediaelemente etc. entsprechend markiert werden. Diese Markierungen werden von Browsern interpretiert und dargestellt.

HTML GUI-Elemente
Eine Übersicht der verschiedenen HTML GUI-Elemente findet man unter folgenden Quellen:
http://en.wikipedia.org/wiki/HTML_element
http://www.w3.org/TR/2014/REC-html5-20141028/

JavaScript (ECMAScript)

JavaScript wurde entwickelt um HTML um dynamische Komponenten erweitern zu können. Ursprünglich wurde JavaScript ausschließlich von Browsern interpretiert und ausgeführt. Mittlerweile kann JavaScript auch Server-seitig genutzt werden.
Weiterführende Links:
http://de.wikipedia.org/wiki/JavaScript

JavaScript Frameworks

Frameworks unterstützen den Entwickler mit bereits vorgefertigten Funktionen und Methoden, sie erweitern und ergänzen den Umfang einer Programmiersprache und unterstützen den Entwickler mit Werkzeugen, welche immer wiederkehrende Befehle zusammenfassen. Häufig verwendete JavaScript Frameworks sind z. B. JQuery, Prototype, MooTools, AngularJS, YUI, Google Web Toolkit und viele mehr.

Microsoft Silverlight

siehe http://www.microsoft.com/silverlight/

Responsive Web Design

HTML Oberflächen, welche sich flexibel in ihrer Größe an das Anzeigegerät anpassen.

Rich Internet Applications

vgl. Client-zentrierte Webanwendungen.

Server-zentrierte Webanwendungen

Klassischer Architektur von Webanwendungen, bei der fast alle Funktionen Server-seitig realisiert sind. Benutzereingaben führen in der Regel zu einem kompletten Neuaufbau der Website im Browser.

SQL-Injection

Die SQL-Injection beschreibt das Ausnutzen einer Sicherheitslücke in einer (Web-)Anwendung, die auf der Nutzung von SQL-Datenbanken basiert. Durch unzureichende Verarbeitung von Dateneingaben versucht ein Angreifer eigene Befehle an eine Datenbank weiterzugeben und so Daten einzusehen, die Kontrolle über den Server zu erhalten oder Schaden anzurichten.

Threat Risk Modeling (Threat Modeling)

Threat Modeling ist ein Verfahren, um systematisch alle potentiellen Bedrohungen für ein IT-System, eine Anwendung oder einen Prozess zu erfassen. Ein vollständiges Threat-Model (Bedrohungsmodell) hilft Entwicklern und Betreibern einer Anwendung oder Infrastruktur adäquate Maßnahmen zu entwickeln und anzuwenden.

White-Listing

Eine weiße Liste enthält Daten und Informationen, welche in Bezug auf die Nutzung einer Anwendung oder eines Systems vertrauenswürdig sind und somit zu einem Zugriff berechtigt sind. Im Gegensatz dazu enthält eine schwarze Liste (Black List) Daten und Informationen, die nicht vertrauenswürdig sind und denen somit ein Zugrif verwehrt wird.

Quellen

Links

- Allgemein
  - AIDA-Modell auf Wikipedia: http://de.wikiedia.org/wiki/AIDA-Modell
  - Webanwendung auf Wikipedia: http://de.wikipedia.org/wiki/Webanwendung
  - Six Sigma auf Wikipedia: https://de.wikipedia.org/wiki/Six_Sigma
  - Responsive Webdesign: http://alistapart.com/article/responsive-web-design/
- Testen
  - ISTQB Glossar auf German Testingboard: http://glossar.german-testing-board.info/de/
- Security
  - CVE – Common Vulnerabilities and Exposures: http://cve.mitre.org/
  - OWASP – Open Web Application Security Project: https://www.owasp.org/index.php/Main_Page
- Barrierefreiheit
  - Techniques for WCAG 2.0: http://www.w3.org/TR/WCAG20-TECHS/
- Wichtige gesetzliche Grundlagen
  - BITV 2: http://www.gesetze-im-internet.de/bitv_2_0/
  - Bundesdatenschutzgesetz: http://www.gesetze-im-internet.de/bdsg_1990/
  - Bürgerliches Gesetzbuch: http://www.gesetze-im-internet.de/bgb/
  - Telekommunikationsgesetz: http://www.gesetze-im-internet.de/tkg_2004/
  - Telemediengesetz: http://www.gesetze-im-internet.de/tmg/
- Empfehlungen
  - SAGA – Standards und Architekturen für E-Government-Anwendungen: http://www.cio.bund.de/DE/Architekturen-und-Standards/SAGA/saga_node.html
- relevante Leitfäden des Bitkom
  - Agiles Software Engineering Made in Germany: http://www.bitkom.org/files/documents/20150415_LF_Agiles_Software_Engineering_web.pdf
  - Apps & Mobile Services, 2. erweiterte Auflage: http://www.bitkom.org/files/documents/Leitfaden_Apps_und_Mobile_Services_2014.pdf
  - Best Practices für die Entwicklung und den Test mobiler Anwendungen: http://www.bitkom.org/files/documents/140131_Best_Practices_Mobile_App_Quality.pdf
  - Industrielle Softwareentwicklung: http://www.bitkom.org/files/documents/Industrielle_Softwareentwicklung_web.pdf
  - Softwareorientierte Architekturen in der Cloud: http://www.soa-know-how.de/soa-in-der-cloud
- BSI Leitlinien für sichere Webentwicklung
  - Leitfaden zur Entwicklung sicherer Webanwendungen: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Webanwendungen/Webanw_Auftragnehmer_pdf.pdf;jsessionid=C2889278D132A96A9310E3E958181FF8.2_cid359?__blob=publicationFile

Literatur

- Basiswissen Softwaretest, 4. Auflage, A. Spillner / T. Linz, dpunkt.verlag 2010
- Web-Controlling – Analyse und Optimierung der digitalen Wertschöpfungskette mit Web Analytics. D. Zumstein / A. Meier, 2010.

Bitkom vertritt mehr als 2.300 Unternehmen der digitalen Wirtschaft, davon gut 1.500 Direktmitglieder. Sie erzielen mit 700.000 Beschäftigten jährlich Inlandsumsätze von 140 Milliarden Euro und stehen für Exporte von weiteren 50 Milliarden Euro. Zu den Mitgliedern zählen 1.000 Mittelständler, 300 Start-ups und nahezu alle Global Player. Sie bieten Software, IT-Services, Telekommunikations- oder Internetdienste an, stellen Hardware oder Consumer Electronics her, sind im Bereich der digitalen Medien oder der Netzwirtschaft tätig oder in anderer Weise Teil der digitalen Wirtschaft. 78 Prozent der Unternehmen haben ihren Hauptsitz in Deutschland, 9 Prozent kommen aus Europa, 9 Prozent aus den USA und 4 Prozent aus anderen Regionen. Bitkom setzt sich insbesondere für eine innovative Wirtschaftspolitik, eine Modernisierung des Bildungssystems und eine zukunftsorientierte Netzpolitik ein.

Bundesverband Informationswirtschaft,
Telekommunikation und neue Medien e.V.

Albrechtstraße 10 | 10117 Berlin

Frank Termer | Projektleiter Software, Technologien und Märkte
T 030 27576-232 | f.termer@bitkom.org

www.bitkom.org

bitkom