Zur Nutzerfreundlichkeit von API-Dokumentation in der Praxis:

Überprüfung von Heuristiken zur Erstellung von API-Dokumentation

Masterarbeit im Fachgebiet

Informationsdesign und Medienmanagement

an der

Hochschule Merseburg (FH)

Vorgelegt von:	Chilja Speransky
Matrikelnummer:	21527
Erstgutachter:	Prof. Dr. Michael Meng
Zweitgutachter:	Dipl. Technik-Redakteur (FH) Stefanie Steinhardt
Datum:	16.08.2017

Abstract

Der Einsatz von APIs als Schnittstellen wird immer wichtiger für die Nutzung des Webs 3.0. Nur mithilfe von modularen Systemen kann die Menge an Daten und technischen Möglichkeiten im Internet bewältigt werden. Für diesen Zweck werden Web-APIs eingesetzt, die bereits vorhandene Datenbanken, Funktionen etc. für andere Anwendungen zugänglich machen. Jede API verfügt über ihre eigene Logik und Namensgebung, die für Entwickler verständlich dargestellt werden muss. API-Dokumentation ist das Instrument zur Bewältigung dieser Informationsmenge.

In dieser Arbeit werden existente API-Dokumentationen mithilfe von vorhandenen empirischen Forschungserkenntnissen untersucht. Dabei handelt es sich um acht konkrete Heuristiken, die in verschiedenen Studien im Rahmen eines Forschungsprojekts der Hochschule Merseburg erarbeitet wurden. Die Heuristiken beziehen sich auf eine konsistente Gestaltung, die Grundregeln guter Dokumentation, eine transparente Navigation, die Struktur, den schnellen Einstieg, den selektiven Zugriff auf konzeptuell Informationen, eine intelligente Suchfunktion sowie die codenahe Platzierung von Informationen. Methodisch wird eine qualitative, heuristische Untersuchung durchgeführt, die dem Ansatz von Watson (2013) entspricht. Untersucht werden 30 Dokumentationen von REST-APIs, die im Internet frei zugänglich sind.

Durch die Untersuchung der 30 API-Dokumentationen konnte aufgezeigt werden, dass besonders die Heuristiken erfüllt wurden, die sich auf die Grundlagen der Dokumentationserstellung beziehen. Heuristiken mit stark usability-orientierten oder interaktiven Kriterien wurden eher seltener erfüllt. Darüber hinaus hat die Untersuchung zahlreiche Fragestellungen für zukünftige Untersuchungen aufgezeigt. Die möglichen Hypothesen beschäftigen sich mit der Komplexität/Linearität von Dokumentation, der Gewichtung der Kriterien in einer Untersuchung, der Anwendung von Frameworks für API-Dokumentation sowie der Auslagerung von Inhalten der Dokumentation auf externe Webseiten beschäftigen.

Inhalt

1. Einleitung

> *„Wir befinden uns an der Schwelle zu einem Zeitalter vernetzter Intelligenz – einem Zeitalter, das eine neue Wirtschaft, eine neue Politik und eine neue Gesellschaft hervorbringen wird. Die Wirtschaft ist im Begriff, sich grundlegend zu ändern, die staatlichen Behörden müssen sich erneuern, und selbst der einzelne* [sic!] *wird sich neu definieren können – all das mit Hilfe der Informationstechnologie.“*
> (Tapscot 1997: 18)

Bereits im Jahr 1997 sah Tapscot die weitreichenden Auswirkungen von vernetzter Intelligenz und der Ausbreitung der Digitalisierung hervor. Heute hat das Internet tatsächlich einen sehr hohen Stellenwert im Leben der meisten Menschen eingenommen. Viele Deutsche können sich nicht mehr vorstellen, ohne die Vorzüge der dauerhaften Erreichbarkeit und der unendlichen technischen Möglichkeiten des Webs zu leben (vgl. Generation What o. D.). Aufgrund der hohen Nachfrage wird der Markt mit Apps und Anwendungen überschwemmt, die vom Freizeitspaß bis zu komplizierten Rechenleistungen alles zu bieten haben. Dabei zählen aber nicht nur Innovation und Interaktion, sondern im gleichen Maße auch die intelligente Vernetzung von Daten und Funktionen. Immer mehr Anbieter nutzen dafür bereits bestehende Code und Datenbestände anderer, um ihre eigenen Anwendungen zu verschlanken. Diese ökonomische Entwicklung ermöglicht es, umfangreiche Leistungen anzubieten, die in Eigenleistung häufig zu aufwendig oder teuer wären. Möglich ist das Teilen von Datenbeständen, Daten, Code etc. über Schnittstellen, sogenannte APIs.

APIs bilden als Werkzeug die Grundlage für eine Vielzahl von digitalen Möglichkeiten. Ideen und Projekte wie das Internet der Dinge, Smart Cities, aber auch anbieterübergreifende Bezahl- und Login-Funktionen wären ohne APIs nicht realisierbar. Sie ermöglichen erst die vereinfachte Kommunikation, die Vernetzung und den potenziellen Zugriff von und auf unterschiedliche Systeme und Anwendungen. Häufig greifen dabei externe Anbieter auf APIs zu, um deren Inhalt in den eigenen zu integrieren.

APIs bestehen aus Code, so dass sie vor allem von Softwareentwicklern bzw. Programmierern genutzt werden. In diesem Fall nutzen die Entwickler eine fremde API, bei der sie weder den Aufbau noch die Logik kennen. Diese Informationen soll API-Dokumentation liefern. Die Umsetzung von API-Dokumentation zeigt ebenso viele Varianten wie die Bandbreite an Apps, tatsächlich gibt es keine allgemein verpflichtenden Spezifikationen für deren Erstellung. Die Realisierung von Vernetzungen und Digitalisierungen ist jedoch maßgeblich abhängig von verständlichen und schnell einsetzbaren APIs, so dass API-Dokumentation ein besonderer Stellenwert zukommt.

In dieser Arbeit wird untersucht, inwiefern bestehend API-Dokumentation empirisch erarbeiteten Qualitätsmerkmalen entspricht. Dafür wird auf die Erkenntnisse eines Forschungsprojekts der Hochschule Merseburg zurückgegriffen, das sich mit Entwickler-Dokumentation beschäftigt. Aufbauend auf den im Projekt entwickelten Heuristiken werden 30 API-Dokumentationen anhand eines Kriterienkatalogs geprüft. Das Ziel ist es herauszufinden, welche Kriterien bereits von API-Dokumentationen in welchem Maß erfüllt werden und welche eher nicht.

Nachfolgend werden die Begriffe API und API-Dokumentation näher erklärt. Anschließend werden die Erkenntnisse des Forschungsprojekts der Hochschule Merseburg dargelegt. Zum Schluss werden die Untersuchung vorgestellt und deren Ergebnisse detailliert diskutiert.

2. Theoretische Grundlagen: APIs und API-Dokumentation

In diesem einleitenden Kapitel werden die Begriffe API sowie API-Dokumentation vorgestellt und erklärt. Damit das Konzept von APIs bestmöglich dargestellt werden kann, werden sie zum einen aus Sicht der Informatik und zum anderen als technisches Produkt im wirtschaftlichen Kontext dargestellt.

Anschließend wird der Begriff API-Dokumentation definiert und eine Reihe von Forschungsansätzen und Studien bezogen auf die Erstellung von API-Dokumentation vorgestellt.

2.1. APIs

Der Begriff API kommt aus dem Englischen und ist ein Akronym[1]. Er steht für „Application Programming Interface“, was auf Deutsch übersetzt „Schnittstelle zur Anwendungsprogrammierung“ bedeutet. Bedingt durch die Dominanz des Englischen in der Informatik wird in Deutschland in den meisten Fällen die englische Bezeichnung API verwendet.

APIs ermöglichen den Zugriff auf Software, Datenbanken und Hardware (Festplatte, Grafikkarte etc.) sowie das Erstellen von Komponenten grafischer Benutzeroberflächen und von ergänzender Software (vgl. Gründerszene o. J.). Sie können aus mindestens zwei Perspektiven betrachtet werden. Zum einen bestehen sie aus Code, weswegen eine technische Betrachtung aus Sicht der Informatik naheliegend ist. Zum anderen haben APIs eine große Bedeutung für das Web 2.0[2], so dass sie ebenfalls im wirtschaftlichen Kontext beleuchtet werden müssen.

2.1.1. APIs aus Sicht der Informatik

Generell gehören APIs aus technischer Sicht in die Informatik, die sich wiederum in viele verschiedene Disziplinen aufgliedert (vgl. Fischer/Hofer 2010: 430 ff.). APIs können Gegenstand aller Disziplinen sein, besondere Anwendung finden sie aber in der praktischen Informatik[3].

Stylos beschreibt eine API als Möglichkeit, Code wiederzuverwenden, ohne dass der Entwickler den Code verstehen oder verändern muss. Ganz im Gegenteil dazu interagiert der Entwickler ausschließlich mit der für ihn vorgesehenen Schnittstelle.

> *“With APIs, programmers can reuse code without needing to modify, understand or even see the implementation, and instead interacting only with the programming interface.”*
> (Stylos 2009: 1)

Im Gegensatz zu Binärschnittstellen, den ABIs, die u. a. *„(...) die Vereinbarungen dafür* [definieren]*, wie Code generiert werden soll“* (Galloway 2014: 55), bezieht sich die API als Schnittstelle auf den reinen Zugriff auf den Quellcode.[4] Tritsch spezifiziert das Prinzip von APIs eingehender:

> *„Ein[e] API stellt den Satz von Funktionen dar, mit der ein Anwendungsentwickler das Verhalten eines Systems beeinflussen kann oder aber Dienste eines Systems in Anspruch nehmen kann, ohne dieses System auf der Quellcode-Ebene zu modifizieren.“*
> (Tritsch 1997: 7)

---

[1] Ein Akronym ist ein *„aus Anfangsbuchstaben, Silben und anderen Wortteilen einer Wortgruppe […] gebildetes Kurzwort.“* (Wörterbuch zur Lexikographie und Wörterbuch-Forschung 2010: s. v. Akronym).

[2] Das Web 2.0 bezeichnet die Veränderung des Nutzerverhaltens von einer passiven hin zu einer aktiven Rolle. Der Begriff Web 2.0 wird im Kapitel Wirtschaftliche Aspekte von APIs näher beleuchtet.

[3] Die praktische Informatik widmet sich vor allem der Softwareentwicklung sowie deren Anwendung (vgl. Fischer/Hofer 2010: 430).

[4] Neben den protokollorientierten APIs existieren außerdem funktions-, objekt- und datenorientierte Schnittstellen (vgl. ITWissen.info 2013).

Tritsch benennt Funktionen als zentrales Element, um durch APIs einen Zugang zu einem fremden System zu ermöglichen. Durch sie werden grundsätzlich die Fähigkeiten und die Arbeitsweise einer API festgelegt. Der API-Entwickler beschränkt und regelt anhand der Fähigkeiten der Funktionen den Zugriff auf seine Software, Hardware und/oder Datenbanken. Der Nutzer erhält keinen Zugriff auf den Quellcode und kann nur auf die Eigenschaften und Fähigkeiten zugreifen, die zuvor durch den Entwickler festgelegt wurden. Das gilt auch für Daten, die durch den Entwickler zur Verfügung gestellt werden. Der Nutzer hat dementsprechend keinen Zugriff auf das Back-End[5] der API.

Funktionen werden nicht nur für APIs, sondern in der gesamten Programmierung genutzt. Sie enthalten eine Sammlung an Befehlen und Aufrufen, die einen bestimmten Effekt erzielen. Dafür können beispielsweise Objekte miteinander kommunizieren[6] sowie Parameter, Variablen usw. festgelegt werden (vgl. Highscore 2010). Die Namensgebung für Parameter und Funktionen ist frei wählbar und wird durch den Entwickler bestimmt.

Alle Funktionen einer API werden durch den Entwickler in einer bestimmten Sprache definiert. Diese Sprache wird auch als Protokoll bezeichnet.

2.1.1.1.API-Protokolle

Die technische Umsetzung von APIs ist hochindividuell, da für jede Schnittstelle neuer Code geschrieben werden muss. Aus Zeit- und Kostengründen wird häufig auf standardisierte Protokolle wie z. B. SOAP, XML-RPC oder RESTful-http zurückgegriffen. Die häufigsten Datenformate sind XML[7] und JSON[8] (vgl. Gründerszene 2009). Nachfolgend werden die drei genannten Protokolle kurz beschrieben.

XML-RPC ist ein standardisiertes Protokoll zum Funktions- oder Methodenaufruf in verteilten Systemen[9]. Es steht für „Extensible Markup Language Remote Procedure Call“ und kann flexibel in vielen Systemumgebungen[10] auch bei der Anwendung unterschiedlicher Programmiersprachen implementiert werden. Das ist möglich, weil die Darstellung der Daten in XML und die Übertragung der Daten im http-Standard erfolgen. Mit XML-RPC können auf diese Weise „*(...) complex data structures (...) be transmitted, processed and returned.*“ (UserLandSoftware.inc 1999). Die Vorteile dieses Protokolls liegen in seiner Einfachheit sowie in seiner Flexibilität bezogen auf Systemumgebungen und Programmiersprachen.

SOAP ist der Nachfolger von XML-RPC. Das Akronym steht für „Simple Object Access Protocol”, wobei häufig kritisiert wird, das SOAP keineswegs einfach ist (vgl. SoapUI by Smartbear 2017). Ebenso wie XML-RPC ist SOAP ein XML-basierter Standard. Die Elemente und Attribute in XML werden durch ein

[5] Das Back-End ist die Programmierung hinter dem Front-End (die API). Ein passender Vergleich für ein Back-End sind bei einem Theaterstück alle Handlungen, die hinter der Bühne geschehen (vgl. Hery-Mossmann 2016[a]).

[6] Dieser Ansatz wird auch „objektorientierte Programmierung“ genannt (vgl. Highscore 2010).

[7] XML ist eine Auszeichnungssprache, mit der Daten übersichtlich gespeichert und transportiert werden können. Das Besondere an XML und anderen Auszeichnungssprachen ist, dass es gleichzeitig maschinen- und menschenlesbar ist (vgl. w3schools 2017[a]).

[8] JSON ist eine JavaScript Bibliothek und bezeichnet Text, der mit der JavaScript Notation ausgezeichnet ist. Mit JSON können Daten gespeichert und ausgetauscht werden (vgl. w3schools 2017[b]).

[9] Ein verteiltes System hat *„(…) eine Reihe einzelner Funktionseinheiten, die miteinander über ein Transportsystem verbunden sind* [und] *in Zusammenarbeit Anwendungen bewältigen.“* (Bengel 2015: 4). Die Funktionseinheiten können PCs, Server, Datenbanken oder Geräte wie Drucker o. ä. sein (vgl. ebd.).

[10] Systemumgebungen können in der Informatik viele Ausprägungen annehmen. Generell handelt es sich um eine Zusammenfassung von IT-Systemen, Programmen und Ressourcen. Die Bezeichnung der Systemumgebung hängt von ihrem Zweck ab, z. B. Entwicklungsumgebung, Testumgebung oder Produktionsumgebung. Sie können über Schnittstellen miteinander kommunizieren oder isoliert existieren (vgl. Gräsel 2012).

XML-Schema[11] explizit definiert, so dass eine strikte und klar definierte Struktur entsteht. Ein speziell für SOAP definiertes XML-Schema ist WSDL. Obwohl die Anwendung von WSDL nicht Pflicht ist, wird sie trotzdem empfohlen, um die Kommunikation zu vereinfachen (vgl. ebd.). WSDL enthält alle Eigenschaften und Verbindungen der Elemente und Attribute, so dass das Schema als grundlegende Dokumentation genutzt werden kann. Die Vorteile von WSDL sind gleichzeitig auch die Nachteile von SOAP (vgl. ebd.). Jede Änderung im XML-Schema muss ebenfalls von den Nutzern der API übernommen werden, ansonsten sind die Anfragen ungültig. Das Updaten oder Pflegen der API gestaltet sich dementsprechend schwierig.

Trotz des Pflegeaufwands verfügt SOAP über eine besondere Flexibilität, da es mit unterschiedlichen Transport-Protokollen kombiniert werden kann, z. B. http, SMTP, TCP, oder JMS. In der Realität wird allerdings meist nur http als Transport-Protokoll benutzt. Das liegt auch an der Entwicklung des Internets, die sich bisher stark auf http fokussiert hat (vgl. SoapUI by Smartbear 2017). SOAP eignet sich vor allem für APIs, bei denen die Kommunikation sehr stark reglementiert werden muss oder die Länge der Anfragen kein Entscheidungskriterium ist. Für einfachere und schnellere Anfragen ist ein anderes Protokoll besser geeignet, nämlich REST.

Der Begriff „Representational State Transfer“, kurz REST, wurde von Fielding im Rahmen seiner Dissertation eingeführt (vgl. Fielding 2000: 76 ff.). Er beschreibt eine Reihe von Designmechanismen für die Architektur von Web-Services[12] und stellt eine Abstraktion der Architektur des Internets dar. Die Architekturform wird auch als RESTful-http bezeichnet (vgl. Tilkov 2009). Einer der wichtigsten Mechanismen ist der Stateless-Zustand, der die Unabhängigkeit einer Anfrage beschreibt. Die Anforderung enthält alle notwendigen Informationen selber, wodurch sie zustandslos und leichter zu handhaben wird.

> *“Stateless server-side components [...] are less complicated to design, write, and distribute across load-balanced servers.”*
> (Rodriguez 2015)

Der maßgebliche Unterschied zwischen REST und anderen Architekturstilen für das Web liegt in der Forderung nach einer einheitlichen Schnittstelle. Die einheitliche Schnittstelle ergibt sich aus der Directory Structure[13] von REST, die sich an Ressourcen orientiert. Aus diesem Grund wird REST auch „Resource Orientated Archtitecture“ genannt (vgl. Tilkov 2009). Der Entwickler kann alle möglichen Teile eines Service mit einem „Uniform Resource Identifier“, kurz URI, versehen und so eindeutig identifizierbar machen.

> *„Jede Bestellung, Filiale, Region, jedes Produkt, jeder Artikel kann mit einem eigenen URI identifiziert werden. Auch die ´Liste aller Regionen, in denen der Umsatz größer als 5 Mio. Euro war´, die ´Produkte aus der Kategorie Spielwaren´, alle ´Bestellungen aus dem Jahr 2008´ sind gültige Ressourcen und sollten einen möglichst stabilen und langlebigen URI erhalten.“*
> (Tilkov 2009)

Ein URI kann eine abstrakte oder physische Ressource identifizieren. Er beinhaltet die Komponenten Uniform Ressource Locator (URL) und Uniform Ressource Name (URN). Ein URL lokalisiert ein Objekt

[11] Mithilfe von XML Schema können die Struktur, der Inhalt und die Semantik von XML-Dokumenten detailliert definiert werden. Beispielsweise kann festgelegt werden, in welcher Reihenfolge Elemente vorkommen dürfen oder eine begrenzte Auswahlmöglichkeit für den Inhalt angegeben werden (vgl. W3C 2000).

[12] Web-Services bezeichnen die Funktionsweise von Web-APIs in Verbindung mit den vorgestellten Protokollen XML-RPC, SOAP und REST. Der Begriff wird im Kapitel Das Internet - die digitale Welt von heute detailliert erklärt.

[13] Die „Directory Structure“ ist ein Begriff der Informatik und beschreibt, wie Datenverzeichnisse und Daten für den Nutzer abgebildet werden (vgl. dpBestflow 2015).

während ein URN dauerhaft und ortsunabhängig beschreibt, was ein Objekt ist (vgl. t3n 2013). Ein Beispiel für einen URI ist „*t3n.de*“, für einen URL „*http://t3n.de*“ und für einen URN „*urn:isbn:3827370191*“ (vgl. ebd.)

Zusammengefasst werden die identifizierbaren Ressourcen im RESTful-http über Links miteinander verknüpft. Eine solche Architektur bietet neben der Auffindbarkeit durch Suchmaschinen auch programmiertechnische und sicherheitsbasierte Vorteile (vgl. t3n 2013). Die Länge der Anfragen verkürzt sich ebenfalls deutlich. Damit die eindeutige Identifizierung der Ressourcen funktioniert, müssen sie über eine einheitliche Schnittstelle verfügen. Dafür wurden Methoden konkretisiert, die als http-Kontrakt zwischen spezifischen Ressourcen und speziell zugeschnittenen Clients oder auch generischen Clients (z. B. einem Browser) genutzt werden. Die REST-Architektur und die http-Methoden werden für den Aufbau von und die Kommunikation mit APIs genutzt.

In dieser Arbeit werden ausschließlich APIs untersucht, die mit RESTful-http programmiert wurden. Die Auswahl wurde aufgrund der aktuellen technischen Entwicklung und der zunehmenden Nutzung von REST getroffen (vgl. MSXFAQ 2017), nicht aufgrund von qualitativen Überlegungen. Generell kann kein Protokoll für einen Web-Service als besser oder schlechter bezeichnet werden, die Wahl ist in jedem Fall abhängig von der Aufgabe der API. SOAP wird beispielsweise tendenziell für die Kommunikation zwischen verschiedenen Sprachen und Tools genutzt, während REST hauptsächlich identifizierbare Ressourcen überträgt (vgl. Rocher/Brown 2010: 425).

REST-APIs können sich trotz des einheitlichen Protokolls stark in ihrer Anbindung unterscheiden. Die verschiedenen Anbindungstypen werden nachfolgend vorgestellt.

2.1.1.2.API-Typen

APIs können auf unterschiedliche Weise als Schnittstelle fungieren. Zum einen gibt es die weit verbreitete Nutzung im Web und zum anderen die Nutzung innerhalb von Hardware und Software. Generell können Entwickler per API Software, Hardware, Datenbanken, graphische Interfaces usw. an ein System anbinden. Diese Anbindung kann online oder offline erfolgen. APIs, die online zur Verfügung stehen, werden auch Web-APIs genannt. In dieser Arbeit liegt der Fokus auf Web-APIs, weswegen sie im Zentrum der Ausführungen stehen. Andere APIs werden inhaltlich nur angeschnitten.

APIs können angelehnt an die Art der Anbindung in interne APIs, externe APIs, Plattform-APIs und Authentifizierungs-/Autorisierungs-APIs unterteilt werden (vgl. Gründerszene 2009). Nachfolgend werden die verschiedenen API-Typen näher beleuchtet.

Interne APIs werden verwendet, um eine Software zu modularisieren. Das bedeutet, dass der komplexe Code einer Software in unterschiedlichen, voneinander unabhängigen Paketen abgelegt wird. Die Pakete können später über interne APIs miteinander kommunizieren. Diese Art der Softwarearchitektur wird auch SOA genannt, ein Akronym für „Service Orientated Architecture“. Der Begriff „Service“ beschreibt neben der Verringerung der Komplexität einer Software den größten Nutzen einer solchen Architektur. Durch die Unabhängigkeit voneinander können die Module einfach gepflegt und angepasst werden, ohne dass Konsequenzen für andere Module eintreten (vgl. Gründerszene 2009).

Externe APIs sind die häufigste Form von Web-APIs. Sie ermöglichen den indirekten Zugriff auf den Code eines anderen. Will z. B. ein App-Anbieter eine Karte oder einen Routenplaner in seine

Applikation[14] integrieren, kann er die API von Google Maps dafür nutzen. Das Vermischen von eigenen und extern bereitgestellten Inhalten wird auch Mash-Up genannt (vgl. Gründerszene 2009.).

Plattform-APIs sind ebenfalls Web-APIs. Sie ermöglichen es Entwicklern, ihre Applikationen bzw. Plug-Ins in andere Webseiten, Apps etc. zu integrieren. Häufig stellen die Betreiber APIs bereit, mit deren Hilfe das Graphical User Interface (GUI) der zu integrierenden Apps/Plug-Ins ebenfalls angepasst wird (vgl. Gründerszene 2009).

Authentifizierungs-/Autorisierungs-APIs regeln die „*[...] Identifikation und Zugriffsrechtsgewährung von Nutzern*" (vgl. Gründerszene 2009). Ein prominentes Beispiel dafür ist Facebook, das eine API für den Nutzerlogin bereitstellt. Der Nutzer kann sich mit seinen Facebook-Nutzerdaten in Apps oder auf Webseiten anmelden, die nicht zu Facebook gehören. Facebook wirbt damit, dass die Webseite TripAdvisor durch die Einführung der Plattform-API für den Nutzerlogin im Jahr 2015 doppelt so viele Neukunden gewinnen konnte wie im Vorjahr (vgl. Facebook o. J.).

Der Fokus dieser Arbeit liegt vor allem auf externen Web-APIs, die im RESTful-http programmiert wurden. Die Einschränkung auf einen API-Typen von Web-APIs gewährleistet in der Untersuchung eine bessere Vergleichbarkeit der Ergebnisse. Die durchgeführte Untersuchung kann aber auch auf andere API-Typen angewendet werden.[15]

2.1.2. APIs als technisches Produkt im wirtschaftlichen Kontext

Die Betrachtung von APIs im wirtschaftlichen Kontext basiert vor allem auf deren Bedeutung für die stark verbreitete Nutzung des Internets. Bevor die wirtschaftliche Betrachtung erfolgen kann, muss deshalb vorerst der Begriff des Internets beleuchtet werden.

2.1.2.1. Das Internet – die digitale Welt von heute

Das Internet spielt heutzutage eine tragende Rolle im Alltag der Menschen. Egal, ob es sich um Social Media, Online-Shopping oder Informationsbeschaffung handelt, das Internet bietet frei zugängliche Informationen unabhängig von Ort und Zeit. Die Architektur des Internets ist jedoch weit komplexer, als viele Anwender glauben. Tatsächlich wird meist nur das World Wide Web genutzt, so dass der Begriff WWW häufig synonym für das Internet gebraucht wird (vgl. Elektronik Kompendium o. J.). Nachfolgend werden der Begriff des Internets und die Rolle von Web-APIs erklärt.

Das Internet ist ein „*(...) globales System aus miteinander verbundenen Computernetzwerken.*" (Digitale Gesellschaft 2012: 4). Die Kommunikation dieser Netzwerke funktioniert über das Internet Protocol, kurz IP. Das IP wiederum enthält je nach Anwendungszweck ein weiteres Protokoll für einen spezifischen Anwendungsfall, z. B. HTTP, SMTP oder FTP. Während das SMTP-Protokoll für das Versenden von E-Mails und das FTP-Protokoll für die Übertragung von Daten genutzt wird, dient das http-Protokoll dem Zugriff auf Webseiten o. ä. im World Wide Web. Das World Wide Web, kurz WWW, wird häufig mit dem Internet gleichgesetzt, bildet tatsächlich aber nur einen Teil der möglichen Internetanwendungen[16].

[14] Eine Applikation ist in der IT *"(…) the use of a technology, system or product."* (Rouse 2017[a]). Häufig wird eine Applikation in diesem Zusammenhang auch als Anwendung bezeichnet. Darüber hinaus gibt es Applikationsprogramme, kurz Apps, die für eine spezifische Aufgabe für den User gebaut werden. Beispiele dafür sind Bildbearbeitungsprogramme, Datenbanken, Web-Browser usw. (vgl. ebd.).

[15] Die in der Untersuchung geprüften Kriterien wurden für APIs allgemein, nicht für einen bestimmten API-Typen oder ein bestimmtes Protokoll empirisch erarbeitet. Aus diesem Grund sind die Kriterien auch auf andere API-Typen oder Protokolle anwendbar.

[16] Weitere Internetdienste sind u.a. Peer-to-Peer-Systeme, Internettelefonie, Internetradio, Usenet und Telnet. Generell sind alle Dienste, die über die Architektur des Internets verfügbar gemacht werden, Internetanwendungen (vgl. Laudon et al. 2010: 374 ff.).

Aus Sicht der Informatik ist das WWW ist ein verteiltes System. Verteilte Systeme bestehen aus mehreren Funktionseinheiten, die über ein Transport-Protokoll miteinander kommunizieren und gemeinsam Anwendungen bewältigen können (vgl. Bengel 2015: 4). Das Transport-Protokoll im WWW ist http, was „Hypertext Transfer Protocol" bedeutet. Die Funktionseinheiten des WWW sind Web-Server und Web-Clients. Web-Server sind große Datenbanken, die unendlich viele Dokumente bzw. Webseiten enthalten können. Diese Webseiten wiederum können unendlich viele Unterseiten besitzen, die über Verlinkungen im internen HTML Code[17] zugänglich gemacht werden. Der Zugriff auf einen Web-Server erfolgt durch einen Web-Client. Web-Clients sind z. B. Browser wie Mozilla Firefox, Opera oder Internet Explorer, die über das Internet im http-Protokoll mit dem jeweiligen Web-Server kommunizieren.

Neben Web-Browsern können auch andere Funktionseinheiten des WWW Web-Clients sein. Entscheidend ist, dass die Funktionseinheit eine Anfrage stellen kann. Integriert z. B. ein Anbieter mittels der Facebook API den Facebook Login in seine Webseite, schickt die Webseite bei einem Login eine Anfrage an Facebook. So wird der Anbieter zum Web-Client und Facebook zum Web-Server, der den Login durchführt. Die Funktion des Logins von Facebook durch eine API wird auch als Web-Service bezeichnet (vgl. Thattil 2014).

Für die Nutzung von Web-Services sind zwei Dinge Voraussetzung: der Einsatz von http als Transport-Protokoll und das Verwenden einer gemeinsamen Sprache wie SOAP oder REST (vgl. Thattil 2014). SOAP/REST ermöglichen die Kommunikation zwischen unterschiedlichen Programmiersprachen, z. B. .NET und Java, in denen Anwendungen programmiert werden können. Tatsächlich könnte eine Anfrage, die in der Programmiersprache Java an eine Anwendung in .NET geschickt wird, nicht von .NET verarbeitet werden. Java und .NET können aber beide SOAP/REST als Vermittler, sozusagen als kleinsten gemeinsamen Nenner nutzen. Auf diese Weise können Web-Services und speziell APIs als Brücke zwischen unterschiedlichen Programmiersprachen genutzt werden.

2.1.2.1.1. Beispiel für einen Web-Service

Die Funktionsweise von APIs als Web-Service kann durch das Beispiel des Einbindens einer Google Karte auf einer externen Webseite gut veranschaulicht werden. Der Betreiber der Webseite stellt an die API eine Anfrage für eine „Route über zwei Wegpunkte". Damit die Anfrage gültig ist, also von der API verstanden wird, müssen die richtige Sprache in Form eines bestimmten Protokolls sowie die korrekte Funktion samt der geforderten Parameter verwendet werden. Parameter können z. B. der Ausgangspunkt, das Ziel, die Wegpunkte, das Verkehrsmittel etc. sein. Wenn alle Voraussetzungen für eine gültige Anfrage erfüllt sind, sendet die API eine Antwort mit den geforderten Inhalten. In diesem Fall wäre das die berechnete Route über die geforderten Wegpunkte. Das dazugehörige Code-Beispiel ist in Form von Request und Response (in JSON und XML) im Anhang unter API Beispielanfrage zu finden.

In dem Beispiel wird dem Betreiber der Webseite durch die API kostenlos[18] ein Navigationsdienst in Form eines Web-Services zur Verfügung gestellt, der auf ein komplexes System zurückgreift. Durch

[17] HTML, oder „Hypertext Markup Language", ist die Auszeichnungssprache für das WWW. Sie wird standardmäßig für das Strukturieren von Webseiten eingesetzt (vgl. W3C 2016). Die Verlinkung von Unterseiten wird in HTML üblicherweise mit dem <a> Element sowie dem href-Attribut durchgeführt.

[18] Google Maps kann für die nicht kommerzielle Nutzung kostenlos verwendet werden. Im Fall der gewerblichen Nutzung ist der Karten- und Navigationsdienst von Google kostenpflichtig (vgl. Google 2015).

den simplen Zugriff per API auf die Server und das System[19] von Google kann der Betreiber der Webseite diesen Dienst jedoch ohne großen Aufwand bereitstellen. Im Vergleich dazu wäre es für den Betreiber sehr zeit- und kostenintensiv, selber einen Navigationsdienst zu programmieren und zu unterhalten. Die Bereitstellung der Dienste von Google Maps erfordert riesige Server und Datenbanken, die nicht ohne weiteres realisiert werden können. Tatsächlich könnte allein mit dem Strom, den alle Google Server im Jahr verbrauchen, eine Kleinstadt versorgt werden (vgl. Süddeutsche Zeitung 2012).

Die Bedeutung von APIs ist basierend auf den bisherigen Ausführungen in technischer Hinsicht sehr groß. Ohne APIs wäre die heute übliche großflächige Verknüpfung von Web-Inhalten im WWW nicht denkbar. Aus diesem Grund ist eine Betrachtung der wirtschaftlichen Aspekte rund um APIs naheliegend.

2.1.2.2. Wirtschaftliche Aspekte von APIs

APIs leisten einen großen Beitrag zu der engmaschigen Verknüpfung von eigenen und fremden Diensten in Webseiten, Apps etc. Moderne Webseiten greifen zunehmend auf APIs zurück, um ihren Inhalt mit fremdem zu vermischen oder einzelne externe Anwendungen in ihre Webseite einzubinden.

> *"Market Conditions have changed in ways that makes APIs relevant to any business with assets that others would like to use."*
> (Jacobson et al. 2012: 12)

Prominente Beispiele dafür sind der Like-Button von Facebook oder das Prinzip von Pinterest, bei dem eine individualisierte Webseite aus zusammengepinnten Inhalten entsteht (vgl. Pinterest o. J.). Weiterhin gibt es Webseiten, die gesammelte Angebote generieren, z. B. Urlaubssuchen. Daraus entstehen sogenannte *„[...] Mash-Ups oder Executive Dashboards, die Daten aus zwei oder mehr Quellen abrufen"* (Rouse 2017[b]).

Die Betreiber von solchen Webseiten/Apps stellen zahlreiche Informationen und Dienstleistungen zur Verfügung, die meist kostenlos sind. Ihren Gewinn erwirtschaften sie durch Werbeeinnahmen (Youtube) oder den Verkauf von Daten (Google). Je mehr Nutzer eine Webseite/App hat, desto höher werden Ihre Werbeeinnahmen ausfallen. Analog dazu steigt die Menge an verkäuflichen Daten mit der Anzahl der Nutzer.

Die Gewinnerwirtschaftung der Betreiber hat sich parallel zu der Rolle der Nutzer im vergangenen Jahrzehnt entwickelt. Die Rolle der Nutzer im Web hat sich von einer passiven hin zu einer aktiven und gestaltenden verändert. Gemeinhin wird diese Entwicklung vom Konsumenten zum Teilnehmer auch als Web 2.0 bezeichnet (vgl. Walsh et al. 2011: 3 f.). Explizit durch die nutzergesteuerte Gestaltung von Webseiten, z. B. Youtube, Twitter, Facebook usw., können große Gewinne durch Werbemaßnahmen erzielt werden. Das liegt vor allem an den sehr hohen Nutzerzahlen (vgl. ebd.).

Entscheidend für den Erfolg einer Webseite/App ist letztendlich auch deren Interoperabilität und Vernetzung, *„erst dadurch wird eine isolierte Anwendung zu einer offenen Plattform"* (Walsh et al. 9). Dementsprechend groß ist das Interesse daran, Produkten durch eine weite Streuung und

[19] Google nutzt neben der Ortung über GPS auch die Standortdaten von mobilen Geräten. Jedes Gerät, auf dem Google Maps installiert ist und das über eine GPS-Funktion verfügt, sendet Daten an Google. Mithilfe dieser Daten kann u. a. die aktuelle Verkehrslage abgebildet werden (vgl. Berkemeyer 2017).

Vernetzung zu mehr Bekanntheit und Beliebtheit zu verhelfen. An dieser Stelle wird die Bedeutung von APIs als Schnittstelle zwischen Apps, Webseiten usw. im Web 2.0 deutlich.

Neben der reinen Gewinnerwirtschaftung im WWW spielt auch die Einsparung von Zeit und Kosten eine große Rolle im Zusammenhang mit APIs. Prinzipiell ermöglichen sie einen vereinfachten Zugriff auf Quellcode, was viele Vorteile für Nutzer und Anbieter birgt. Auf der Nutzerseite handelt es sich um ökonomische Aspekte wie die Verringerung der zu lernenden bzw. zu programmierenden Codemasse, die auch zu einer Zeitersparnis führt. Der Anbieter wiederum zieht seinen Vorteil aus der Verbreitung seiner Anwendung, so dass er seine Gewinne steigern kann.

Bezogen auf externe APIs ist es besonders wichtig, dass das Angebot eines anderen in das eigene integriert werden kann, z. B. ein Login oder eine Karte. Durch die Kombination von eigenen und fremden Inhalten können simpel und schnell umfangreiche Angebote generiert werden. Je nach API-Typ kann dieser Aspekt auch für den Anbieter interessant sein. Beispielsweise im Fall einer internen API würde sich die Verringerung der zu lernenden Codemasse auf die „internen“ bzw. eigenen Entwickler beziehen und so Kosten sparen. Ein weiterer zeit- und kostensparender Aspekt ist in jedem Fall die leichtere Wartung, da der Code „hinter“ der API ohne Konsequenzen für das Front-End[20] gepflegt und angepasst werden kann.

Durch die Einsparung von Zeit und Personalkosten ergibt sich ein finanzieller Vorteil für Unternehmen, egal ob sie im Web oder softwarebasiert mit APIs arbeiten. Zusätzlich erhöht sich auch die Sicherheit der Systeme, da Fehlerquellen schneller geortet und behoben werden können (vgl. Omkt o. J.).

2.1.2.3.Trends

APIs werden neben dem häufigen Einsatz als Webservices auch in anderen Kontexten eingesetzt. Die Möglichkeit über Schnittstellen sprachenunabhängig zu kommunizieren eröffnet informationstechnologische Trends wie Microservices oder das Internet der Dinge.

Microservices beschreibt einen Programmieransatz, bei dem Anwendungen aus modularen Paketen zusammengesetzt werden. Die Kommunikation zwischen den als „Microservices“ bezeichneten Paketen wird über APIs gelöst. Der Ansatz Anwendungen in der Programmierung zu modularisieren ist nicht neu (siehe auch SOA; interne APIs), das unabhängige Deployment der Microservices aber schon. Tatsächlich kann jeder Microservice unabhängig von den anderen Paketen in den Produktivstand gehen, was eine große Unabhängigkeit in der Entwicklung bedeutet. Die großen Vorteile von Microservices sind außerdem die Übersichtlichkeit des Codes, die strikte Trennung von Prozessen, die Vermeidung von ungewollten Abhängigkeiten und die Unabhängigkeit der Entwickler-Teams (vgl. Wolff 2017). Besonders wichtig ist ebenfalls die Robustheit des Systems, da ein ausfallender Microservice keine oder keine grundsätzliche Auswirkung auf die anderen Services hat. Im Fall eines Bugs wäre nur der betroffene Service nicht mehr abrufbar. Ein bekanntes Beispiel für die Anwendung von Microservices ist der Streamingdienst Netflix (vgl. ebd.)

Das Internet der Dinge beschreibt eine Vernetzung von realen Dingen über das Internet, die Daten in eine Cloud senden. Die Verbindung zwischen den Dingen, beispielsweise einem Auto, und der Cloud gewährleisten APIs. Besonders interessant an dem Internet der Dinge sind die Daten, die die Dinge

[20] Das Front-End ist das, was der Nutzer sieht (vgl. Hery-Mossmann 2016[a]). Im Fall der API sind es die Anfragen an und die Antworten von der API.

senden. Die Menge an auswertbaren Daten wird zukünftig weiter ansteigen, sodass viele Lebenssituationen verbessert werden können, z. B. Müllbeseitigung, Verkehr, industrielle Fertigung, Gesundheit, Handel usw. (vgl. ebd.).

> *"Cloud-based services are the way in which the IoT* [Internet of Things – Anm. d. Autorin] *is connected to data. APIs are the skybridge — IoT on one side, useful information and plentiful data crunching capabilities on the other. APIs make IoT useful, turning limited little* ***things*** *into powerful portals of possibilities."*
> (Spencer/Riggins 2015)

Ohne APIs wäre das Internet der Dinge demzufolge nicht in der Form realisierbar, in der es heute besteht. Manche Experten bezeichnen die momentane Entwicklung des WWWs bereits als Web 3.0. Im Unterschied zum Web 2.0 zeichnet sich das Web 3.0 dadurch aus, dass die eingebundenen Systeme von Menschen erstellte Inhalte eigenständig verstehen können. Dieser Ansatz der künstlichen Intelligenz, der den Anspruch auf selbstdenkende Maschinen erhebt, ist maßgeblich für das Web 3.0 (vgl. Upon Online Marketing 2014).

Experten schätzen, dass die Anzahl der mit dem Internet verbundenen Geräte bis zum Jahr 2020 von derzeit 15 Millionen auf 34 Millionen steigen wird. Dadurch können Menschen, Prozesse, Daten und Dinge in einer ganz neuen Größenordnung verbunden werden. Diese Nutzung des Internets wird auch als „Internet of Everything" bezeichnet (vgl. Chambers 2012). Analog zu dem Internet der Dinge wird die Kommunikation durch APIs ermöglicht werden, wodurch deren Bedeutung für die Entwicklung des Internets noch einmal unterstrichen wird.

2.2. API-Dokumentation

Durch die zuvor geschilderte Entwicklung des Webs hin zum Web 2.0 und 3.0 erhalten APIs auf technischer Ebene zunehmend mehr Aufmerksamkeit. Die Anzahl an neuen APIs steigt rasant an, deswegen müssen Entwickler sich schnell in sie einarbeiten können. Für diesen Zweck wird API-Dokumentation bereitgestellt. Ihre Erstellung wird zum Teil von der Informatik selbst geleistet, teilweise werden zusätzlich Technische Redakteure miteingebunden. Das Ziel der Technischen Redaktion ist es, technische Inhalte zielgruppengerecht medial zu vermitteln. Diese Arbeit verortet sich in der Technischen Redaktion.

2.2.1. Definition und Zweck von API-Dokumentation

APIs sind programmierte Produkte bzw. Produkte bestehend aus maschinenlesbarem Code. Ihre Dokumentation hat somit einen technischen Anspruch, weswegen sie als „Technische Dokumentation" eingestuft werden kann.

Die Technische Dokumentation bezeichnet

> *„(...) die geordnete Zusammenstellung ausgewählter Dokumente und Sprachmaterialien des Herstellers zu einem von ihm erstellten technischen Produkt."*
> (Grupp 2008, zitiert nach Zehrer 2014: 12)

Zu Dokumenten und Sprachmaterialien gehören Daten und Informationen, die über den gesamten Lebenszyklus des Produkts entwickelt und verwendet werden (vgl. Muranko/Drechsler 2006). Der wichtigste Aspekt der Technischen Dokumentation ist die Gebrauchstauglichkeit von Inhalt und Form, die auch als Usability bezeichnet wird. Usability kann u. a. durch die nutzerzentrierte Gestaltung von Inhalten erreicht werden (vgl. Lehrner-Mayer 2016).

> *"Software providers want their products to be usable. Usability is the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use'."*
> (Von ISO/IEC 25064:2013, zitiert nach Reilly 2015)

Die übliche Vorgehensweise in der Technischen Dokumentation ist auf Geräte, Maschinen oder ähnliche Produkte spezialisiert. Der Dokumentation von Software und verwandten Produkten widmet sich eine eigene Teildisziplin der Technischen Dokumentation, die Software-Dokumentation[21]. Sie hat sich auf die Bedürfnisse der Informatik eingestellt.

> *"Software-Dokumentation umfasst alle Arten von Informationen, die einen Benutzer beim Gebrauch eines Produktes am Bildschirm unterstützen."*
> (Grünwied 2013: 24)

Die Rolle des Benutzers ist im Hinblick auf Vorwissen und Kenntnisse variabel. Die grundlegendste Unterscheidung besteht darin, ob es sich um einen klassischen Anwender einer Software oder um einen Anwender auf der Codebene handelt, der programmierbezogene Aufgaben lösen muss. Grünwied unterscheidet in Bezug auf Anwenderdokumentation Benutzer in Laien, unterwiesene Personen und Fachleute (vgl. Grünwied 2013: 74). Ihre Ausführungen sind vor allem für Anwenderdokumentation relevant, Entwickler-Dokumentation spielt eher keine Rolle.

Die Anforderungen an Software-Dokumentation sind in verschiedenen Normen und Richtlinien zusammengefasst.[22] Trotzdem gibt es kontinuierlich Diskussionen zwischen Entwicklern und Technischen Redakteuren über den Umfang von qualitativ hochwertiger Software-Dokumentation.

> *„Obwohl der Wert professioneller und fundierter Dokumentation hinreichend bekannt ist, wird immer wieder gegen den Grundsatz verstoßen, dass Software ausreichend beschrieben werden muss."*
> (Nowak 2002: 1)

Tatsächlich bietet eine vollständige Software-Dokumentation attraktive Vorteile, die einen höheren Zeitaufwand in der Entwicklungsphase rechtfertigen. Neben der kosteneffizienten Wartung und Weiterentwicklung spielt ebenso die Motivation der Entwickler eine Rolle. Durch das Anlegen einer Dokumentation wird z. B. die Leistung des Entwicklers verdeutlicht und so seine Motivation gesteigert (vgl. Nowak 2002: 1).

Die Bestandteile einer vollständigen Software-Dokumentation können grob umrissen werden, im Einzelfall kann es aber zu Abweichungen kommen (vgl. Parson 2013). Die Informatik unterscheidet die Bestandteile generell in High-Level- und Low-Level-Informationen. High-Level-Informationen bezeichnen die Architektur der Software, also die Konzepte und Überlegungen hinter dem Code. Low-Level-Informationen bezeichnen die Aufschlüsselung des Codes sowie seine Referenzen. Während Low-Level-Informationen häufig für einen schnellen Einstieg genutzt werden, werden High-Level-Informationen für die intensivere oder auch effizientere Nutzung des Codes benötigt.

API-Dokumentation ist neben Architektur-, Design- und Referenzdokumentation[23] etc. einer der Bestandteile von vollständiger Software-Dokumentation (vgl. Parson 2013). Im Gegensatz zu den

[21] Die Disziplin der Software-Dokumentation widmet sich u.a. den Themengebieten Anwenderdokumentation, Tutorials (Video, Text o.ä.), Erste-Schritte-Anleitungen, Entwickler-Dokumentation usw.

[22] Dazu gehören beispielsweise ISO IEC IEEE 26514, DIN EN 82079-1 (IEC 82079-1), ISO IEC IEEE 26515, ISO IEC 15289 etc. (vgl. Schmeling/Gocke 2013).

[23] Architektur- bzw. Designdokumentation beschreibt den gesamten Aufbau der Software im technischen Sinn sowie alle Entscheidungen, die während der Entwicklung des Systems getroffen wurden (vgl. Parson 2013). Referenzdokumentation hingegen enthält alle Informationen zu den Schnittstellen, Klassen, Methoden, Funktionen, Parametern usw. (vgl. ebd.).

anderen Bestandteilen kann API-Dokumentation in vielen Fällen sogar als abgegrenzter Bereich verstanden werden. Häufig sind APIs so komplex, dass sie selber eine umfangreiche Dokumentation benötigen. Das liegt daran, dass sie sowohl High-Level- als auch Low-Level-Informationen enthalten.

2.2.1.1. Umsetzung von API-Dokumentation

Aufbauend auf den vorherigen Ausführungen kann API-Dokumentation analog zu Software-Dokumentation als geordnete Zusammenstellung ausgewählter Informationen über die Funktionen einer API und dessen Anwendung klassifiziert werden. Dazu gehört die Dokumentation aller Low-Level- und High-Level-Informationen. Die Erstellung von API-Dokumentation folgt bisher noch keinem offiziellen Standard. Marvin liefert einen allgemeinen Ansatz für API-Dokumentation:

> *"APIs are diverse in terms of language and functions, but what they share is the fact that the user sends a request and the API (sometimes) returns a response. The technical writer's job is to detail what goes into a request, what form the request takes, and what data gets returned."*
> (Marvin 2014: 7)

Er identifiziert die Form und Referenzierung von Anfragen sowie die zurückgegebenen Daten als zentrale Inhalte von API-Dokumentation. Einige Entwickler/Forscher sind sogar der Ansicht, dass diese Informationen schon in dem Code der API zu finden sind und keine weitere Dokumentation nötig ist.

> *"In many cases APIs can achieve a goal of being ´self documenting', where users can learn the APIs simply by trying to use them."*
> (Stylos et al. 2009: 119)

Im Idealfall ist eine API kompakt und logisch strukturiert, so dass sie weitestgehend selbsterklärend ist. Selbsterklärender Code ist in der Praxis tatsächlich eher die Ausnahme, häufig ist der Code nicht so eindeutig wie es dem Entwickler scheint.[24] Mihaly vermutet hinter dem Wunsch nach selbsterklärendem Code mangelnde Kenntnisse der Entwickler in Bezug auf die Erstellung von Dokumentation.

> *"Developers prefer writing code over documentation, rarely showing the same enthusiasm and thoughtfulness for the latter. Some developers claim they write self-documenting code. Others like to point out that ´nobody reads the documentation´. Such excuses create a vicious circle: we dislike documenting because we are not skilled enough and our skills do not improve because we pass up on opportunities to practice them."*
> (Mihaly 2011)

Eine Verbesserung der Situation kann für ihn nur durch die offensive Auseinandersetzung mit der Problematik erfolgen (vgl. Mihaly 2011). Aus diesem Grund empfiehlt es sich, API-Dokumentation als festen Bestandteil der Technischen Dokumentation in der Informatik einzustufen.[25]

2.2.2. Der Problemfall API-Dokumentation

API-Dokumentation ist von zentraler Bedeutung für das Web 3.0. Rasante Weiterentwicklungen, komplexe Vernetzungen und immer neue Funktionen verlangen Entwicklern bereits heute viel ab.

> *"Each new API includes new features, which software developers must learn and apply quickly and correctly. This rapid growth shows no sign of abating, and the demand for increasingly short*

[24] Auf diesen Aspekt wird unter Wie verstehen Entwickler fremden Code im Kapitel Forschungsfeld API-Dokumentation näher eingegangen.

[25] Die Dokumentation von APIs kann sowohl in agilen als auch in normalen Projekten erfolgen. Ergänzend zu den hier dargestellten Konzepten geben z. B. Bednarczyk/Queins (2013) und Rüping (2013) Denkanstöße für eine erfolgreiche Software-Dokumentation unter agilen Bedingungen.

> *time-to-market puts tremendous pressure on today's software developers to learn and apply these new APIs."*
> (Watson et al. 2013: 165)

Die Bedeutung von qualitativ hochwertiger und vollständiger API-Dokumentation wird für das Erlernen neuer APIs häufig unterschätzt. In vielen Fällen wird zwar eine Dokumentation angefertigt, diese ist dann aber unvollständig oder wird nicht ausreichend gepflegt.

> *„Die gegenwärtige Praxis bei Software-Projekten ist dadurch gekennzeichnet, dass der größte Teil der Dokumentation (Projekt-, Entwicklungs-, System-, Benutzerdokumentation) nicht aktuell, unvollständig und oft schwer zu lesen und zu verstehen ist. Übergeordnete Konzepte und Strukturübersichten fehlen und sind deshalb nach einiger Zeit nicht mehr vollständig rekonstruierbar."*
> (qz-online 2016)

Diese Aussage deckt sich auch mit empirischen Forschungsergebnissen, die u. a. die Qualität von API-Dokumentation als zentrales Lernhindernis identifizierten (Watson 2012, Robillard/DeLine 2011 u. v. m.). Tatsächlich beschäftigen sich sowohl externe Anwender als auch interne Entwickler mit API-Dokumentation (vgl. Parson 2013). Eine hochwertige Dokumentation spart somit nicht nur Zeit und Geld, sondern kann sich auch als Kundenbindungsinstrument erweisen.

Die Forschung unterscheidet klar zwischen guter und schlechter API-Dokumentation. Dafür stützt sie sich auf umfassende empirische Forschungserkenntnisse. Häufig auftretende Probleme in schlechter API-Dokumentation sind eine unübersichtliche Struktur, eine schwer verständliche Navigation, eine inadäquate inhaltliche Gestaltung von Abschnitten, das Fehlen von High-Level-Informationen, das Fehlen von Lösungsstrategien bei Bugs, eine unvollständige Referenzierung der Parameter, das Gestalten ohne Fokus auf eine Zielgruppe usw. (s. a. Buse/Weimer 2012; Stylos et al. 2009; Robillard 2009; Robillard/DeLine 2011; Nykaza et al. 2002; Novick/Ward 2006[a]; Novick/Ward 2006[b]). Auf die Forschung an der Gestaltung von API-Dokumentation wird im Folgenden näher eingegangen.

2.2.3. Forschungsfeld API-Dokumentation

Die Technische Redaktion und die Informatik beschäftigen sich schon seit der Etablierung des PC in den 1980er Jahren mit der simplen sowie nutzerorientierten Erstellung und Dokumentation von Code bzw. APIs. Bis heute sind aus dieser Forschung viele verschiedene Ansätze und Erkenntnisse hervorgegangen, die im Folgenden zusammengefasst erläutert werden. Der nachfolgende Forschungsüberblick entspricht dem Rahmen dieser Arbeit und erhebt keinen Anspruch auf Vollständigkeit.

In der empirischen Forschung zu API-Dokumentation gibt es verschiedene Herangehensweisen. Einige Forscher nähern sich der Thematik mittels einer Expertenbefragung oder Tests, andere forschen anhand bereits vorhandener Dokumentationen. In diesem Forschungsüberblick werden Studien aus allen Bereichen nach Themen gegliedert vorgestellt.

2.2.3.1. Wie verstehen Entwickler fremden Code

Bereits 1977 begann Shneiderman sich mit der Frage zu beschäftigen, wie Entwickler Code lernen und verstehen (vgl. Shneiderman 1977). In einer Studie in Zusammenarbeit mit Mayer stellte er fest, dass das Entwicklerverständnis von konkreten einzelnen Codeschnipseln und der Erfahrung der Programmierer abhängt (vgl. Shneiderman/Mayer 1979: 222).

1983 identifizierte Brooks strukturbildende Codeschnipsel und benannte sie als „Beacons", die erfahrene Entwickler erkennen und daraus Ableitungen über den Aufbau des Codes anstellen

können[26] (vgl. Brooks 1983; s. a. Wiedenbeck 1986; Ko/Riche 2011; Robillard 2011). Gellenbeck/Cook fanden 1991 heraus, dass die deskriptive Namensgebung[27] eine wichtige Rolle für die Erkennung von Beacons spielt. Den positiven Einfluss einer deskriptiven Namensgebung hat Crosby 2002 besonders bei Entwicklern mit wenig Erfahrung nachgewiesen (s. a. Teasly 1994). In der neueren Forschung wurde gezielt der Einfluss von deskriptiver/nicht deskriptiver Namensgebung auf Lernsituationen untersucht. Dabei stellte sich heraus, dass APIs, die mit deskriptiver Namensgebung konzipiert und in der Lernsituation mit einer Dokumentation ergänzt wurden, das beste Verständnis erzielten. APIs mit nicht deskriptiver Namensgebung und ohne zusätzliche Dokumentation schnitten am schlechtesten ab (vgl. Blinman/Cockburn 2005: 4 ff.).

2.2.3.2. Gestaltung von API-Dokumentation

Friendly beschäftigt sich seit 1995 mit der Gestaltung von API- bzw. Entwicklerdokumentation. Neben studienspezifischen Zielen legte sie auch allgemeine Ziele für die Usability von API-Dokumentation fest:

> *"(...) we postulated some usability goals: readability, browsability, and ease of navigation. Our design had to support users well enough that their focus never had to shift from understanding the API to navigating its documentation."*
> (Friendly 1995: 2)

Friendly benennt die Lesbarkeit, die Browserkompatibilität[28] und eine einfache Navigation als generelle Ziele. Weiterhin betont sie die Bedeutung einer durchdachten Struktur und übersichtlichen Navigation in API-Dokumentation. Sobald der Leser sich mehr damit beschäftigen muss in der Dokumentation nach dem richtigen Inhalt zu suchen als die API zu verstehen, hat die Dokumentation ihren Zweck verfehlt.

Mit der Problematik der Struktur und Navigation von API-Dokumentation beschäftigten sich auch Jeong et al. In ihrer Studie suchten Entwickler eigenständig in API-Dokumentation nach Funktionen und Parametern, um einen Anwendungsfall zu lösen (vgl. Jeong et al. 2009: 89 f.). Das Vorwissen der Entwickler variierte sehr stark, die untersuchte Dokumentation war aber allen unbekannt. Tatsächlich fanden nur zwei der acht teilnehmenden Entwickler alle Services für die gesuchte Anfrage. Jeong et al. beobachteten vor allem Probleme in Bezug auf die Eindeutigkeit der Navigation (vgl. ebd. 94 ff.). Bereits auf der Startseite waren die Entwickler unsicher, welcher Menüpunkt der richtige ist. Unsicherheiten entstanden vornehmlich durch unbekannte Terminologie, fehlende Erklärungen oder nicht-deskriptive, teilweise kryptische Namensgebung. Die Namensgebung erwies sich ebenfalls als Problem in der Suchfunktion, da Begriffe, die in der Dokumentation genutzt wurden, in der Suche keine Ergebnisse brachten (vgl. ebd.). Auffällig war, dass die Entwickler mit spezifischem Vorwissen die gestellte Aufgabe tendenziell besser lösen konnten (vgl. ebd. 92).

[26] Vermutlich beruht die Erwartung an selbstdokumentierende Software-Dokumentation auf dem Erkennen von Beacons. Das Erkennen von Beacons ist vor allem bei unerfahrenen Programmierern nicht gewährleistet, bei erfahrenen Programmierern besteht wiederum die Gefahr der falschen Schlüsse (vgl. Teasly 1994: 766 ff.).

[27] Die deskriptive Namensgebung bezieht sich auf die Benennung der Funktionen, Parameter etc. durch den Entwickler. Als deskriptiv werden Benennungen angesehen, die die Fähigkeiten oder die Aufgabe bzw. den Inhalt aufgreifen. Im Gegensatz dazu stehen kryptische Namensgebungen, die aus einer zufälligen Aneinanderreihung von Zahlen und Buchstaben oder nicht in Relation stehenden Begriffen bestehen (s. a. Gellenbeck/Cook 1991; Crosby 2002; Teasly 2004; Blinman/Cockburn 2005).

[28] Die Studie von Friendly beschäftigt sich mit automatisiert generierter API-Dokumentation, die im WWW veröffentlich werden soll. Aus diesem Grund räumt sie der Browserkompatibilität einen hohen Stellenwert ein (vgl. Friendly 1995: 1 f.).

Ko/Riche untersuchten den Einfluss von konzeptuellem Vorwissen[29] in Bezug auf API-Dokumentation genauer (vgl. Ko/Riche 2011: 1). In ihrer Studie mussten Entwickler mit und ohne Vorwissen anhand von API-Dokumentation herausfinden, ob eine API bestimmte technische Anforderungen erfüllt. Sie stellten fest, dass Entwickler mit Vorwissen einen einfacheren und schnelleren Zugang zu der Dokumentation fanden, als solche ohne Vorwissen (vgl. ebd. 3). Das wurde vor allem an den Suchbegriffen und der Bewertung der Suchergebnisse deutlich. Entwickler ohne Vorwissen scheiterten bereits an der Terminologie, die für das Auffinden von relevanten Inhalten nötig gewesen wäre. Häufig waren die Entwickler ohne Vorwissen auch unsicher in Bezug auf die Brauchbarkeit der gefundenen Ergebnisse (vgl. ebd. 3 f.).

Für die Probleme in der Namensgebung sowie in Bezug auf das fehlende konzeptuelle Wissen haben Stylos et al. eine Lösung entwickelt. Sie schlagen vor „Placeholder" in API-Dokumentation zu integrieren, die die Anwendung der API erleichtern sollen (vgl. Stylos et al. 2009: 119).

> "*The idea behind our API ´placeholders´ is that the documentation should also list the classes and methods that programmers expect to exist, and these placeholders should contain forward references to the actual parts of the APIs that should be used instead.*"
> (Stylos et al. 2009: 121)

Placeholder sind Funktionen, Parameter etc., die als leerer Container mit den tatsächlich anzuwendenden Bezeichnungen indexiert werden. Stylos et al. haben diesen Ansatz in ihrer Studie überprüft und festgestellt, dass Entwickler durch die Bereitstellung von Placeholdern die gestellten Aufgaben dreimal schneller lösen konnten (vgl. Stylos et al. 125). Das liegt vor allem daran, dass sie durch die eingesetzten Suchbegriffe häufig sofort ein Ergebnis erhielten. Auf diese Weise konnte die umständliche Suche nach der korrekten Terminologie mithilfe von Placeholdern umgangen werden. Außerdem konnten die Entwickler bei Bedarf Placeholder hinzufügen, was ebenfalls sehr gut von ihnen angenommen wurde (vgl. ebd.).

Watson verfolgt eine andere Lösungsstrategie für das Problem der Namensgebung. Er schlägt vor, dass Technische Redakteure schon früher in den Entwicklungsprozess von Software-Projekten eingebunden werden sollen (vgl. Watson 2009: 1). Auf diese Weise könnten Inkonsistenzen im Code schon im Entwicklungsprozess aufgedeckt und im Development[30] beseitigt werden. Watson legte in seiner Studie eine Liste aller Parameter an und überprüfte diese manuell auf Inkonsistenzen in der Namensgebung (Watson 2009: 5 f.).

> "*To the developers, the inconsistent names might be physically or functionally distant in their view of the API, such as in the 10,000-line header file, the specification document, or the many different source code files in the source code library. In a sorted list of parameter names, however, the names line up one after another and the discrepancies speak for themselves.*"
> (Watson 2009: 5)

Für die Entwickler ist es im Entwicklungsprozess von Code schwierig, die Namensgebung aller Funktionen, Parameter etc. zu überblicken. Angesichts einer Vielzahl von Elementen und einer großen Menge an Codezeilen ist dieser Umstand nachzuvollziehen. Die Auflistung von Parameter-Bezeichnungen, die Watson durchgeführt hat, ist eine schnelle und aussagekräftige Methode für das Auffinden von Inkonsistenzen. Der Mehrwert dieser Vorgehensweise liegt nicht nur in der

[29] Der Begriff „konzeptuelles Vorwissen" umfasst bei Ko/Riche *„(…) developers' conceptual knowledge about a requirement* (…)" (Ko/Riche 2011:1). Mit „requirements" bzw. Anforderungen ist speziell technisches Wissen gemeint wie die Kenntnisse über eine Programmiersprache, ein Bildverarbeitungssystem etc.

[30] Development ist eine Bezeichnung für den Prozess des Programmierens von Software und anderen codebasierten Produkten.

Vereinfachung der Anfertigung einer Dokumentation, sondern auch in einer transparenten und konsistenten Namensgebung der API (vgl. Watson 2009: 5 f.).

Lethbridge et al. wählten einen allgemeineren Ansatz für ihre Studie. Sie befragten Entwickler zu guten, schlechten und untragbaren Eigenschaften von API-Dokumentation. Als gute Eigenschaften wurden High-Level-Informationen genannt, die z. T. unabhängig von kleinen Änderungen an der API aktuell bleiben oder als Hilfestellung genutzt werden können (vgl. Lethbridge et al. 2003: 36). Außerdem wurden Inline-Kommentare[31] als hilfreich für Pflege- oder Update-Arbeiten benannt. Auf der anderen Seite ist Dokumentation häufig nicht mehr aktuell, zu umfangreich, schlecht geschrieben, überladen mit nutzlosen Informationen oder schlicht zu aufwendig in der Erstellung. Die Entwickler empfinden darüber hinaus den Großteil der vorhandenen API-Dokumentation als untragbar, da sie „*untrustworthy*" (Lethbridge et al. 2003: 36) sei.

Einen sehr umfangreichen allgemeinen Ansatz verfolgten Nykaza et al. (2002), die durch qualitative Interviews mit Entwicklern notwendige Inhalte von Software-Dokumentation identifizierten. Nykaza et al. fragten gezielt nach den Wünschen der Entwickler. Die Antworten wurden zu einem Software Development Kit (SDK) zusammengefasst.

> „*Software Development Kits sind Pakete, die Programmcodes, Schnittstellen und in der Regel auch Anleitungen zur Verfügung stellen. Damit kann Software entwickelt werden, die auf einem bestimmten Betriebssystem, einer Hardware oder in Kombination mit bestimmter Software funktioniert.*"
> (Ziemer 2013)

Das Anleitungs-Paket von Nykaza et al. besteht aus: Support, Installation sowie Nutzung und Hilfe (vgl. Nykaza 2002: 8 f.). Zu dem Bereich Support gehören u. a. der Zweck des SDK, die Beschreibung der Komponenten, der Anwendungsbereich und Grenzen der Anwendbarkeit. Für die Installation/Nutzung werden How to´s, What if´s, Code-Beispiele, Bug-Lösungen, u. v. m. benötigt. Die Hilfe sollte aus FAQs, Kontaktdaten, einem technischen Support und Code-Bibliotheken bestehen. Zusätzlich zu den Inhalten bieten Nykaza et al. außerdem Vorschläge für die strukturelle Umsetzung an (vgl. ebd. 9).

Eine weitere Problematik von API- und generell Software-Dokumentation ist der Mythos, dass niemand Dokumentation liest. Mihaly widmete sich dieser grundlegenden Frage mit einem kritischen Ansatz:

> "*(...) documentation is referenced, not read. If reading the documentation cover-to-cover line-by-line is the only way to find information, developers will not find it, creating the false impression that they don't read the documentation.*"
> (Mihaly 2011)

Er identifiziert die Vorstellung des „Lesens" als Problem in Bezug auf API-Dokumentation. Entwickler lesen Dokumentation nicht, sondern scannen[32] sie auf relevante Informationen. Informationen, die in langen oder unübersichtlichen Textblöcken versteckt sind, können deshalb schwer gefunden werden.

[31] Die Bezeichnung „Inline" steht für einen Kommentar, der direkt in den Code geschrieben wird. Auf diese Weise können wichtige Informationen genau an der Stelle im Code zugänglich gemacht werden, für die sie relevant sind. Dabei kann es sich um Hinweise, Erklärungen o.ä. handeln.

[32] Der Begriff „Scannen" steht für eine an das Web angepasste Lesetechnik. Im Web dargestellter Text verliert durch den dynamischen Aufbau von Webseiten einen Teil seiner Linearität. Das bedeutet, dass die Leser den Text nicht von Anfang bis Ende bzw. von oben nach unten oder links nach rechts durchlesen. Vielmehr überfliegt der Leser zuerst den Text und andere Inhalte und bewertet anschließend die Relevanz des Gefundenen. Diese Prozesse werden als Skimmen und Scannen bezeichnet, wobei Scannen sich auf die Bewertung der Relevanz und Skimmen auf das Überfliegen der Inhalte bezieht. Skimmen und Scannen bilden meist die Voraussetzung für das u. U. darauffolgende vollständige Erschließen des Webinhalts (vgl. Antos et al. 2011).

Die Problematik liegt also nicht darin, dass die Entwickler keine Dokumentationen lesen, sondern darin, dass Dokumentationen nicht nutzerzentriert[33] gestaltet sind (vgl. Mihaly 2011). Die Gestaltung sollte auch den Zeitfaktor miteinbeziehen. Entwickler verbringen tendenziell maximal zehn Minuten mit dem Studium von Dokumentation, bevor sie sich wieder dem Code zuwenden (vgl. ebd.). Mihalys Annahmen, dass Entwickler sehr wohl Dokumentation lesen, wird durch eine Studie von Stack Overflow[34] gestützt (vgl. Stack Overflow 2017). Tatsächlich nutzen mehr als 80 % der Befragten die offiziellen Dokumentationen in ihrer täglichen Arbeit. Die Webseite Stack Overflow wird ebenfalls zu 80 % genutzt.

2.2.3.3. Lernprozesse in API-Dokumentation

Das Erlernen von neuem Code bzw. von Code-Architektur, -Logik und -Aufbau ist ein besonderer Prozess, der in der Forschung eine gezielte Zuwendung erfährt.

Ko et al. führten eine Studie zu den allgemeinen Informationsbedürfnissen von Entwicklern durch. Dafür wurden 49 Entwickler jeweils 90 Minuten lang von einem Beobachter begleitet, der ihr Handeln notierte (vgl. Ko et al. 2007: 345). Anhand dieser Untersuchung konnten 21 verschiedene Informationsbedürfnisse festgestellt werden (vgl. ebd. 346). Die folgende Abbildung zeigt alle gefundenen Informationsbedürfnisse sowie deren Suchfrequenz, die Bewertung des Informationstyps, die Häufigkeit und das Ergebnis der Suche sowie die Häufigkeit der angefragten Quellen.

	information type	search times min	mid	max	% agreed info is... import.	unavail.	inacc.	frequency and outcome of searches (acquired ▪ deferred ▫ gave up ⸜ beyond obs. -)	frequency of sources (br = bug report, dbug = debugger)
s1	Did I make any mistakes in my new code?	0	1	6	59	7	12		*dbug* 10 *compile* 26 *intuition* 6 *unit test* 4
a2	What have my coworkers been doing?	0	1	11	17	10	10		***coworker*** 20 *email* 13 *tool* 4 *bug alert* 4 *im* 2
u3	What code caused this program state?	0	2	21	90	49	32		*dbug* 16 ***br*** 3 *intuition* 3 *log* 3 *tools* 3 *code* 2 ***coworker*** 1
r2	In what situations does this failure occur?	0	2	49	80	32	20		***br*** 8 ***coworker*** 8 *inference* 5 *tools* 3 *dbug* 2 *comment* 1
d2	What is the program *supposed* to do?	0	1	21	93	29	29		*spec* 13 ***coworker*** 9 *docs* 5 *email* 1
a1	How have resources I depend on changed?	0	1	9	41	15	15		*tools* 12 ***coworker*** 6 *email* 4 ***br*** 2 *code* 1
u1	What code *could* have caused this behavior?	0	2	17	73	20	22		***coworker*** 5 *intuition* 4 *log* 4 ***br*** 4 *dbug* 2 *im* 1 *code* 1 *spec* 1
c2	How do I use this data structure or function?	0	1	14	71	20	29		*docs* 11 *code* 5 ***coworker*** 4 *spec* 1
d3	Why was this code implemented this way?	0	2	21	61	37	39		*code* 4 *intuition* 4 *history* 3 ***coworker*** 2 *dbug* 2 *tools* 2 *comment* 1 ***br*** 1
b3	Is this problem worth fixing?	0	2	6	44	10	20		***coworker*** 12 *email* 2 ***br*** 1 *intuition* 1
d4	What are the implications of this change?	0	2	9	85	44	49		***coworker*** 13 *log* 1
d1	What is the *purpose* of this code?	1	1	5	56	24	29		*intuition* 5 *code* 2 *dbug* 2 *tools* 2 *spec* 1 *docs* 1
u2	What's statically related to this code?	0	1	7	66	27	27		*tools* 8 *intuition* 2 *email* 1
b1	Is this a legitimate problem?	0	1	2	49	17	34		***br*** 5 ***coworker*** 1 *log* 1
s2	Did I follow my team's conventions?	0	7	25	41	10	15		*docs* 2 *tools* 2 *memory* 1
r1	What does the failure look like?	0	0	2	88	24	23		***br*** 3 *screenshot* 2
s3	Which changes are part of this submission?	0	2	3	61	7	5		*tools* 2 *memory* 2
c3	How I can coordinate this with this other code?	1	1	4	75	28	30		*docs* 2 *code* 1 ***coworker*** 1
b2	How difficult will this problem be to fix?	2	2	4	41	15	32		*code* 1 ***coworker*** 1 *screenshot* 1
c1	What can be used to implement this behavior?	2	2	2	61	27	22		*memory* 1 *docs* 1
a3	What information was relevant to my task?	1	1	1	59	15	13		*memory* 2

Figure 3. Types of information developers sought, with search times in minutes; perceptions of the information's importance, availability, and accuracy; frequencies and outcomes of searches; and sources, with the most common in boldface.

Abb. 1: Informationstypen nach Ko et al. (Ko et al. 2007: 350)

Der am häufigsten gesuchte Informationstyp war die Frage, ob Fehler im eigenen Code gemacht wurden. Direkt danach kam die Frage, was die anderen Entwickler machen. Auf den letzten Positionen befinden sich Fragen nach relevanten Informationen für die Umsetzung einer Aufgabe sowie nach der codebasierten Umsetzung derselben. Eine Auffälligkeit ist, dass die Bewertung der

[33] Userzentrierte Gestaltung bezeichnet die Gestaltung von Dokumentation anhand detaillierter Informationen über den Nutzer. Dazu gehören u. a. Zielgruppenanalysen und die Strukturierung der Inhalte anhand der Bedürfnisse des Users. Eine nicht userzentrierte Gestaltung bezieht den Nutzer nicht als Rezipienten mit ein, so dass z. B. die Gliederung nicht anhand von Anwendungsfällen aufgebaut wird (Lehrner-Mayer 2016). Nicht userzentrierte Dokumentationen sind für den Nutzer häufig umständlich und nicht nachvollziehbar.

[34] Stack Overflow ist eines der weltweit meistgenutzten Foren für Entwickler.

Entwickler der Informationstypen in einem Fall nicht den Beobachtungsergebnissen entspricht. Die Teilnehmer bewerteten die Frage „Was machen die anderen Entwickler“ als eher unwichtig. Diese Bewertung steht im Gegensatz zu den Beobachtungsergebnissen, in denen dieser Informationstyp am zweithäufigsten vorkommt (vgl. Ko et al. 2007: 350). Tatsächlich waren Kollegen auch die am häufigsten frequentierte Quelle für Nachfragen, was eine große Relevanz des Austauschs unter Entwicklern nahelegt (vgl. ebd. 351).

Die Frage nach dem Informationsbedürfnis von Entwicklern wird in Hinsicht auf API-Dokumentation noch von Parnin zugespitzt. Er untersuchte über 11 Wochen die Internethistorie von 17 Android-Entwicklern. Dabei stellte er fest, dass die offizielle Entwickler-Webseite von Android fast vier Mal seltener aufgerufen wurde als die Webseite Stack Overflow, bei der es sich um ein hochfrequentiertes Forum für Crowd Documentation[35] handelt (vgl. ebd.). Entwickler können hier Einträge und Fragen posten sowie auf die Fragen anderer Mitglieder antworten. Auf diese Weise entstehen ganze Dokumentationen für APIs und andere Software-Produkte.

> “*(...) a crowd of Android developers can cover as much as 88 % of the API classes in discussions on Stack Overflow.*”
> (Parnin 2013)

Tatsächlich können besonders bezogen auf Android fast 90 % der API-Referenzdokumentation[36] von Stack Overflow abgedeckt werden. Parnin fand auch heraus, dass auf Stack Overflow zwei Mal mehr Code-Beispiele als in der originalen Dokumentation zu finden waren. Ergänzend zu den Forschungsergebnissen von Ko et al. bezogen auf die Orientierung an anderen Entwicklern (vgl. Ko et al. 2007: 351) ist ein Randergebnis der Studie von Parnin besonders interessant: Entwickler können auf Stack Overflow bei anderen Entwicklern in eine Lehre gehen, um Code zu lernen (vgl. Parnin 2013). Diese Lehrmöglichkeit unterstreicht das Bedürfnis von Entwicklern in der Lernphase, sich mit anderen Entwicklern auszutauschen.

2.2.3.4. Spezielle Lernhindernisse bei APIs

Neben allgemeinen Studien zu der Erstellung von Software-Dokumentation und dem Lernverhalten von Entwicklern gibt es auch Studien, die sich speziell mit den Lernhindernissen in API-Dokumentation beschäftigen. Robillard und Robillard/DeLine fragten in ihren Studien gezielt nach Problemen bei dem Erlernen von neuen APIs.

In seiner Studie von 2009 stellte Robillard 83 Software-Entwicklern die allgemeine Frage „What makes APIs hard to learn“ (vgl. Robillard 2009: 27 ff.). Grundlage für die Untersuchung waren die Erfahrungen der teilnehmenden Entwickler. Robillard fand heraus, dass es fünf Kriterien gibt, anhand derer eine gute API-Dokumentation ausgemacht werden kann (vgl. ebd. 29). Die fünf Kriterien sind: gute Beispiele, Vollständigkeit, viele komplexe Anwendungsbeispiele und -szenarien, eine übersichtliche Struktur und relevante High-Level-Informationen (vgl. ebd. 29). Insgesamt wurde eine fehlende oder unzureichende API-Dokumentation am häufigsten als Problem genannt (vgl. ebd. 30). Andere Gründe waren die Architektur der API, die Kenntnisse des Anwenders, Diskrepanzen im technischen Umfeld oder prozessbezogene Probleme (Zeit, Unterbrechungen usw.).

[35] „Crowd Documentation“ beschreibt die Erarbeitung und Pflege von Dokumentation in einer frei zugänglichen Community. Entwickler können in einer Art Forum Einträge und Fragen posten sowie auf die Fragen anderer Mitglieder antworten. Auf diese Weise entstehen ganze Dokumentationen für APIs und andere Software-Produkte. (vgl. Parnin 2013)

[36] API-Referenzdokumentation ist die Bezeichnung für die Darstellung der Low-Level-Informationen der API wie Funktionen, Parameter etc. Meist wird sie als Tabelle oder verlinkte Liste umgesetzt (vgl. Parson 2013).

Robillard/DeLine führten 2011 eine weitere Studie zu Lernhindernissen bei APIs durch. Mit einer Kombination aus Umfragen und Interviews wurden insgesamt 440 Entwickler befragt. Als kritischer Punkt im Lernprozess konnte erneut API-Dokumentation ausgemacht werden (Robillard/DeLine 2011: 715). Robillard/DeLine entdeckten, dass fünf Aspekte in API-Dokumentation besonders häufig fehlen oder inadäquat ausgeführt werden. Sie werden als die fünf zentralen Lernhindernisse bezeichnet. Dazu gehören Intent Documentation, Code Examples, Matching APIs with Scenarios, Penetraibility und Documentation Format (vgl. Robillard/De Line 2011: 715 ff.). Übersetzt handelt es sich um die vorgesehene Nutzung der API, Code-Beispiele, Referenzierung von Code und Szenarien, High-Level-Informationen und die Makrostruktur der Dokumentation.

2.2.3.5. Qualität von bereits vorhandener API-Dokumentation

Eine Vielzahl von Studien beschäftigt sich mit positiven und negativen Kriterien zur Erstellung von API-Dokumentation. Ein Großteil dieser Studien wird als Expertenbefragungen durchgeführt, in denen die Entwickler subjektive Empfehlungen abgeben. Einen anderen Ansatz verfolgt die Untersuchung von bereits vorhandener API-Dokumentation. Die Forscher wenden dafür einen heuristischen qualitativen Ansatz an, der auf das Entdecken von Strukturen und nicht das Arbeiten innerhalb eines Paradigmas ausgerichtet ist.

Robillard/Maalej untersuchten in ihrer Studie API-Referenzdokumentation. Sie wollten herausfinden, welche Inhalte in der Dokumentation vorhanden sind und wie sie strukturiert sind. Dafür stellten sie verschiedene Knowledge Types bzw. Wissenstypen zusammen, anhand derer sie die Referenzdokumentationen untersuchen konnten.

TABLE 1
Taxonomy of Knowledge Types (Summary)

Knowledge Type	Description (Excerpt)
Functionality and Behavior	Describes what the API does (or does not do) in terms of functionality or features. Describes what happens when the API is used (a field value is set, or a method is called).
Concepts	Explains the meaning of terms used to name or describe an API element, or describes design or domain concepts used or implemented by the API.
Directives	Specifies what users are allowed / not allowed to do with the API element. Directives are clear contracts.
Purpose and Rationale	Explains the purpose of providing an element or the rationale of a certain design decision. Typically, this is information that answers a "why" question: Why is this element provided by the API? Why is this designed this way? Why would we want to use this?
Quality Attributes and Internal Aspects	Describes quality attributes of the API, also known as non-functional requirements, for example, the performance implications. Also applies to information about the API's internal implementation that is only indirectly related to its observable behavior.
Control-Flow	Describes how the API (or the framework) manages the flow of control, for example by stating what events cause a certain callback to be triggered, or by listing the order in which API methods will be automatically called by the framework itself.
Structure	Describes the internal organization of a compound element (e.g. important classes, fields, or methods), information about type hierarchies, or how elements are related to each other.
Patterns	Describes how to accomplish specific outcomes with the API, for example, how to implement a certain scenario, how the behavior of an element can be customized, etc.
Code **Examples**	Provides code examples of how to use and combine elements to implement certain functionality or design outcomes.
Environment	Describes aspects related to the environment in which the API is used, but not the API directly, e.g., compatibility issues, differences between versions, or licensing information.
References	Includes any pointer to external documents, either in the form of hyperlinks, tagged "see also" reference, or mentions of other documents (such as standards or manuals).
Non-information	A section of documentation containing any complete sentence or self-contained fragment of text that provides only uninformative boilerplate text.

Abb. 2: Knowledge Types nach Robillard/Maalej (Robillard/Maalej 2013: 5)

In einem aufwendigen Prüfverfahren fanden sie heraus, dass der Knowledge Type „Functionality" am weitesten verbreitet war. Auch Informationen zu „Structure" waren üblich (vgl. Robillard/Maalej 2013: 20). Die Types „Concepts" und „Purpose" hingegen waren eher selten zu finden. Überraschenderweise war der Type „Non-Information" bei zwei APIs sogar sehr häufig zu finden,

obwohl es sich dabei um weniger nützliche Informationen handelt. Robillard/Maalej empfehlen die Knowledge Types für die Erstellung von API-Referenzdokumentation zu verwenden. Die Types könnten dafür sogar als Template bzw. Vorlage genutzt werden (vgl. ebd.).

Watson et al. spannten den Rahmen für ihre Untersuchung noch etwas weiter. Sie verglichen empirisch erarbeitete Kriterien für die Erstellung von API-Dokumentation mit vorhandener Crowd Documentation. Das Ziel war es herauszufinden, inwiefern die Wünsche und Empfehlungen von Entwicklern aus Experteninterviews etc. in Dokumentationen zu finden sind, die von Open Source Communities entwickelt wurden (vgl. Watson et al. 2013: 165). Basierend auf einer Vielzahl von vorausgegangenen Studien erstellten Watson et al. einen umfassenden Kriterienkatalog. Er enthält u. a. die Umsetzung und Funktion von Code-Beispielen, die nutzerzentrierte Umsetzung der Dokumentation, Fehler- und Bug-Management sowie allgemeinere Kriterien wie beispielsweise eine korrekte, vollständige und akkurate Umsetzung der Dokumentation (vgl. ebd. 167 f.). Die Ergebnisse der Untersuchung entsprachen der Vermutung von Watson et al., dass die übernommenen Kriterien auch in der Praxis zu finden seien.

> *Except for the API overviews, our findings support our hypothesis in that the documentation elements listed as required or desired by software developers in API documentation were found in most of the API documentation we studied.*
> (Watson et al. 2013: 168)

Obwohl viele API-Dokumentationen aus Sicht der erfüllten Kriterien die Einstufung sehr gut oder hervorragend erhielten, stellten die Forscher eine Divergenz in der visuellen und sprachlichen Umsetzung fest. Um diese Unterschiede ebenfalls darstellen zu können, entwickelten Watson et al. ein zweites Bewertungsschema für die „Design and Writing Quality“ (vgl. Watson et al. 2013: 169). Insgesamt wiesen 82 % der untersuchten Dokumentationen eine hohe Textqualität auf, während eine hohe Designqualität bei 62 % zu finden war. Etwas mehr als die Hälfte der untersuchten Dokumentation wiesen gleichzeitig eine hohe Design- und Textqualität auf (vgl. ebd.). Watson et al. sehen darin einen zusätzlichen Beweis für die hohe Qualität von Crowd Documentation (vgl. ebd. 171).

2.2.3.6. Automatisierte Erstellung von API Dokumentation

Ein großer Streitpunkt in der Software-Dokumentation ist neben der Gestaltung die Frage nach dem lohnenden Aufwand. Tatsächlich existieren viele Ansätze für die automatisierte Dokumentation von Code, mit der sich auch wissenschaftliche Studien beschäftigen. Die automatisierte Dokumentation von APIs bzw. Code wird durch Dokumentationswerkzeuge wie Javadoc, PHPDoc oder Doxygen umgesetzt. Dokumentationswerkzeuge können anhand von verschiedenen Quellen eine umfangreiche Dokumentation erzeugen. In den meisten Fällen wird der Quellcode ausgelesen und zu Parameterlisten, Kommentarsammlungen usw. verarbeitet (s. a. Oracle o. J.[a]).

Friendly beschäftigte sich bereits Mitte der neunziger Jahre mit der automatisierten Dokumentation der Programmiersprache Java. Eines der Dokumentationswerkzeuge für Java ist Javadoc. Friendly hat in ihrer Studie die API der „HotJava alpha1“ Software dokumentiert. Sie wollte auf diese Weise die Programmiersprache Java und die Anwendung HotJava für ein breiteres Spektrum an Entwicklern zugänglich machen (vgl. Friendly 1995: 12). Javadoc liest den Quelltext von Java Code und erzeugt daraus ein HTML-Dokument. Obwohl die Dokumentation mit Javadoc automatisiert abläuft, kann der Anwender Tags für die weitere Beschreibung von Klassen, Parametern etc. hinzufügen. Auch die Konvertierung in andere Formate wie PDF, XML oder Windows Help Formate ist mithilfe von Doclets

möglich (vgl. Oracle o. J.[b]). Die Herausforderung bestand für Friendly darin, eine Dokumentation zu erstellen, die mehr Informationen als der Quelltext enthält.

> *"The document which resulted from the design process is much more than a subset of the Java source code. It contains essential information that can not* [sic!] *be readily obtained by reading the source code. It is on-line, hyperlinked, easily navigated and supports full text searching."*
> (Friendly 1995: 11)

Nach mehreren Testläufen und User-Tests konnte Friendly eine Dokumentation automatisiert erstellen, die laut dem Feedback der User einen Mehrwert im Gegensatz zum Quelltext darstellt (vgl. Friendly 1995: 12). Ihre Dokumentation wurde sowohl für HotJava alpha1 im Dezember 1994 als auch für HotJava alpha2 im März 1995 veröffentlicht.

Die Studie von Sridhara et al. beschäftigt sich ebenfalls mit Javadoc als Dokumentationswerkzeug. Sridhara et al. entwickelten eine neue Technik, um automatisiert deskriptive Kommentare für Parameter im Java-Quelltext zu generieren (vgl. Sridhara et al. 2011: 1). Ziel war es, neben den Low-Level-Informationen wie Bezeichnungen, Werten etc. auch High-Level-Informationen darzustellen. High-Level-informationen beziehen sich in diesem Fall auf die Rolle eines Parameters in der Gesamtheit der Funktionen, sprich Abhängigkeiten, Beziehungen etc. In diesem Zusammenhang müssen auch der Hauptzweck eines Parameters und seine weiteren Einsatzgebiete dargestellt werden (vgl. ebd. 2). Sridhara et al. entwickelten einen Ansatz für die Erstellung von Parameter-Kommentaren und implementierten ihn in Javadoc (vgl. ebd. 4). Dann ließen sie Javadoc eine automatisierte Dokumentation erstellen und legten das Ergebnis Entwicklern zur Bewertung vor.

> *"According to nine experienced developers, the generated comments are accurate and are useful in providing a quick perspective on the parameter's purpose in accomplishing the method's intent."*
> (Sridhara et al. 2011: 10)

Alle neun Entwickler bewerteten das Ergebnis als akkurat und hilfreich. Die Parameter-Kommentare schufen einen schnellen Überblick über den Zweck eines Parameters und seine Rolle in der Ausführung einer bestimmten Methode.

Buse/Weimer entschieden sich in ihrer Studie noch einen Schritt weiterzugehen als Sridhara et al. Sie entwickelten einen neuen Algorithmus für die automatisierte Erstellung von API Code-Beispielen. Dokumentationswerkzeuge wie Javadoc u. a. können zwar Code-Beispiele erstellen, Buse/Weimer beschreiben diese aber als unzureichend.

> *"In general, mined examples are long, complex, and difficult to understand and use. Good human written examples, on the other hand, often present only the information needed to understand the API and are free of superfluous context."*
> (Buse/Weimer 2012: 782)

Im Gegensatz zu automatisierten sind manuell erstellte Code-Beispiele kürzer, prägnanter und einfacher zu verstehen. Die manuelle Erstellung von Code-Beispielen hat aber auch Nachteile:

> *"Human written documentation has two important disadvantages, however: it requires a significant human effort to create, and is thus often not created; and it may not be representative of, or up-to-date with, actual use."*
> (Buse/Weimer 2012: 782)

Die manuelle Erstellung ist häufig sehr aufwendig, weswegen sie in vielen Fällen gar nicht erst durchgeführt wird. Außerdem sind manuelle Beispiele schwerer zu updaten. Buse/Weimer haben aus diesem Grund einen Algorithmus entwickelt, mit dem die Qualität von manuellen Beispielen auch in

einer automatisierten Erstellung von Dokumentation umgesetzt werden kann. Unter Anwendung des Algorithmus erstellten sie Code-Beispiele, die 150 IT-Studenten vorgelegt wurden (vgl. Buse/Weimer 2012: 790). Vergleichsgrößen bildeten manuell und automatisiert erstellte Code-Beispiele. Rund 82 % der Teilnehmer bewerteten die Code-Beispiele von Buse/Weimer als genauso gut wie die manuell erstellten, 94 % bewerteten sie sogar als besser (vgl. ebd. 791).

Die automatisierte Erstellung von API-Dokumentation und Software-Dokumentation eröffnet neue Möglichkeiten in Bezug auf Zeit- und Kosteneffizienz. Neben der rein automatisierten Erstellung ist auch die Anwendung von Vorlagen, Frameworks usw. eine mögliche Arbeitsersparnis.[37]

2.2.4. API-Dokumentation in dieser Arbeit

API-Dokumentation ist essentiell für das effiziente und schnelle Erlernen von APIs. Trotzdem in den letzten 40 Jahren eine Vielzahl an wissenschaftlichen Studien zu Software- und API-Dokumentation durchgeführt wurden, werden nach wie vor Diskussionen über den Wert und die Form von API-Dokumentation geführt. Entscheidend für die Akzeptanz von Dokumentation und deren Notwendigkeit ist letztendlich aber der damit einhergehende Mehrwert. Ein Mehrwert von Dokumentation kann jedoch nur erzielt werden, wenn die Qualität sehr hoch ist und sie gleichzeitig eine gute Usability aufweist. Diese Arbeit verfolgt den Ansatz, einen Beitrag zu der qualitativen Erstellung von API-Dokumentation zu leisten. Dafür wird eine Untersuchung durchgeführt, die auf einem Forschungsprojekt der Hochschule Merseburg basiert. Dieses Projekt und seine Ergebnisse werden nachfolgend detailliert vorgestellt.

3. Forschungsprojekt der Hochschule Merseburg

Die HS Merseburg bietet als vielseitiger Standort und mit langjähriger Tradition in der Ausbildung von Technischen Redakteuren optimale Voraussetzungen für die Forschung an API-Dokumentation (s. a. Frank 2016; KIW 2017[a]). Die hier angeführten empirischen Untersuchungsergebnisse wurden im Rahmen des Projekts „*Expertenorientierte Optimierung von Software-Entwicklerdokumentation*“ erhoben. Das Projekt wurde im Jahr 2014 initiiert und wird seitdem erfolgreich durchgeführt.

Zwar gibt es bereits eine Vielzahl von empirischen Untersuchungen zu API-Dokumentation aus den USA bzw. dem englischsprachigen Raum, Projekte und Studien aus Deutschland sind jedoch eher selten zu finden. Aus diesem Grund bildet das Projekt der HS Merseburg einen Versuch, diese Lücke zu schließen.

Nachfolgend wird das Projekt detailliert vorgestellt. Neben allgemeinen Informationen zu den Projektzielen werden auch bereits durchgeführte Studien sowie daraus erarbeitete Heuristiken[38] zur Erstellung von API-Dokumentation vorgestellt.

3.1. Ziele des Projekts

Das Projekt der Hochschule Merseburg verfolgt vor allem Ziele im Hinblick auf die Verbesserung der Usability und Effizienz von Entwickler-Dokumentation. Dusold stellt die Fragestellungen des Projekts im Rahmen ihrer Masterarbeit vor:

[37] Im Web gibt es zahlreiche Open-Source-Angebote für Vorlagen bzw. Templates zur Erstellung von API-Dokumentation (vgl. Techslides 2016). Die Untersuchung der angebotenen Templates in Bezug auf empirisch erarbeitete Kriterien zur Erstellung von API-Dokumentation könnte eine interessante Perspektive für zukünftige Forschungsprojekte sein.

[38] Das Wort „Heuristik“ stammt aus dem Griechischen und bedeutet entdecken oder finden (vgl. Michalkiewicz 2015). In der Psychologie bezeichnet eine Heuristik eine „*Allgemeine Strategie zur Lösung von Problemen.*“ (Woolfolk 2008: 365).

- „*Wie sollte die Dokumentation für die Zielgruppe der Software-Entwickler aussehen, damit der Lernprozess optimal unterstützt wird?*
- *Welche Inhalte sollten in einer effektiven Entwicklerdokumentation enthalten sein, und wie sollten diese Informationen dargestellt und strukturiert werden?*
- *Wie können die Anbieter der Dokumentation das Wissen optimal für die Nutzung durch Benutzer von APIs, die auch außerhalb des Unternehmens bereitgestellt werden, aufbereiten?*
- *Wie kann der Informationsaustausch zwischen Experten durch die Entwicklerdokumentation verbessert werden?*"
  (Dusold 2016: 19)

Das Hauptaugenmerk des Projekts richtet sich auf die optimierte Gestaltung von Software-Dokumentation bzw. Entwicklerdokumentation, was auch API-Dokumentation miteinschließt.[39] Entscheidend für eine optimierte Gestaltung ist es, den Lernprozess bestmöglich zu unterstützen. In diesem Zusammenhang soll nach notwendigen Inhalten gesucht (s. a. Nykaza et al. 2002; Robillard 2009; Robillard/DeLine 2011) sowie deren mögliche Darstellung und Strukturierung evaluiert werden. Zur Beantwortung der Forschungsfragen wurden bereits zwei Studien durchgeführt, die nachfolgend vorgestellt werden.

3.2. Studien

Die erste Studie fragte in qualitativen Experteninterviews[40] gezielt nach typischen Problemen und Fragestellungen bei der Einarbeitung in eine API. Die Ergebnisse wurden zusätzlich in einer Beobachtungsstudie überprüft. Die zweite Studie beschäftigte sich ebenfalls in Expertenbefragungen mit Best- und Worst-Practices in API-Dokumentationen.

3.2.1. Software-Entwicklerdokumentation: Was wollen Entwickler wirklich?

In der Studie „*Software-Entwicklerdokumentation: Was wollen Entwickler wirklich?*" (Steinhardt et al. 2015, s. a. Meng et al. 2016; Meng et al. 2017) wurde ein breites Themenspektrum untersucht. Ziel war es herauszufinden, wie Entwickler methodisch an die Einarbeitung in eine neue API herangehen. Dazu wurde nach typischen Problemen, Fragestellungen zu Beginn sowie aufkommenden Fragen im Verlauf der Einarbeitung und benötigten Informationen gefragt. Außerdem sollten die Teilnehmer von guten und schlechten Erfahrungen sowie ihren persönlichen Erwartungen an eine API-Dokumentation berichten (vgl. Steinhardt et al. 2015). Durch qualitative Experteninterviews und quantitative Fragebogenuntersuchungen konnte eine Reihe an Forschungserkenntnissen gewonnen werden.

Auffällig an den Ergebnissen der beiden Untersuchungen war, dass sich zwei verschiedene Ansätze bei der Einarbeitung in APIs herauskristallisierten. Während die eine Gruppe sich vor allem auf konzeptuelle Informationen stützte, bevorzugte die andere den direkten Einstieg in den Code (vgl. Meng et al. 2016; Meng et al. 2017: 14 f.). Eine konzeptorientierte Vorgehensweise wird allgemein durch die Bevorzugung von High-Level-Informationen gekennzeichnet. Entwickler mit diesem Lernansatz versuchen den Aufbau und die Arbeitsweise der API möglichst ganzheitlich zu verstehen (vgl. Steinhardt et al. 2015). Ein codeorientierter Lernansatz hingegen kennzeichnet sich durch ein

[39] Entwickler-Dokumentation bezeichnet die Dokumentation von Code speziell für Entwickler und ist dementsprechend ein Teil von Software-Dokumentation. Das Gleiche gilt für API-Dokumentation. Weitere Ausführungen dazu sind unter Definition und Zweck von API-Dokumentation in dem Kapitel API-Dokumentation zu finden.

[40] Qualitative Experteninterviews sind Befragungen von Personen mit Fachwissen auf einem für die Befragung relevanten Gebiet (Borchardt/Göthlich 2009: 38). Das Ziel dieser Befragungen ist es, eine subjektive Einschätzungen zu erheben.

aktives exploratives Verhalten. Solche Entwickler bevorzugen Optionen zum Ausprobieren (z. B. Code Generatoren) und benötigen hauptsächlich Low-Level-Informationen.

Die verschiedenen Lernansätze wurden auch bei der Frage nach dem weiteren Vorgehen in der Einarbeitung deutlich. Die meisten Teilnehmer arbeiteten im weiteren Verlauf mit Code-Beispielen von ausgewählten Funktionen oder lasen die Getting-Started Dokumentation (vgl. Meng et al. 2016). Während das Arbeiten mit Code-Beispielen ein sehr praktischer Ansatz ist, enthält die Getting-Started-Dokumentation vor allem grundlegende Informationen zu der ersten Nutzung der API. Dazu gehören auch konzeptuelle Erklärungen in Form von High-Level-Informationen. Diese beiden häufigsten Arbeitsweisen spiegeln gleichzeitig auch die unterschiedlichen Lernansätze wieder (vgl. Meng et al. 2017: 21).

Die Teilnehmer wurden in der Studie ebenfalls zu Problemen beim Erlernen einer API sowie nach typischen Problemen von API-Dokumentation befragt. Als häufigste Probleme wurden falsche, unverständliche und unvollständige Dokumentationen genannt (vgl. Meng et al. 2016). Im Detail waren vor allem eine unklare Struktur und Navigation der Dokumentationen problematisch (vgl. Meng et al. 2016). Diese Ergebnisse decken sich mit denen der empirischen Untersuchungen von Robillard (2009) und Robillard/DeLine (2011).

Meng et al. heben darüber hinaus besonders die Problematik des Einstiegspunkts in die Dokumentation hervor.

> "*They have a problem they need to solve, they know the API will potentially solve their problem, but they don't know where and how to begin.*"
> (Meng et al. 2017: 22)

Obwohl die Entwickler bereits ihren Anwendungsfall und die API als brauchbar identifiziert haben, besteht ein Problem mit dem Einstieg. Sobald der Einstiegspunkt gefunden wurde, ist die Einarbeitung vergleichsweise einfach (vgl. Meng et al. 2017: 22 ff.). Obwohl häufig konzeptuelle Dokumente wie ein Überblick o. ä. für die Erleichterung eines Einstiegs geliefert werden, werden sie ebenso häufig übergangen oder nicht beachtet (vgl. ebd.).

3.2.1.1. Beobachtungsstudie

Im Anschluss an die qualitativen und quantitativen Befragungen wurden die Ergebnisse zusätzlich in einer Beobachtungsstudie überprüft. In der Beobachtungsstudie wurden Entwickler bei der Bearbeitung einer Aufgabe bezogen auf eine API gefilmt.

Die Beobachtungsstudie zeigte anhand der Aufnahmen eines Eye-Trackers[41], dass die Teilnehmer Texte nicht lesen, sondern scannen (s. a. Mihaly 2011). Besondere Aufmerksamkeit galt Überschriften, Code-Blöcken und Verlinkungen (vgl. Meng et al. 2016). Textblöcke wurden dementsprechend nicht sofort gelesen, sondern erst auf Eignung gescannt oder komplett übersprungen. Aus diesem Grund ist es entscheidend, wichtige Informationen für die Anwendung des Codes so zu platzieren, dass sie vom Leser wahrgenommen werden können.

Ein weiterer Faktor, der das Scannen maßgeblich unterstützt, ist eine klare, konsistente und transparente Navigation (vgl. Meng et al. 2016). Im Fall einer nicht nachvollziehbaren oder

[41] Eye-Tracking ist eine Messmethode für die Augenbewegungen eines Probanden. Durch die technisch gestützte Aufzeichnung der Augenbewegungen kann festgestellt werden, wo und für welche Zeitspanne der Proband zu einem bestimmten Zeitpunkt hingeschaut hat (Bergstrom, Schall 2014: 3). Außerdem kann die gesamte Augenbewegung als Pfad dargestellt werden.

unlogischen Struktur wird dem Leser das Auffinden des gewünschten Inhalts erschwert. Aus diesem Grund sollten die Kapitel bzw. Unterseiten anhand des Inhalts und nicht aufgrund des formalen Typs des Dokuments[42] benannt werden.

Die Suchfunktion wurde vor allem verwendet, wenn der Teilnehmer den gewünschten Inhalt nicht eigenständig finden konnte (vgl. Meng et al. 2016). Aus diesem Grund ist eine intelligente und komplexe Suchfunktion sehr wichtig. Bei einfachen Suchfunktionen gibt es u. a. keine Möglichkeit der Singular- und Pluralerkennung, was die Ergebnisliste für ein Schlagwort bereits erheblich einschränkt. Darüber hinaus kann nicht gewährleistet werden, dass der Leser explizit nach dem Begriff sucht, der in der API-Dokumentation verwendet wird (s. a. Stylos et al. 2009). Diese Problematik könnte durch Synonymerkennung oder verwandte Suchvorschläge umgangen werden.

Durch die Beobachtungsstudie konnte weiterhin gezeigt werden, dass Code-Beispiele bei den ersten Schritten der Einarbeitung eine besonders große Bedeutung haben (vgl. Meng et al. 2016). Bei allen Teilnehmern begann die Einarbeitung basierend auf einem Code-Beispiel, von dem ausgehend individuelle Lösungsstrategien entwickelt wurden. Analog dazu steigt die Relevanz der Low-Level-Informationen in einer Referenzdokumentation mit zunehmendem Wissen über die API (vgl. ebd.). Dieses Vorgehen sollte bei der Gestaltung eines Getting-Started-Dokuments o. ä. unbedingt beachtet werden.

Im Hinblick auf die Lernansätze stellte sich heraus, dass explizites Vorwissen bezogen auf die Branche der API[43] oder die Umgebung keinen Einfluss auf die Bevorzugung eines Informationstyps hatte (konzeptorientiert vs. codeorientiert). Das Vorwissen spielte aber in Bezug auf die Lösung von Problemen eine große Rolle, da strukturelles Vorwissen das Auffinden von Lösungen erleichterte (vgl. Meng et al. 2016).

3.2.2. API-Dokumentation aus der Perspektive von Software-Entwicklern: Was macht sie besonders hilfreich?

Mit der Durchführung der Studie „*API-Dokumentation aus der Perspektive von Software-Entwicklern: Was macht sie besonders hilfreich?*“ (Dusold 2016) wurden weitere Erkenntnisse zu der Gestaltung von API-Dokumentation gewonnen. Das geschah durch Experteninterviews, in denen die Teilnehmer eigenständig Best-Practice und Worst-Practice-Beispiele vorstellten (vgl. Dusold 2016: 22 ff.). Durch dieses Vorgehen konnte eine Reihe von Aspekten identifiziert werden, die für die Gestaltung von API-Dokumentation wichtig sind.

Generell hatten die Teilnehmer hohe Ansprüche, die sich in einer Erwartungshaltung zeigten (vgl. Dusold 2016: 33). Sie erwarteten eine vollständige, selbsterklärende, durchsuchbare, logisch gegliederte und mit Production Code[44] versehene Dokumentation (vgl. ebd. 34). Eine logische Gliederung bestand für die Teilnehmer darin, dass bestimmte Informationsblöcke vorhanden waren. Dazu gehörten allgemeine Konzepte[45], eine Übersicht der Funktionen und Dienste, die Beschreibung

[42] Es gibt keine offizielle Bezeichnung für unterschiedliche Formate von Dokumentation, weshalb diese Bezeichnungen zu Verwirrung führen können. Typische Bezeichnungen sind z. B. Cookbook, Recipe, Getting-Started, Sand Box, Developers Guide usw.

[43] Die API der Beobachtungsstudie stammt aus der Branche des E-Commerce, so dass Entwickler mit Vorwissen bereits Kenntnisse über typische Funktionen in Bezug auf explizite Anwendungsfälle hatten. Ein Anwendungsfall konnte z. B. das Versenden eines Pakets sein.

[44] Production Code ist ein umfangreiches, komplexes und vollständiges Code-Beispiel für die Anfrage an eine API und umfasst auch deren Antwort (vgl. Robillard/DeLine 2011: 717).

[45] Die allgemeinen Konzepte sollen Informationen dazu enthalten, wie die Schnittstelle als Ganzes funktioniert (vgl. Dusold 2016: 35). Es geht dementsprechend nicht um einzelne Parameter, sondern um High-Level-Informationen wie Zusammenhänge der Funktionen und die Arbeitsweise und Architektur der API.

der Parameter, die Typisierung[46], das Laufzeitverhalten[47], Fehlermeldungen und Code-Beispiele. Entscheidend für die Darstellung der Low-Level-Informationen ist es dabei, dass nicht nur die Aufgabe und die Fähigkeit eines Parameters beschrieben werden, sondern auch die Regeln für Werte[48] und andere Vorgaben[49]. Insgesamt sollen die Inhaltsblöcke klar strukturiert sowie kurz und prägnant formuliert sein (vgl. ebd. 35). Der Gesamtaufbau der API-Dokumentation soll insbesondere die Auffindbarkeit und Verständlichkeit von einzelnen Anwendungsfällen ermöglichen (vgl. Dusold 2016: 33).

Besonders ausschlaggebend ist laut den Teilnehmer der Aufbau der Dokumentation, der sich auch an Zielgruppen orientieren sollte (vgl. Dusold 2016: 52). Die Auslagerung und Verteilung von Informationen ermöglicht es, unterschiedliche Zielgruppen mit variierendem Vorwissen anzusprechen. So wird Referenzdokumentation als wichtig für Entwickler mit Vorwissen bezeichnet, die eine schnelle, codebasierte Lösung suchen. High-Level-Informationen wiederum wären vor allem für eine Zielgruppe ohne Vorwissen interessant.

Als häufig problematisch wurde die Aktualität der Dokumentation, der Verlinkungen und der Code-Beispiele benannt (vgl. Dusold 2016: 39 ff.). Auch eine inkonsistente Struktur, Gestaltung und Namensgebung machen Dokumentation unübersichtlich (vgl. ebd. 42 ff.). In diesem Rahmen erwähnten zwei Teilnehmer den Anbieter Swagger als positives Beispiel, der ein Open Source Framework[50] für die Gestaltung von API-Dokumentation offeriert (vgl. Swagger 2016). Swagger bietet die Möglichkeit, jede API unabhängig von der Programmiersprache etc. einheitlich zu dokumentieren. Die Einheitlichkeit besteht nicht nur in der Gestaltung der Dokumentation, sondern auch in der Gliederung des Inhalts und des gesamten Aufbaus (vgl. Dusold 2016: 42 ff.). Die Teilnehmer sehen einen besonderen Vorteil in der Konsistenz eines solchen Dokumentationssystems, da Inhalte leichter auffindbar sind und ihnen generell der Aufbau und die Gestaltung gefällt (vgl. ebd. 42). Die Auffindbarkeit wird durch die einfache, übersichtliche und konsistente Gestaltung von Swagger Dokumentation erreicht. So gibt es eine klare Farbgebung für den Anlass einer Funktion, z. B. Löschen in Rot, Ändern in Orange usw. (vgl. Swagger o. J.). Entwickler, die den Aufbau von Swagger Dokumentationen bereits kennen, finden sich deshalb sofort zurecht.

Die Teilnehmer benannten ebenfalls besonders hilfreiche Eigenschaften von Dokumentation. Dazu gehörten eine Suchfunktion, Interaktivität, die Auszeichnung besonderer Inhalte, Code als Dokumentation und Crowd Documentation (vgl. Dusold 2016: 38 ff.). Interaktivität wurde auf zwei Weisen definiert: zum einen handelt es sich um das Generieren von Code innerhalb der Dokumentation, zum anderen um interaktive Steuerungs- und Designelemente wie ausklappbare Inhalte, Verlinkungen und Buttons (vgl. Dusold 2016: 38 f.). Bezogen auf Code als Dokumentation wurde ein schlüssiger und konsistenter Code von den Teilnehmern als genauso wichtig wie eine gute

[46] Der Begriff „Typisierung“ bezieht sich auf die Namensgebung der API. Er bezeichnet ein stark individualisiertes Interface, das auf übliche Bezeichnungen oder Strukturen verzichtet. Die Teilnehmer wünschten sich im Fall von stark typisierten Interfaces, dass sich die Namensgebung in der API wiederspiegelt (vgl. Dusold 2016: 36).

[47] Das Laufzeitverhalten ist die Dauer, die ein durch einen Rechner ausgeführtes Programm zu der Bewältigung einer Aufgabe benötigt. Häufig stehen Funktionen und Parameter im Zusammenhang mit dem Laufzeitverhalten der API bzw. Software. Manche Funktionen können das Laufzeitverhalten durch eine große Beanspruchung von Ressourcen wie Datenbanken o. ä. verlängern.

[48] Die Werte von Parametern können an verschiedene Regeln gebunden sein. Es kann z. B. für einen Parameter ein String (eine Zeichenkette) gefordert sein, die nur eine bestimmte Anzahl an Zeichen enthalten darf. Ebenso können Zahlen oder bestimmte Zahlenformate gefordert werden u. v. m.

[49] Andere Vorgaben sind optional/nicht optional, darf null sein/darf nicht null sein usw. (vgl. Dusold 016: 35).

[50] Ein Framework ist ein Programmiergerüst für Entwickler. Das Gerüst besteht aus fertigen Code-Bausteinen, die sich für die Wiederverwendung eignen. Beispiele dafür sind der Zugriff auf eine Schnittstelle oder im Fall von Web-Frameworks das Design einer Webseite (vgl. Hery-Mossmann 2016b).

Dokumentation eingeschätzt. Tatsächlich benötigen einige keine Dokumentation, wenn sie über ein entsprechendes Vorwissen verfügen und auf einen qualitativ hochwertigen Code zugreifen können (vgl. Dusold 2016: 52).[51] Die Dokumentation durch den Code an sich ist nach Erfahrungen der Teilnehmer weniger fehleranfällig, als andere Dokumentation. Durch den Compiler[52] entfalle außerdem eine manuelle Überprüfung. Ein Compiler zeigt Fehler im Code direkt an, so dass der bereitgestellte Code unkompliziert überprüft und ggf. korrigiert werden kann (vgl. Dusold 2016: 53).

Die Ergebnisse der Studie lieferten auch eindeutige Informationen dazu, was schlechte Dokumentation ist. Grundsätzlich handelt es sich um schlechte Dokumentation, wenn zu wenig Zeit eingeplant wird (vgl. Dusold 2016: 40). Häufig wird aber auch gar keine Dokumentation zur Verfügung gestellt oder die vorhandene ist veraltet. Schlecht sind auch unnötige Informationen wie z. B. zum Thema Caching[53], zu der Implementierung der API, den aufgerufenen Dienste und Technologien sowie der Duplizierung oder übertriebene Erklärung von Code (vgl. Dusold 2016: 37). Die Teilnehmer empfanden darüber hinaus inhaltlich das Fehlen von wichtigen Informationen zu Fehlermeldungen, der Befüllung von Parametern oder auch fehlende Erklärungen für Funktionen als schlecht. Auch das Fehlen von Grundlagen wurde als problematisch empfunden. Entwickler ohne Vorwissen müssten in diesem Fall das notwendige Wissen eigenständig zusammentragen, was sehr zeitaufwendig wäre.

Die vorgestellten Studienergebnisse werden nachfolgend als Heuristiken für die Erstellung von API-Dokumentation zusammengefasst und spezifiziert.

3.3. Heuristiken zur Erstellung von API-Dokumentation

Von den empirischen Erhebungen des Forschungsprojekts der HS Merseburg wurden Heuristiken zur Optimierung von API-Dokumentation abgeleitet. Sie wurden als Maximen für die Erstellung von API-Dokumentation erhoben und werden im Folgenden unter Heranziehung von Praxisbeispielen erläutert.

Die Heuristiken wurden in vier Themenblöcke mit unterschiedlichen Ausprägungen aufgeteilt, die nacheinander vorgestellt werden. Bei den Themenblöcken handelt es sich um die „Unterstützung des Lernansatzes“, die „Gestaltung, Struktur und Navigation“, den „schnellen Einstieg“ und die „Grundregeln guter Dokumentation“. Die Heuristiken werden nachfolgend einzeln erklärt und mit Umsetzungsbeispielen veranschaulicht.

3.3.1. Unterstützung des Lernansatzes: konzeptorientiert vs. codeorientiert

Der Lernansatz von Software-Entwicklern bei der Sichtung von Dokumentation kann sehr unterschiedlich sein (vgl. Steinhardt et al. 2015). Einige Entwickler bevorzugen eine konzeptorientierte Vorgehensweise, andere eine codeorientierte.[54]

[51] Entwickler schätzen ihren Code häufig als selbstdokumentierend ein, was in den meisten Fällen nicht zutrifft. Probleme in der Namensgebung, undurchsichtige Designentscheidungen und eine nicht schlüssige Architektur erschweren das Verständnis für den Nutzer (vgl. Mihaly 2011). Die Erstellung von selbstdokumentierendem Code ist zwar noch nicht der aktuelle Zustand, kann aber durch das Beachten von Designprinzipien für APIs durchaus erreicht werden. Weitere Ansätze dazu liefern Stylos et al. (Stylos et al. 2009: 119).

[52] Ein Compiler ist ein Programm, das aus dem Quelltext eines Entwicklers maschinenlesbaren Code erstellt (vgl. Aho et al. 2008: 3). Programmiersprachen sind rechnerunabhängig, aber in dieser Form nicht maschinenlesbar. Damit ein Rechner den geschriebenen Quellcode zu einem Programm verarbeiten kann, muss ein Compiler ihn zuerst übersetzen (vgl. ebd.).

[53] Caching bezeichnet den Vorgang, dynamisch generierter Webseiten mittels zwischengespeicherten Inhalten die Ladezeiten zu verkürzen. Ursprünglich mussten die Inhalte dieser Webseiten bei jedem Aufruf über einen Server aus ggf. unterschiedlichen Datenbanken zusammengetragen werden. Durch das Zwischenspeichern der Inhalte auf einem Server können Ladezeiten verkürzt und so die Performance gesteigert werden (vgl. Cremer 2016).

[54] Siehe Kapitel 3.2 Studien.

Der duale Aufbau einer Dokumentation ermöglicht das bestmögliche Lernumfeld für beide Lernansätze. Umgesetzt und unterstützt werden kann der Aufbau durch eine intelligente Suchfunktion, den selektiven Zugriff auf konzeptuelle Informationen und die codenahe Platzierung von wichtigen Informationen.

3.3.1.1. Intelligente Suchfunktion

Die Suchfunktion ist eine sehr wichtige Eigenschaft von Dokumentation. Sie hilft dem Nutzer dabei, schnell und möglichst einfach den gewünschten Inhalt zu finden. In der Realität gestalten sich Suchen häufig problematisch oder sehr aufwendig, da besondere Funktionen und Eigenschaften aufwendig programmiert werden müssen.[55] In vielen Fällen werden deshalb fertige Suchfunktionen für Webseiten genutzt, die im Baukasten eines Content-Management-Systems[56] mitgeliefert werden. Diese Suchen sind meist unflexible Standardversionen, die beispielsweise keine Singular- und Pluraleingabe ermöglichen, Synonyme nicht erkennen usw. (vgl. Mutschler 2015).

Entwickler-Dokumentation ist häufig in Form von Online-Medien verfügbar, so dass die Problematik der Suchfunktion für sie besonders relevant ist. Tatsächlich kann eine nicht zielführende Suche zur Frustration des Nutzers führen (s. a. Jacobsen 2005; Steinhardt et al. 2015).

Eine intelligente Suchfunktion kann über besondere Features wie eine Fehlertoleranz und die Erkennung von Singular, Plural und/oder Synonymen verfügen (vgl. 1&1 2017). Hilfreich ist ebenfalls eine Suchhistorie oder das Angebot von verwandten Suchmöglichkeiten[57]. Im Bereich der Autovervollständigung werden durch die Suche Vorschläge anhand des bereits eingetippten Texts gemacht.

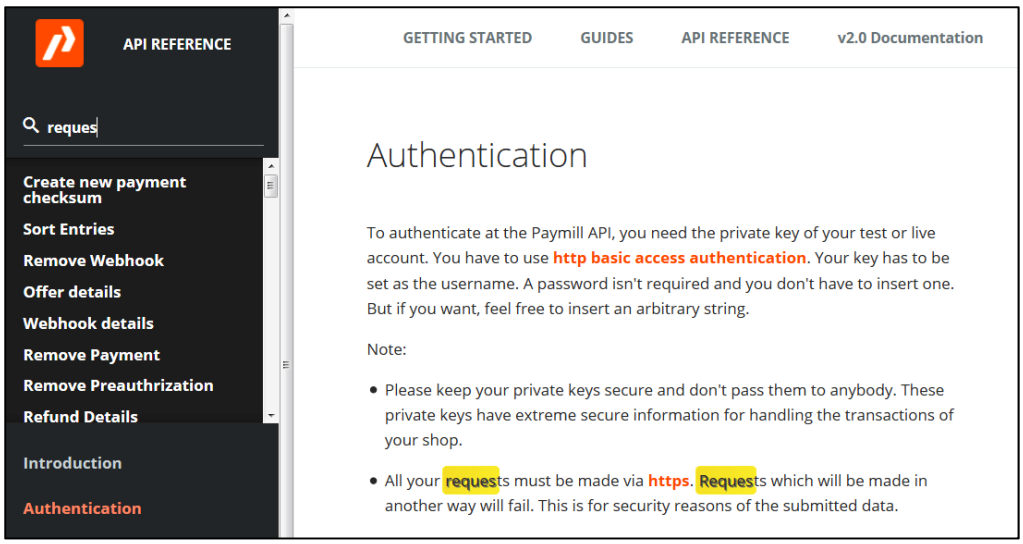


Abb. 3: Intelligente Suche mit Autovervollständigung (Paymill o. J.)

[55] Es gibt verschiedene Möglichkeiten, Suchfunktionen zu programmieren. Häufig werden MySQL-Datenbanken in Verbindung mit Skripten der Webprogrammiersprache PHP genutzt. MySQL ist ein relationales Open Source Datenbanksystem, dass eine Vielzahl von Funktionen für Abfragen und gezielte Datenlieferungen etc. bereitstellt (vgl. MySQL 2017). Relationale Datenbanksysteme enthalten viele tendenziell klein skalierte Datenbanken, in denen eine eigenen Tabellenarchitektur und spezifisch definierte Beziehungen zwischen Feldern festgelegt werden können (vgl. ebd.).

[56] Ein Content-Management-System (CMS) dient der einfachen Verwaltung von Inhalten im Web oder offline. Häufig werden CMS für Webseiten eingesetzt, so dass Inhalte und Medien zentral gepflegt und angepasst werden können (vgl. CMS made simple 2017). Die größten Vorteile eines CMS sind, dass Inhalte in einer Quelle zusammengeführt und ohne Entwicklerfachkenntnisse bearbeitet werden können.

[57] Einen ähnlichen Ansatz verfolgten Stylos et al. mit der Einführung von Placeholdern. Typische Funktionen und Parameter wurden als Verlinkungen zu den tatsächlich anzuwendenden Funktionen etc. genutzt. Irritationen durch die Namensgebung konnten so umgangen werden (vgl. Stylos et al. 2009: 121).

Für den Suchbegriff „reques“ werden nicht nur die Stichwortergebnisse angezeigt, sondern auch damit in Relation stehende Themen, die ebenfalls diesen Suchbegriff enthalten. Die Suchergebnisse werden parallel in Echtzeit gelb im Text hervorgehoben. Neben der Stichwortmarkierung können die Suchergebnisse auch in Listenform präsentiert werden.

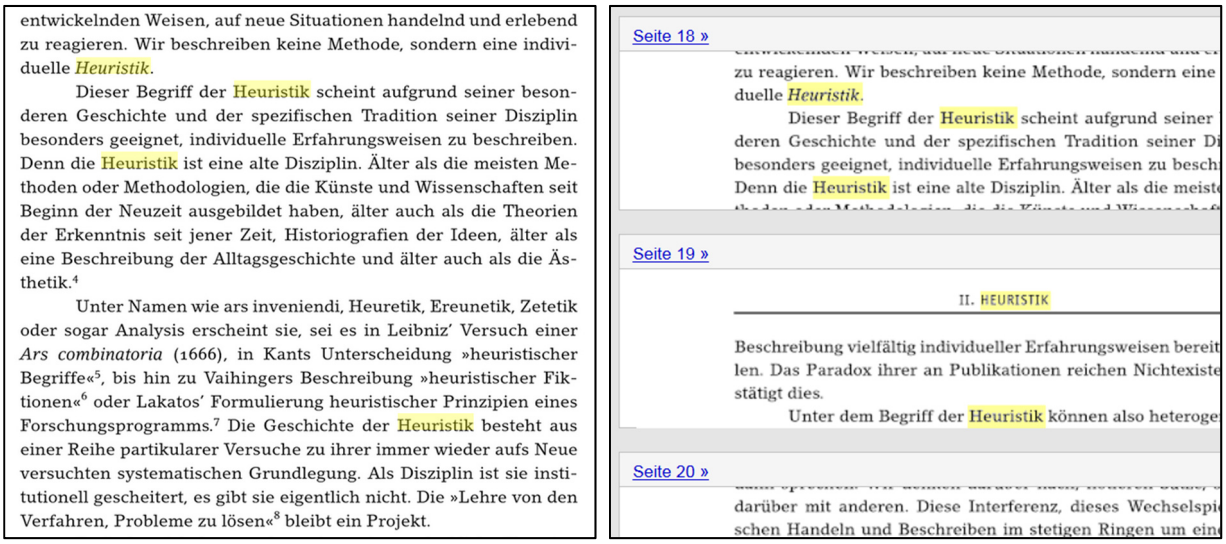


Abb. 4: Anzeige der Suchergebnisse als Liste und als Stichworte (v. l. n. r., Google Books o. J.)

Die Abbildung zeigt verschiedene Anzeigemöglichkeiten für einen Suchbegriff in einem Buch auf Google Books (vgl. Google Books o. J.). In der linken Darstellung der Suchergebnisse in Listenform werden die Ergebnisse optisch durch einen grauen Balken voneinander getrennt. Zusätzlich zu der Liste werden die Stichwörter in den angezeigten Textpassagen gelb hervorgehoben. Durch den blau unterlegten Link kann der Nutzer direkt auf die gewünschte Seite springen. Im Gegensatz dazu ist auf der rechten Seite eine Stichwortsuche im Volltext zu sehen. Über die blau unterlegten Schaltflächen „Weiter“ und „Zurück“ kann der Nutzer zwischen den Stichwortergebnissen chronologisch hin- und herspringen.

3.3.1.2. Selektiver Zugriff auf Informationen

Der selektive Zugriff auf Informationen kann durch unterschiedliche Maßnahmen umgesetzt werden. Empfohlen wird die Interaktivität von Text- oder Codeblöcken mittels Buttons, Pfeilen oder Links. In der originalen Ansicht der Dokumentation wären in diesem Fall nur Informationen zu einer Funktion und deren Parametern zu sehen. Wünscht sich der Nutzer mehr Informationen, kann er per Klick weiteren Text einblenden oder per Link auf eine referenzierte Seite bzw. zu einer referenzierten Textstelle wechseln. Analog dazu könnte für den konzeptuellen Ansatz ein Textblock zu sehen sein, bei dem durch einen Klick Code-Beispiele eingeblendet werden.

Der selektive Zugriff kann auch auf allgemeinere Inhalte übertragen werden. Nicht jeder Nutzer benötigt beispielsweise allgemeine Informationen oder solche zur Inbetriebnahme (Installation, Keys, Authentifizierung etc.). Diese Informationstypen könnten ebenfalls selektiv eingeblendet oder in einem Fenster geöffnet werden.

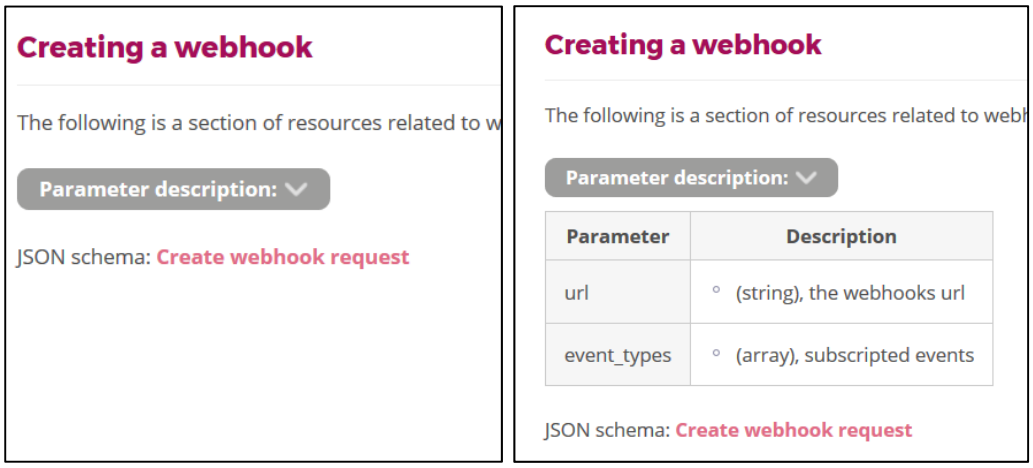


Abb. 5: Selektives Einblenden von Informationen durch eine Panelfunktion (Shipcloud o. J. [a])

Im Rahmen des Forschungsprojekts an der HS Merseburg wurde eine eigene API Dokumentation entworfen, die sowohl als Testobjekt als auch als Positivbeispiel dient (vgl. Shipcloud o. J. [a]). Diese Dokumentation in Form einer Webseite ist in der obenstehenden Abbildung zu sehen. Die Abbildung zeigt außerdem die mögliche Umsetzung des selektiven Zugriffs auf Informationen über ein per Klick zu öffnendes Panel. Das geöffnete Panel in der rechten Abbildung enthält Spezifikationen zu den Parametern, die in der ausgewählten Funktion verwendet werden müssen. Zusätzlich ist ein externer Link unterhalb des Panels verfügbar, der auf ein Schema in JSON weiterleitet (Abb. 6, Shipcloud o. J. [b]). Mithilfe des Schemas kann der Nutzer unkompliziert anhand einer Vorlage eine komplette Anfrage erarbeiten.

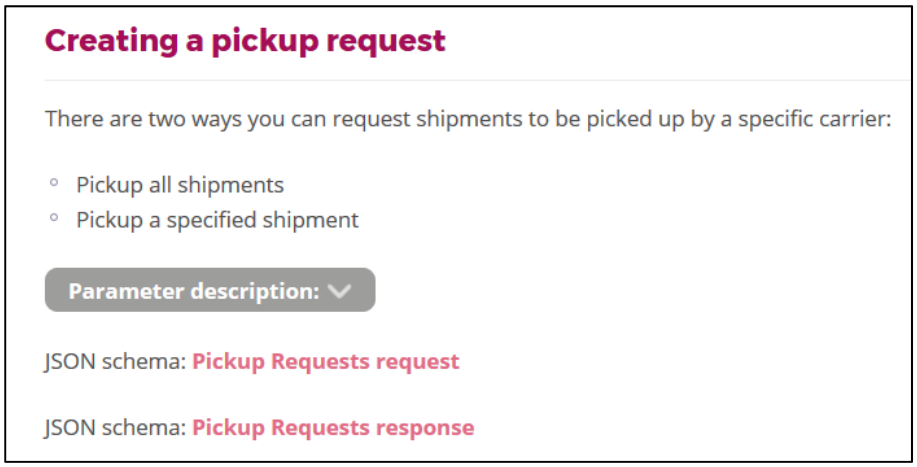


Abb. 6: Selektiver Zugriff auf ein JSON Schema durch eine externe Verlinkung (Shipcloud o. J. [b])

Eine Problematik der Trennung von Text und Code kann die Auffindbarkeit sein. Ungünstig platzierte Verlinkungen, unzureichendes Design von Panels oder fehlende Rückwege können sowohl die Navigation als auch das Auffinden erheblich erschweren. Grundsätzlich vermeiden lässt sich diese Problematik durch eine besondere Art der Referenzierung von Text und Code in einem zweispaltigen Layout. In solch einem Layout können in einer Spalte die konzeptuellen und in der anderen die codebasierten Informationen angezeigt werden. Diese Gestaltung ermöglicht das schnelle Auffinden von referenzierten Informationen, ohne dass ein Lernansatz benachteiligt wird.

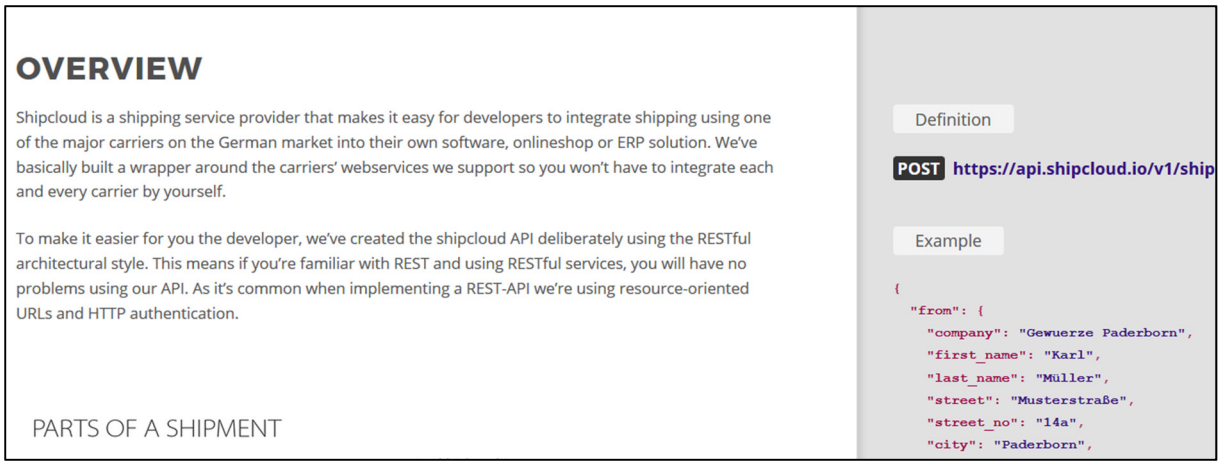


Abb. 7: Zweispaltiges Layout (Shipcloud o. J. [a])

In der Abbildung sind die konzeptuellen und codebasierten Informationen der beiden Spalten referenziert (vgl. Shipcloud o. J. [a]). Das bedeutet, dass der Nutzer durch den Text der Dokumentation scrollen kann, während die Spalte mit den Code-Beispielen automatisch mitgescrollt wird.[58] Auf diese Weise wird immer das passende Code-Beispiel zu dem aktiven Kapitel angezeigt. Zusätzlich kann auf technischer Ebene zur besseren Orientierung die aktuelle Überschrift der Textspalte mit einem farbigen Balken hervorgehoben werden (vgl. Twilio 2016). Durch die Hervorhebung wird der Blick des Nutzers direkt auf die richtige Stelle gelenkt.

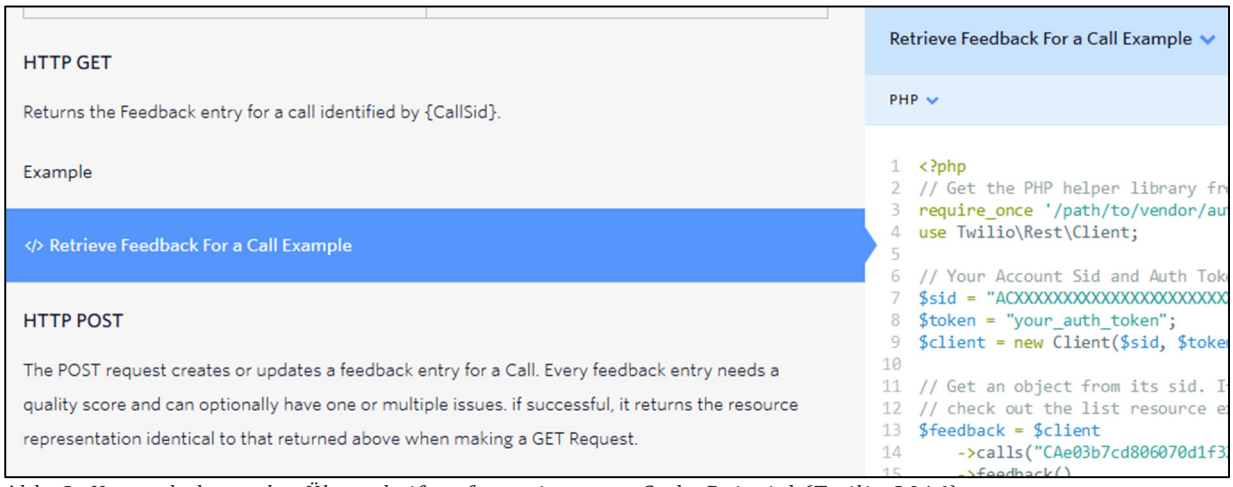


Abb. 8: Hervorhebung der Überschrift referenziert zum Code-Beispiel (Twilio 2016)

Ein weiterer Ansatz für den selektiven Zugriff auf konzeptuelle Informationen ist das Anbieten von zwei separaten Navigationssträngen. Beispielsweise könnte es die Stränge „Problemlösung" und „Gesamtverständnis" geben. Unter „Problemlösung" wären codebasierte Informationen vorhanden, die gezielt für einen bestimmten Anwendungsfall geeignet sind. Unter dem Navigationsstrang „Gesamtverständnis" wiederum wären vorrangig High-Level-Informationen zu der API und deren Nutzung zu finden. Die Aufteilung der API-Dokumentation auf zwei Navigationsstränge könnte durch Verlinkungen in den anderen Navigationsstrang oder Panelfunktionen optimal unterstützt werden.

Die Gestaltung von Dokumentation anhand von zwei Navigationsstränge geht über die Trennung der Lernansätze hinaus. Sie bezieht sich zusätzlich auf das Ziel des Nutzers, dessen Lösungsstrategie

[58] Diese Gestaltung kann durch simples Webdesign mithilfe von CSS umgesetzt werden.

jedoch durch seinen Lernansatz bedingt sein kann. Ein konzeptorientierter Lerner wird z. B. eher auf konzeptuelle Informationen zugreifen. Benötigt er aber beispielsweise nur Informationen zu einem Parameter, werden die konzeptuellen Informationen eher eine untergeordnete Rolle spielen. Im Endeffekt handelt es sich bei dem Lernansatz um die grundsätzliche Tendenz einer Person, nicht um ein fixes Verhalten.

3.3.1.3. Codenahe Platzierung von wichtigen Informationen

Ergänzend zu einer intelligenten Suche und dem selektiven Zugriff auf konzeptuelle Informationen ist die codenahe Platzierung von wichtigen Informationen sinnvoll. Die Grundlage für diese Überlegung bildet ein stark codeorientierter Lernansatz, bei dem Entwickler ausschließlich Code-Zeilen lesen. Diese Gruppe zeigt eine Antipathie gegenüber Fließtexten, so dass konzeptuelle Informationen in Form von Textblöcken gar nicht erst wahrgenommen werden (s. a. Meng et al. 2016; Meng et al. 2017).

Entwickler mit einem stark codeorientierten Lernansatz laufen Gefahr, entscheidende konzeptuelle Informationen schlichtweg zu übersehen.[59] Aus diesem Grund müssen konzeptuelle Informationen so platziert werden, dass sie auch mit einem codebasierten Lernansatz gelesen werden. Um gleichzeitig auch den konzeptorientierten Lernansatz zu unterstützen, empfiehlt es sich, wichtige Informationen redundant in Text und Code zu platzieren.

Eine codenahe Platzierung kann z. B. durch eine Funktion ähnlich IntelliSense[60] geleistet werden. IntelliSense liefert Informationen zu möglichen Membern[61] und Parametern, bietet eine Quickinfo und kann die Eingabe vervollständigen. Beispielsweise könnte der Nutzer in einem Code-Beispiel oder einer Auflistung in der Referenzdokumentation auf eine Funktionsbezeichnung klicken oder fahren, um eine Liste der Parameter zu öffnen.

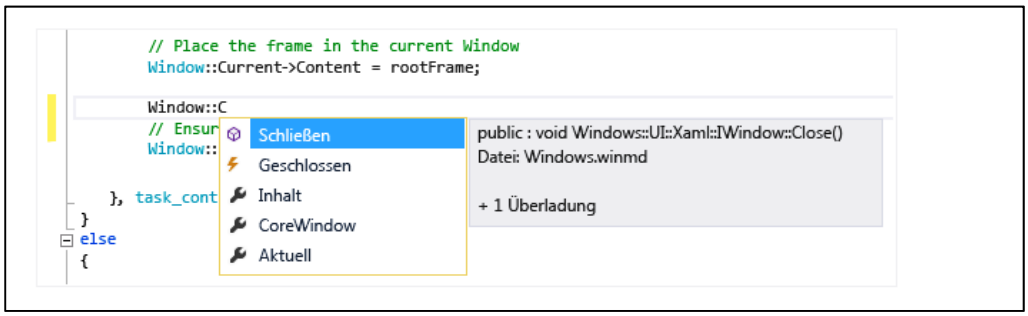


Abb. 9: Member Liste von IntelliSense im Visual Studio (Microsoft 2017a)

Bei Klick auf einen Parameter wird die Parameterinfo angezeigt, die Regeln für den Wert des aktuellen und den jeweils nachfolgenden Parameter liefert.

[59] Die Bedeutung von konzeptuellen Informationen für das Erlernen einer API wurde in mehreren Studien nachgewiesen (s. a. Meng et al. 2016; Meng et al. 2017; Ko/Riche 2011; Jeong et al. 2009).

[60] IntelliSense ist eine Anwendung in der Entwicklungsumgebung „Visual Studio“ von Microsoft (vgl. Microsoft 2017a).

[61] Die Bezeichnung „Member“ bezieht sich auf alle erlaubten Funktionen usw., die nachfolgend auf den eingetippten Code verwendet werden dürfen. Bezogen auf APIs könnte es sich dabei um eine Liste möglicher Parameter für eine Funktion handeln.

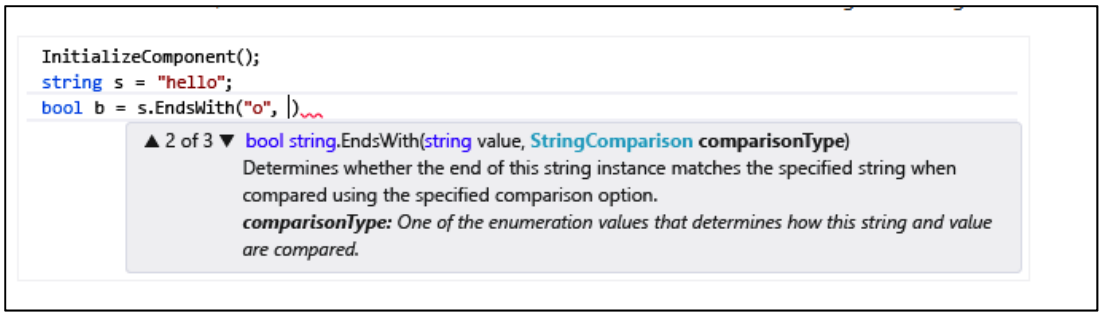


Abb. 10: Parameterinfo von IntelliSense im Visual Studio (Microsoft 2017[a])

Eine ähnliche Funktion ist die Quickinfo für Werte, die allerdings nur die Spezifikation anzeigt. Sie wird eingeblendet, wenn der Wert für einen Parameter eingetippt wird.

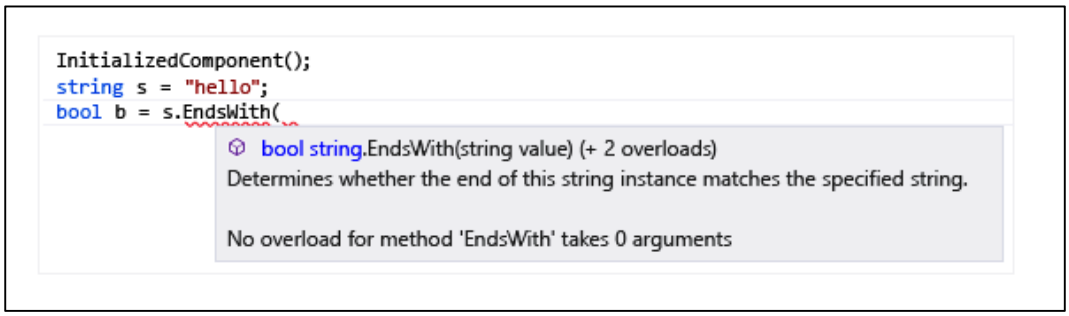


Abb. 11: Quickinfo von IntelliSense im Visual Studio (Microsoft 2017[a])

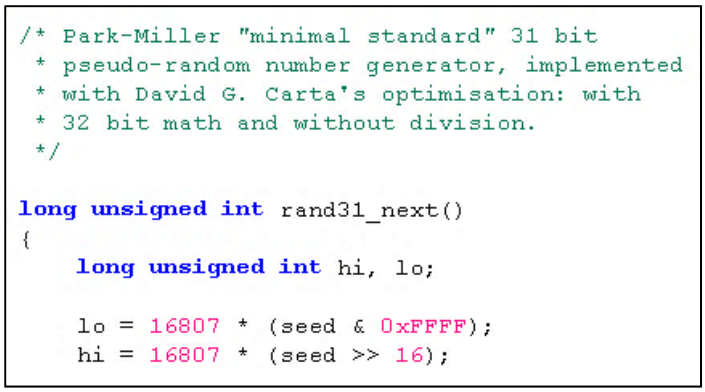
Gesammelt vereinfachen die Funktionen von IntelliSense nicht nur das Programmieren, sondern erleichtern auch den Zugriff auf referenzierte Informationen enorm. Durch einen Klick sind ergänzende Informationen direkt an der Stelle im Code sichtbar, wo sie gebraucht werden.

Die Umsetzung dieser Funktionen in API-Dokumentation müsste an das jeweilige Format angepasst werden. Eine naheliegende Form der Umsetzung für Webseiten wären Panels oder Hover-Boxen[62]. Die Entwickler müssten dementsprechend den Code nicht verlassen, um auf konzeptuelle Informationen zuzugreifen. Eine solche Funktion könnte die Usability von API-Dokumentation um ein Vielfaches steigern.

Ein verwandter Ansatz ist die farbige/ typografische Hervorhebung von Inline-Kommentaren. Der Kommentar wird mit Tags oder anderen Markierungszeichen versehen, die ihn eindeutig für die Entwicklungsumgebung und den Leser als Kommentar kennzeichnen.[63] In Entwicklungsumgebungen werden voreingestellte Farben verwendet, um Elemente, Werte etc. optisch voneinander abzugrenzen. Anhand des Tags werden Kommentare dementsprechend automatisch in einer anderen Farbe markiert als der Code.

[62] Die Hover-Funktion bezeichnet in CSS das Erscheinen einer Textbox, wenn der Nutzer mit der Maus über einen bestimmten Bereich oder ein bestimmtes Wort auf einer Website fährt (vgl. w3schools 2017c).

[63] Die Kennzeichnung eines Kommentars hängt von der Programmier- oder Auszeichnungssprache ab. In der Auszeichnungssprache HTML werden Kommentare mithilfe von Tags, sprich spitzen Klammern, und einer spezifischen Zeichenfolge kenntlich gemacht: *<! - - Kommentar - - >* (vgl. SelfHTML 2017[b]). In der Programmiersprache Java dagegen wird ein Kommentar so gekennzeichnet*: // Einzeiliger Kommentar* bzw. */* Mehrzeiliger Kommentar */* (vgl. TEIA 2017).



```
/* Park-Miller "minimal standard" 31 bit
 * pseudo-random number generator, implemented
 * with David G. Carta's optimisation: with
 * 32 bit math and without division.
 */

long unsigned int rand31_next()
{
    long unsigned int hi, lo;

    lo = 16807 * (seed & 0xFFFF);
    hi = 16807 * (seed >> 16);
```

Abb. 12: Grüne Farbe für Inline-Kommentar in der Programmiersprache C (Carta 2013)

Der Inline-Kommentar ist durch die grüne Farbe und die Auszeichnung /* Text */ gekennzeichnet (Carta 2013). Inline-Kommentare können dementsprechend direkt in den geschriebenen Code eingefügt werden und sind durch die Farbigkeit trotzdem leicht auffindbar. Auf diese Weise können High-Level-Informationen direkt an der Stelle im Code platziert werden, für die die Informationen benötigt werden. Eine redundante Platzierung wird dadurch gewährleistet, dass die Information ebenfalls in Textform zu finden ist. Beispielsweise in einem zweispaltigen Layout können wichtige Hinweise sowohl im Text als auch in der Codespalte per Inline-Kommentar platziert werden.

Mit Inline-Kommentaren könnte eine Antipathie gegen Textblöcke umgangen werden, indem Informationen sehr kurz und prägnant „eingeschoben“ werden. Eine zu intensive Nutzung der Kommentare könnte deren Mehrwert jedoch untergraben, da der Code dadurch zerstückelt wirken würde (s.a. Dusold 2016: 47 f.).

3.3.2. Gestaltung, Struktur und Navigation

API-Dokumentation muss über eine nachvollziehbare und übersichtliche Struktur verfügen. Das schnelle Zurechtfinden sollte zusätzlich durch eine transparente Navigation und eine konsistente Gestaltung gestützt werden.

3.3.2.1. Struktur

Die Strukturierung von Dokumentation bildet eine zentrale Frage im Gestaltungsprozess. Die bloße Aneinanderreihung von unabhängig nebeneinanderstehenden Kapiteln sollte in jedem Fall vermieden werden. Die Dokumentation sollte stattdessen prozessbasiert gestaltet werden, so dass die Anwendung der API und nicht die Systematik der Dokumentation im Vordergrund steht. Häufig wird die Systematik der Dokumentation priorisiert, indem Kapitel nach Textsorten gestaltet werden. Bei Textsorten handelt es sich um typische Formate wie Referenzdokumentation, Developers Guide, Cookbook, Sand Box usw.

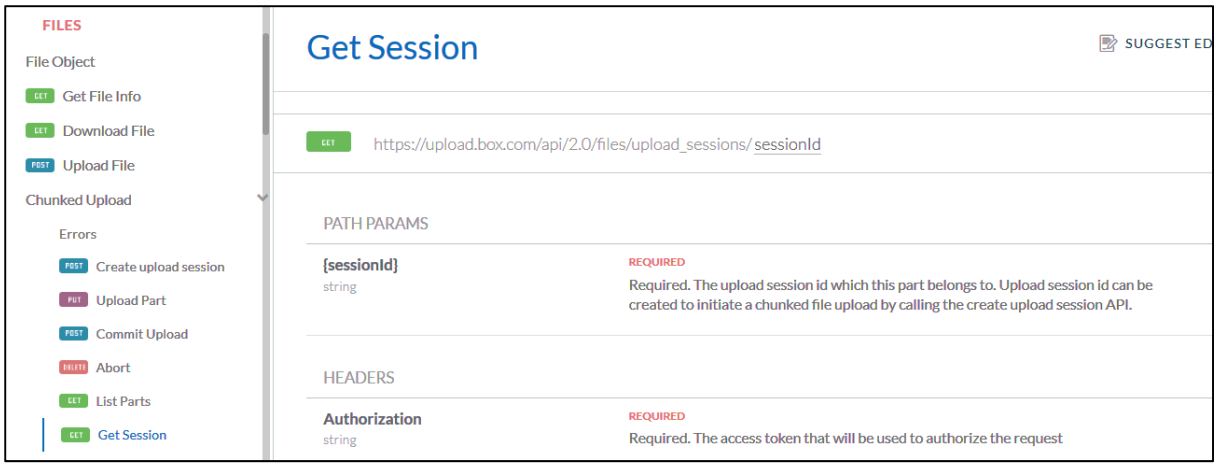


Abb. 13: Prozessbasierte Strukturierung (boxDevelopers o. J.)

Die Abbildung zeigt die Gliederung einer API-Dokumentation nach Prozessen anstatt von Textsorten. Durch diese Struktur sind alle mit einem Anwendungsfall verknüpften Informationen schnell zu finden. Aufbauend auf der Struktur ist die konsistente Feinstruktur der Kapitel sehr wichtig. Grundsätzlich sollten Inhalte gleichen oder ähnlichen Typs auch analog oder zumindest ähnlich dargestellt werden. Die Darstellung bezieht sich auf alle Aspekte der visuellen und inhaltlichen Gestaltung, sprich Typografie, Layout, Aufbau, Überschriften, Farbigkeit, Verweise, Tabellen etc.

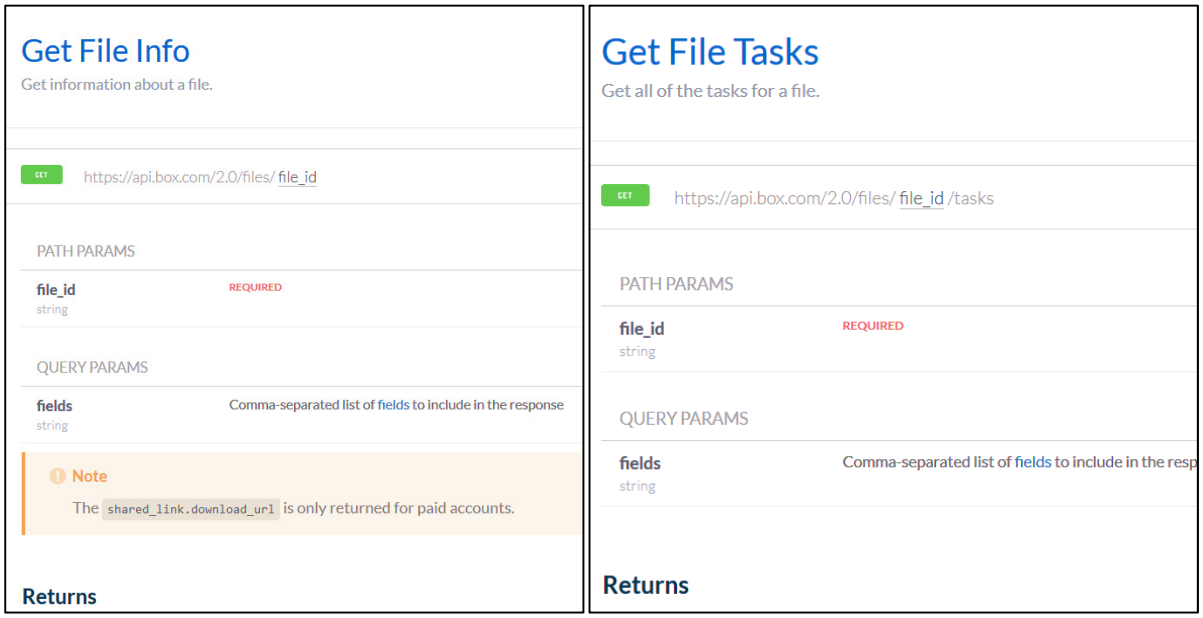


Abb. 14: Konsistente Feinstruktur der Kapitel (boxDevelopers o. J.)

3.3.2.2.Transparente Navigation

Eine transparente Navigation muss nicht nur inhaltlich nachvollziehbar sein und aussagekräftige Überschriften besitzen, sondern auch aus technischer Sicht sauber konstruiert sein. In vielen Fällen verfügen Webseiten von API-Dokumentation über ein unzureichendes Webdesign, so dass Navigationsleisten beim Scrollen verschwinden und/oder eine Webseite unübersichtlich in Unterseiten verschachtelt ist.

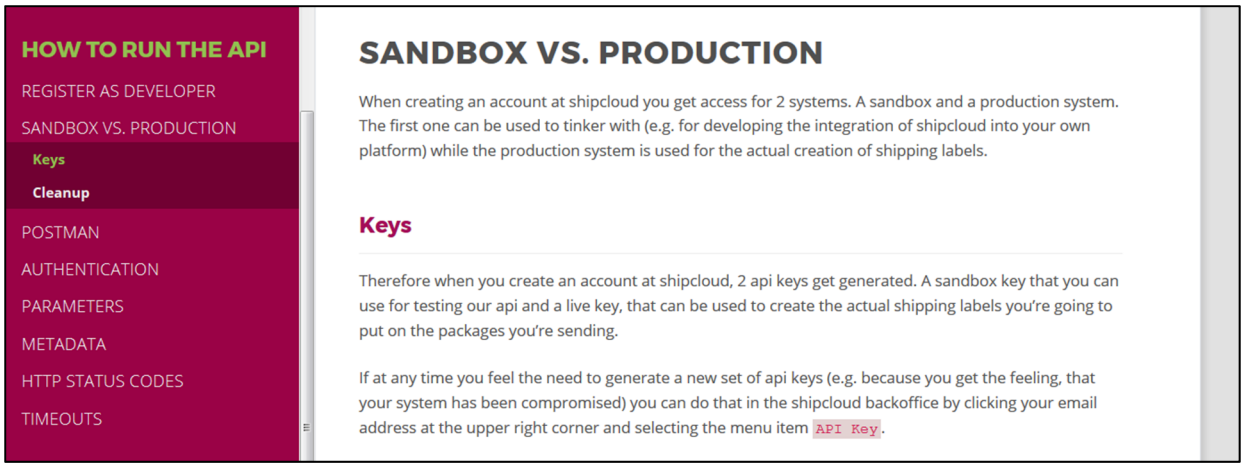


Abb. 15: Transparente Navigation (Shipcloud o. J. [a])

Die Navigation in der Abbildung ist fest auf der linken Seite platziert, so dass eine konsistente Optik in einem dreispaltigen Layout entsteht (vgl. Shipcloud o. J. [a]). Zusätzlich kann sie über einen separaten Balken vertikal gescrollt werden. Eine weitere hilfreiche Funktion für Navigationsleisten ist das automatische Öffnen und Schließen von Unterkapiteln parallel zum Scrollen (vgl. Paymill o. J.). Das bedeutet, dass in der Navigation parallel zum Scrollen die aktuelle Überschrift farbig markiert wird und ggf. Unterkapitel dynamisch geöffnet werden.

3.3.2.3.Konsistente Gestaltung

Die konsistente Gestaltung bezieht sich auf ein einheitliches visuelles und inhaltliches Bild. Einheitlich bedeutet in diesem Zusammenhang, dass die Dokumentation insgesamt, sowohl optisch als auch inhaltlich, stimmig ist. Bei den optischen Aspekten handelt es sich um das allgemeine Design, die Farbigkeit, die Typografie und das Layout. Die inhaltlichen Aspekte beziehen sich vor allem auf das inhaltliche aufeinander Aufbauen der einzelnen Teile der Dokumentation. Das kann durch Verweise, Verlinkungen, Hinweise und Bezüge im Text umgesetzt werden. Generell beschreibt es außerdem die mögliche Gesamtheit einer Dokumentation, d. h. inwiefern die einzelnen Teile zu einem Ganzen werden oder ob sie unabhängig nebeneinander stehen.

Besonders wichtig ist die konsistente Trennung von Code-Beispielen und Text bzw. Code und Codekommentar. Teilweise sind diese Bereiche nicht klar abgesetzt, so dass sie beim Scannen des Dokuments durch den Nutzer leicht übersehen werden können. Eine klare optische Abgrenzung durch Absätze, Farbigkeit oder Typografie kann hier Abhilfe schaffen.

3.3.3. Schneller Einstieg

Viele Nutzer brauchen einen schnellen Einstieg in eine API, damit sie gezielt ein bestimmtes Anwendungsszenario realisieren können. Für diese Nutzer ist es wichtig, mithilfe der Dokumentation einen zeitnahen Produktivstand zu erreichen. Im Gegensatz zu der Unterscheidung der Lernansätze handelt es sich hierbei nicht um die Beschränkung auf einen bestimmten Typen von Informationen, sondern um die kompakte Aufbereitung von allen notwendigen Informationen.[64]

Im besten Fall sollte als Einstiegsszenario ein kompakter API-Überblick vorhanden sein, der den Zweck der API, die wichtigsten Fähigkeiten, die Lizensierung/das Preismodell usw. beinhaltet. Auf

[64] Sprich es werden Low-Level und High-Level-Informationen gleichermaßen dargestellt.

diese Weise kann der Nutzer gleich feststellen, ob die API für seinen Zweck geeignet ist und welche Bedingungen[65] er erfüllen muss, um sie einsetzen zu können.

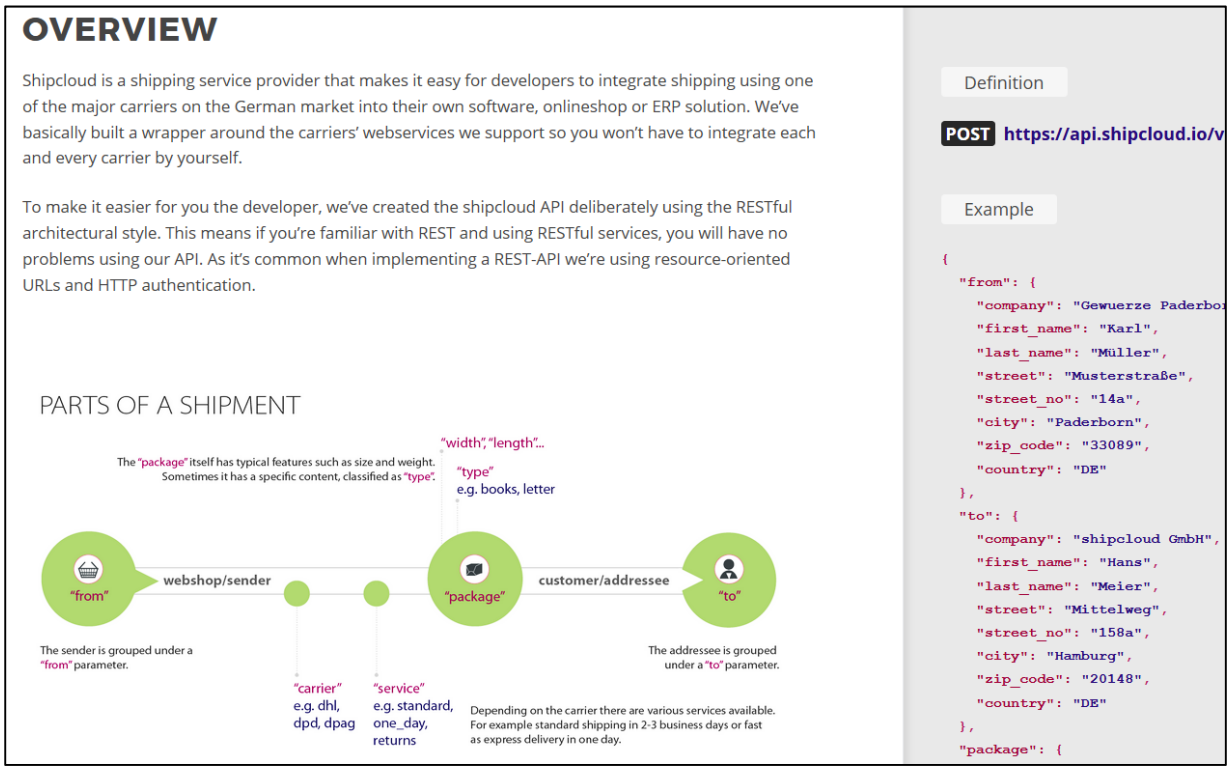


Abb. 16: Überblick der Shipcloud Dokumentation (Shipcloud o. J. [a])

Der Überblick zeigt dem Nutzer sofort, welche allgemeinen Voraussetzungen er für die Nutzung der API erfüllen muss und was die API kann (vgl. Shipcloud o. J. [a]). Durch die darauffolgenden Kapitel „Versioning and Prices" und „Help and Information" werden außerdem wichtige organisatorische Informationen geliefert. Inhaltlich sollten in einem Überblick in jedem Fall die häufigsten/wichtigsten Anfragen an die API sowie technische und wirtschaftliche Voraussetzungen enthalten sein. Diese Aspekte werden in der Shipcloud Dokumentation übersichtlich in einer Prozessgrafik dargestellt (siehe Abb. 16). Die Grafik visualisiert zum einen die zentrale Anwendung der API und situiert zum anderen Funktionen und Parameter im Prozess. Auf diese Weise kann der abstrakte Prozess der Arbeitsweise der API anschaulich als realer Nutzen dargestellt werden. Die Vermittlung des realen Nutzens durch grafische Mittel wie Prozessgrafiken, Icons o. ä. ist ein probates Mittel, um komplexe Zusammenhänge simplifiziert darzustellen.

Ein anderes Beispiel für die Vermittlung des realen Nutzens ist die Darstellung der verfügbaren Paketdienste in einer Tabelle mit den passenden Parametern (vgl. Shipcloud o. J. [a]). Die Paketdienste sind als Logos abgebildet, wodurch die Verknüpfung mit dem realen Nutzen sehr deutlich wird. Durch die Darstellung in der Tabelle ist außerdem schnell zu erkennen, welcher Paketdienst für welche Leistungen geeignet ist.

[65] Bedingungen könnten das Betriebssystem, Programmiersprachen, Server-Voraussetzungen usw. sein.

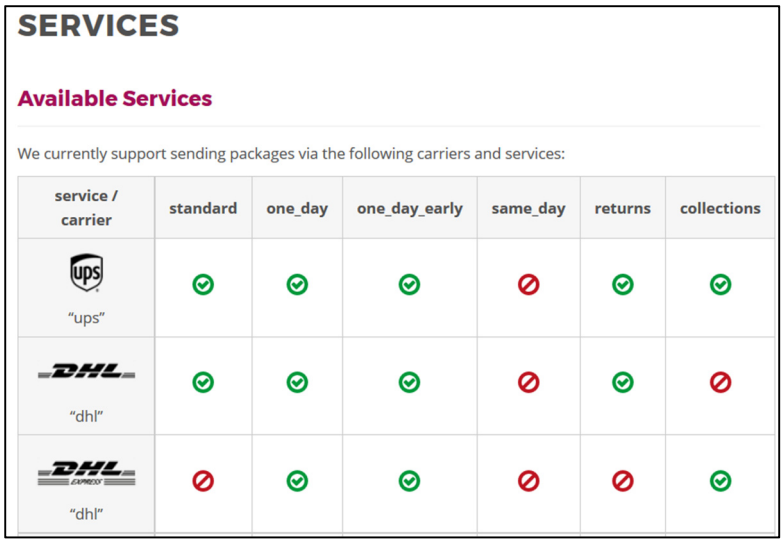


Abb. 17: Zusammenhang zwischen dem realen Nutzen einer API und deren Code (Shipcloud o. J. [a])

Alternativ zu einer Prozessgrafik können die wichtigsten/häufigsten Anfragen auch als verlinkte Liste abgebildet werden. Durch eine verlinkte Liste kann der Nutzer sofort an die richtige Stelle in der Dokumentation springen (siehe Abb. 18). In dem Beispiel verweist die große Überschrift links unten zusätzlich auf ein Anwendungsszenario, das vermutlich am häufigsten genutzt wird.

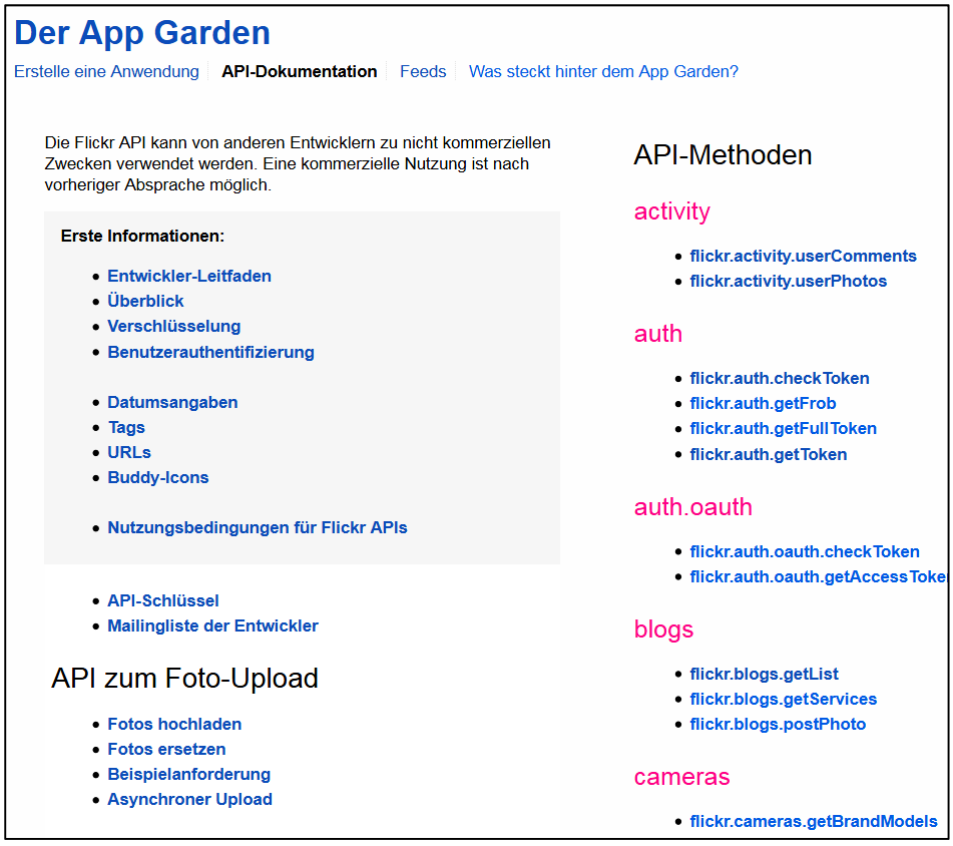


Abb. 18: Übersichtsseite mit integrierter Referenzdokumentation (flickr o. J. [a])

In Ergänzung zu dem Überblick ist ein Getting-Started-Dokument sehr wichtig. Es sollte die technischen Voraussetzungen sowie die Darstellung von Prozessen in den Mittelpunkt rücken. Im besten Fall ist das Getting-Started eine kurze und übersichtliche Anleitung für die erste Inbetriebnahme der API.

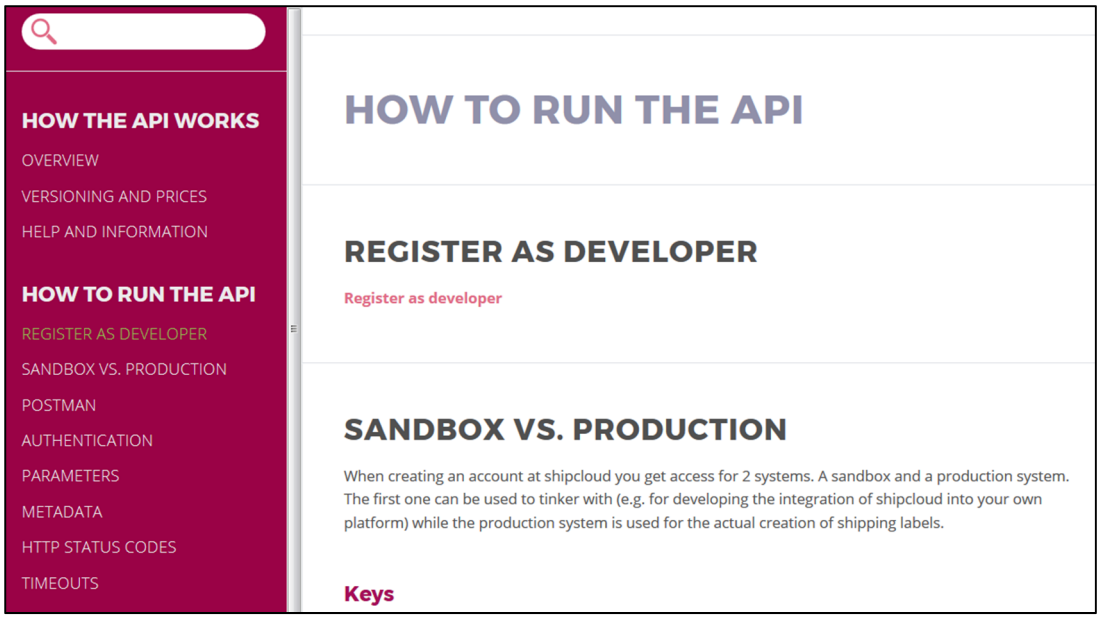


Abb. 19: Getting-Started mit prozessorientierter Überschrift (Shipcloud o. J.)

Mit der ersten Inbetriebnahme der API geht auch das Ausprobieren des Codes einher. Entwickler wollen schnell mit Code arbeiten und nicht nur theoretische Code-Beispiele lesen. Für diesen Fall bieten sich Code-Generatoren oder interaktive Übungsfelder an, die durch den Nutzer gefüllt werden können. Innerhalb der API-Dokumentation werden eine Anfrage und die passende Antwort der API im Test-Modus generiert.

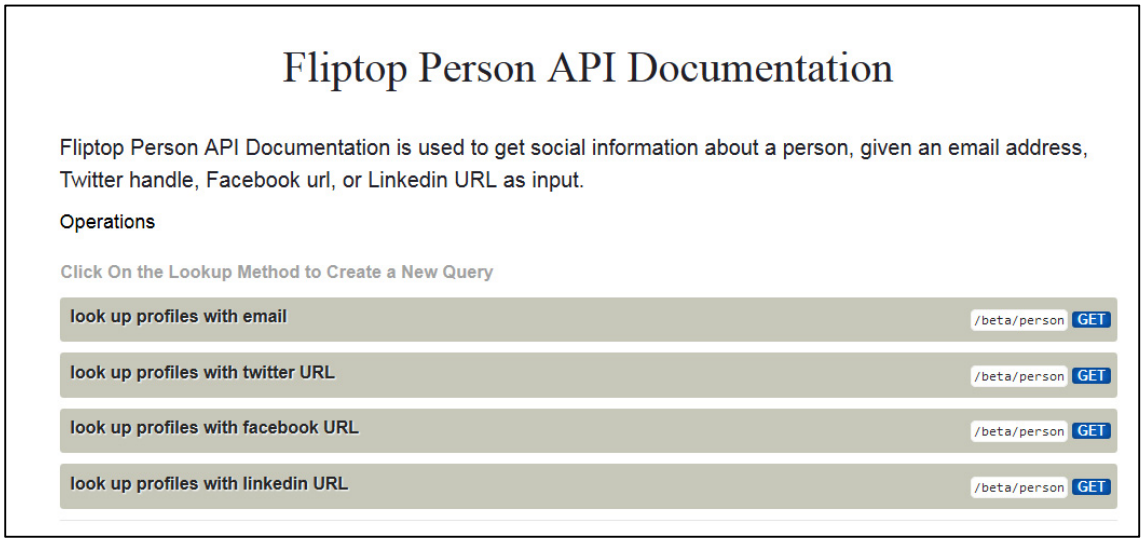


Abb. 20: Code-Generator mit nutzerzentriertem Aufbau (D&B Developer o. J.)

In der Abbildung sind die grau unterlegten Use-Cases vor der restlichen Dokumentation platziert (D&B Developer o. J.). Per Klick auf einen Use-Case öffnet sich das Panel, in dem die Werte für die geforderten Parameter der Funktion eingegeben werden können. Nach erfolgreicher Eingabe werden die gewünschte Anfrage sowie die Antwort der API generiert. Es gibt auch andere Möglichkeiten Code-Generatoren in Dokumentationen zu integrieren, z. B. ausgelagert per Link, parallel in einem mehrspaltigen Layout oder direkt in die Kapitel.

3.3.4. Grundregeln guter Dokumentation

Ergänzend zu den spezifischen Heuristiken für die Erstellung von API-Dokumentation sollten auch generelle Qualitätsmerkmale guter Dokumentation beachtet werden. Richtungsweisend ist, dass den Nutzern die Qualität von Dokumentation keineswegs egal ist. Zu ähnlichen Schlüssen gelangen auch Watson et al., die generell eine hohe Qualität von Crowd Dokumentation feststellen konnten (vgl. Watson et al. 2013: 169 f.).

Bei den Grundregeln guten Dokumentierens handelt es sich nicht nur um designtechnische Aspekte wie Layout oder textliche Qualität, sondern auch um den realen Mehrwert von Dokumentation. Eine fehlerhafte oder unvollständige Dokumentation kann den Nutzer zu falschen Schlüssen verleiten oder in Sackgassen führen (s. a. Dusold 2016). Damit eine Dokumentation vollständig und korrekt ist, müssen die Funktionen, Parameter etc. korrekt und umfassend beschrieben werden. Außerdem müssen alle Code-Beispiele korrekt, vollständig und aktuell sein.

Die Problematik der Aktualität könnte durch die Kennzeichnung der Version umgangen werden. Der Nutzer könnte durch eine Versionierung oder ein Datum der letzten Änderung an der Dokumentation sofort absehen, ob sie aktuell ist oder nicht. Die Aktualität bezieht sich neben den Code-Beispielen auch auf alle anderen Inhalte der Dokumentation. Ändert sich z. B. die Bezeichnung einer Funktion, kann der Nutzer ohne versionierte Änderungen seine Anfragen nicht anpassen bzw. korrekt erstellen.

Einer der wichtigsten Aspekte in der Erstellung von Dokumentation ist die Verwendung von kurzen und prägnanten Formulierungen. Der Nutzer möchte schnell zu der Lösung seines Problems geleitet werden, ohne Unmengen von Text oder Informationen lesen bzw. aufnehmen zu müssen. Aus diesem Grund sollte auf eine kompakte Syntax geachtet und auf zweideutige oder umständliche Formulierungen verzichtet werden.[66]

3.4. Zusammenfassung

Von den Ergebnisse der Studien *„Software-Entwicklerdokumentation: Was wollen Entwickler wirklich?“* (Steinhardt et al. 2015; Meng et al. 2016; Meng et al. 2017) und *„API-Dokumentation aus der Perspektive von Software-Entwicklern: Was macht sie besonders hilfreich?“* (Dusold 2016) wurden die dargelegten Heuristiken abgeleitet. Die Heuristiken beziehen sich vor allem auf die Unterstützung unterschiedlicher Lernansätze sowie auf die Gestaltung, Struktur und Navigation von Dokumentation. Zusätzlich wurden besondere Formate zum Schnelleinstieg und Grundregeln für gutes Dokumentieren berücksichtigt.

Die aufgestellten Heuristiken dienen als Grundlage für die Untersuchungen dieser Arbeit. Nachfolgend werden sie zu einem Kriterienkatalog zusammengefasst, der zur Überprüfung derselben in der Praxis angewendet wird.

4. Überprüfung der Studienergebnisse

Die in den vorherigen Kapiteln ausführlich dargestellten Studienergebnisse werden in Form der daraus erarbeiteten Heuristiken in dieser Arbeit überprüft. Basierend auf den Heuristiken wird ein

[66] Ein geeigneter Ratgeber für die anwenderfreundliche Texterstellung ist die Tekom Leitlinie für regelbasiertes Schreiben (vgl. tekom-AG „Regelbasiertes Schreiben“ 2013).

Kriterienkatalog angelegt, mit dem Dokumentationen aus der Praxis untersucht werden können. Anschließend werden die Ergebnisse vorgestellt und diskutiert.

4.1. Untersuchung

Dieses Kapitel beinhaltet alle Informationen zu der in dieser Arbeit durchgeführten Untersuchung. Einleitend werden die Methode und die Forschungsfrage dargelegt. Aufbauend auf den bisherigen Ausführungen wird dann ein Kriterienkatalog erstellt, der der Überprüfung der Heuristiken dient. Im Anschluss daran werden in einer kurzen Übersicht alle Forschungsobjekte zuerst allgemein beschrieben und dann nach den formalen Kriterien „Umfang“ und „Medium der Umsetzung“ unterteilt. Zuletzt wird das Vorgehen erklärt.

4.1.1. Methode

Diese Arbeit orientiert sich methodisch an der Studie „*API Documentation and Software Community Values: A Survey of Open-Source API Documentation*“ (Watson et al. 2013). Watson et al. überprüften eine Auswahl empirischer Erkenntnisse hinsichtlich der Strukturen von API-Dokumentation in der Praxis. Anders als vorherige Studien wählten sie bereits vorhandene API-Dokumentationen als Forschungsobjekte aus. Das Einbeziehen von existenter API-Dokumentation ist an sich zwar kein neuer Ansatz, die Untersuchungsmethode von Watson et al. dafür umso mehr. Sie evaluierten die ausgewählten Dokumentationen eigenständig anhand eines eigens zusammengestellten Kriterienkatalogs und verzichteten so auf die üblicherweise herangezogenen Entwicklerexpertisen (vgl. ebd. 2013: 165).

Die Vorgehensweise dieser Untersuchung ist an den qualitativen heuristischen Forschungsansatz von Watson et al. (2013) angelehnt. Anders als quantitative Forschungsmethoden, die vor allem die Überprüfung von Hypothesen verfolgen, zielt die qualitative heuristische Methode auf das Erforschen und Aufklären der Strukturen, Beziehungen und Abhängigkeiten eines Gegenstands ab (vgl. Gerken 2003: 13).

> „*Qualitative Techniken sollen dem Forscher unbekannte, von ihm auch nicht vermutete Beziehungen aufdecken. [...] Insofern ist quantitative Sozialforschung mehr ‚beschreibend‘ als entdeckend, weil sie Daten innerhalb eines vorgegebenen Kategorienschemas liefert und nicht das Kategorienschema selbst.*“
> (Kleining 1994: 19)

Die Theorie der Sozialwissenschaften lässt sich auch auf den Bereich der Usability-Methoden anwenden. Anders als in der Sozialwissenschaft gibt es im Usability-Testing keinen relevanten Widerspruch zwischen quantitativer und qualitativer Forschung (vgl. Gerken 2013: 21). Vielmehr werden qualitative und quantitative Untersuchungsmethoden gemischt. Bei einem Test mit einem qualitativen Vorgehen, z. B. der Think-Aloud-Methode[67], werden parallel quantitative Daten gewonnen und ausgewertet (vgl. ebd.).

Im Sinne des heuristischen Vorgehens werden durch die durchgeführten qualitativen Untersuchungen dieser Arbeit neue Hypothesen erarbeitet, die wiederum in der weiteren Forschung eruiert werden können. Aus diesem Grund beschränkt sich diese Arbeit auf die Untersuchung einer Forschungsfrage und stellt keine konkrete Hypothese auf.

[67] Bei der Think-Aloud-Methode werden die Probanden dazu aufgefordert, alle ihre Gedanken laut zu äußern (vgl. Häder 2015: 402 ff.). Durch die laute Äußerung der Gedanken können Lösungswege für Fragen vollständig nachvollzogen und qualitative Überlegungen mitgehört werden.

4.1.2. Forschungsfrage

Ziel dieser Arbeit ist es, die vorgestellten Heuristiken anhand von API-Dokumentationen aus der Praxis zu prüfen. Fraglich ist, ob und inwiefern die Heuristiken in den Dokumentationen wiederzufinden sind.

Forschungsfrage:
Können die erarbeiteten Heuristiken in API-Dokumentationen aus der Praxis nachgewiesen werden?

Die Forschungsfrage bezieht sich auf die Ergebnisse der Studien der HS Merseburg, die im Rahmen des Projekts „*Expertenorientierte Optimierung von Software-Entwicklerdokumentation*“ durchgeführt wurden. Die zu untersuchenden API-Dokumentationen werden nachfolgend als Forschungsobjekte bezeichnet.

4.1.3. Erstellung eines Kriterienkatalogs

4.1.3.1. Definition

Für die Überprüfung der Studienergebnisse in der Praxis wird ein Kriterienkatalog verwendet. Er umfasst eine Auflistung aller zentralen Anforderungen, die von den Teilnehmern in den beiden Studien der HS Merseburg genannt wurden.[68] Ziel ist es, das Vorkommen der Heuristiken in den Forschungsobjekten zu überprüfen.

> „*Kriterienkataloge enthalten Zusammenstellungen von Fragen und Einschätzungsskalen zur standardisierten Beschreibung und Beurteilung von Aspekten der technischen, inhaltlichen und didaktischen Qualität von Lernangeboten.*“
> (Tergan 2004: 143)

Lernangebote können digitale, multimediale oder analoge Instrumente zur Vermittlung von unbekannten Informationen sein (vgl. Tergan 2004: 131 ff.). Bei der Einarbeitung in eine neue API mithilfe von Dokumentation sind dem Entwickler die API und alle damit verbundenen Informationen unbekannt. Bevor er sie einsetzen kann, muss er in einem Prozess erlernen, wie sie funktioniert. Der stattfindende Lernprozess zeigt, dass API-Dokumentationen ebenfalls Lernangebote sind. Die Prüfung von API-Dokumentation mit einem Kriterienkatalog ist dementsprechend geeignet, um deren Qualität zu bewerten.

Ein Kriterienkatalog kann eine Vielzahl von Eigenschaften bei Lernangeboten abfragen. Er muss ein standardisiertes Werkzeug sein, damit alle Forschungsobjekte vergleichend untersucht werden können. Standardisiert bedeutet in diesem Zusammenhang, dass der Kriterienkatalog klare Vorgaben zu den zu untersuchenden Kriterien enthalten muss. Die Kriterien dürfen nicht für abweichende Forschungsobjekte angepasst oder abgeschwächt werden.

4.1.3.2. Erstellung

Für die nachfolgende Untersuchung wurde ein Kriterienkatalog erstellt. Für die Erstellung des Katalogs wurden die vorherigen Ausführungen bezüglich der Heuristiken für API-Dokumentation herangezogen.

Der Kriterienkatalog wurde in Form einer Tabelle angelegt. Er enthält vier Spalten, die die Heuristik, die Ausprägung, das Vorhandensein und Bemerkungen vorsehen. Unter „Heuristik“ werden alle Themengebiete eingetragen, die im Kapitel Heuristiken zur Erstellung von API-Dokumentation vorgestellt

[68] Siehe Kapitel 3 Forschungsprojekt der Hochschule Merseburg.

wurden. In der Spalte „Ausprägung“ sind die Eigenschaften der jeweiligen Heuristik zu finden. Das Vorkommen der Heuristiken wird in der Spalte „Vorhanden“ mit „Ja“ oder „Nein“ markiert. Falls notwendig, können andere Beobachtungen oder Auffälligkeiten in der letzten Spalte „Bemerkungen“ eingetragen werden. Unterhalb der letzten Heuristik „Grundregeln guter Dokumentation“ ist ein Block für weitere Beobachtungen platziert. Hier können alle Aspekte notiert werden, die relevant sind und die nicht von den Heuristiken abgedeckt werden.

Heuristik	Ausprägung (ggf. Abhängigkeit)	Vorhanden?	Bemerkungen
Intelligente Suchfunktion	Erweiterte Suchfunktion (Fehlertoleranz, Erkennung von Singular/Plural)	Ja	
		Nein	
	Komplexe Suchfunktion (Anbieten von verwandten Suchbegriffen, Erkennen und Anbieten von Synonymen, Autovervollständigung)	Ja	
		Nein	
	Suchhistorie	Ja	
		Nein	
	Suchergebnisse als Stichwort im Text hervorgehoben oder als Textstellen in Listenform	Ja	
		Nein	
Selektiver Zugriff auf Informationen	Zweispaltiges Layout Text - Code	**Ja**	
		Nein	
	Weitere Funktionen im zweispaltigen Layout (Referenzierung von Text und Code-Beispiel, dynamische Anpassung des Code-Beispiels an das Kapitel, Hervorheben der aktuellen Überschrift)	Ja	
		Nein	
	Navigationsstrang Problemlösung (Angebot von einer Überblicksseite in die konkrete Problemlösung zu springen, gezieltes Hinführen ohne Basisinformationen o. ä.)	Ja	
		Nein	
Codenahe Platzierung von wichtigen Informationen	Interaktive Verfügbarkeit von wichtigen Informationen (Hover-Funktion/ Verlinkung der Memberliste, Parameterinfo oder Quickinfo)	Ja	
		Nein	
	Inline-Kommentare (inklusive farbiger/typografischer Hervorhebung)	Ja	
		Nein	
	Redundante Präsentation wichtiger Informationen (in Text und Code-Beispiel o. ä.)	Ja	
		Nein	
Struktur	Gliederung prozessorientiert	Ja	
		Nein	
	Konsistente Feinstruktur der Kapitel	Ja	
		Nein	

Abb. 21: Kriterienkatalog

Heuristik	Ausprägung (ggf. Abhängigkeit)	Vorhanden?	Bemerkungen
Transparente Navigation	Logischer Aufbau, intuitive Bedienbarkeit	Ja	
		Nein	
	Prozess- oder inhaltsbasierte Überschriften	Ja	
		Nein	
	Keine übermäßige oder unübersichtliche Verschachtelung	Ja	
		Nein	
	Mitwandernde Navigationsleiste beim Scrollen (kein Verschwinden)	Ja	
		Nein	
	Dynamische Anpassung der Navigation bei Änderung des Kapitels durch Scrollen	Ja	
		Nein	
Konsistente Gestaltung	Visuelle Konsistenz	Ja	
		Nein	
	Inhaltliche Konsistenz	Ja	
		Nein	
	Visuelle Abgrenzung von Text und Code-Beispielen bzw. Code und Code-Kommentar	Ja	
		Nein	
Schneller Einstieg	Überblick generell (Ziel/Zweck der API, wichtigste Features, Lizenzierug, Preismodell etc.)	Ja	
		Nein	
	Getting-Started (schneller Einstieg, technische Voraussetzungen, Schwerpunkt auf Darstellung von Prozessen)	Ja	
		Nein	
	Fähigkeiten der API	Ja	
		Nein	
	Grundlegendes Prozess- und Hintergrundwissen	Ja	
		Nein	
	Voraussetzungen der API (technisch, wirtschaftlich etc.)	Ja	
		Nein	
	Wichtigste/ häufigste Anfragen an die API	Ja	
		Nein	
Grundregeln guter Dokumentation	(Umfangreiche) Beschreibung der Parameter, Funktion etc.	Ja	
		Nein	
	Kurze und prägnante Formulierungen	Ja	
		Nein	
	Kennzeichnung der Version	Ja	
		Nein	
Weitere Beobachtungen			

Abb. 21: Fortsetzung des Kriterienkatalogs

Die Ausprägungen der Heuristik „Grundregeln guter Dokumentation“ konnten nicht vollständig in den Kriterienkatalog übernommen werden. Nicht übernommen wurden die „vollständige und korrekte Beschreibung der Funktionen, Parameter etc.“ und „korrekte Code-Beispiele“. Diese Aspekte können in der Untersuchung nicht beurteilt werden, da die APIs nicht angewendet werden, sondern nur deren Dokumentation gelesen wird. Ob die Beschreibungen und Beispiele korrekt und vollständig sind, kann dementsprechend nicht beurteilt werden.

4.1.4. Forschungsobjekte

Die Forschungsobjekte dieser Arbeit sind 30 im Web frei zugängliche API-Dokumentationen. Nachfolgend werden die zu untersuchenden Dokumentationen kurz vorgestellt. Recherchiert wurden sie über die Webseite ProgrammableWeb, die sich mit APIs und API-Dokumentation beschäftigt (vgl. ProgrammableWeb o. J.). Sie enthält neben allgemeinen Informationen zu APIs auch aktuelle Meldungen, Best Practices für die Anfertigung von Dokumentation sowie eine API-Suche. Mithilfe der API-Suche wurden REST-APIs recherchiert, die über eine öffentlich zugängliche Dokumentation verfügen. Die Auswahl der APIs geschah zufällig und folgte keinem Muster. Nachfolgend werden REST-APIs als APIs bezeichnet.

4.1.4.1. Allgemeine Beschreibung der Forschungsobjekte

(1) OpenFEC API

Die OpenFEC API bietet einen Überblick darüber, wie politische Wahlkandidaten und Komitees in den USA ihre Kampagnen finanzieren (vgl. OpenFEC API o. J.). Entwickelt wurde sie von dem Unternehmen 18F, das Teil der General Service Administration ist (vgl. 18F o. J.).

(2) Udemy API

Die Udemy API bezieht sich auf eine Online-Kursplattform und ermöglicht es, Applikationen und Integrationen mit einem Zugriff auf Udemy zu bauen. Die Kursplattform ist in verschiedenen Sprachen verfügbar und bietet mehr als 16.000 Kurse an (vgl. Udemy o. J.).

(3) GuideStar Search API

GuideStar ist eine Webseite, die Informationen über Non-Profit Organisationen zur Verfügung stellt. Sie greift auf eine Datenbank zu, in der u. a. Informationen zu den Finanzen, der politischen Ausrichtung, der Reputation usw. gespeichert sind (vgl. Guide Star 2015). Die Search API ermöglicht die gefilterte Suche nach Organisationen (vgl. Guide Star 2016).

(4) ANX Exchange API v3

Mithilfe der Webseite ANXPRO können Nutzer bargeldlos mit einer Kryptowährung im Internet bezahlen. Anders als Bitcoin unterstützt ANXPRO auch andere Kryptowährungen wie etwa Litecoin, Dogecoin etc. (vgl. ANXPRO 2016). Mithilfe der Exchange API können u. a. der Markt für Kryptowährungen beobachtet und ein Kauf oder Verkauf von Coins realisiert werden (vgl. ANX Exchange API v3 o. J.).

(5) Google Maps Directions API

Die Google Maps Directions API ermöglicht das Einbinden eines Routenplaners in Webseiten oder mobile Applikationen. Im Routenplaner kann beispielsweise eine Route mit verschiedenen Transportmitteln inklusive Anpassung der Fahrzeit berechnet werden (vgl. Google Maps APIs o. J.).

(6) Flickr API

Flickr ist eine Plattform, auf der Fotos hochgeladen, geteilt und um Metadaten ergänzt werden können. Metadaten sind die Geo-Position, Personennamen etc. (vgl. flickr o. J. [b]). Die Flickr API unterstützt neben Webdiensten auch mobile -und Desktopanwendungen (vgl. flickr o. J. [b]).

(7) Box Content API

Die Box Content API ermöglicht den Zugriff auf ein gesichertes Content Management System (vgl. boxDevelopers o. J.). In diesem System können Daten aufbewahrt und geteilt werden. Der Zugriff ist von mobilen Endgeräten und anderen Geräten möglich.

(8) Klipfolio API

Klipfolio bietet mit seiner API die Möglichkeit, unkompliziert Dashboards zu bauen. Dafür ermöglicht die API den Zugriff auf vorgefertigte Metriken und Visualisierungen sowie betriebswirtschaftliche Kennzahlen (vgl. Klipfolio o. J.).

(9) Etsy API

Auf der E-Commerce-Plattform Etsy können Nutzer handgemachte Sachen kaufen und verkaufen. Die API ermöglicht den Zugriff auf Verkaufsinformationen wie Fotos, Preise und Beschreibungen (vgl. Etsy 2017).

(10)Zazzle API

Zazzle bietet als E-Commerce-Plattform die Möglichkeit, personalisierte Geschenkideen umzusetzen. Beispielsweise können bedruckte T-Shirts, Tassen usw. gestaltet werden. Durch die Zazzle API können externe Anbieter die Erstellung von Grafiken o. ä. anbieten, die von Zazzle automatisch in deren Produkte eingefügt werden (vgl. Zazzle 2014).

(11)Open Calais API

Thomson Reuter bietet mit der Open Calais API die Möglichkeit, unstrukturierten Text in strukturierten umzuwandeln. Das geschieht, indem der Text auf semantische Stichwörter gescannt und als RDF-Dokument ausgegeben wird. Aus den Stichwörtern werden Metadaten gewonnen, die Aufschluss über die Zusammenhänge mit anderen Dokumenten liefern können (vgl. Thomson Reuter Open Calais 2017).

(12)Reddit API

Reddit ist eine Plattform, auf der angemeldete Nutzer Texte und Bilder als Artikel veröffentlichen, die von anderen Nutzern bewertet werden können (vgl. reddit 2016). Durch die Reddit API kann auf alle Funktionen der Webseite zugegriffen werden, z. B. Inhalte veröffentlichen, kommentieren und bewerten (vgl. reddit 2017).

(13)Open Secrets API

Die Webseite Open Secrets bietet einen Überblick über Geld, das in der amerikanischen Politik fließt. Besonderes Augenmerk liegt dabei auf dem Effekt des Geldflusses auf Wahlen und dem Gefüge Politik-Wirtschaft-Gesellschaft (vgl. Open Secrets o. J. [a]). Durch die Open Secrets API kann auf diese Daten zugegriffen und z. B. ein Mash-Up generiert werden (vgl. Open Secrets o. J. [b]).

(14)Spotify Web API

Spotify stellt kostenlos Musikdateien für den individualisierten Zugriff zur Verfügung. Mithilfe der Spotify Web API kann auf den gesamten Musikkatalog und persönliche Playlisten, gespeicherte Musik etc. zugegriffen werden (vgl. Spotify Developer 2016).

(15)Photobucket API

Photobucket ist eine Webseite für das Hochladen, Teilen und Runterladen von Fotos und Videos. Durch die Photobucket API kann auf alle Funktionen der Webseite zugegriffen werden (vgl. Photobucket 2010).

(16)NYT Books API

Die amerikanische Zeitung New York Times bietet mit der NYT Books API die Möglichkeit, auf deren Best-Seller-Listen und Buchrezensionen zuzugreifen (vgl. NYT Developers Network o. J.).

(17)Visual Search API

Mithilfe der Visual Search API können anhand eines hochgeladenen Bildes ähnliche Bilder im Web gesucht werden. Gesucht wird auf Basis von Farbe, Formen, Geometrie und Bildeigenschaften (vgl. VisualSearchApi 2016).

(18)Tinfoil Security API

Tinfoil Security ist ein Anbieter für Sicherheitssysteme für Webseiten. Ihre Programme scannen Webseiten auf Schwachstellen und helfen dabei, diese einfach und schnell zu beseitigen (vgl. Tinfoil Security 2017[a]). Durch die API kann auf alle Funktionen zugegriffen werden, die in der Webanwendung verfügbar sind (vgl. Tinfoil Security 2017[b]).

(19)Arlo REST Public API

Arlo ist ein individualisiertes Managementsystem für Termine und Veranstaltungen wie Tagungen, Weiterbildungen o. ä. im Business-Sektor. Die Arlo REST Public API ermöglicht den Zugriff auf kommende Termine, Terminvorlagen, Veranstaltungsorte, eine Übersicht der Redner u. v. m. (vgl. Arlo 2015).

(20)Watchful REST API

Watchful ist eine Anwendung, mit der alle Webseiten in einem Interface zusammengefasst werden können, die mit den CMS Joomla und Wordpress erstellt wurden (vgl. Watchful 2017[a]). Außerdem unterstützt Watchful die Sicherheit der Webseiten und das Update-Management. Durch die API kann auf alle Funktionen von Watchful zugegriffen werden (vgl. Watchful 2017[b]).

(21)The Budget Your Trip API

Die Webseite Budget Your Trip ermöglicht die finanzielle Planung für Reisen auf der ganzen Welt. Zusätzlich biete sie einen Währungsrechner und eine lokale Suche nach Namen oder Geodaten an (vgl. Budget Your Trip o. J.). Die API ermöglicht den Zugriff auf alle Funktionen der Webseite.

(22)Kaltura API

Kaltura ist ein Anbieter für den Aufbau von webbasierten Videoplattformen. Die Kaltura API bietet u. a. die Möglichkeit Videos bereitzustellen und abzuspielen sowie den Zugriff auf statistische Daten wie Klicks (vgl. KalturaDeveloper 2016).

(23)Oanda API

Oanda ist ein Online-Dienstleister mit umfangreichen Datenbanken zu Devisenkursen. Sie bieten Devisen- und CDF-Handel, Wechselkursdaten und den Devisentransfer für Unternehmen an (vgl. Oanda 2017). Die API ermöglicht einen umfangreichen Zugriff auf aktuelle und archivierte Marktdaten sowie Trading-Prozesse (vgl. Oanda API 2014).

(24)Microsoft Outlook Mail REST API

Microsoft Outlook ist eines der meistgenutzten E-Mail Systeme. Die Microsoft Outlook Mail Rest API ermöglicht das Lesen, Schreiben und Senden von Nachrichten und Anhängen, das Lesen und Beantworten von Veranstaltungseinladungen, die Ordnerverwaltung u. v. m. (vgl. Microsoft 2017[b]). Die gleichen Funktionen sind auch für die Anbieter Hotmail.com, Live.com, MSN.com, Outlook.com und Passport.com verfügbar (vgl. ebd.).

(25)Docker Cloud REST API

Docker ist ein Dienstleister für Software-Container, der das modulare und weitestgehend unabhängige Bearbeiten und Nutzen von Anwendungen, Software etc. ermöglicht (vgl. Docker 2017). Mit der Docker Cloud REST API kann auf alle Funktionen der Docker Cloud zugegriffen werden (vgl. Docker Cloud o. J.).

(26)Here Weather API

Here ist einer der führenden Anbieter für Navigations-, Karten- und andere standortbezogene Informationsdienste (vgl. Here 2017[a]). Die Here Weather API ermöglicht den Zugriff auf aktuelle Wettervorhersagen, Unwetterwarnungen sowie Mond- und Sonnendaten für einen bestimmten Standort (vgl. Here 2017[b]).

(27)SNAP PAC REST API

SNAP PAC ist ein integriertes Software- und Hardwaresystem von Opto 22 für die industrielle Anwendung. Es ermöglicht die Industriesteuerung, Fernüberwachung, Datenerfassung und Vernetzung im Internet of Things (vgl. Opto 22 2017). Die Snap Pac REST API vereinfacht den Einsatz von SNAP PAC im klassischen Sinn einer API (vgl. Opto 22 Developer 2017).[69]

(28)Commerce.js API

Mithilfe der Commerce.js API des Anbieters Chec kann auf alle notwendigen Funktionen für den Aufbau und Betrieb eines E-Commerce zugegriffen werden (vgl. Commerce.js 2016). Die API kann alleine oder in Kombination mit anderen APIs von Chec verwendet werden (vgl. Commerce.js o. J.).

(29)TryMyUI 2 API

TryMyUI ist ein Anbieter für Usability- und Nutzer-Tests für das WWW (vgl. TryMyUI 2015). Privatpersonen und Unternehmen können durch TryMyUI ihre Webseiten, Apps etc. von echten Nutzern ausprobieren und bewerten lassen. Die TryMyUI API bietet vollen Zugriff auf alle Funktionen der Webseite, z. B. der Auswahl von Testern oder das Stellen von Aufgaben.

(30)Streetlayer API

Die Streetlayer API bietet die Möglichkeit international Adressen und Adressfragmente zu validieren (streetlayer 2017). Basierend auf einer wöchentlich aktualisierten Datenbank können z. B. Unternehmen ihren Adressdatenbestand und neue Adressen verifizieren und bereinigen (vgl. ebd.)

4.1.4.2. Einteilung der Forschungsobjekte nach formalen Kriterien

Die Forschungsobjekte werden vor der Untersuchung nach formalen Kriterien unterschieden. Die Kriterien beziehen sich auf das Medium der Umsetzung und den Umfang. Diese Einteilung ermöglicht in der Auswertung eine bessere Vergleichbarkeit und das Erkennen von Abhängigkeiten oder Mustern.

[69] Die Argumente für den Einsatz von APIs werden in dem Kapitel APIs aus Sicht der Informatik dargelegt.

Das Medium der Umsetzung kann eine Webseite, eine Onepage oder ein PDF sein. Eine Webseite verfügt über eine Navigation, die durch Verlinkungen auf Unterseiten weiterleitet. Alle Unterseiten müssen in die Navigation eingebunden sein, andere Verlinkungen werden als extern bzw. Sackgassen angesehen. Eine Sackgasse bezeichnet in diesem Zusammenhang eine Seite ohne Navigation bzw. strukturelle Einbindung in die Webseite. Der Nutzer kann dementsprechend nur über den Zurück-Button des Browsers auf die vorherige Seite zurückkehren.

Eine Onepage ist ein spezieller Typ einer Webseite. Bei ihr werden alle Inhalte linear durch Scrollen oder nicht-linear durch Springen mithilfe von Zwischenüberschriften erreicht. Eine Onepage hat keine Verlinkungen auf Unterseiten, kann aber externe Verlinkungen aufweisen. Beispiele dafür wären die Verlinkung eines Begriffs zu Wikipedia oder die Verlinkung einer Funktion zu einem Code-Beispiel in Form einer Sackgasse.

Das dritte Medium der Umsetzung sind PDF-Dateien, die eine buchähnliche, lineare Struktur aufweisen. PDFs können online oder bei vorheriger Speicherung offline gelesen werden. Häufig enthalten sie eine Navigation und/oder ein Inhaltsverzeichnis, diese Strukturen müssen jedoch gezielt beim Export aus einem Editor ausgewählt werden.

Der Umfang der Forschungsobjekte konnte kurz, mittel oder lang sein. Für die Zuordnung des Umfangs wurden die Dokumentationen in Seiten eingeteilt. Eine Seite entsprach dabei einer Seite im PDF oder einer Seite auf einer Webseite bzw. einem Kapitel in einer Onepage. Auf den Seiten/in den Kapiteln wurde ein kurzes Scrollen toleriert, allerdings bis maximal der doppelten Länge, die ein Bildschirmausschnitt betrug. Bei längerem Text wurden zwei oder ggf. mehr Seiten gezählt. Die Einteilung in kurz, mittel und lang wurde von der Gesamtanzahl der Seiten abhängig gemacht.

Kurz	Mittel	Lang
Max. 10 Seiten	11 - 30 Seiten	Mehr als 30 Seiten

Abb. 22: Abstufung des Umfangs in den formalen Kriterien

Nachfolgend werden alle Forschungsobjekte in einer Tabelle den formalen Kriterien zugeordnet. Sie konnten entweder kurz, mittel oder lang sowie eine Webseite, Onepage oder ein PDF sein. Doppelnennungen waren nicht möglich.

Nr.	API	Umfang			Medium		
		Kurz	Mittel	Lang	Webseite	Onepage	PDF
1	OpenFEC API			x		x	
2	Udemy API		x		x		
3	Guide Star Search API	x			x		
4	ANX Exchange API v3			x		x	
5	Google Maps Directions API		x		x		
6	Flickr API			x	x		
7	Box Content API			x	x		
8	Klipfolio API			x	x		
9	Etsy API			x	x		
10	Zazzle API		x				x

Abb. 23: Einteilung der Forschungsobjekte nach den formalen Kriterien

Nr.	API	Umfang			Medium		
		Kurz	Mittel	Lang	Webseite	Onepage	PDF
11	Thomson Reuter Open Calais API			x			x
12	Reddit API			x		x	
13	Open Secrets API		x		x		
14	Spotify Web API			x	x		
15	Photobucket API			x	x		
16	NYT Book API	x			x		
17	Visual Search API	x				x	
18	Tinfoil Security API	x				x	
19	Arlo REST Public API			x	x		
20	Watchful REST API			x		x	
21	The Budget Your Trip API			x	x		
22	Kaltura API			x	x		
23	Oanda API			x	x		
24	Microsoft Outlook Mail REST API			x	x		
25	Docker Cloud REST API			x		x	
26	Here Weather API			x	x		
27	SNAP PAC REST API			x	x		
28	Commerce.js API			x		x	
29	TryMyUI 2 API			x	x		
30	Streetlayer API	x				x	
Summe		**5**	**4**	**21**	**19**	**9**	**2**

Abb. 23: Fortsetzung Einteilung der Forschungsobjekte nach den formalen Kriterien

In absoluten Zahlen gibt es 19 Forschungsobjekte, die als Webseite umgesetzt wurden. Darauf folgen neun, die als Onepage und zwei, die als PDF realisiert wurden. Bezogen auf den Umfang gibt es fünf kurze, vier mittlere und 21 lange Forschungsobjekte. Die Verteilung der Forschungsobjekte auf die formalen Kriterien wird für die bessere Übersicht zusätzlich grafisch in Prozentwerten dargestellt.

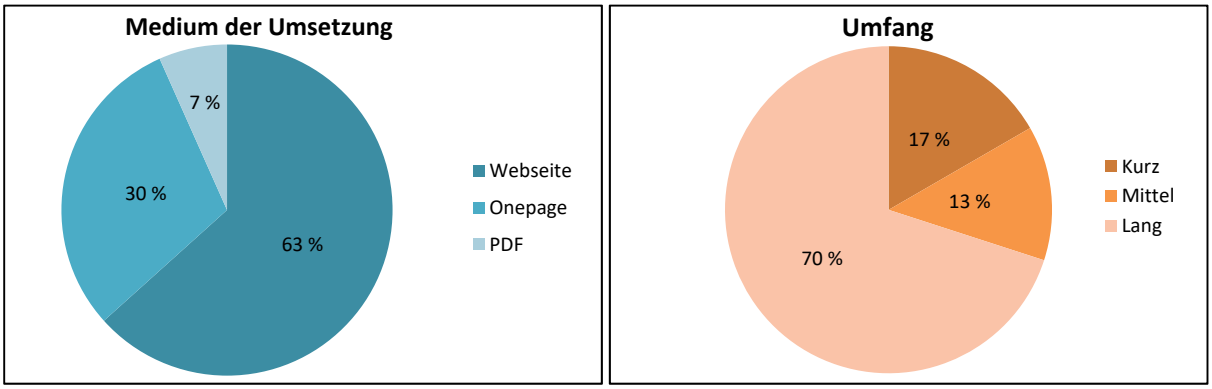
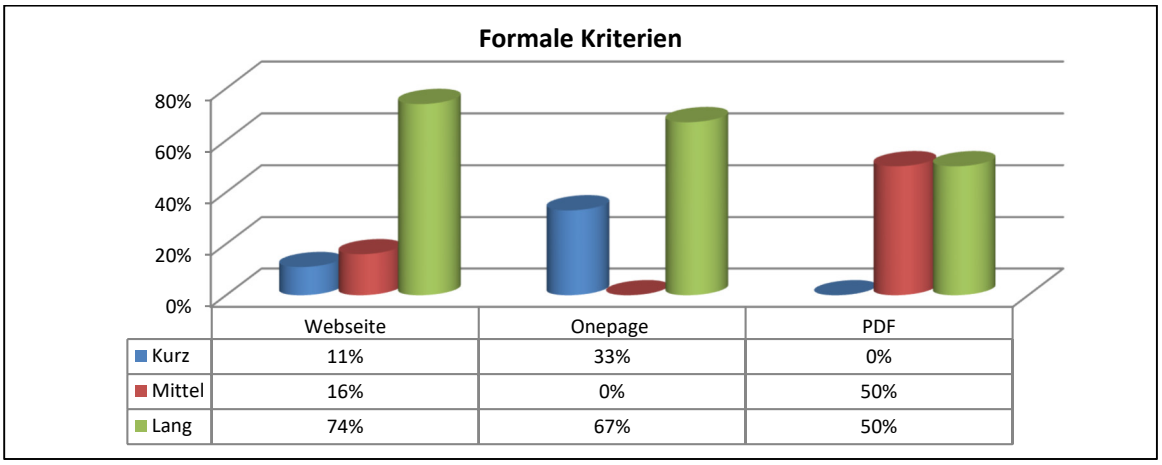


Abb. 24: Prozentuale Verteilung der Forschungsobjekte anhand der formalen Kriterien

Etwas mehr als die Hälfte der Forschungsobjekte sind mit 63 %, Webseiten, das Vorkommen von Onepages liegt mit 30 % deutlich darunter. PDFs sind nur zu 7 % vertreten. Unabhängig vom Medium

hat die Mehrheit der Forschungsobjekte mit 70 % einen langen Umfang. Ein langer Umfang ergab sich häufig aus einer großen Anzahl von Funktionen, so dass APIs mit einem längeren Umfang meist auch komplexer waren. Die Komplexität kann in dieser Untersuchung jedoch nicht miteinbezogen werden, da sie durch die Autorin nicht zweifelsfrei beurteilt werden kann.[70] 17 % der Forschungsobjekte weisen einen mittleren und 13 % einen kurzen Umfang auf.

Das Medium der Umsetzung und der Umfang können auch in Beziehung zueinander betrachtet und auf diese Weise verglichen werden.



	Webseite	Onepage	PDF
Kurz	11%	33%	0%
Mittel	16%	0%	50%
Lang	74%	67%	50%

Abb. 25: Verteilung der Forschungsobjekte auf die Medien in Bezug zum Umfang

Die meisten Forschungsobjekte, die als Webseite umgesetzt wurden, haben mit 74 % einen langen Umfang. Kurze Webseiten gab es nur zu 11 %, mittlere Webseiten zu 16 %. Eine sehr starke Tendenz zu langen Dokumentationen ist auch bei dem Medium Onepage zu erkennen. Hier weisen 67 % der Forschungsobjekte einen langen Umfang auf, während die restlichen 33 % kurze Onepages sind. Mittlere Onepages sind nicht vorhanden. Bei dem Medium PDF verteilt sich der Umfang zu jeweils 50 % auf mittel und lang, kurze PDFs kommen nicht vor.

Die Einteilung der Forschungsobjekte nach formalen Kriterien wird in der Auswertung wieder aufgegriffen.

4.1.5. Vorgehen

Das Vorgehen der Untersuchung gestaltet sich für alle Forschungsobjekte analog. Die Untersuchung wird durch die Autorin dieser Arbeit durchgeführt.

Nachdem alle notwendigen Angaben im Kopf des Kriterienkatalogs gemacht wurden, wird die jeweilige Dokumentation komplett gelesen. Anschließend wird sie erneut unter Beachtung des Kriterienkatalogs in zwei Durchgängen durchgearbeitet. Für die Bewertung der Kriterien wird keine Einschätzungsskala verwendet, sie können lediglich „erfüllt“ oder „nicht erfüllt“ sein. Alle Abweichungen oder Auffälligkeiten müssen in einem gesonderten Textfeld vermerkt werden. Für jedes Forschungsobjekt wird der Kriterienkatalog komplett ausgefüllt und als MS Office Word-Dokument (2010) gespeichert.

[70] Für die Beurteilung der Komplexität der APIs wären Programmierkenntnisse notwendig, über die die Autorin nicht verfügt. Die Untersuchung der Zusammenhänge von Kriterien zur Gestaltung von API-Dokumentation und der Komplexität von APIs ist aber eine interessante Perspektive für zukünftige Forschungen.

Die Ergebnisse der Untersuchung werden in einer MS Office Excel-Datei gesammelt. Sämtliche Diagramme für die Diskussion werden ebenfalls in MS Office Excel (2010) erstellt.

4.2. Auswertung

Die Resultate der Untersuchung werden im Rahmen der Auswertung sowohl quantitativ dargestellt als auch qualitativ diskutiert. Die Betrachtung erfolgt in dem Abschnitt „Ergebnisse“ anhand der Forschungsobjekte, der Heuristiken und weiterer Ansätze. In dem Abschnitt „Diskussion“ werden die Ergebnisse für jede einzelne Heuristik zuerst mit den formalen Kriterien in Beziehung gesetzt. Darauf aufbauend werden die Ausprägungen der Heuristik betrachtet, die ebenfalls in Abhängigkeit von den formalen Kriterien diskutiert werden. Zuletzt werden die Problematiken für jede Heuristik dargestellt, die während der Untersuchung aufgetreten sind.

4.2.1. Ergebnisse

Die Untersuchung hat ergeben, dass die Heuristiken mit einer knappen Mehrheit in den Forschungsobjekten vorkommen. In absoluten Zahlen wurden von 900[71] untersuchten Kriterien 455 mit „Ja“ und 445 mit „Nein“ bewertet. Prozentual gesehen ergibt das einen Anteil von 50,6 % für die Übereinstimmung der Forschungsobjekte mit den Heuristiken.

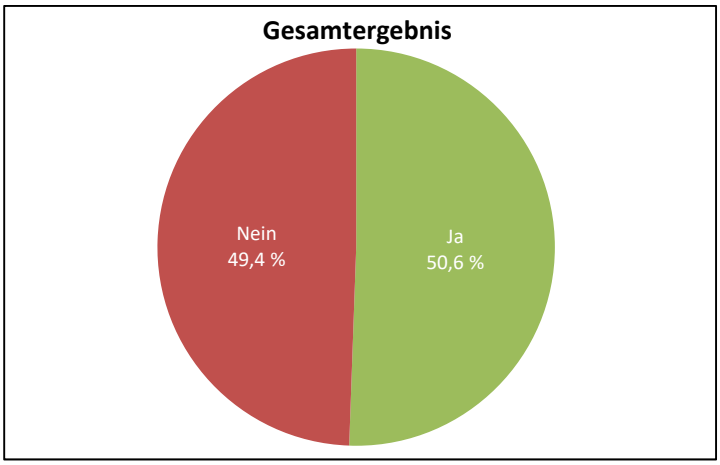


Abb. 26: Gesamtbetrachtung der Ergebnisse nach Ja/Nein

Die Verteilung der mit „Ja“ eingestuften Heuristiken und deren einzelner Kriterien variierte stark zwischen den Forschungsobjekten und Heuristiken. Aus diesem Grund werden die Ergebnisse nachfolgend einzeln vorgestellt.

4.2.1.1. Betrachtung nach Forschungsobjekten

Gemessen an den Forschungsobjekten liegt das Maximalergebnis der Übereinstimmungen bei 24 von 30 möglichen Ausprägungen, was einem Wert von 80 % entspricht. Die geringste Übereinstimmung umfasst 10 Kriterien bzw. 33 %. Die Hälfte aller Forschungsobjekte weist eine Übereinstimmung von 50 % oder mehr auf. 10 % haben sogar ein Ergebnis über 70 %. Die detaillierte Auflistung der Ergebnisse für alle Forschungsobjekte ist der nachfolgenden Abbildung zu entnehmen.

[71] Die Gesamtanzahl von 900 Kriterien ergibt sich aus der Anzahl der Forschungsobjekte multipliziert mit der Anzahl der untersuchten Ausprägungen: 30 x 30 = 900.

Forschungsobjekt	Platz	Anzahl positiv	%	Anzahl negativ	%
MOMR	**1**	24	80 %	6	20 %
GMD	**2**	23	77 %	7	23 %
BC	**3**	22	73 %	8	27 %
SW	**4**	20	67 %	10	33 %
OANDA	**5**	19	63 %	11	37 %
Etsy	**6**	18	60 %	11	37 %
Kaltura	**7**	17	57 %	12	40 %
CJS	**7**	17	57 %	13	43 %
PH	**7**	17	57 %	13	43 %
ANXE	**7**	17	57 %	13	43 %
HW	**11**	15	50 %	13	43 %
HW	**11**	15	50 %	13	43 %
DCR	**11**	15	50 %	15	50 %
TROC	**11**	15	50 %	15	50 %
Reddit	**11**	15	50 %	15	50 %
Flickr	**11**	15	50 %	15	50 %
ARP	**16**	14	47 %	15	50 %
WR	**16**	14	47 %	16	53 %
TBYT	**16**	14	47 %	16	53 %
SPR	**16**	14	47 %	16	53 %
GSS	**16**	14	47 %	16	53 %
Klipfolio	**21**	13	43 %	16	57 %
NYTB	**21**	13	43 %	17	57 %
TMU	**23**	12	40 %	17	60 %
OFEC	**23**	12	40 %	18	60 %
OS	**23**	12	40 %	18	60 %
VS	**26**	11	37 %	19	63 %
SL	**26**	11	37 %	19	63 %
Udemy	**26**	11	37 %	19	63 %
TS	**26**	11	37 %	19	63 %
Zazzle	**30**	10	33 %	17	67 %

Abb. 27: Gesamtergebnis nach Forschungsobjekten geordnet nach Häufigkeit Ja/Nein

Die höchste Übereinstimmung der Forschungsobjekte liegt bei 80 % und wurde von der Microsoft Office Mail Rest API (MOMR)[72] erreicht. Das zweithöchste Ergebnis beträgt 77 %, das Forschungsobjekt ist die Google Maps Directions API (GMD). Die geringste Übereinstimmung liegt bei 33 % und betraf die Forschungsobjekte Tinfoil Security (TS) und Zazzle. Zwischen den meisten und den wenigsten Übereinstimmungen liegen fast 50 % Unterschied.

4.2.1.2. Betrachtung nach Heuristiken

Die Betrachtung der Ergebnisse nach den Heuristiken ist ebenso heterogen wie die Betrachtung nach Forschungsobjekten. Die Spannweite der Übereinstimmungen reicht von 91 bis zu 19 in absoluten Zahlen. Anhand der Betrachtung der absoluten Zahlen kann in diesem Zusammenhang jedoch keine klare Aussage getroffen werden, da die einzelnen Heuristiken unterschiedlich viele Ausprägungen aufweisen. Aus diesem Grund können die absoluten Zahlen nicht für eine Betrachtung herangezogen werden, weshalb die prozentuale Verteilung dargestellt wird.

[72] Eine Übersicht der Forschungsobjekte und den jeweiligen Abkürzungen ist im Anhang zu finden.

	Intelligente Suchfunktion	Selektiver Zugriff auf konzeptuelle Informationen	Codenahe Platzierung von wichtigen Informationen	Struktur	Transparente Navigation	Konsistente Gestaltung	Schneller Einstieg	Grundregeln guter Dokumentation	Gesamt
Ja	29	35	19	36	96	86	90	64	455
Nein	91	85	71	24	54	4	90	26	445
Gesamt	120	120	90	60	150	90	180	90	**900**
Prozent „Ja“	24 %	29 %	21 %	60 %	64 %	96 %	50 %	71 %	
Ranking nach „Ja“	**7**	**6**	**8**	**4**	**3**	**1**	**5**	**2**	
Prozent „Nein“	76 %	71 %	79 %	40 %	36 %	4 %	50 %	29 %	
Ranking nach „Nein“	**2**	**3**	**1**	**5**	**6**	**8**	**4**	**7**	

Abb. 28: Gesamtbetrachtung der Ergebnisse nach Heuristiken

Die Abbildung zeigt die Verteilung der Übereinstimmungen auf alle Heuristiken. Die Heuristiken „Grundregeln guter Dokumentation“ und „Konsistente Gestaltung“ haben mit deutlichem Abstand am meisten Übereinstimmungen. Besonders sticht die „Konsistente Gestaltung hervor“, die mit 96 % einen sehr hohen Wert erreicht. Die „Codenahe Platzierung von Informationen“, die „Intelligente Suchfunktion“ und der „Selektive Zugriff auf Informationen“ weisen mit zwischen 20 % - 30 % den geringsten Wert von Übereinstimmungen auf. Die übrigen drei Heuristiken bewegen sich im oberen Mittelfeld zwischen 50 % und 65 %. Insgesamt weist die Hälfte der Heuristiken Übereinstimmungen von über 50 % auf. Das entspricht einer absoluten Anzahl von 4 Heuristiken, die 50% oder mehr Übereinstimmung mit den Forschungsobjekten aufweisen.

4.2.1.3. Weitere Betrachtungen

Die Varianz der Übereinstimmungen konnte nicht nur in Bezug auf Forschungsobjekte und Heuristiken festgestellt werden, sondern auch in Bezug auf die einzelnen Kriterien. In einigen Heuristiken gab es Kriterien, die besonders häufig bzw. besonders selten erfüllt wurden. Ein Beispiel dafür ist die Heuristik „Transparente Navigation“.[73]

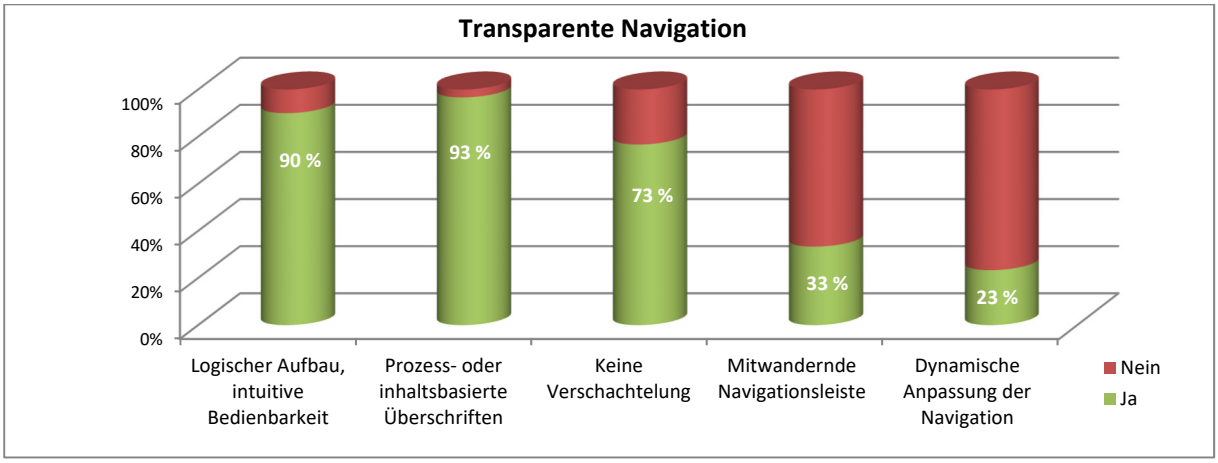


Abb. 29: Ergebnisse für Ja/Nein in Prozentzahlen

[73] In der Diskussion werden die Ergebnisse für alle Heuristiken einzeln ausführlich dargestellt und besprochen.

In dieser Heuristik gibt es große Unterschiede der Übereinstimmungen für die Kriterien. Auffällig ist, dass die ersten drei Kriterien eine große Anzahl an Übereinstimmungen haben, während die letzten beiden Kriterien (v. l. n. r.) deutlich weniger aufweisen. Die ungleichmäßige Verteilung der Übereinstimmungen auf die Kriterien zeigt einen generellen Trend, der auch bei den anderen Heuristiken beobachtet werden konnte. Ein Diagramm visualisiert die unterschiedliche Verteilung für die Gesamtheit der Kriterien sehr gut. Die Kriterien werden ohne Bezug zu der jeweiligen Heuristik dargestellt. Das Diagramm dient lediglich der Visualisierung der Verteilung der Übereinstimmungen, ein Ablesen von genauen Werten ist nicht vorgesehen. Die Vorstellung der genauen Prozentzahlen wird nachfolgend in der Diskussion durchgeführt.

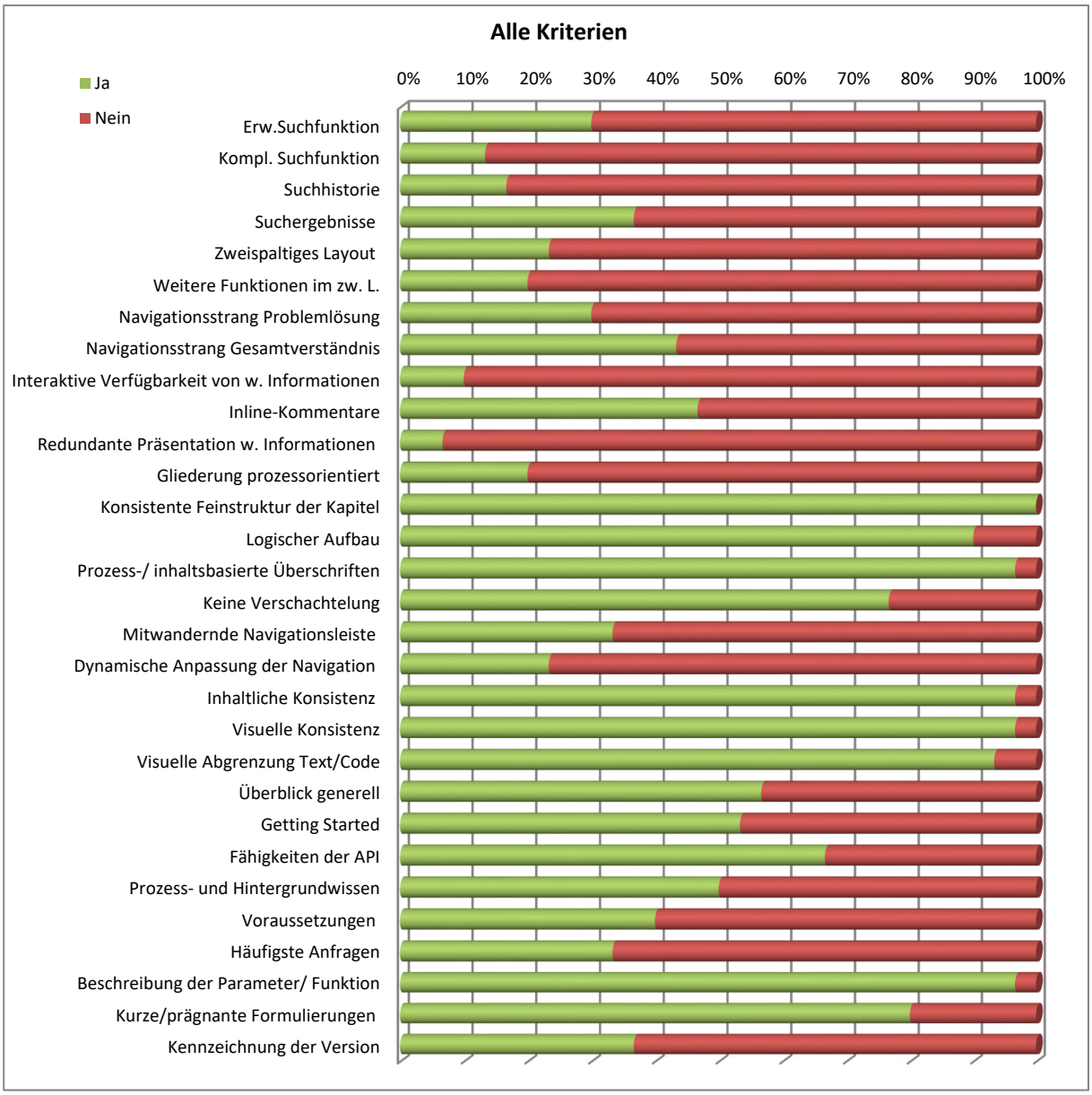


Abb. 30: Ergebnisse für „Ja“ nach Kriterien in Prozentzahlen

4.2.2. Diskussion

Die Betrachtung der Ergebnisse erfolgt ausgehend von den vorgestellten Heuristiken[74]. Alle Heuristiken werden allgemein, anhand der formalen Kriterien, anhand der Ausprägungen bzw. Kriterien der Heuristik und bezogen auf ihre Problematik dargestellt. Die Reihenfolge der Heuristiken in der Diskussion richtet sich nach deren Übereinstimmungen in absteigender Reihenfolge.

4.2.2.1. Konsistente Gestaltung

Die Heuristik „Konsistente Gestaltung“ wird zu 96 % erfüllt, womit sie eindeutig und mit großem Abstand den ersten Platz einnimmt. Die zweithäufigste Heuristik sind die „Grundregeln guter Dokumentation“, die 71 % der Kriterien erfüllen. Der Abstand zwischen dem Vorkommen des ersten und zweiten Platzes beträgt demnach ganze 25 %.

Die „Konsistente Gestaltung“ bezieht sich auf die einheitliche Gestaltung von Dokumentation. Wie bereits erwähnt, wird Dokumentation häufig von mehreren Autoren erstellt bzw. mehrere Autoren liefern Inhalt, der dann als Dokumentation zusammengefasst wird. Durch die unterschiedlichen Quellen können Abweichungen in der inhaltlichen und visuellen Gestaltung auftreten. Gründe dafür können unklare Absprachen sowie das Fehlen eines Hauptautors oder eines Konzepts sein. Auch ein inkonsequent umgesetztes Konzept kann zu Unstimmigkeiten führen, die von Nutzern als unangenehm und aufwendig empfunden werden (s. a. Steinhardt et al. 2015; Dusold 2016; Meng et al. 2016; Meng et al. 2017). Aus diesem Grund ist die Konsistenz ein sehr wichtiger Aspekt für die Usability von API-Dokumentation.

Die Umsetzung einer konsistenten Gestaltung verlangt einen Mehraufwand in Form der Erstellung eines Konzepts, Leitfadens etc. Dieser Mehraufwand wird aber durch viele Vorteile im Hinblick auf die Erstellung und Pflege ausgeglichen. Bereits durch die Erstellung anhand eines Konzepts wird Zeit eingespart, da die gleiche Vorgehensweise für alle Kapitel angewendet werden kann. Auf diese Weise wird ein zusätzlicher Zeit- und Arbeitsaufwand für die Neuerstellung jedes einzelnen Themas vermieden. Tatsächlich muss der Autor ohne Konzept bei jedem Kapitel neu überlegen, wie er es aufbauen, gliedern und umsetzen möchte. Dieser Aufwand wird durch ein Konzept vorweggenommen und bringt durch die damit verbundene Konsistenz zusätzlich Vorteile für den Nutzer.

Im Hinblick auf die Pflege ist es wesentlich einfacher, Inhalte in einer konsistenten Dokumentation wiederzufinden und anzupassen. Sind beispielsweise die Kapitel für die Funktionsbeschreibungen alle unterschiedlich aufgebaut, muss der Autor jedes Mal nach dem gewünschten Inhalt suchen, da er an unterschiedlichen Stellen zu finden ist. Dieser Mehraufwand wird durch ein zuvor erstelltes Konzept vermieden.

4.2.2.1.1. Betrachtung nach formalen Kriterien

Die Betrachtung nach formalen Kriterien zeigt teilweise starke Abweichungen von dem sehr hohen Durchschnittswert von 96 %. Vor allem bei mittleren PDFs kommt es mit lediglich 67 % Übereinstimmung zu derart großen Abweichungen.

[74] Siehe Kapitel 3.3 Heuristiken zur Erstellung von API-Dokumentation.

	Webseite	Onepage	PDF	Gesamt
Kurz	83 %	100 %	-	**93 %**
Mittel	100 %	-	67 %	**92 %**
Lang	95 %	100 %	100 %	**97 %**
Gesamt	**95 %**	**100 %**	**83 %**	**Gesamt: 96 %**

Abb. 31: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Ebenfalls auffällig sind kurze Webseiten mit 83 % Übereinstimmung. Onepages, lange PDFs und mittlere Webseiten hingegen sind zu 100 % konsistent. Die Ergebnisse werden unter Betrachtung der einzelnen Kriterien weiter analysiert. Fraglich ist, warum ausgerechnet kurze Webseiten und mittlere PDFs derart große Abweichungen von der durchschnittlichen Übereinstimmung zeigen.

4.2.2.1.2. Betrachtung nach Kriterien

Die Betrachtung nach Kriterien zeigt ein relativ ausgeglichenes Verhältnis in der Verteilung der Vorkommen.

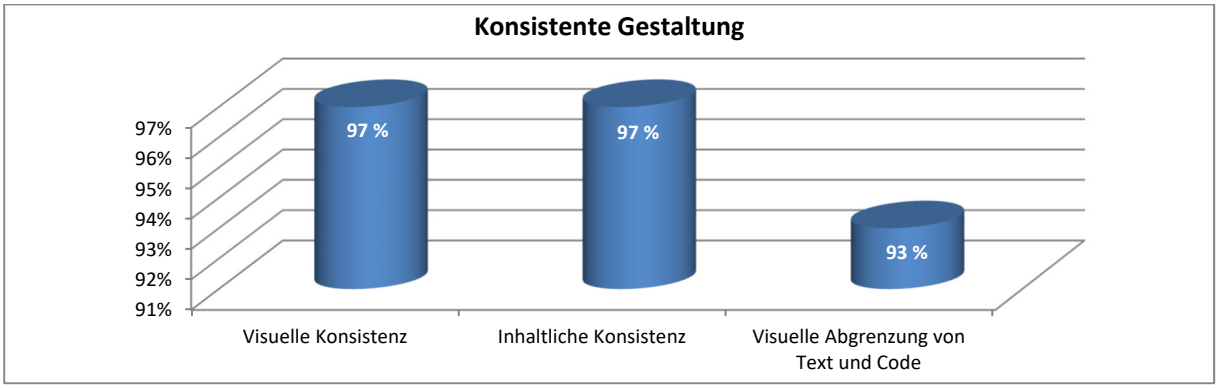
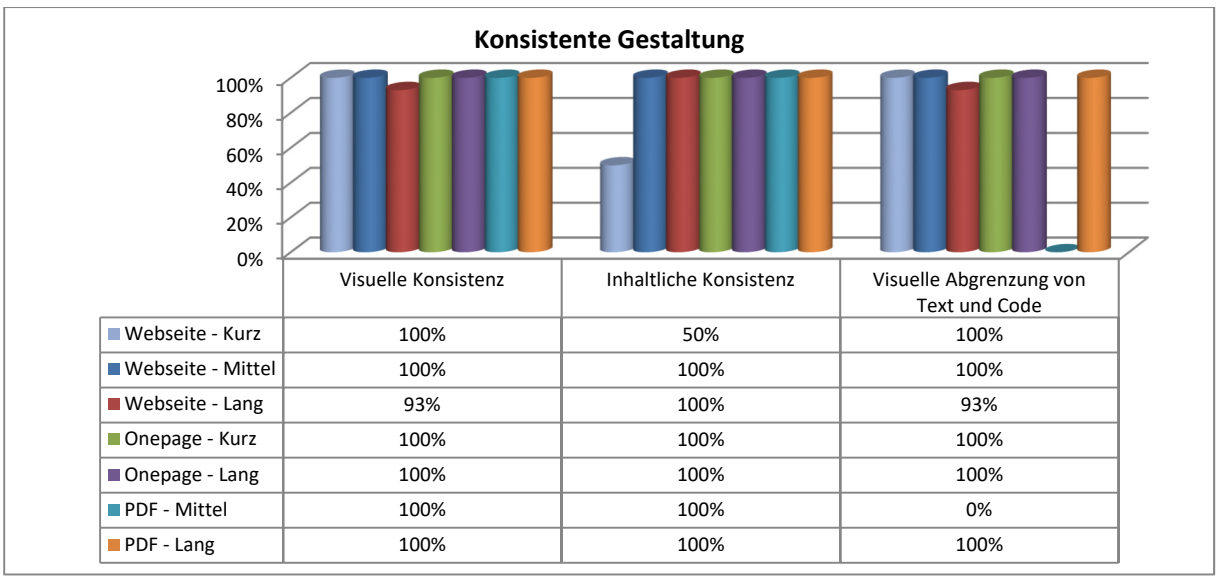


Abb. 32: Gesamtvorkommen der Kriterien für „Ja“

Es gibt kein Kriterium, das auffällig positiv oder negativ ausschlägt, vielmehr weisen alle Kriterien eine sehr hohe Übereinstimmung auf. Das Kriterium „Visuelle Abgrenzung von Text und Code“ hat mit 93 % am wenigsten Übereinstimmungen. Diese Abweichung lässt sich eindeutig an bestimmten Kombinationen der formalen Kriterien festmachen.



	Visuelle Konsistenz	Inhaltliche Konsistenz	Visuelle Abgrenzung von Text und Code
Webseite - Kurz	100%	50%	100%
Webseite - Mittel	100%	100%	100%
Webseite - Lang	93%	100%	93%
Onepage - Kurz	100%	100%	100%
Onepage - Lang	100%	100%	100%
PDF - Mittel	100%	100%	0%
PDF - Lang	100%	100%	100%

Abb. 33: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“[75]

[75] Es ist zu beachten, dass es unter den Forschungsobjekten keine kurzen PDFs und Webseiten sowie keine mittleren Onepages gab. Aus diesem Grund werden diese Kombinationen nicht in den Grafiken dargestellt.

Das Kriterium „Visuelle Konsistenz“ wird von allen Kombinationen bis auf lange Webseiten zu 100 % erfüllt, sie erfüllen es nur zu 93 %. Die Abweichung entsteht durch ein Forschungsobjekt, dass das Kriterium nicht erfüllt (TBYT). Bei der The Budget Your Trip API (TBYT) fällt vor allem die unterschiedliche Gestaltung der Kapitel auf. In sich sind die Kapitel zwar stimmig, die Gesamtbetrachtung der Dokumentation wirkt aber sehr unruhig und inkonsistent. Das liegt an nicht konsequent verwendeten hierarchischen Überschriften, unterschiedlichen Schriftgrößen für ähnliche Inhalte, der vereinzelte Bereitstellung von zusätzlichen Navigationen sowie der sporadischen Anwendung von Grafiken für eine Paneloptik der Überschriften.

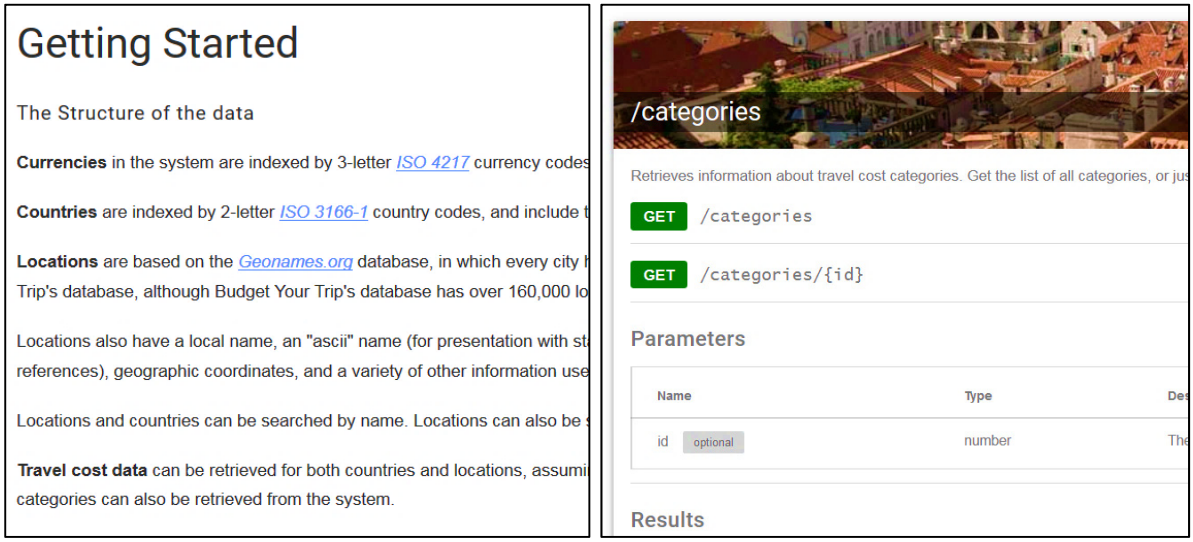


Abb. 34: Inkonsistente visuelle Gestaltung in unterschiedlichen Kapiteln (TBYT)

Bei dem Kriterium „Inhaltliche Konsistenz“ gibt es ebenfalls nur eine Kombination, die keine 100 % Übereinstimmung aufweist. Dabei handelt es sich um eine kurze Webseite (NYTB). Inhaltliche Konsistenz wurde vor allem daran festgemacht, ob die Kapitel aufeinander aufbauen, aufeinander verweisen oder sonstige Verbindungen haben. Das reine Nebeneinanderstehen von Inhalten wurde deshalb mit „Nein“ für dieses Kriterium bewertet. Ein Beispiel dafür ist die Dokumentation der New York Times Books API (NYTB), die aus drei unterschiedlichen Seiten besteht. Die Startseite bildet die „Documentation“, die als Referenzdokumentation der Funktionen zu werten ist. Bei den beiden anderen Seiten handelt es sich um das „Readme“ und die „Console“. Es gibt keine inhaltlichen Verweise zwischen den Seiten und kein inhaltliches aufeinander Aufbauen.

Bei dem Kriterium „Visuelle Abgrenzung von Text und Code“ gibt es zwei Kombinationen aus Umfang und Medium, die es nicht zu 100 % erfüllen. Dabei handelt es sich um lange Webseiten und mittlere PDFs. Für mittlere PDFs gibt es 0 % Übereinstimmung, bei den langen Webseiten gibt es ein Forschungsobjekt, dass das Kriterium nicht erfüllt. Dabei handelt es sich um die Photobucket API (PH), die als klassische Online-Hilfe umgesetzt wurde. Mit dem Begriff Online-Hilfe wird gemeinhin eine Optik ähnlich einem Programm wie HTML Help bezeichnet, mit dem aus einzelnen HTML-Seiten automatisch eine Dokumentation inklusive Navigation, Index usw. erstellt wird.

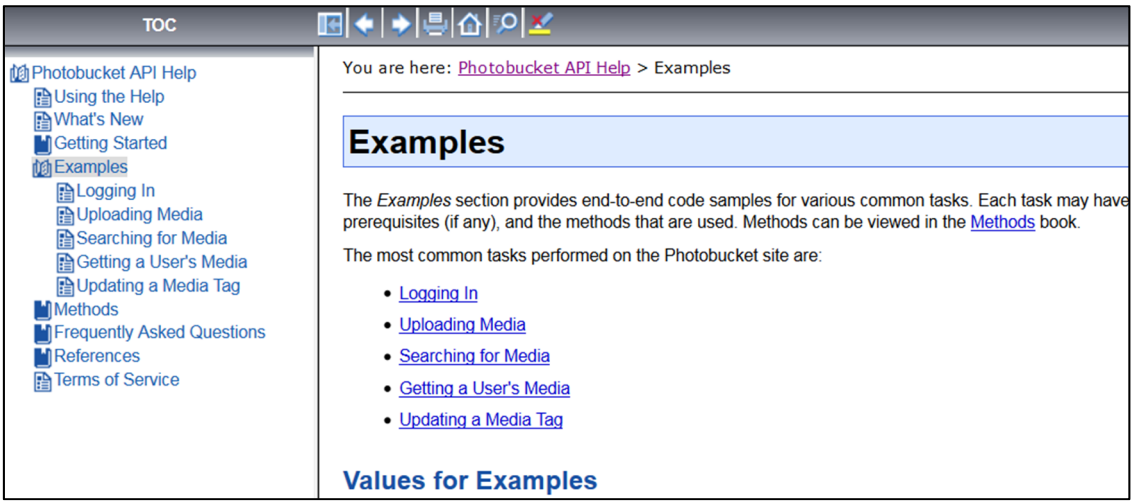


Abb. 35: Klassische Optik einer Online-Hilfe (PH)

Bei den mittleren PDFs ist es die Zazzle API, die keine visuelle Abgrenzung von Text und Code aufweist. Die nachfolgende Abbildung zeigt die fehlenden Abgrenzungen in beiden Forschungsobjekten.

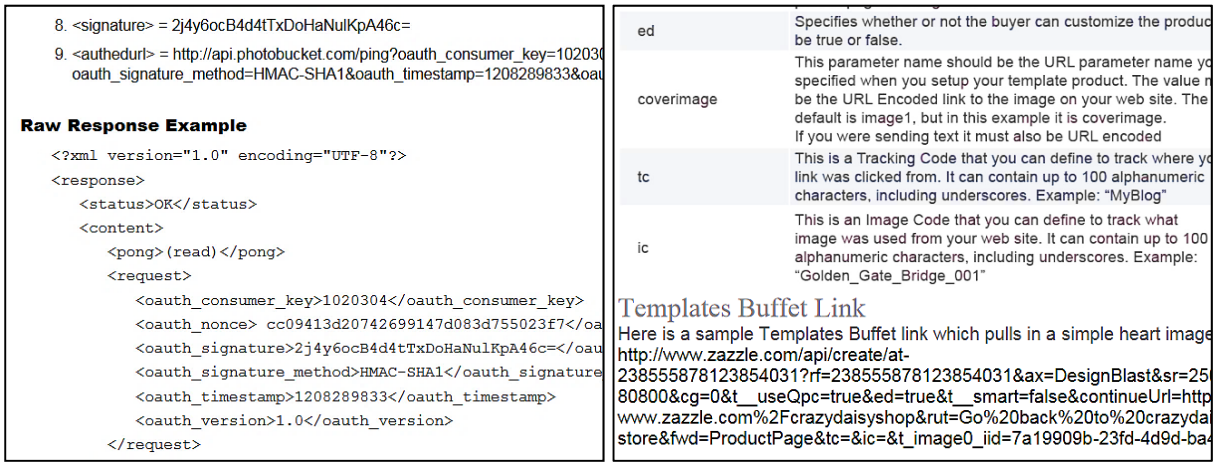


Abb. 36: Fehlende visuelle Abgrenzung von Text und Code (v. l. n. r. PH, Zazzle)

In den anderen Dokumentationen wird die optische Trennung von Text und Code unterschiedlich gelöst. Nachfolgend werden Beispiele für eine erfolgreiche Abgrenzung zum Vergleich gezeigt.

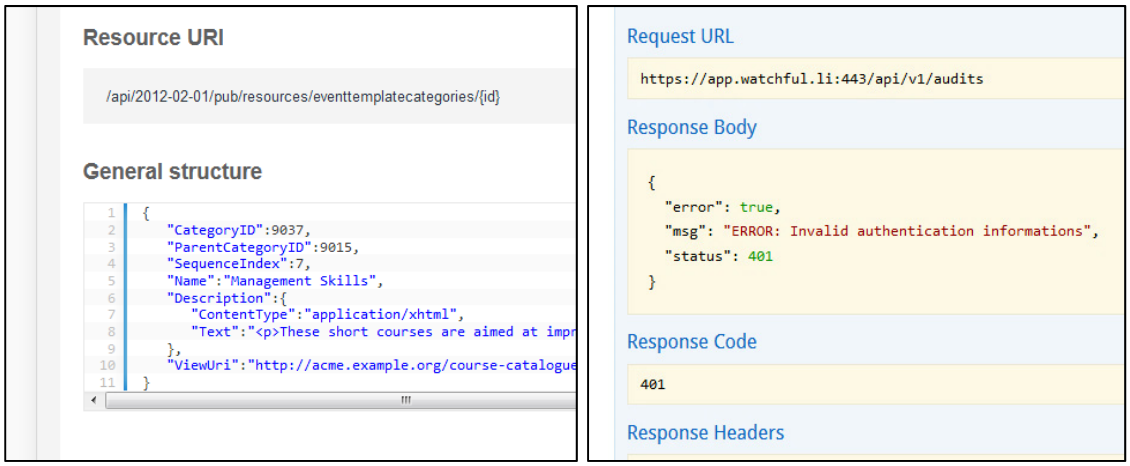


Abb. 37: Erfolgreiche Abgrenzung von Text und Code (v. l. n. r. ARP, WR)

Die erfolgreiche Abgrenzung von Text und Code wurde vor allem durch Farbigkeit, Boxen und/ oder veränderte Typografie erreicht.

4.2.2.1.3. Problematiken

Die Bewertung der visuellen Konsistenz fiel teilweise schwierig und oftmals auch grenzwertig aus. In einem Fall war die visuelle Konsistenz zwar gegeben, allerdings war die Feinstruktur der Kapitel recht unterschiedlich (GMD). Das Äußerte sich durch Videos, Links oder sehr lange Texte, die für ein unruhiges Bild beim Scrollen sorgten, obwohl im Detail die visuelle Konsistenz vorhanden war. In einem anderen Fall war die Dokumentation an sich visuell konsistent, nur ein Kapitel wich durch eine andere Typografie der Überschriften ab (Klipfolio). Neben der reinen Konsistenz spielte außerdem die Frage nach dem Layout und der generellen Typografie eine Rolle. Die durchgehende Verwendung der gleichen Schrift für unterschiedliche Textbausteine wie Überschriften, Fließtexte, Listen usw. war zwar in einem Fall konsistent, aber von der Gestaltung her weder ansprechend noch übersichtlich (Zazzle).

Ähnlich verhielt es sich bei der Bewertung der inhaltlichen Konsistenz. Oftmals bauten die Dokumentationen nur teilweise aufeinander auf oder enthielten nur geringfügig Verweise, so dass vereinzelt Kapitel isoliert nebeneinander standen (GSS, VS, SPR, TMU, Streetlayer, TBYT). Durch das generelle Vorhandensein einer inhaltlichen Konsistenz wurden diese Forschungsobjekte mit „Ja“ für die inhaltliche Konsistenz bewertet.

In Bezug auf die Bewertung der visuellen Abgrenzung von Text und Code gab es keine Problematiken.

4.2.2.2. Grundregeln guter Dokumentation

Die Heuristik „Grundregeln guter Dokumentation“ kommt am zweithäufigsten in den Forschungsobjekten vor, insgesamt wurde sie zu 71 % gefunden. Sie enthält maßgebliche Grundlagen für die Erstellung von Dokumentation, z. B. die umfangreiche Beschreibung aller Funktionen/Parameter, kurze und eindeutige Formulierungen sowie die Kennzeichnung der Version.

4.2.2.2.1. Betrachtung nach formalen Kriterien

Das Vorhandensein der Heuristik könnte in Abhängigkeit von Umfang und Medium der Forschungsobjekte stehen, da sich bei kurzen Medien die Textlänge, Anzahl der Codebeispiele etc. nicht so stark auf die Übersichtlichkeit auswirken wie bei langen Medien. Eine lange Dokumentation wird durch verschachtelte Sätze und unverständliche inhaltliche Bezüge quasi un- oder zumindest schwer lesbar.

	Webseite	Onepage	PDF	Gesamt
Kurz	67 %	33 %	-	**47 %**
Mittel	67 %	-	33 %	**58 %**
Lang	83 %	78 %	33 %	**79 %**
Gesamt	**79 %**	**63 %**	**33 %**	**Gesamt: 71 %**

Abb. 38: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Die Betrachtung nach formalen Kriterien zeigt eine ungleichmäßige Verteilung der Vorkommen auf die verschiedenen Medien und Umfänge. Besonders häufig sind die Grundregeln des guten Dokumentierens in langen Webseiten mit 83 % und langen Onepages mit 78 % zu finden. In etwa im Durchschnitt liegen mittlere und kurze Webseiten mit 67 %, kurze Onepages und PDFs haben nur ein Drittel Übereinstimmungen. Insgesamt haben wie vermutet kurze Forschungsobjekte mit 47 % am wenigsten Vorkommen. In Bezug auf das Medium haben PDFs mit 33 % die Heuristik am wenigsten erfüllt.

4.2.2.2.2. Betrachtung nach Kriterien

Die Betrachtung nach Kriterien spezifiziert die Ergebnisse zusätzlich und zeigt ebenfalls eine ungleichmäßige Verteilung der Vorkommen.

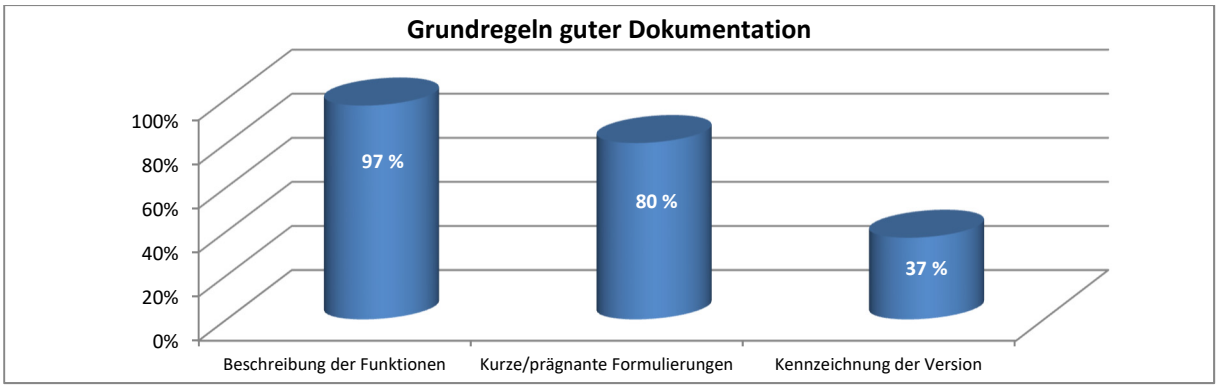
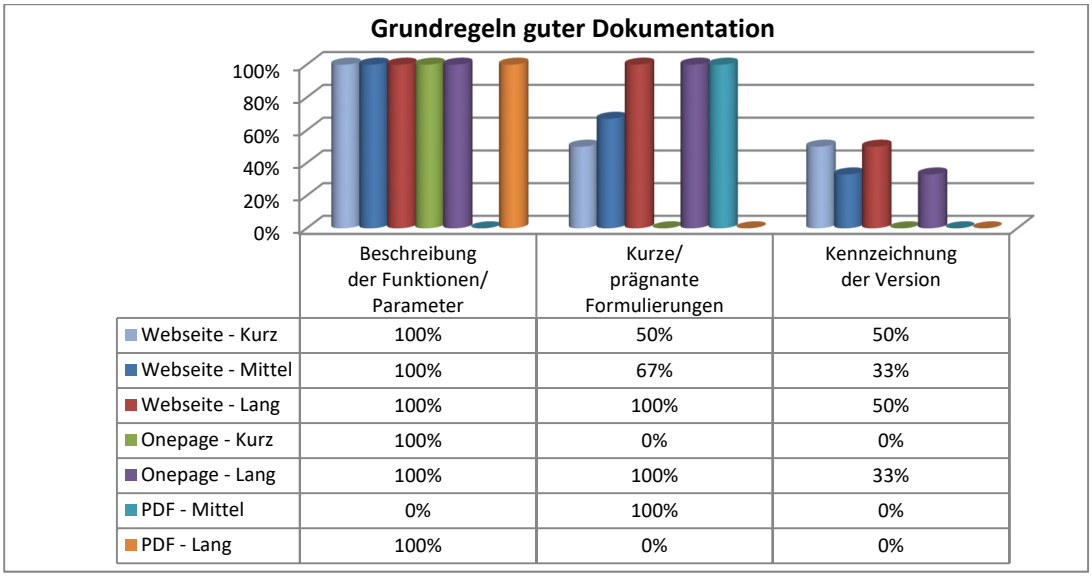


Abb. 39: Gesamtvorkommen der Kriterien für „Ja“

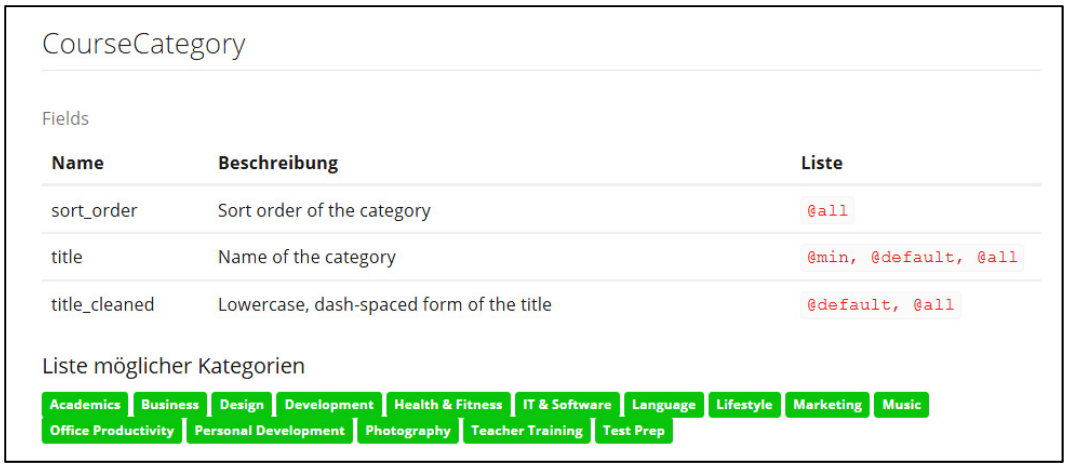
Die Beschreibung der Parameter kommt mit insgesamt 97 % mit großem Abstand am häufigsten vor. Darauf folgen die kurzen/prägnanten Formulierungen mit 80 % und weit abgeschlagen die Kennzeichnung der Version mit 37 %. Die Betrachtung der Kriterien in Abhängigkeit von Medium und Umfang verdeutlicht deren Polarisierung. Vor allem in langen PDFs und kurzen Onepages schwankt das Vorkommen sehr stark. Diese Ergebnisse werden im Folgenden zuerst grafisch visualisiert und anschließend nacheinander durch Beispiele veranschaulicht.



	Beschreibung der Funktionen/ Parameter	Kurze/ prägnante Formulierungen	Kennzeichnung der Version
Webseite - Kurz	100%	50%	50%
Webseite - Mittel	100%	67%	33%
Webseite - Lang	100%	100%	50%
Onepage - Kurz	100%	0%	0%
Onepage - Lang	100%	100%	33%
PDF - Mittel	0%	100%	0%
PDF - Lang	100%	0%	0%

Abb. 40: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Die „Beschreibung der Funktionen, Parameter etc.“ ist in mittleren PDFs gar nicht zu finden (Zazzle), in allen anderen Kombinationen kommt sie zu 100 % vor. Die Zazzle API wurde für den Anwender stark vereinfacht, so dass durch vorgefertigte Templates auf die Funktionen zugegriffen werden kann. Da keine Anfragen formuliert werden müssen, benötigt der Anwender auch keine Parameterbeschreibungen. Bei den anderen Forschungsobjekten kam es zu starken Schwankungen in der Qualität der Beschreibungen.



CourseCategory

Fields

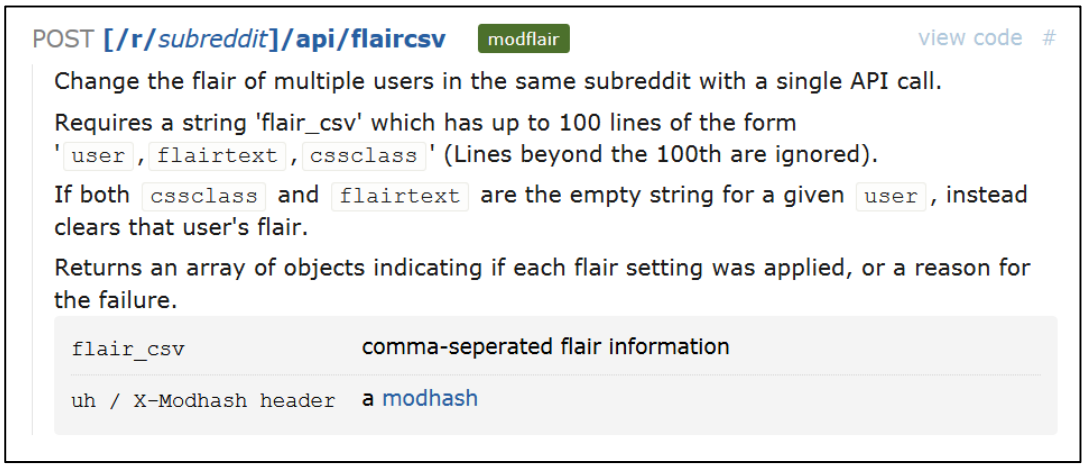
Name	Beschreibung	Liste
sort_order	Sort order of the category	`@all`
title	Name of the category	`@min, @default, @all`
title_cleaned	Lowercase, dash-spaced form of the title	`@default, @all`

Liste möglicher Kategorien

Academics | Business | Design | Development | Health & Fitness | IT & Software | Language | Lifestyle | Marketing | Music | Office Productivity | Personal Development | Photography | Teacher Training | Test Prep

Abb. 41: Umfangreiche Beschreibung der Parameter (Udemy)

In den umfangreichen Beschreibungen wurden nicht nur alle notwendigen Informationen bereitgestellt, sondern auch übersichtlich präsentiert. Bei den notwendigen Informationen handelte es sich meist um die Parameterbezeichnung, eine Erklärung/Beschreibung, Informationen zur Befüllung und Abhängigkeiten usw. Die Darstellung erfolgt am häufigsten in Tabellenform (OFEC, Udemy, GSS, ANXE, BC, Klipfolio, Etsy, SW, PH, VS, ARP, WR, TBYT, MOMR, DCR, HW, SPR). Manche Parameterbeschreibung enthalten zwar alle notwendigen Informationen, wirken aber durch eine ungünstige visuelle Umsetzung unübersichtlich.



POST **[/r/*subreddit*]/api/flaircsv** modflair view code #

Change the flair of multiple users in the same subreddit with a single API call.

Requires a string 'flair_csv' which has up to 100 lines of the form '`user`,`flairtext`,`cssclass`' (Lines beyond the 100th are ignored).

If both `cssclass` and `flairtext` are the empty string for a given `user`, instead clears that user's flair.

Returns an array of objects indicating if each flair setting was applied, or a reason for the failure.

`flair_csv`	comma-seperated flair information
`uh / X-Modhash header`	a modhash

Abb. 42: Unübersichtliche Parameterbeschreibung (Reddit)

Kurze/prägnante Formulierungen sind in langen Webseiten, langen Onepages und mittleren PDFs zu 100 % vertreten. Mittlere Webseiten besitzen sie in 67 % der Fälle, kurze Webseiten nur in 50 %. Kurze Onepages und lange PDFs erfüllen das Kriterium gar nicht. Die Werte bestätigen die Annahme, dass lange Medien eher über dieses Kriterium verfügen als kurze. Ein Ausnahme bildet das lange PDF (TROC). Bei längeren Textpassagen sind die Sätze vereinzelt sehr lang und stark verschachtelt. Die Verschachtelung entsteht vornehmlich durch Aufzählungen im Fließtext, entweder/oder - Beziehungen usw.

1.2.2.2 Topic Tags (DocCat)

Open Calais identifies the topic or topics that are being discussed in the document. For example, “Macroeconomics,” “Equities,” “Sports,” “Entertainment,” “Politics,” “Oil & Gas Products,” “Mergers/Acquisitions/Takeovers,” “Computer Hardware,” “Consumer Financial Services,” “Software and IT Services,” etc. A DocCat (topic) tag is designed to give a general notion of what an input document is about. There is no specific entity recognition in the text, but rather deduction about what the text is about.

The reference list of topics is defined by the Thomson Reuters Coding Schema (TRCS) and/or by the International Press Telecommunications Council (IPTC) news taxonomy and/or by the Self Service Classification project taxonomy.

Each identified topic results in a Topic (DocCat) tag. It is possible that multiple topics will be identified, or that no topic will be identified if the content does not discuss anything currently defined by the relevant taxonomies.

Following are some of the Topic tags that were extracted by Open Calais from the story about the Apple Watch Launch.

Abb. 43: Verschachtelte und lange Sätze (TROC)

Das Kriterium „Kennzeichnung der Version“ ist nur bei vier der sieben Kombinationen vorhanden. Die höchste Übereinstimmung weisen kurze und lange Webseiten mit 50 % auf. Dahinter folgen mittlere Webseiten und lange Onepages mit 33 %. Der geringe Wert für dieses Kriterium könnte sich daraus ergeben, dass ein Teil der APIs über Crowd Development weiterentwickelt wird. Das bedeutet, dass alle Änderungen auf der jeweiligen Plattform für alle Nutzer sichtbar sind. In solchen Fällen existieren im Prinzip keine verschiedenen Versionen, sondern nur der aktuellste Entwicklungsstand. Diese Vorgehensweise würde eine Versionierung dementsprechend überflüssig machen. Trotzdem müssten Code-Beispiele etc. der Dokumentation an den aktuellen Entwicklungsstand angepasst werden. Dafür wären Tools zur automatisierten Erstellung von Dokumentation gut geeignet. Ein Beispiel für ein solches Vorgehen ist die Reddit API, die als reine Referenzdokumentation verfügbar ist. Alle weiteren Informationen, insbesondere konzeptuelle, sind auf Github zu finden.

Etwas mehr als ein Drittel der Forschungsobjekte verfügt über eine Versionierung. Bei zwei Objekten sind sogar verschiedene Versionen der Dokumentation verfügbar (TMU, MOMR). Eine Dokumentation verfügt über ein „Changelog“, das alle Änderungen an der API übersichtlich zusammenfasst und verlinkt (SW).

Web API Change Log

This page lists significant updates and bug fixes to the Web API. In order to keep track of the changes we recommend you to subscribe to the Spotify page on API Changelog, subscribe to our RSS or follow @SpotifyPlatform on Twitter.

SPOTIFY API CHANGELOG
Powered by Hitch.
Jun 5 2017 (Last update) | (Size of update) | 1134 (Followers)
FOLLOW

Search:

DATE	DESCRIPTION	ANNOUNCEMENT
July 26th, 2017	Added endpoint to Upload a Custom Playlist Cover Image	Blog post
May 29th, 2017	Started requiring access tokens across all endpoints.	Tweet, Blog post

Abb. 44: Change Log für Änderungen an der API (SW)

Ansonsten beschränkt sich die Kennzeichnung auf einen Hinweis im Footer, bei der Überschrift oder im Header der Webseite.

4.2.2.2.3. Problematiken

Die Bewertung der Parameterbeschreibung war aufgrund der qualitativen Unterschiede grenzwertig. Für die Bewertung wurde der Schwerpunkt auf das Vorhandensein der Informationen gelegt, nicht auf deren Gestaltung.

Die Angaben für die Versionierung der Dokumentationen waren teilweise schwer zu finden, weil sie im Header/Footer integriert waren oder als Teil einer Überschrift fungierten, so dass sie beim Wechsel der Seite oder durch Scrollen verschwanden (Udemy, ANXE, Klipfolio, TBYT).

4.2.2.3. Transparente Navigation

Die Heuristik „Transparente Navigation" liegt mit 61 % Übereinstimmungen auf dem dritten Platz. Sie bezieht sich vor allem auf die Navigierbarkeit und generelle Übersichtlichkeit der Dokumentationen. Obwohl Kriterien wie „Logischer Aufbau", „Keine Verschachtelung" oder „Mitwandernde Navigationsleiste" in der Dokumentationserstellung grundsätzlich eine Rolle spielen sollten, wurden sie in der Untersuchung nur in etwas mehr als der Hälfte der Fälle erfüllt.

4.2.2.3.1. Betrachtung nach formalen Kriterien

Auch diese Heuristik wird in Abhängigkeit der formalen Kriterien dargestellt. In diesem Fall sind die Abweichungen jedoch relativ gering.

	Webseite	Onepage	PDF	Gesamt
Kurz	90 %	47 %	-	**64 %**
Mittel	73 %	-	40 %	**65 %**
Lang	61 %	60 %	80 %	**68 %**
Gesamt	**66 %**	**56 %**	**60 %**	**Gesamt: 63 %**

Abb. 45: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja"

Am häufigsten kommt die Heuristik in den langen Forschungsobjekten unabhängig vom Medium vor. Der Abstand zu den kurzen und mittleren Medien fällt aber mit unter 5 % sehr gering aus. Eine hohe Übereinstimmung erzielen vor allem kurze Webseiten mit 90 % und lange PDFs mit 80 %. Mittlere Webseiten liegen mit 73 % knapp dahinter, mittlere PDFs haben mit 40 % am wenigsten Übereinstimmungen. Kurze Onepages haben mit 47 % am zweitwenigsten. In der Gesamtwertung des Mediums verfügen Webseiten über am meisten Übereinstimmungen.

Überraschend an den Ergebnissen ist die geringe Übereinstimmung für Webseiten und Onepages. Der durchschnittliche Wert liegt bei 66 % und 56 %, dabei sind Webseiten/Onepages sehr gut als Medium für die Umsetzung der Heuristik geeignet. Dynamische und mitwandernde Navigationen können in Webseiten und Onepages sehr einfach mit Webdesign integriert werden. Im Gegensatz dazu ist eine Umsetzung in PDFs aus technischen Gründen nur eingeschränkt möglich, trotzdem gibt es für sie eine ebenfalls hohe Übereinstimmung mit 60 %.[76]

4.2.2.3.2. Betrachtung nach Kriterien

Wie bereits erwähnt, unterscheiden sich die Kriterien im Hinblick auf die mediale Umsetzbarkeit. Die Betrachtung der Ergebnisse für die einzelnen Kriterien kann darüber zusätzlich Aufschluss liefern.

[76] In PDFs gibt es generell nur die Navigation, die der eingesetzte PDF-Reader anbietet. Da Überschriften etc. aber in dem erstellten Dokument gekennzeichnet und gezielt als Lesezeichen ausgegeben werden müssen, werden diese Navigationen ebenfalls als „mitwandernd" angesehen. Die Umsetzung einer dynamischen Navigation ist in einem PDF nicht möglich.

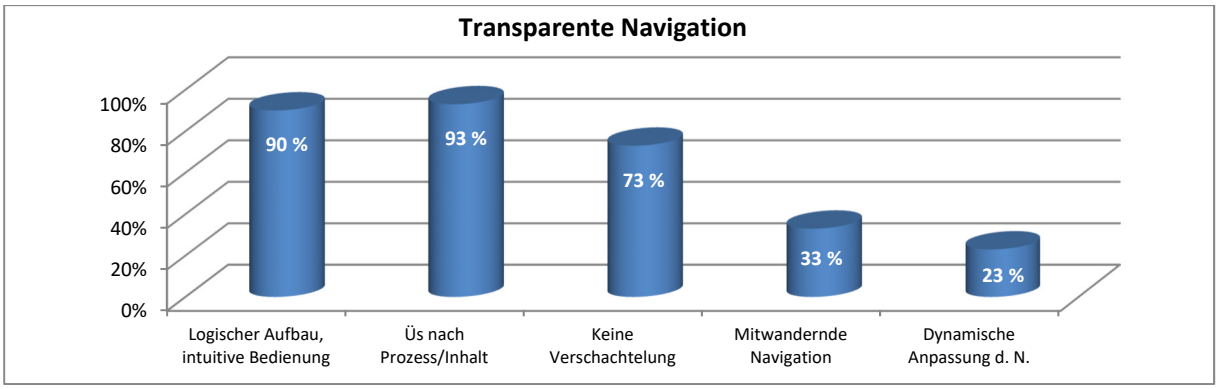
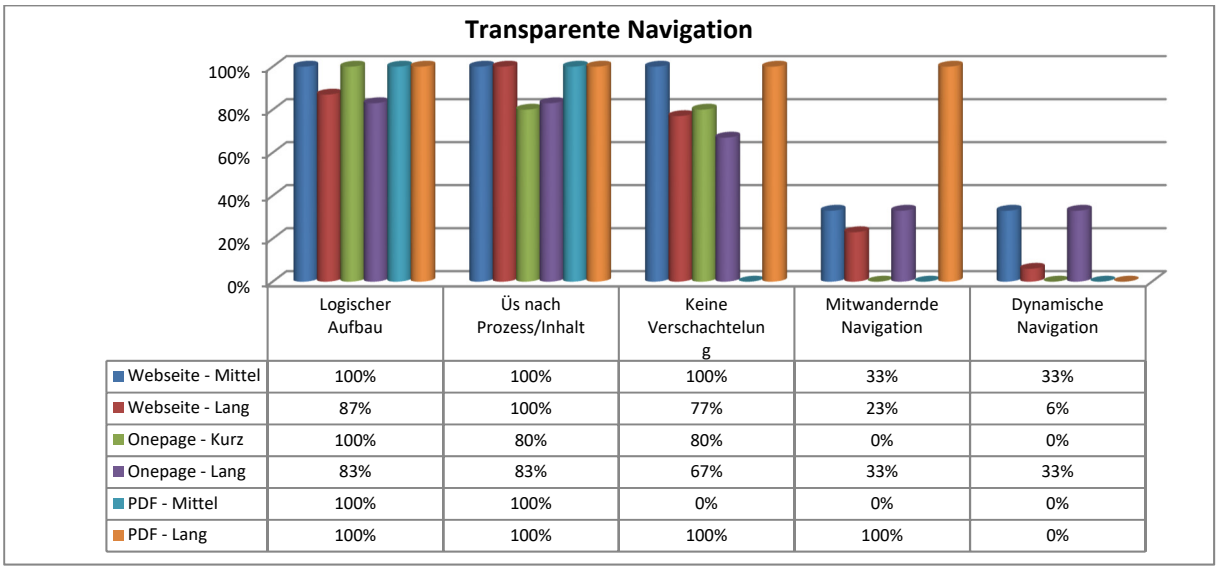


Abb. 46: Gesamtvorkommen der Kriterien für „Ja“

Die ersten drei Kriterien weisen mit großem Abstand die meisten Übereinstimmungen auf. Weit abgeschlagen sind hingegen Kriterien mit Bezug zur Navigation. Nur etwas mehr als ein Drittel der Forschungsobjekte besitzt eine mitwandernde Navigation, während die dynamische Anpassung sogar nur bei 23 % vorhanden ist. Die Darstellung in Beziehung zu den formalen Kriterien verdeutlicht die Unterschiede zusätzlich.



	Logischer Aufbau	Üs nach Prozess/Inhalt	Keine Verschachtelung	Mitwandernde Navigation	Dynamische Navigation
Webseite - Mittel	100%	100%	100%	33%	33%
Webseite - Lang	87%	100%	77%	23%	6%
Onepage - Kurz	100%	80%	80%	0%	0%
Onepage - Lang	83%	83%	67%	33%	33%
PDF - Mittel	100%	100%	0%	0%	0%
PDF - Lang	100%	100%	100%	100%	0%

Abb. 47: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Die Grafik zeigt eine sehr ungleichmäßige Verteilung der Übereinstimmungen, insbesondere für die Kriterien der Navigation. Für die bessere Übersichtlichkeit werden die Kriterien nachfolgend einzeln besprochen.

Die ersten drei Kriterien der Heuristik werden sehr häufig erfüllt, bei dem logischen Aufbau fallen nur lange Onepages (GSS) und lange Webseiten etwas heraus (Flickr, NYTB). Im Fall von Guide Star Search (GSS) erscheint der Aufbau auf den ersten Blick logisch, bei näherem Hinsehen werden aber extrem redundante Inhalte deutlich. Obwohl drei verschiedene Kapitel verfügbar sind, ist der Inhalt der Kapitel zum großen Teil der gleiche.

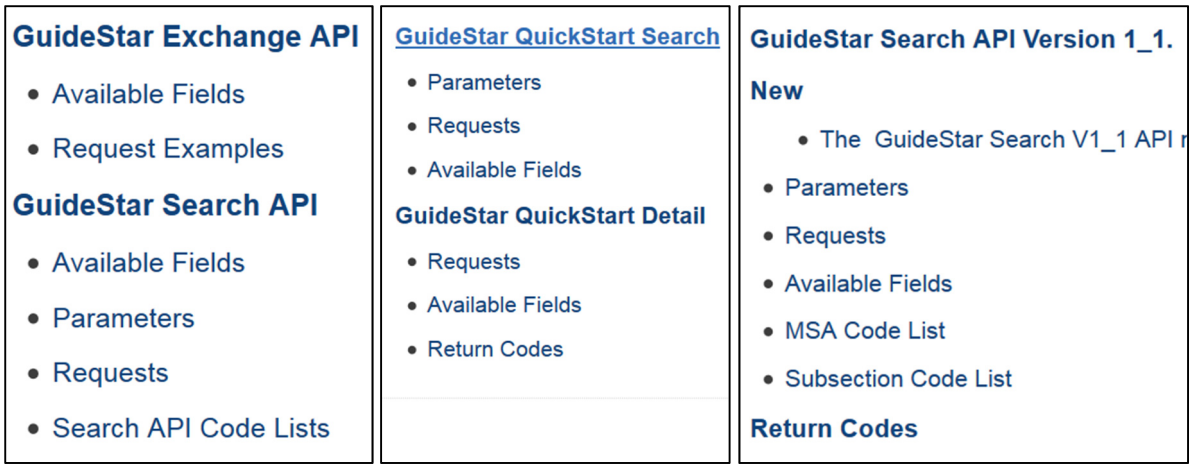


Abb. 48: Menüs der Unterpunkte für Guide Star APIs, Quickstart API und Sandbox APIs (v. l. n. r.; GSS)

Die Qualität eines solchen Vorgehens wie bei dem Forschungsobjekt GSS ist fragwürdig. Bei einem so geringen Mehrwert in den Kapiteln wäre eine inhaltlich durchdachte Zusammenfassung vermutlich leichter für den Nutzer zu handhaben. Auf diese Weise müssten nicht drei verschiedene Menüpfade angeklickt werden, um den gesuchten Inhalt zu finden.

Bei dem Kriterium „Überschriften nach Prozessen/Inhalt“ weisen alle Kombinationen bis auf kurze Onepages 100 % Übereinstimmung auf. Die Ausnahme ist das Forschungsobjekt Visual Search (VS). Wie bereits bei der Gliederung erwähnt, gibt es bei VS bis auf „Summery“[77] keine weiteren Überschriften.

Das Kriterium „Keine Verschachtelung“ weist mehr Differenzen in der Verteilung der Übereinstimmungen auf. Mittlere PDFs erfüllen es zu 0 %, darauf folgen kurze Onepages, lange Webseiten und Onepages mit um die 70 % - 80 %. Mittlere Webseiten und lange PDFs haben 100 % Übereinstimmung. Verschachtelungen bewirken sowohl in der Navigation als auch im eigentlichen Text Unübersichtlichkeit. Ein Beispiel dafür ist das Forschungsobjekt Kaltura.

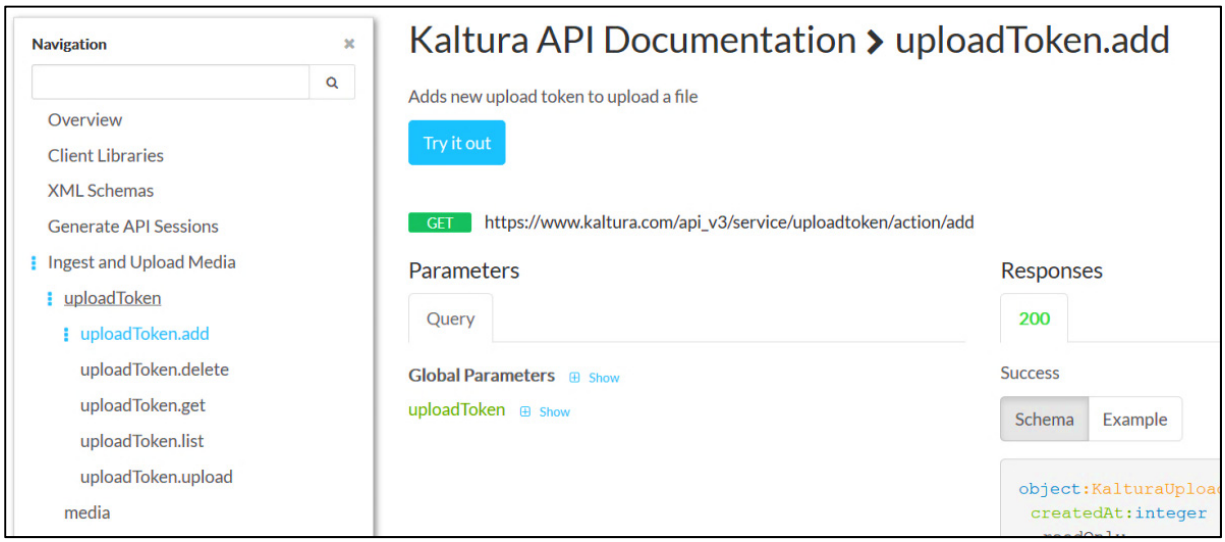


Abb. 49: Unnötige Verschachtelungen in der Navigation und im Text (Kaltura)

[77] Der Rechtschreibfehler wurde original von der Quelle übernommen (vgl. VisualSearchApi 2016).

Bei Kaltura sind die Verschachtelungen besonders schwierig, da es viele interaktive Möglichkeiten zum Aus-/Einklappen, Verlinkungen o. ä. gibt. Zusätzlich kann die Navigation nur durch das Anwählen der Seite geöffnet werden, so dass sich die inaktiven Kapitel und Unterkapitel sofort wieder schließen. Die Navigationspfade, die im Text angeboten werden (Expand, Try it out etc.), sind nicht nachvollziehbar und teilweise Sackgassen.

Die „Mitwandernde Navigationsleiste“ gibt es vor allem bei langen PDFs mit 100 % und langen Webseiten mit 57 % Übereinstimmungen. Knapp ein Drittel Übereinstimmung haben mittlere Webseiten. Am wenigsten Übereinstimmungen weisen lange Webseiten mit 23 % und kurze Onepages mit 20 % auf. Der niedrige Wert für die kurzen Onepages ist nicht überraschend, da diese vermutlich durch kurze Scrollwege auch ohne mitwandernde Navigation noch übersichtlich sind (TS).

In den Fällen, in denen eine Navigationsleiste vorhanden ist, ist diese nicht auch zwingend dynamisch gestaltet. Nur bei langen Onepages (ANXE, CSJ, DCR) und mittleren Webseiten (GMD) sind alle vorhandenen Navigationen dynamisch. Bei langen Webseiten ist nur die Hälfte der Navigationen dynamisch (BC, MOMR).

4.2.2.3.3. Problematiken

Die Bewertung des logischen Aufbaus wurde durch die Auslagerung von Inhalten auf externe Webseiten wie Github erschwert. Diese Dokumentationen waren eine Mischung aus offiziell bereitgestellter und Crowd Documentation (OFEC, Kaltura). Meist wurden gezielt konzeptuelle Informationen oder aktuelle Codebeispiele auf Github ausgelagert (Reddit, ANXE, SL). Github wurde aber auch als Diskussions- und Kommunikationsplattform für Bugs, Probleme etc. genutzt (SW, DCR). Die ausgelagerten Inhalte konnten in die Bewertung nur begrenzt miteinbezogen werden, da die Gliederung, der Aufbau, das Layout etc. durch den Hoster der Webseite vorgegebenen sind.

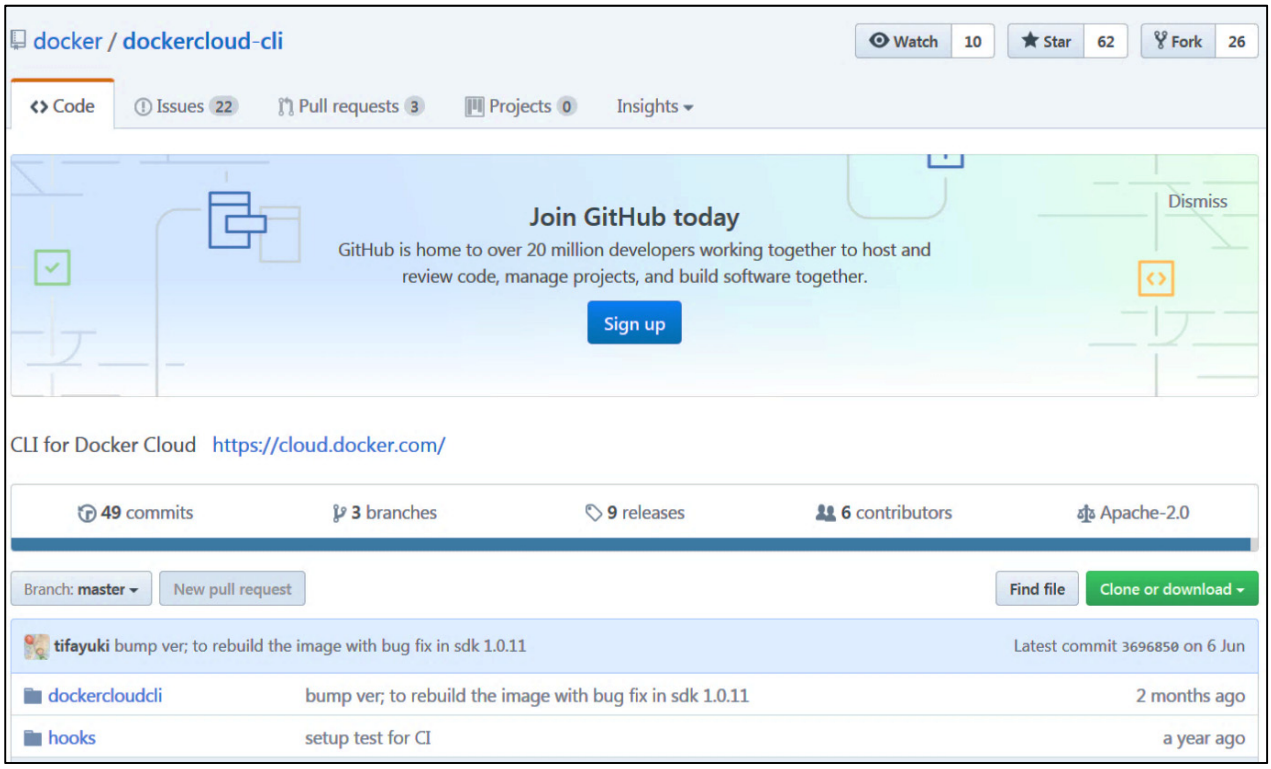


Abb. 50: Vorgegebenes Webdesign für das Docker Cloud Readme auf Github (DCR)

Ebenfalls nicht eindeutig fiel die Bewertung der Überschriften aus. Häufig waren nicht alle Überschriften einer Dokumentation prozess- oder inhaltsbasiert. Das Kriterium wurde mit „Ja“ bewertet, wenn ein Großteil der Überschriften das Kriterium erfüllte.

Abb. 51: Gemischte Überschriften nach Inhalt/Prozessen und nach Textsorte (GMD)

In der Google Maps Directions API (GMD) sind fast alle Überschriften prozess- oder inhaltsbasiert. Eine Ausnahme bildet der „Entwickler-Leitfaden“, bei dem es sich um eine Überschrift nach Textsorte handelt.

4.2.2.4. Struktur

Die Heuristik „Struktur“ wurde zu 57 % in den Forschungsobjekten gefunden, womit Sie insgesamt auf dem vierten Platz liegt. Bei den Kriterien der Heuristik handelte es sich um die Gliederung nach Prozessen und die konsistente Feinstruktur der Kapitel.

4.2.2.4.1. Betrachtung nach formalen Kriterien

Das Vorhandensein der Heuristik könnte von Umfang und Medium der Forschungsobjekte abhängig sein. Es ist vorstellbar, dass gerade für kurze Dokumentationen keine detaillierte Struktur durch den Autor angelegt wird, da häufig der Aufwand als nicht lohnenswert angesehen wird. Gleichzeitig sind bezogen auf das Medium vor allem Webseiten ohne eine Struktur nicht denkbar, da die Navigation der Webseite diese erfordert. Onepages und PDFs können auch ohne Struktur umgesetzt werden, da bei beiden Medien der Inhalt durch Scrollen auf einer Seite ohne zusätzliche Navigation verfügbar ist.

	Webseite	Onepage	PDF	Gesamt
Kurz	50 %	67 %	-	**60 %**
Mittel	50 %	-	100 %	**63 %**
Lang	57 %	67 %	50 %	**57 %**
Gesamt	**55 %**	**67 %**	**75 %**	**Gesamt: 60 %**

Abb. 52: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Die Betrachtung nach formalen Kriterien zeigt, dass die Heuristik „Struktur“ vermehrt in PDFs vorkommt. Die höchste Übereinstimmung erreichen mit 100 % mittlere PDFs, darauf folgen kurze und lange Onepages mit 67 %. Insgesamt erreichen PDFs eine Übereinstimmung von 75 %, was ebenfalls der höchste Wert ist. Den zweithöchsten Wert haben Onepages mit 67 % auch bei unterschiedlichem Umfang, dahinter liegen mit 55 % Webseiten. Bezogen auf den Umfang lässt sich keine klare Aussage treffen, alle Forschungsobjekte erreichen Werte, die deutlich über der Hälfte liegen. Am meisten Übereinstimmungen hat mit geringem Abstand der mittlere Umfang mit 63 %.

Für die Heuristik lassen sich keine Abhängigkeiten in Bezug auf den Umfang feststellen, da alle Werte maximal 10 % auseinander liegen. Im Gegensatz dazu gab es im Hinblick auf das Medium einen klaren Trend für PDFs. Ganz im Gegenteil zu der vorherigen Vermutung, dass PDFs und Onepages eher ohne eine durchdachte Struktur auskommen würden als Webseiten, gab es für diese beiden Medien die meisten Übereinstimmungen. Die Verteilung wird im Hinblick auf die einzelnen Kriterien näher beleuchtet.

4.2.2.4.2. Betrachtung nach Kriterien

Die Verteilung der Übereinstimmungen auf die Kriterien „Gliederung prozessorientiert“ und „Konsistente Feinstruktur“ ist sehr unterschiedlich.

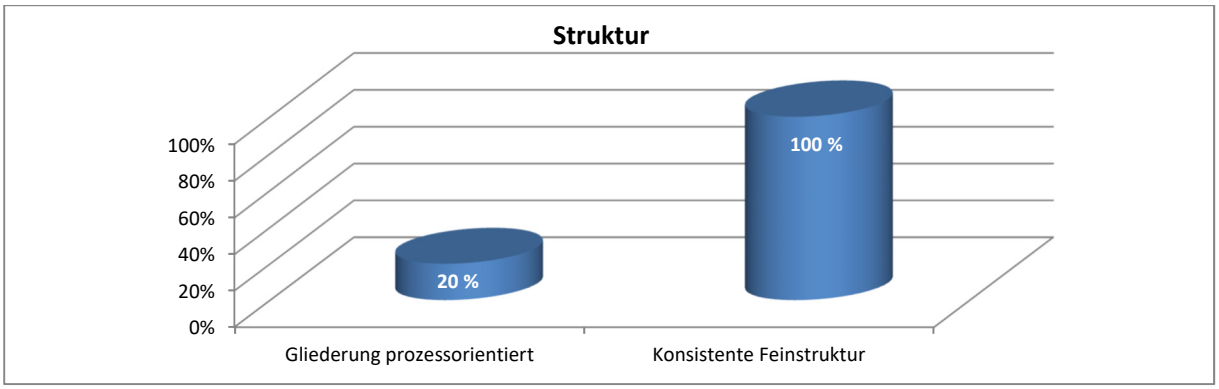
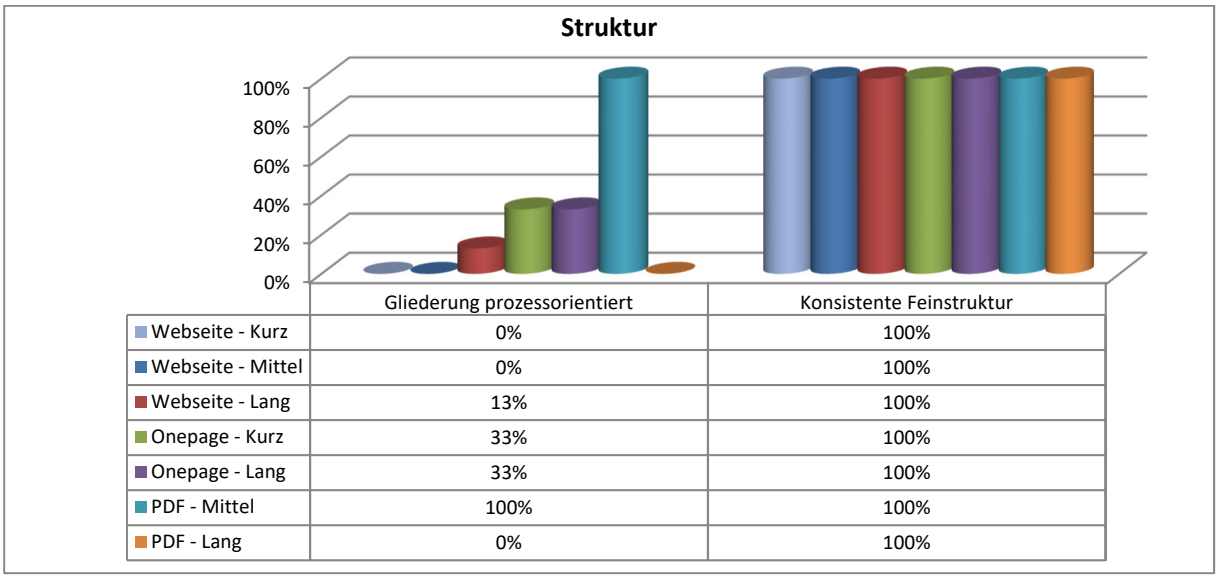


Abb. 53: Gesamtvorkommen der Kriterien für „Ja“

Die konsistente Feinstruktur wird zu 100 % in den Forschungsobjekten erfüllt, während die prozessorientierte Gliederung nur bei 20 % vorhanden ist. Die extreme Abweichung der Kriterien wird unter Einbeziehung von Umfang und Medium näher betrachtet.



	Gliederung prozessorientiert	Konsistente Feinstruktur
Webseite - Kurz	0%	100%
Webseite - Mittel	0%	100%
Webseite - Lang	13%	100%
Onepage - Kurz	33%	100%
Onepage - Lang	33%	100%
PDF - Mittel	100%	100%
PDF - Lang	0%	100%

Abb. 54: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Das Kriterium „Prozessorientierte Gliederung“ ist am häufigsten mit 100 % bei mittleren PDFs vorhanden, darauf folgen lange und kurze Onepages mit 33 %. Lange Webseiten haben nur in 13 % der Fälle eine prozessorientierte Gliederung. Mittlere Webseiten und lange PDFs erfüllen dieses Kriterium gar nicht. Das Kriterium „Konsistente Feinstruktur“ kommt in allen Kombinationen zu 100 % vor.

Als konsistente Feinstruktur wurde der gleichbleibende Aufbau von Kapiteln mit ähnlicher Textsorte gewertet. Kapitel mit beschreibenden/anleitenden Inhalten wie eine Referenzdokumentation können einen anderen Aufbau oder andere Elemente (Tabellen, Listen, Abbildungen) erfordern als Kapitel mit konzeptuellem Hintergrund. Diese Varianz wurde in die Bewertung miteinbezogen, so dass die Konsistenz von konzeptuellen und codeorientierten Kapiteln unterschieden wurde. Zusätzlich wurde vor allem der generelle Aufbau inklusive Hierarchie und allgemeiner Gestaltung bewertet.

Das Kriterium der prozessorientierten Gliederung lag vor, wenn die Kapitel der Dokumentation nicht nach Textsorten wie Referenzdokumentation, Getting-Started, Developers Guide etc., sondern anhand von konkreten Inhalten und Prozessen gestaltet war.

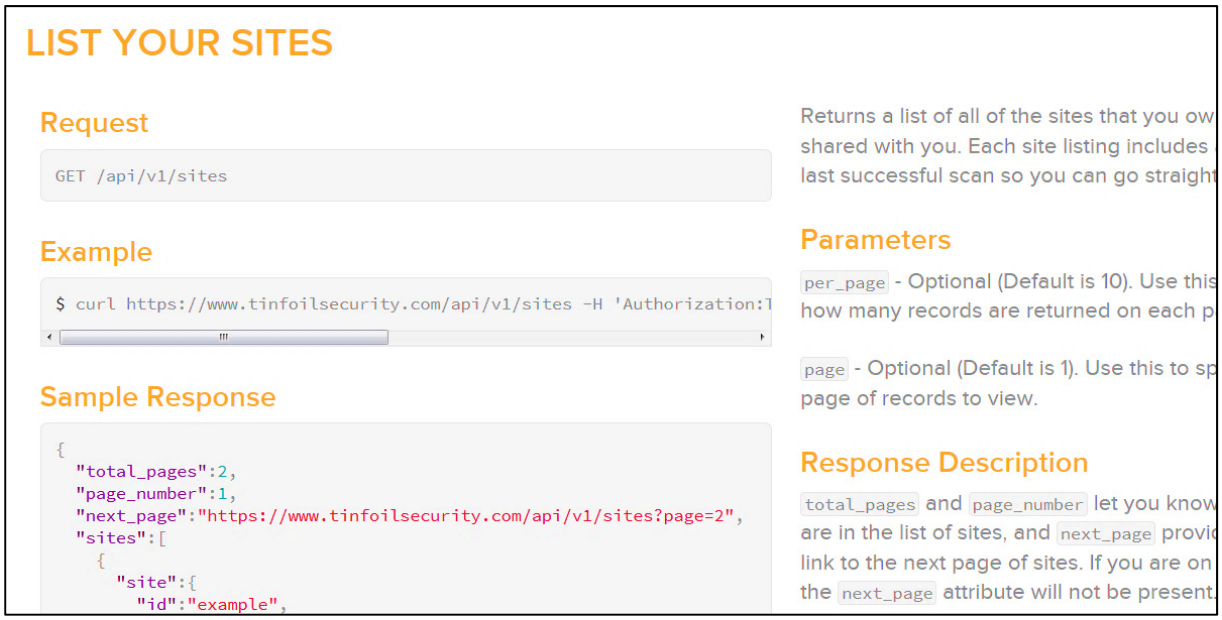


Abb. 55: Prozessorientierte Gliederung auf Onepages ohne Navigation (TS)

Die Onepage von Tinfoil Security (TS) ist anhand von Kapitelüberschriften gegliedert, die nicht noch einmal separat in einer Navigation angezeigt werden.[78] Die Überschriften beschreiben den Prozess, der mit der Funktion ausgeführt werden kann. Im Gegensatz dazu ist die Onepage von Reddit zu betrachten, die über eine Navigation in Form einer Auflistung aller Funktionen verfügt.

[78] Die kritische Betrachtung der Navigationsmöglichkeiten in den Forschungsobjekten erfolgt in der Heuristik „Transparente Navigation“.

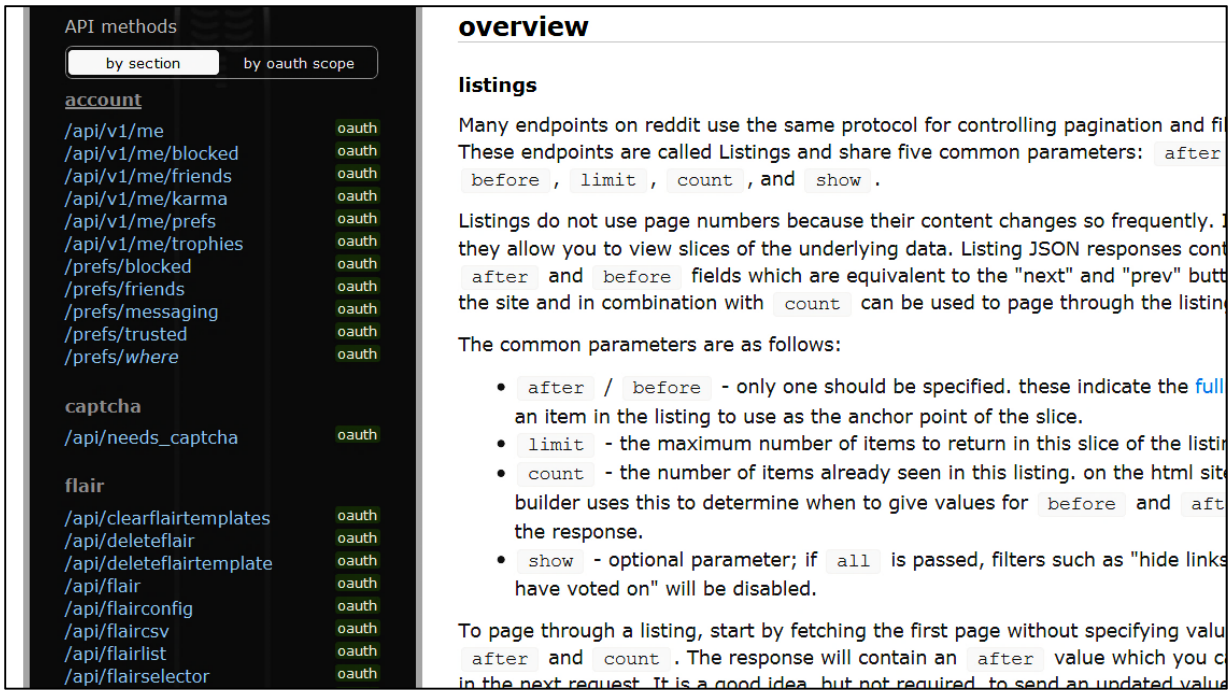


Abb. 56: Prozessorientierte Gliederung auf Onepages mit Navigation (Reddit)

Die Funktionen sind nach Themen wie „Account“, „Likes and Comments“ usw. geordnet. Die Prozesse stehen demnach bei beiden Beispielen der Gliederung im Vordergrund, obwohl sie unterschiedlich umgesetzt wurden.

Bei vielen Forschungsobjekten lag eine Gliederung nach Textsorten vor, die für Nutzer im Vergleich meist schwerer zu verstehen ist. Häufig werden unterschiedliche Bezeichnungen für ähnliche oder sogar gleiche Textsorten verwendet, so dass Verwirrung entstehen kann. Im nachfolgenden Beispiel wurde die Problematik der Gliederung nach Textsorten durch kurze beschreibende Texte gelöst.

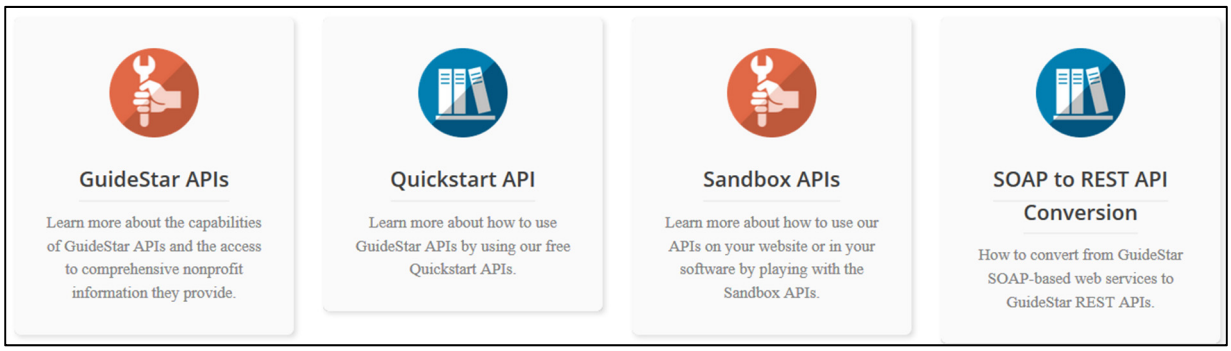


Abb. 57: Gliederung nach Textsorten (GSS)

Erklärende Texte werden auch in anderen Forschungsobjekten genutzt, um die Gliederung nach Textsorten anschaulicher zu machen (SW, TBYT).

4.2.2.4.3. Problematiken

In einigen Fällen wurden die Gliederungsmöglichkeiten miteinander vermischt. Häufig wurden konzeptuelle Informationen eher in Textsortenform präsentiert (Einleitung, User Guide, Tutorial, Quick Start, Development Guide usw.), während codebasierte Informationen zur Anwendung der API prozessual gegliedert waren. Bei diesen Fällen wurde die prozessorientierte Gliederung mit „Nein“ beantwortet.

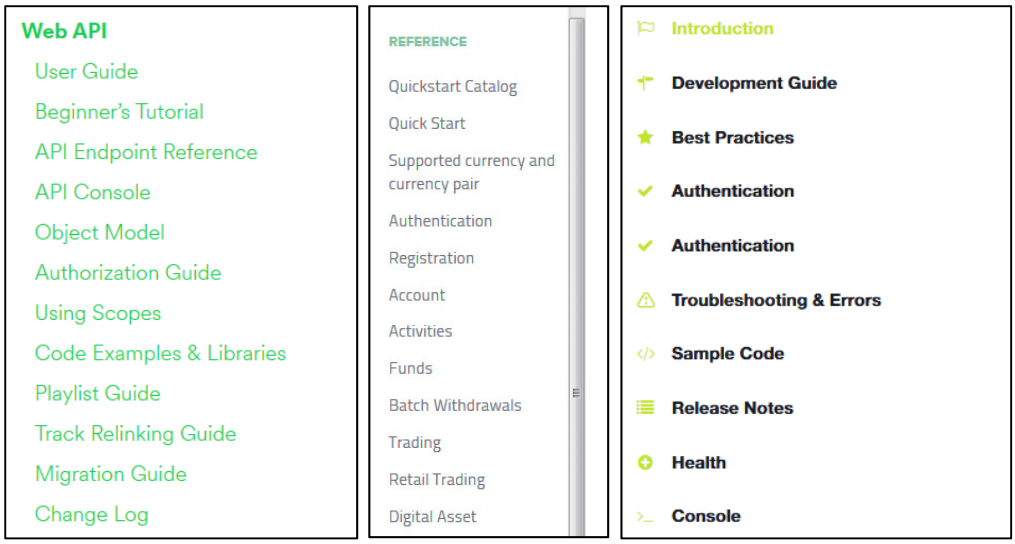


Abb. 58: Vermischung der Gliederungsmöglichkeiten (v. l. n. r. SW, ANXE, Oanda)

Ebenfalls problematisch in der Bewertung waren Dokumentationen, bei denen Inhalte auf andere Webseiten wie Github ausgelagert wurden. Die prozessorientierte Gliederung wurde bei diesen Fällen nur mit „Ja" beantwortet, wenn sie auf der Hauptseite der Dokumentation vorhanden war. Die Gliederung anderer Anbieter wurde nicht berücksichtigt, da sie meist durch die Hoster vorgegeben wird.

Bei einer Dokumentation war sogar gar keine Gliederung vorhanden oder durch Überschriften erkennbar (VS).

4.2.2.5. Schneller Einstieg

Die Heuristik „Schneller Einstieg" weist insgesamt 50 % Vorkommen in den Forschungsobjekten auf. Damit belegt sie den fünften Platz. Sie bezieht sich vor allem auf die Möglichkeit einen schnellen Zugang zu der Dokumentation zu finden. Kriterien waren ein Überblick, ein Getting-Started sowie die übersichtliche Darstellung besonders relevanter Inhalte. Bei diesen Inhalten handelt es sich um die Fähigkeiten der API, grundlegendes Prozess- und Hintergrundwissen, Voraussetzungen und die wichtigsten Anfragen.

4.2.2.5.1. Betrachtung nach formalen Kriterien

Gerade bei umfangreichen Dokumentationen bietet sich die zusätzliche Bereitstellung von spezifischen Informationen an. Auf diese Weise kann der Nutzer schnell einen Überblick gewinnen, ohne eine Masse von Text lesen oder lange nach dem richtigen Kapitel suchen zu müssen.

	Webseite	Onepage	PDF	Gesamt
Kurz	17 %	28 %	-	**23 %**
Mittel	61 %	-	50 %	**58 %**
Lang	64 %	33 %	50 %	**54 %**
Gesamt	**80 %**	**31 %**	**50 %**	**Gesamt: 50 %**

Abb. 59: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja"

Die Betrachtung der Ergebnisse in Abhängigkeit von den formalen Kriterien bestätigt die Annahme, dass eher lange Medien und vor allem lange Webseiten die Heuristik „Schneller Einstieg" erfüllen. Mit 64 % weisen lange Webseiten das höchste Vorkommen auf, dahinter folgen mittlere Webseiten mit 61 %. Kurze Webseiten und kurze Onepages haben sehr viel weniger Übereinstimmungen, allerdings auch lange Onepages. PDFs erfüllen die Heuristik jeweils zu 50 %, unabhängig von ihrem Umfang.

Insgesamt liegen Webseiten mit 80 % Vorkommen eindeutig vorne, Onepages weisen mit 30 % am wenigsten Vorkommen auf. In Bezug auf den Umfang liegen mittlere Forschungsobjekte mit 58 % knapp vor langen mit 54 %. Der kurze Umfang weist mit Abstand am wenigsten Übereinstimmungen mit der Heuristik auf. Das könnte daran liegen, dass der Mehrwert eine schnellen Einstiegs in einer kurzen Dokumentation nicht zwangsläufig gegeben wäre. Der Mehrwert bezieht sich in diesem Fall auf die Einsparung bzw. kompakte Präsentation von Informationen, die potenziell nicht sehr groß bei ohnehin kurzen Dokumentationen ausfallen würde.

4.2.2.5.2. Betrachtung nach Kriterien

Es gibt zwei Kriterien der Heuristik, die etwas nach oben und zwei Kriterien, die etwas nach unten ausschlagen.

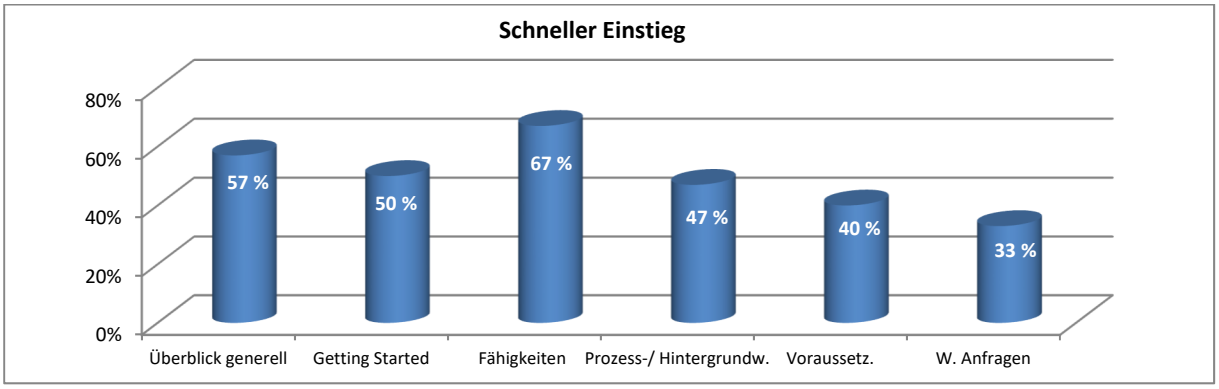
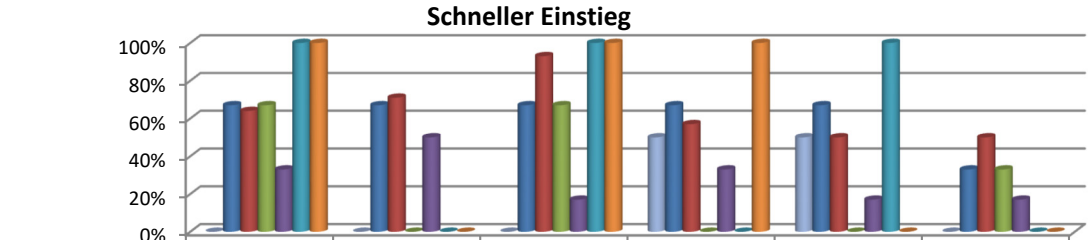


Abb. 60: Gesamtvorkommen der Kriterien für „Ja“

Mit 67 % sind die „Fähigkeiten der API“ besonders häufig vertreten. Ebenfalls oft vertreten ist der „Überblick generell“ mit 57 %. Eher selten vorhanden sind die „Wichtigsten Anfragen“ mit 33 % und die „Voraussetzungen der API“ mit 40 %. Die Kriterien „Getting-Started“ und „Prozess- und Hintergrundwissen“ liegen bei etwa 50 %.

Der große Unterschied im Vorkommen der Kriterien „Fähigkeiten der API“ und „wichtigste Anfragen“ ist überraschend, da sie sich inhaltlich sehr nah sind. Unter Fähigkeiten wurden Angaben zu dem möglichen Einsatzgebiet der API, den möglichen Aufgaben usw. gezählt. Theoretisch könnten die wichtigsten Anfragen aus den Fähigkeiten abgeleitet und so ohne großen Mehraufwand bereitgestellt werden.

Die Betrachtung des Vorkommens der Kriterien in Abhängigkeit von Medium und Umfang verdeutlicht die insgesamt sehr ungleichmäßige Verteilung der Übereinstimmungen. Die Kriterien werden nachfolgend einzeln besprochen.



	Überblick generell	Getting Started	Fähigkeiten	Prozess-/ Hintergrundw.	Voraussetz.	W. Anfragen
Webseite - Kurz	0%	0%	0%	50%	50%	0%
Webseite - Mittel	67%	67%	67%	67%	67%	33%
Webseite - Lang	64%	71%	93%	57%	50%	50%
Onepage - Kurz	67%	0%	67%	0%	0%	33%
Onepage - Lang	33%	50%	17%	33%	17%	17%
PDF - Mittel	100%	0%	100%	0%	100%	0%
PDF - Lang	100%	0%	100%	100%	0%	0%

Abb. 61: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Das Kriterium „Überblick generell“ ist zu 100 % in PDFs vertreten, außerdem zwischen 60 % und 70 % in mittleren und langen Webseiten sowie kurzen Onepages. In langen Onepages ist es zu 33 %, in kurzen Webseiten gar nicht vorhanden.

Der Überblick ist eine Zusammenfassung von z. B. dem Ziel/Zweck der API, wichtigen Features, Preisen, Lizenzen etc. Die Kriterien „Voraussetzungen der API“, „Fähigkeiten der API“ und „Wichtigste/häufigste Anfragen“ können ebenfalls Teil des Überblicks sein. Ein Überblick kann auf unterschiedlichen Wegen umgesetzt werden, z. B. als Grafik, Text, Tabelle, Video oder als Mischung aus diesen Elementen. In den meisten Forschungsobjekten wurde er als Text umgesetzt.

OpenFEC API (Beta) Documentation

This API allows you to explore the way candidates and committees fund their campaigns.

The FEC API is a RESTful web service supporting full-text and field-specific searches on FEC data. Bulk downloads are available on the current site. Information is tied to the underlying forms by file ID and image ID. Data is updated nightly.

There is a lot of data, but a good place to start is to use search to find interesting candidates and committees. Then, you can use their IDs to find report or line item details with the other endpoints. If you are interested in individual donors, check out contributor information in schedule_a.

Get an API key here. That will enable you to place up to 120 requests per minute. Each call is limited to 100 results per page. You can email questions or comments to 18f-fec@gsa.gov. You can also ask questions and discuss the data in the FEC data Google Group. API changes will also be added to this group in advance of the change.

The model definitions and schema are available at /swagger. This is useful for making wrappers and exploring the data.

A few restrictions limit the way you can use FEC data. For example, you can't use contributor lists for commercial purposes or to solicit donations. Learn more here.

Abb. 62: Mögliche Umsetzung eines Überblicks (OFEC)

Das Kriterium „Getting-Started“ war am häufigsten in langen Webseiten mit 71 % vorhanden, darauf folgen mittlere Webseiten mit 67 % und lange Onepages mit 50 %. In Forschungsobjekten mit kurzem Umfang gab es kein Getting-Started. Getting-Started Dokumente liefern vor allem Informationen für einen schnellen Produktivstand der API. Dazu gehören neben Voraussetzungen auch allgemeine Informationen über Anfragen, die Autorisierung usw.

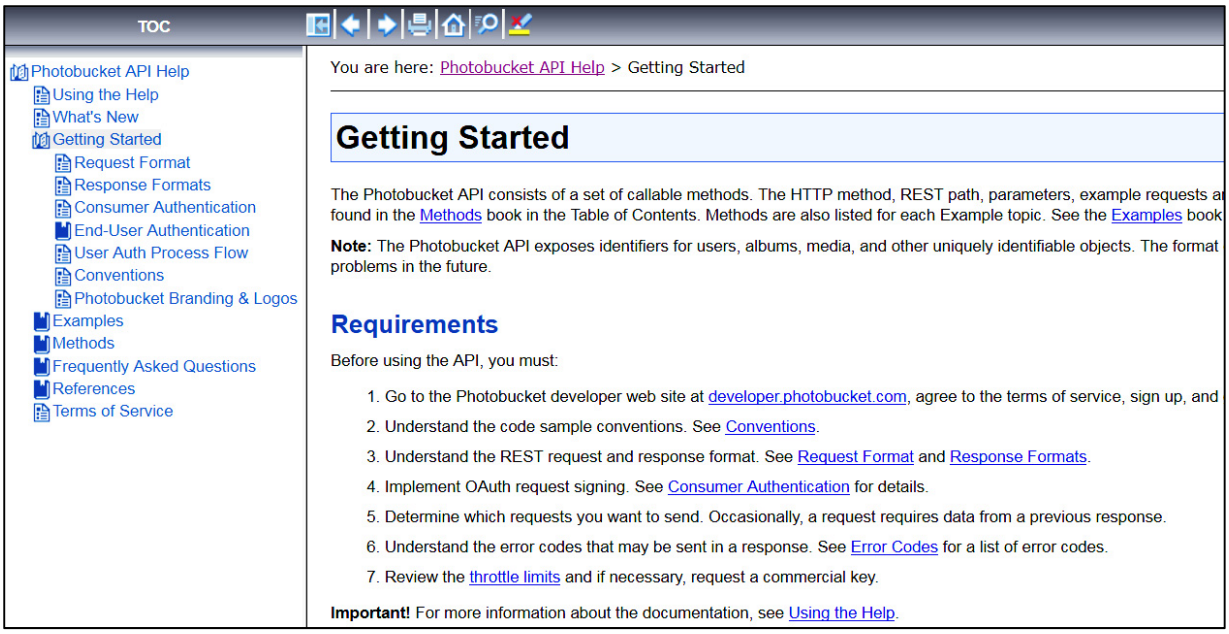


Abb. 63: Beispiel für ein Getting-Started (PH)

Getting-Started Dokumente wurden unterschiedlich umgesetzt, einerseits in Kapitelform wie in der Abbildung oben (PH, Etsy, Reddit, CJS), andererseits als Linkliste (Flickr, ANXE). Der Vorteil einer Linkliste besteht in der direkten Weiterleitung in das jeweilige Kapitel.

Das Kriterium „Fähigkeiten der API“ war ebenfalls nicht in allen Forschungsobjekten vorhanden. Eigentlich handelt es sich bei diesem Kriterium um das Minimum an Informationen, dass ein Entwickler für die Entscheidung über den Einsatz einer API benötigt. Ohne Informationen zu den Fähigkeiten kann er lediglich die Funktionen durchsehen und von diesen auf die Fähigkeiten schließen. Dieses Vorgehen ist jedoch umständlich und als nicht anwenderfreundlich einzustufen.

Besonders häufig waren die Fähigkeiten der API in mittleren und langen PDFs mit 100 % und in langen Webseiten mit 93 % sowie kurzen Onepages zu 67 % vorhanden. In kurzen Webseiten waren sie gar nicht vorhanden, in langen Onepages zu 17 %. Die Verteilung der Vorkommen lässt keinen Rückschluss auf eine Abhängigkeit des Kriteriums mit dem Medium oder dem Umfang zu.

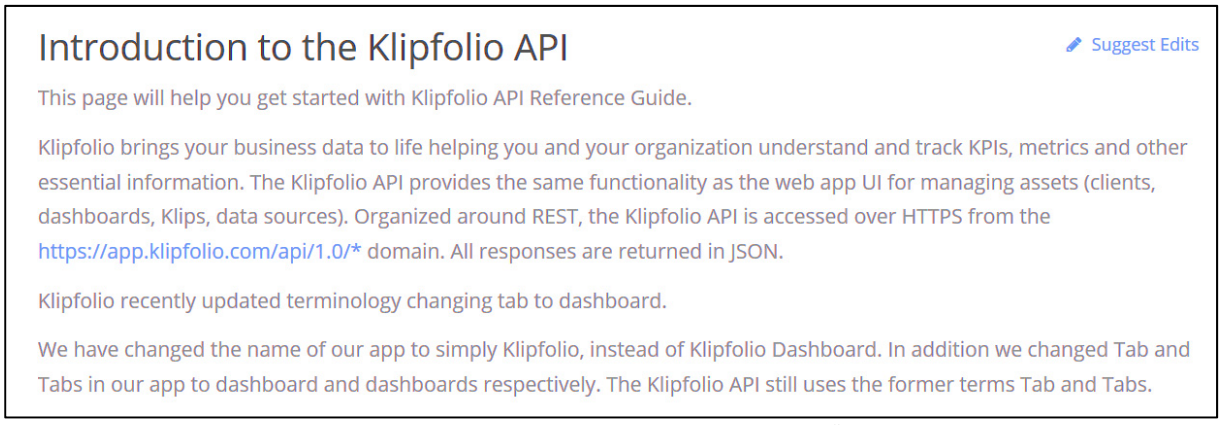


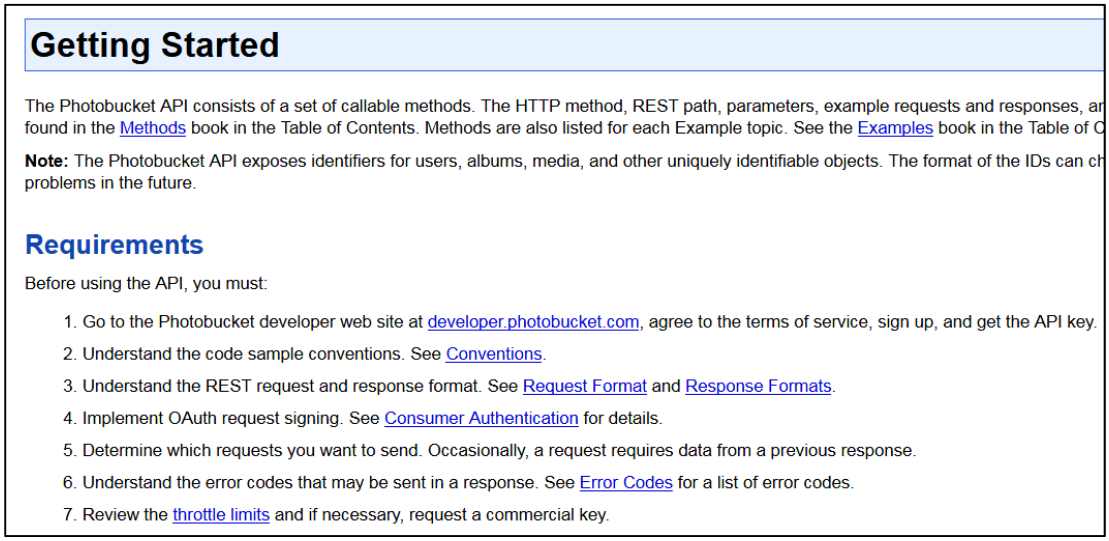
Abb. 64: Beispiel für die Umsetzung des Kriteriums „Fähigkeiten der API“ (Klipfolio)

Das Kriterium „Grundlegendes Prozess- und Hintergrundwissen“ konnte ebenfalls als Teil des Überblicks auftreten. Besonders häufig war es bei langen PDFs (TROC) und besonders wenig bei langen Onepages vorhanden. Bei Webseiten war das Kriterium durchschnittlich zu ca. 50 %

vorhanden, wobei der mittlere und lange Umfang eine höhere Übereinstimmung hatten. Kurze Onepages hatten gar kein Vorkommen, ebenso wie mittlere PDFs.

Die Tendenz bei diesem Kriterium geht dahin, dass in Forschungsobjekten mit längerem Umfang eher Prozess- und Hintergrundwissen vorhanden war. Das kann mindestens zwei Ursachen haben. Zum einen besteht die Möglichkeit, dass Forschungsobjekte mit geringem Umfang nicht so komplex sind, als dass Prozess- und Hintergrundwissen benötigt werden. Zum anderen könnte ebenfalls die Herangehensweise des Autors entscheidend sein: In manchen Dokumentationen war umfassendes Hintergrundwissen vorhanden, das nicht direkt zu der API gehörte. Die Bereitstellung dieser Inhalte bedeutet einen Mehraufwand für den Autor und hängt somit von seiner Motivation ab. Neigt ein Autor ohnehin zu sehr kurzen Formulierungen und gibt nur aus seiner Sicht absolut notwendige Inhalte weiter, wird er tendenziell eher kein zusätzliches Prozess- und Hintergrundwissen liefern. Ein Beispiel dafür waren Erklärungen zu REST-APIs, die als externe Verlinkung oder als integraler Teil der Dokumentation umgesetzt wurden.

Das Vorhandensein des Kriteriums „Voraussetzungen der API“ konnte ebenfalls als Teil des Überblicks auftreten. Sein Vorkommen war sehr unterschiedlich. In mittleren PDFs kam es mit 100 % am meisten vor, am wenigsten in langen Onepages mit 17 %. In kurzen und langen Webseiten war es zu 50 %, in mittleren Webseiten zu 67 % vorhanden. In den anderen Kombinationen kam es nicht vor. Die Voraussetzungen sind besonders wichtig für die schnelle Einschätzung von zu beschaffenden Mitteln, z. B. technischen Voraussetzungen, Wissen usw.



Getting Started

The Photobucket API consists of a set of callable methods. The HTTP method, REST path, parameters, example requests and responses, ar
found in the Methods book in the Table of Contents. Methods are also listed for each Example topic. See the Examples book in the Table of C

Note: The Photobucket API exposes identifiers for users, albums, media, and other uniquely identifiable objects. The format of the IDs can ch
problems in the future.

Requirements

Before using the API, you must:

1. Go to the Photobucket developer web site at developer.photobucket.com, agree to the terms of service, sign up, and get the API key.
2. Understand the code sample conventions. See Conventions.
3. Understand the REST request and response format. See Request Format and Response Formats.
4. Implement OAuth request signing. See Consumer Authentication for details.
5. Determine which requests you want to send. Occasionally, a request requires data from a previous response.
6. Understand the error codes that may be sent in a response. See Error Codes for a list of error codes.
7. Review the throttle limits and if necessary, request a commercial key.

Abb. 65: Beispiel für die Umsetzung des Kriteriums „Voraussetzungen der API“ (PH)

Bei den gefundenen Voraussetzungen handelte es sich meist um eine Mischung aus Regeln des Anbieters und technischen Voraussetzungen. In vielen Fällen gab es laut Dokumentation aber keine spezifischen Voraussetzungen für die Nutzung der API.

Das Kriterium „Wichtigste/häufigste Anfragen an die API“ ist insgesamt wenig vertreten. Nur in vier von sieben Kombinationen tritt es überhaupt auf. Am häufigsten ist es mit 50 % in langen Webseiten vorhanden, darauf folgen mittlere Webseiten und kurze Onepages. Am wenigsten ist es in langen Onepages mit 17 % zu finden.

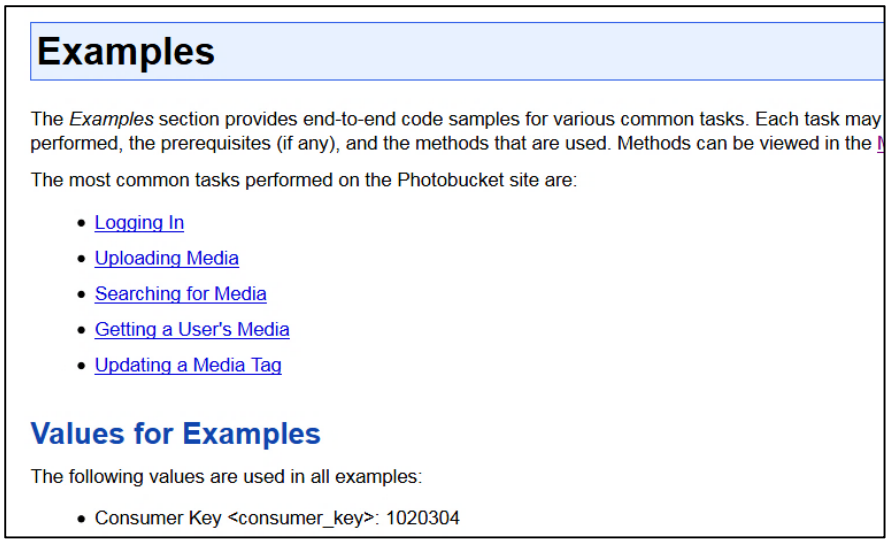


Abb. 66: Beispiel für „Wichtige/häufigste Anfragen der API“ (PH)

In dem Forschungsobjekt Photobucket API (PH) wurden die häufigsten Fähigkeiten als Linkliste umgesetzt. Auf diese Weise kann der Nutzer direkt zu der passenden Stelle in der Dokumentation springen.

4.2.2.5.3. Problematiken

Der Überblick war in einem Fall nicht vor den anderen Inhalten platziert, sondern dazwischen versteckt (Etsy). Außerdem kam es vor, dass die Inhalte des Überblicks auf viele Kapitel verstreut oder in einem nicht eindeutig benannten Kapitel platziert wurden. In diesem Fall konnten die einzelnen Kriterien zwar gewertet werden, es stand aber kein Überblick an sich zur Verfügung (TROC, ARP, MOMR). In einem Fall wurden die Informationen des Überblicks auf Github ausgelagert (Reddit).

Bei dem Getting-Started traten ähnliche Probleme in der Bewertung auf. Teilweise waren die Inhalte auf externe Seiten ausgelagert (OFEC), die Kapitel nicht eindeutig benannt oder Kapitel wurden mit anderen Inhalten vermischt (Udemy, ANXE, Flickr, Klipfolio, Etsy, Reddit, SW, Oanda, MOMR, DCR, HW, CJS).

Die Wertung der Fähigkeiten der API war häufig grenzwertig, da diese nur beiläufig innerhalb eines Satzes erwähnt wurden (OFEC, GSS, BC, Klipfolio, OS). Bei Fällen, die mit „Nein“ bewertet wurden, wurden oftmals Kenntnisse über das Einsatzgebiet der API vorausgesetzt. Beispielsweise bei der Open Secret API (OS) wird darauf hingewiesen, dass die API auf alle Funktionen der gleichnamigen Webseite zugreifen kann.

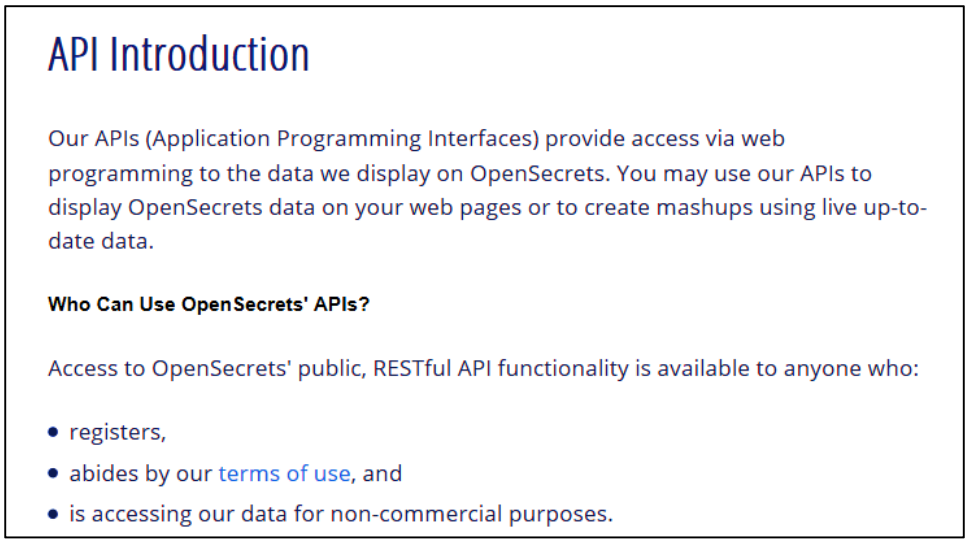


Abb. 67: Fähigkeiten der API werden durch das Produkt definiert (OS)

Der Autor setzt dementsprechend voraus, dass die Anwendung, App oder Webseite dem Nutzer bekannt ist.

4.2.2.6. Selektiver Zugriff auf konzeptuelle Informationen

Der selektive Zugriff auf konzeptuelle Informationen kam zu 29 % in den Forschungsobjekten vor. Die Heuristik belegt damit den drittletzten Platz. Ein genereller Grund für das eher geringe Vorkommen könnte der Mehraufwand in der Umsetzung der Heuristik sein. Sie verlangt die detaillierte Unterscheidung von konzeptuellen und codeorientierten Informationen, die Identifizierung von Verbindungen und Abhängigkeiten dieser Informationen sowie die Erstellung von Konzepten für die Umsetzung einer getrennten Darstellung. Dieses Vorgehen benötigt neben einem finanziellen Aufwand zusätzlich Fachwissen und Motivation.

4.2.2.6.1. Betrachtung nach formalen Kriterien

Ein weiterer Faktor für die Abwägung der Implementierung könnte der Umfang der Dokumentation sein. Bei kurzen Forschungsobjekten müssten eher keine Information separat zugänglich gemacht werden, da beiden Lernansätzen die Auseinandersetzung mit kurzen Passagen anderer Lernstrategien zugemutet werden kann.

Die Verteilung der Übereinstimmungen auf die Forschungsobjekte nach formalen Kriterien ist relativ gleichmäßig. Auffällig ist, dass PDFs keine Kriterien der Heuristik aufweisen, obwohl sie technisch gesehen in ihnen umgesetzt werden könnten.

	Webseite	Onepage	PDF	Gesamt
Kurz	50 %	25 %	-	**35 %**
Mittel	17 %	-	0 %	**13 %**
Lang	32 %	25 %	0 %	**29 %**
Gesamt	**32 %**	**25 %**	**0 %**	**Gesamt: 28 %**

Abb. 68: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Die meisten Vorkommen der Heuristik gibt es mit 50 % in kurzen Webseiten. Am wenigsten Übereinstimmungen haben mittlere Webseiten mit 17 %. Die anderen Kombinationen bewegen sich im Rahmen des Durchschnitts. Insgesamt erfüllen Webseiten die Heuristik am häufigsten, knapp dahinter liegen Onepages. Bezogen auf den Umfang gibt es in kurzen Forschungsobjekten mit 35 % am meisten Übereinstimmungen, mittlere Forschungsobjekte haben mit 13 % am wenigsten.

4.2.2.6.2. Betrachtung nach Kriterien

Am häufigsten vertreten ist das Kriterium „Navigationsstrang Gesamtverständnis“, das bei 40 % der untersuchten Forschungsobjekte vorkommt. Der „Navigationsstrang Problemlösung“ hingegen ist nur in knapp einem Drittel der Fälle vorhanden.

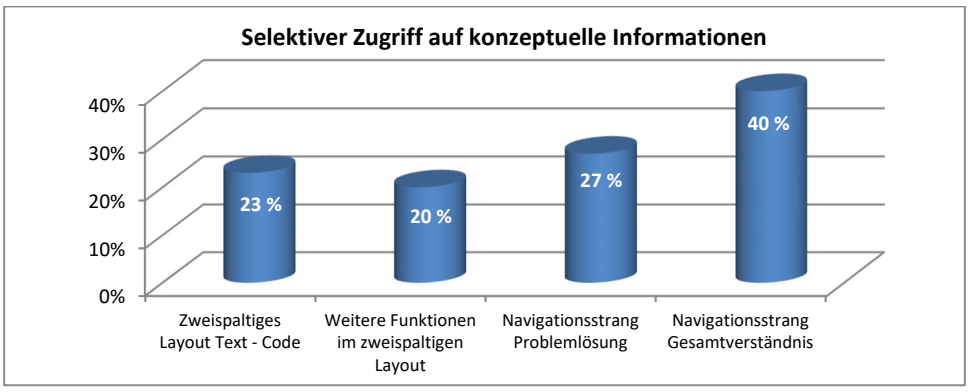
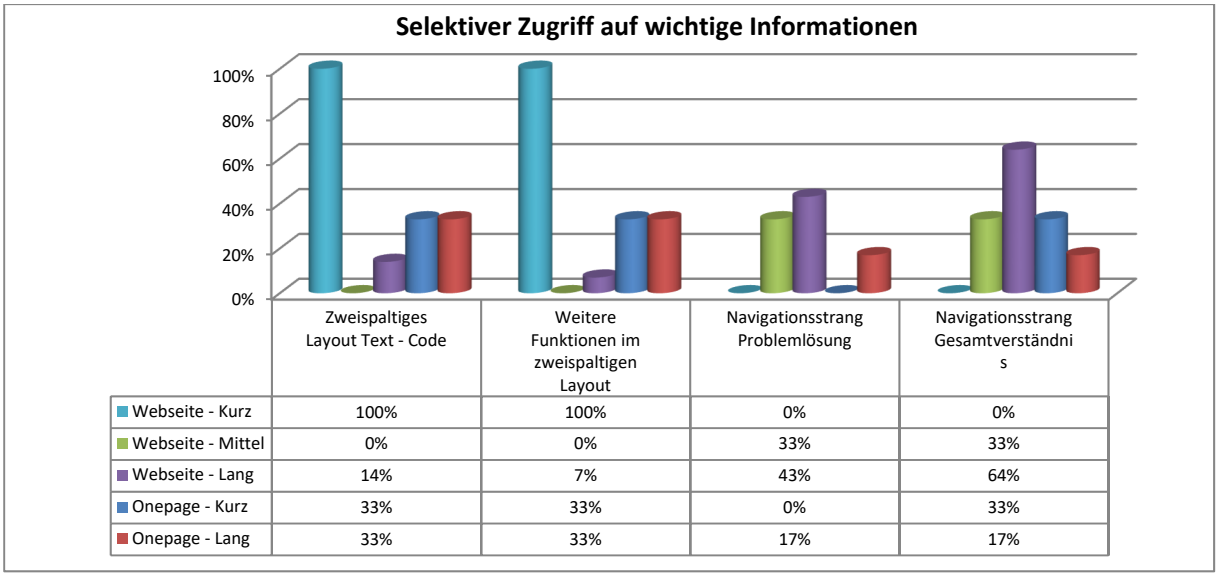


Abb. 69: Gesamtvorkommen der Kriterien für „Ja“

Eine Ursache für das geringere Vorkommen des „Navigationsstrangs Problemlösung“ könnte Referenzdokumentation sein, die in jeder API-Dokumentation vorhanden ist. Viele Entwickler sehen Referenzdokumentation als Werkzeugkasten, in dem alles Notwendige für die Erstellung von Anfragen vorhanden ist. Aus diesem Grund ist es naheliegend, dass konzeptuelle Informationen separat ausgewiesen werden, Inhalte für die Problemlösung aber nicht. Der vorgeschlagene „Navigationsstrang Problemlösung“ soll aber über die Funktion einer Referenzdokumentation hinaus gezielt Ansätze für das schnelle Auffinden von Inhalten bieten. In sehr umfangreichen Referenzdokumentationen kann die Suche nach einem bestimmten Anwendungsfall sehr viel Zeit in Anspruch nehmen, z. B. wenn der Name der Funktion nicht bekannt ist oder diese nicht deskriptiv benannt sind.

Das zweispaltige Layout ist bei knapp einem Viertel der Forschungsobjekte vorhanden gewesen. Angesichts der unterschiedlichen Medien der Umsetzung ist dieser Wert sehr hoch, da ein zweispaltiges Layout in PDFs und auf Webseiten eher selten zu finden ist. Tatsächlich waren nur zwei von den sieben betroffenen Forschungsobjekten Webseiten (Oanda, CJS), bei dem Rest handelte es sich um Onepages. Die Zusammenhänge zwischen Medium, Umfang und Kriterium werden in der folgenden Grafik detailliert dargestellt.[79]

[79] In der Grafik werden nur die Medien inklusive Umfang angezeigt, die Übereinstimmungen mit den Kriterien aufweisen. Die anderen Kombinationsmöglichkeiten wurden aus Platzgründen weggelassen, da sie den Wert „0“ gehabt hätten.



	Zweispaltiges Layout Text - Code	Weitere Funktionen im zweispaltigen Layout	Navigationsstrang Problemlösung	Navigationsstrang Gesamtverständnis
Webseite - Kurz	100%	100%	0%	0%
Webseite - Mittel	0%	0%	33%	33%
Webseite - Lang	14%	7%	43%	64%
Onepage - Kurz	33%	33%	0%	33%
Onepage - Lang	33%	33%	17%	17%

Abb. 70: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Die Verteilung der Vorkommen auf die einzelnen Kriterien in Abhängigkeit von Medium und Umfang ist sehr unterschiedlich. Beispielsweise kommen in kurzen Webseiten das zweispaltige Layout und weitere Funktionen zu 100 % vor, dafür gibt es keine Navigationsstränge. Auch bei kurzen Onepages gibt es eher keine Navigationsstränge. Mittlere Webseiten hingegen weisen kein zweispaltiges Layout auf, haben dafür aber in einem Drittel der Fälle Navigationsstränge. Nur lange Onepages und lange Webseiten erfüllen alle Kriterien der Heuristik, wobei das Maß der Übereinstimmungen auch hier sehr unterschiedlich ist. Lange Onepages haben zu 33 % ein zweispaltiges Layout inklusive weiterer Funktionen, die Navigationsstränge sind nur zu 17 % vertreten. Bei den langen Webseiten ist die Verteilung genau andersherum, das zweispaltige Layout ist nur zu 14 % und weitere Funktionen sind nur zu 7 % vertreten. Die Navigationsstränge sind dahingegen zu 43 % für die Problemlösung und zu 64 % für das Gesamtverständnis vorhanden.

Das zweispaltige Layout wurde in den meisten Fällen durch ein simples zweispaltiges Webdesign umgesetzt, nur in einem Fall wurde das Codebeispiel dynamisch zu dem gescrollten Kapitel generiert (ANXE). Bei ANXE konnte nur die Textspalte gescrollt werden. Die Codespalte scrollte nicht wie in anderen Dokumentationen mit, sondern generierte per Klick auf eine Funktion das passende Beispiel. Auf diese Weise können Text und Code-Beispiel unabhängig voneinander betrachtet werden, so dass im Text gesucht werden kann, ohne das Beispiel zu verlieren.

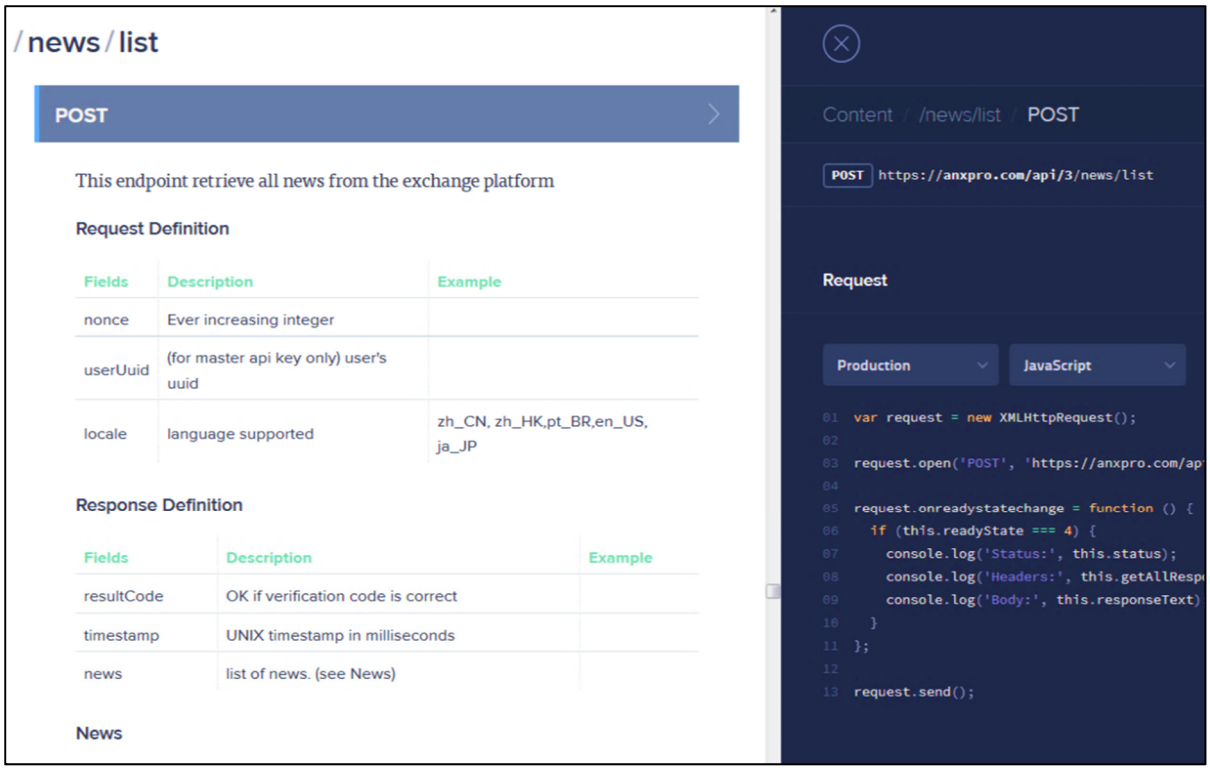


Abb. 71: Zweispaltiges Layout (ANXE)

Mit Ausnahme eines Forschungsobjekts waren in allen Fällen eines „Zweispaltigen Layouts“ auch „Weitere Funktionen im zweispaltigen Layout“ vorhanden. Die Ausnahme war das Forschungsobjekt Oanda, bei dem die Navigationsleiste beim Scrollen nicht mitwanderte. Dadurch konnten die weiteren Funktionen im zweispaltigen Layout nicht bewertet werden.

Die Navigationsstränge wurden auf Webseiten unterschiedlich angeboten. Teilweise waren sie eher in Texte oder längere Auflistungen integriert, in einigen Fällen waren sie aber auch deutlich herausgestellt.

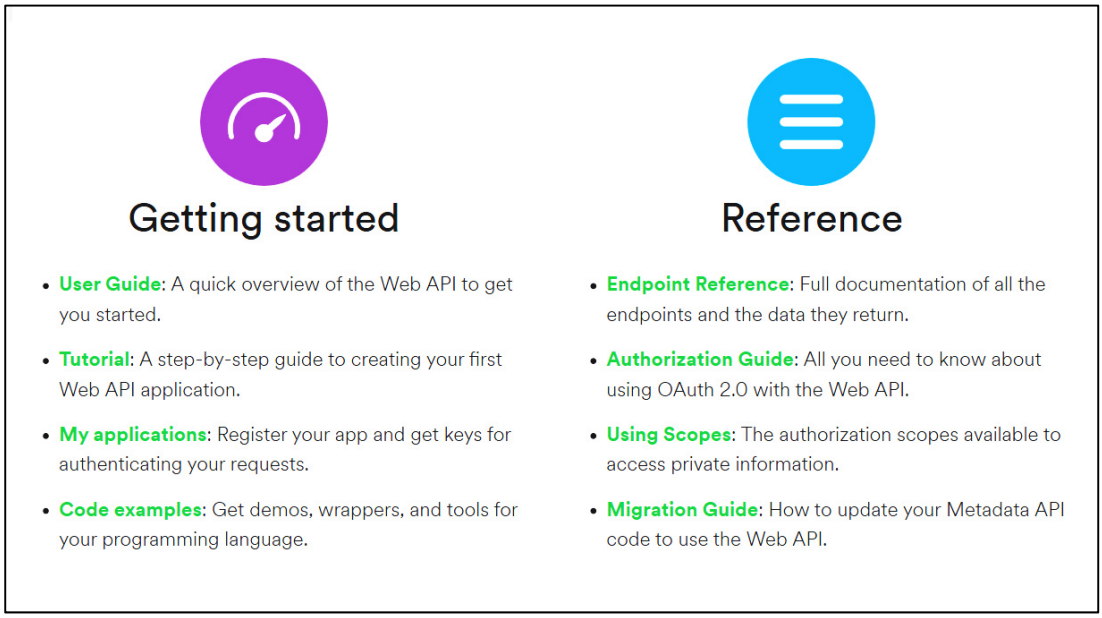


Abb. 72: Navigationsstränge auf der Startseite (SW)

Bei der Spotify Web API (SW) werden die Navigationsstränge sehr übersichtlich und eindeutig auf der Startseite präsentiert. Die grün eingefärbten Menüpunkte dienen als Verlinkungen in die Kapitel.

4.2.2.6.3. Problematiken

In der Bewertung waren vor allem die beiden Kriterien zu den Navigationssträngen problematisch. Häufig gab es passende Überschriften in der Navigation, in der Startseite wurde aber nicht auf diese hingewiesen. In vielen Fällen gab es gar keine Startseite (Udemy, ANXE, Reddit, BC, Klipfolio, OS, NYTB, VS, TS). Bei einem Forschungsobjekt war die API Teil einer Sammlung, so dass die Startseite für mehrere APIs diente (MOMR).

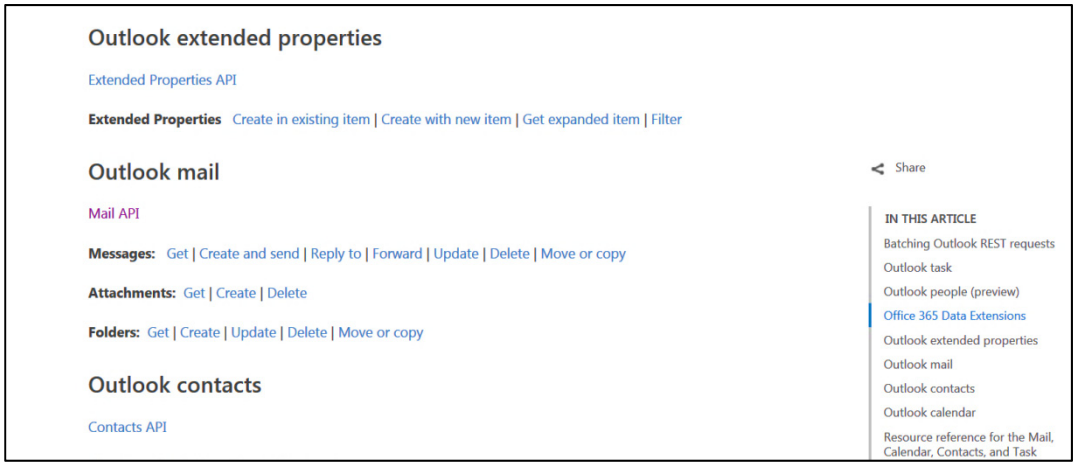


Abb. 73: Navigationsstränge Gesamtverständnis und Problemlösung (MOMR)

Einige Forschungsobjekte besaßen keine Startseite, wiesen aber eine Struktur auf, die den Navigationssträngen Gesamtverständnis und Problemlösung sehr nah war (BC, DCR, Kaltura). Diese Struktur schlug sich in der Navigation nieder, die durch die gebotene Übersichtlichkeit eine Startseite fast überflüssig machte.

Ein weiteres Problem waren überladene Startseiten (Flickr, OFEC, SW, Kaltura, MOMR). Auf den Startseiten gab es zwar die geforderten Verlinkungen, sie zu finden war aber nicht immer ganz einfach. In so einem Fall wäre es besser gewesen, die Startseite so kurz wie möglich zu halten und eine separate Einleitung anzubieten. Das massivste Beispiel ist die Flickr API.

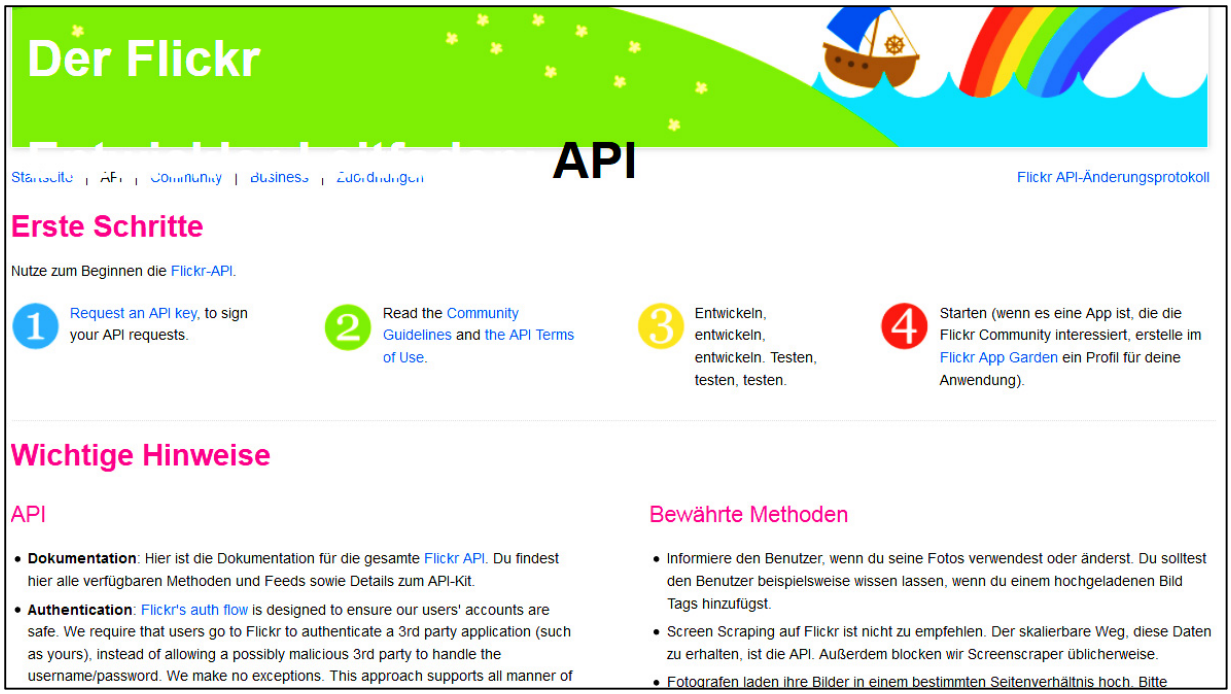


Abb. 74: Überladene Startseite (Flickr)

Die Startseite der Flickr API ist überladen mit Text und Farben, bietet aber trotzdem durch den obenstehenden Punkt „Erste Schritte“ eine Möglichkeit sich schnell zu orientieren. Die Masse an Text und Auszeichnungen wirkt aber dennoch optisch erschlagend. Die großen Textblöcke sind zusätzlich für das Scannen und Skimmen ungünstig. Aufgrund der Gestaltung der nach oben gelagerten Erste-Schritte-Leiste liegt sogar die Vermutung nahe, dass der untenstehende Text übersehen oder sogar ignoriert wird.

4.2.2.7. Intelligente Suchfunktion

Die intelligente Suchfunktion ist mit 24 % Vorkommen die am zweitwenigsten vorkommendste Heuristik der Untersuchung. Ein Grund für das geringe Vorkommen könnte der hohe Aufwand sein, der mit der Bereitstellung einer Suchfunktion zusammenhängt. Dazu kommt, dass einige der Forschungsobjekte mit Frameworks produziert wurden, in denen vermutlich keine Suchfunktion vorgesehen ist (OFEC, ANXE, BC, Klipfolio, NYTB, Kaltura, SPR). Mindestens eine weitere Dokumentation wurde mit einem Werkzeug für die automatisierte Erstellung von Dokumentation erstellt, so dass auch bei diesem Fall eingeschränkte Optionen für den Output vermutet werden können (Reddit). Nur bei Reddit wurde die automatisierte Erstellung ausdrücklich erwähnt, die Erstellung durch Frameworks war ansonsten vor allem durch Logos zu erkennen.

Der hohe Aufwand der Suchfunktion besteht neben den möglichen finanziellen Belastungen für die Erstellung/den Kauf außerdem in der Problematik des Zeitaufwands für deren Aufbau, Konfiguration und Unterhaltung. Suchfunktionen sind komplexe Anwendungen, die mithilfe von Algorithmen und umfangreichen Datenbanken realisiert werden. Für den Abgleich von Singular/Plural, Fehlerkorrekturen, Autovervollständigung etc. muss die Suche sowohl auf eine eigene Datenbank als auch auf eine Logik in Form von Algorithmen zurückgreifen. Diese Komplexität macht intelligente und individualisierte Suchfunktionen so aufwendig. Darüber hinaus exisitiert für Online-Medien die Stichwortsuche mit dem Shortcut „Strg + F“, der gerade für Onepages und PDFs zielführend sein kann. Webseiten hingegen müssen in allen Unterseiten einzeln durchsucht werden, was den Aufwand maßgeblich erhöht.

4.2.2.7.1. Betrachtung nach formalen Kriterien

Eine mögliche Abhängigkeit in dem Vorhandensein einer Suchfunktion könnte in dem Umfang der Dokumentation liegen. Es ist naheliegend, dass eine sehr kurze Dokumentation eher keine Suchfunktion benötigt, da die Inhalte auch ohne Suche schnell auffindbar wären. [80]

Neben dem Umfang könnte ebenfalls das Medium der Umsetzung mit dem Vorhandensein einer Suchfunktion verknüpft sein. In einem PDF wird tendenziell eher keine Suchfunktion zu finden sein, während dies gerade bei Webseiten häufiger der Fall sein könnte. Das liegt an dem Aufbau und den technischen Voraussetzungen der Medien. Die Integration einer Suche ist eine sehr unübliche Methode, da die meisten PDF-Reader eine eigene Suchfunktion besitzen. In Onepages ist der Mehrwert einer simplen Suchfunktion sogar diskutabel, da sie theoretisch per Shortcout „Strg + F“ unkompliziert nach Stichworten durchsucht werden können.

	Webseite	Onepage	PDF	Gesamt
Kurz	50 %	0 %	-	**20 %**
Mittel	25 %	-	0 %	**19 %**
Lang	36 %	8 %	0 %	**23 %**
Gesamt	**36 %**	**8 %**	**0 %**	**Gesamt: 24 %**

Abb. 75: Gesamtvorkommen der Kriterien nach Umfang und Medium[81] für „Ja“

Nach Prozentzahlen betrachtet erfüllen kurze Webseiten mit 50 % am häufigsten die Heuristik. Lange Onepages hingegen haben mit 8 % am wenigsten Übereinstimmungen. Dicht dahinter liegen lange Webseiten mit 36 % und mittlere Webseiten mit 25 %. Am wenigsten Übereinstimmungen weisen kurze Onepages und PDFs auf, die gar keine Suchfunktion haben. Die Vermutungen bezogen auf Umfang und Medium konnten bestätigt werden, insgesamt haben lange Medien mit 23 % und Webseiten mit 36 % am meisten Übereinstimmungen.

4.2.2.7.2. Betrachtung nach Kriterien

Die geringe Gesamtanzahl der Übereinstimmungen mit den Forschungsobjekten verteilt sich sehr unregelmäßig auf die unterschiedlichen Kriterien der Heuristik. Vor allem die „erweiterte Suchfunktion“ und die „Suchergebnisse als Schlagwort/als Liste“ weisen besonders viele Übereinstimmungen auf.

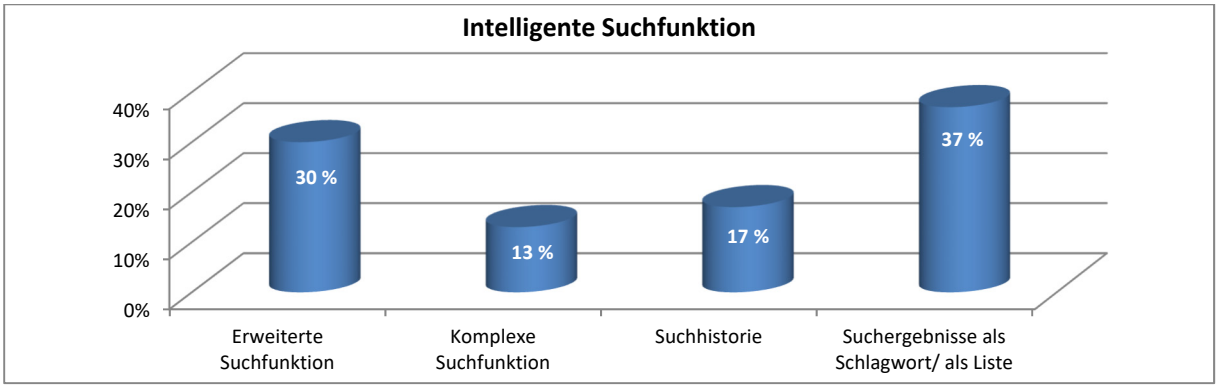


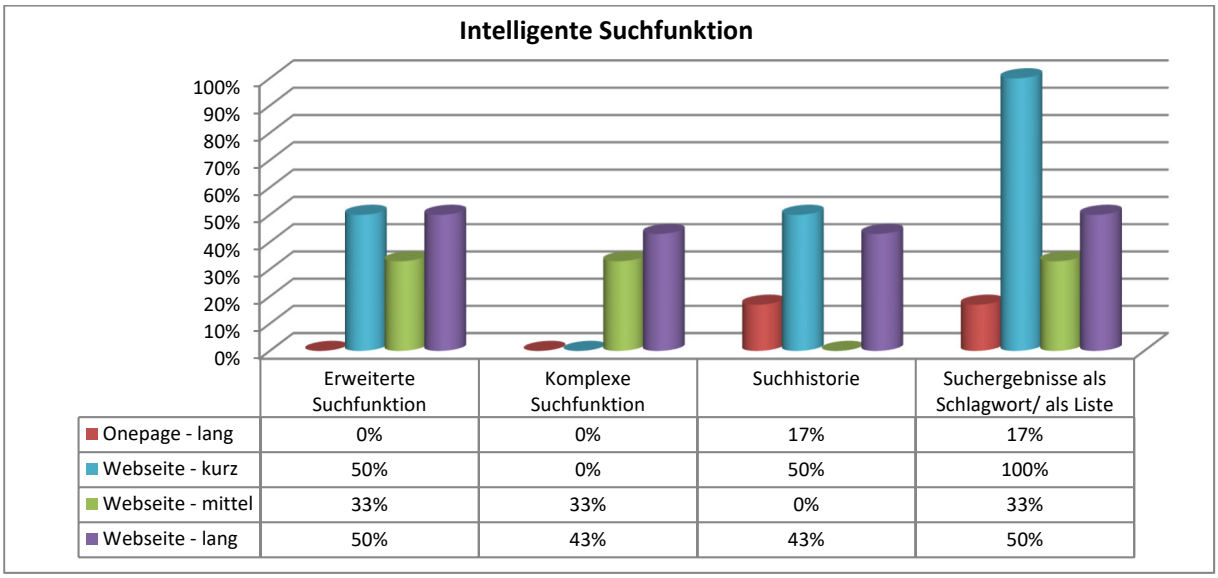
Abb. 76: Gesamtvorkommen der Kriterien für „Ja“

[80] Die Auffindbarkeit von Inhalten ist neben dem Umfang auch von weiteren Aspekten wie der Strukturierung, deskriptiven oder prozessbezogenen Überschriften, der Navigation etc. abhängig.

[81] Die Gesamtangaben beziehen sich auf das Gesamtvorkommen des Umfangs und des Mediums in allen jeweils zugeteilten Forschungsobjekten. Aus diesem Grund bilden die Werte unter „Gesamt“ nicht die reine Summe der anderen Prozentzahlen. Sie wurden nach der Formel „Tatsächliches Vorkommen in absoluten Zahlen geteilt durch 1 % in absoluten Zahlen der Gesamtanzahl der Forschungsobjekte nach Medium/Umfang“ berechnet. Dieses Vorgehen bezieht sich auch auf alle folgenden Tabellen mit diesem Kontext.

Die erweiterte Suche ist in fast einem Drittel der Forschungsobjekte vorhanden, wobei es sehr häufig eine Singular-/ Plural-Erkennung aber keine Fehlertoleranz gibt. Komplexe Suchfunktionen sind mit 13 % am wenigsten zu finden, wobei nur die Autovervollständigung angeboten wird. Das Erkennen von Synonymen oder das Anbieten von verwandten Suchbegriffen wurde in keiner der Suchfunktionen gefunden. Suchhistorien sind mit 17 % Übereinstimmung ebenfalls eher wenig vorhanden. Das Vorkommen der Suchergebnisse liegt höher als das der „erweiterten/komplexen Suchfunktion“, weil nicht alle vorhandenen Suchen auch die geforderten Funktionen erfüllen (DCR, HW). Eine reine Wortsuche ohne Fehlererkennung, Singular-/Plural-Erkennung usw. konnte nicht gewertet werden, die Anzeige der Suchergebnisse hingegen aber schon.

Die Betrachtung nach Kriterien zeigt detailliert die hohen Übereinstimmungen für Webseiten, die sich in fast allen Kriterien fortsetzen. Lange Onepages hingegen verfügen weder über erweiterte noch über komplexe Suchfunktionen.



	Erweiterte Suchfunktion	Komplexe Suchfunktion	Suchhistorie	Suchergebnisse als Schlagwort/ als Liste
Onepage - lang	0%	0%	17%	17%
Webseite - kurz	50%	0%	50%	100%
Webseite - mittel	33%	33%	0%	33%
Webseite - lang	50%	43%	43%	50%

Abb. 77: Gesamtvorkommen der Kriterien nach Umfang und Medium für „Ja“

Die höchste Übereinstimmung erreichen mit 100 % kurze Webseiten bei dem Kriterium der Suchergebnisse. Suchergebnisse sind außerdem zu 50 % in langen und 33 % in mittleren Webseiten vertreten. Onepages stellen nur zu 17 % Suchergebnisse bereit, was gleichzeitig dem Wert für die Suchhistorie entspricht. In keinem Fall gab es Onepages mit erweiterten oder komplexen Suchfunktionen. Insgesamt kommt unabhängig vom Medium die komplexe Suchfunktion am wenigsten vor.

Die Suchfunktion war in den meisten Fällen in die Startleiste der Webseiten/Onepages integriert. Die Eingabe, eventuelle Vorschläge und der Zugriff auf die Suchhistorie erfolgten über dieses Feld. Die Suchhistorie konnte entweder über die Betätigung des Richtungspfeils nach unten auf der Tastatur oder über das Eintippen in das Suchfeld aufgerufen werden.

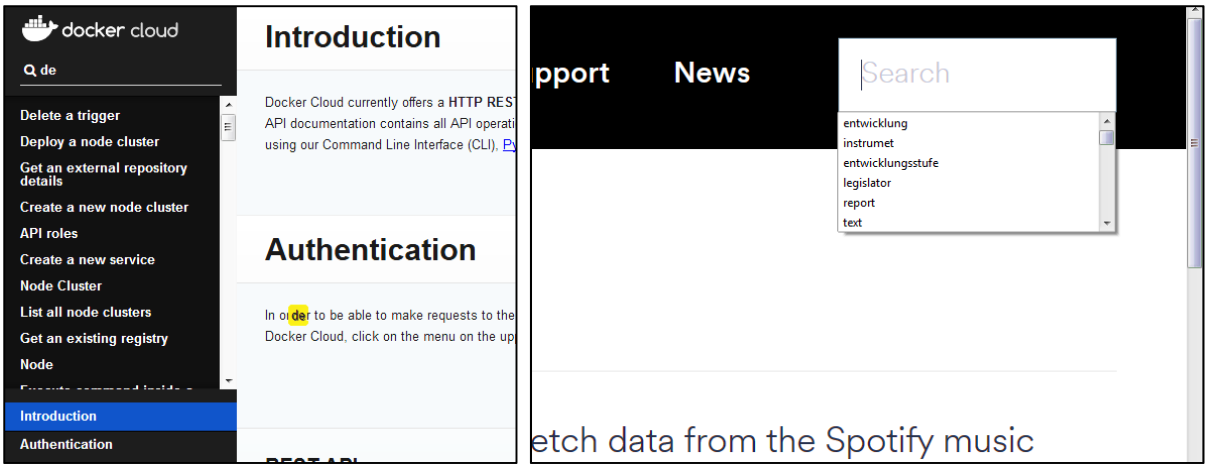


Abb. 78: Suchleisten und Suchhistorien (v. l. n. r. DC, SW)

Die Darstellung der Suchergebnisse erfolgte zu 91 % als Liste (GSS, GMD, BC, SW, PH, Kaltura, Oanda, MOMR, DCR, HW) und zu 9 % als Schlagwort. Das Layout der Listen erinnerte in den meisten Fällen sehr stark an Websuchen, es gab aber auch Mischformen inklusive Schlagwortmarkierung.

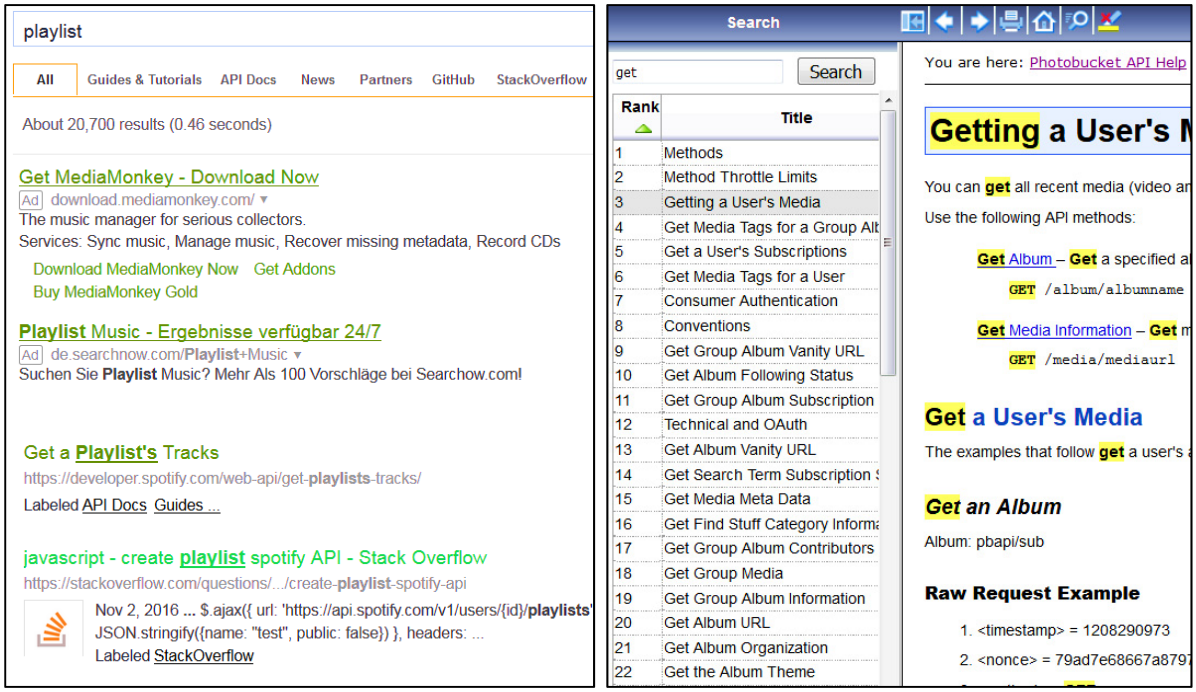


Abb. 79: Suchergebnisse als Liste und Kombination aus Liste und Schlagwörtern (v. l. n. r. SW, PH)

4.2.2.7.3. Problematiken

Eine Problematik der Suchfunktionen war, dass sie nicht nur in der Dokumentation, sondern im gesamten Webauftritt und teilweise noch weiteren Quellen suchten (GSS, Oanda, HW, MOMR). Bei diesen Suchen konnten in den meisten Fällen weitere Filter eingestellt werden, teilweise war eine gezielte Suche in der Dokumentation durch fehende Filter aber auch nicht möglich.

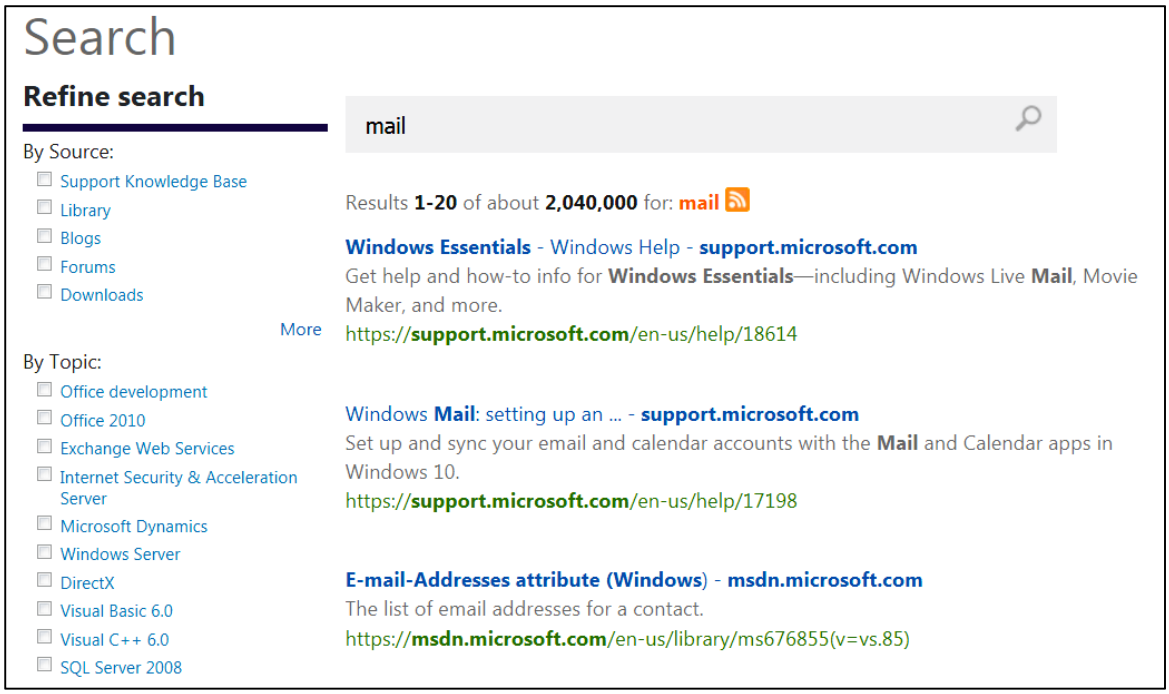


Abb. 80: Liste der Suchergebnisse mit Filtermöglichkeiten (MOMR)

In einigen Dokumentationen war zwar eine Suchleiste vorhanden, diese diente aber nicht der Durchsuchung der Dokumentation, sondern der Durchsuchung der Webseite. Die betroffenen Forschungsobjekte sind Anbieter im E-Commerce, die eine Produktsuche einsetzen (Flickr, Etsy, Zazzle, OS). Da die Dokumentationen in die Hauptwebseite integriert sind, wirkten die Suchen auf den ersten Blick fälschlicherweise als zugehörig.

Manche Suchfunktionen ermöglichten nur die Suche nach Überschriften bzw. in den Kapiteltiteln, nicht aber die Suche in dem eigentlichen Inhalt (Klipfolio, PH).

4.2.2.8. Codenahe Platzierung von wichtigen Informationen

Hinter der „Intelligenten Suchfunktion“ belegt die „Codenahe Platzierung von Informationen“ mit 21 % Übereinstimmungen den letzten Platz. Das Ziel der codenahen Platzierung von Informationen ist es, sowohl den konzept- als auch codeorientierten Lernansatz optimal zu unterstützen. Dafür müssen relevante Informationen identifiziert und redundant im Text und im Code bzw. codenah platziert werden. Allein die Identifizierung der relevanten Informationen bedeutet einen großen Zeitaufwand und erfordert Fachwissen und Einschätzungsvermögen.

4.2.2.8.1. Betrachtung nach formalen Kriterien

Die Umsetzung der Heuristik könnte vom Umfang der Dokumentation abhängig sein. Je umfangreicher eine Dokumentation ist, desto schwieriger wird es bei einem codeorientierten Lernansatz, wichtige konzeptuelle Informationen wahrzunehmen. Bei Dokumentationen mit viel Textanteil oder sehr vielen Funktionen ist die codenahe Platzierung von Hinweisen, Einschränkungen, Besonderheiten usw. ein zielführendes Instrument für die Gleichberechtigung beider Lernansätze. Umgekehrt könnte bei kurzen Dokumentationen eher keine codenahe Platzierung von Informationen zu finden sein, da der Mehrwert des Aufwands fragwürdig wäre.

Für das Medium der Umsetzung gelten abweichende technische Voraussetzungen, da PDFs eingeschränkt sind. Die interaktive Verfügbarkeit von Informationen ist bei ihnen problematisch, da

keine Hover- oder Panel-Funktion integriert werden kann. Interaktivität in PDFs kann in dieser Hinsicht ausschließlich durch Verlinkungen und Querverweise erreicht werden.[82]

Die Betrachtung nach formalen Kriterien zeigt, dass die Heuristik mit 100 % am häufigsten in langen PDFs und ebenfalls insgesamt mit 50 % am häufigsten in PDFs vorkommt.

	Webseite	Onepage	PDF	Gesamt
Kurz	0 %	22 %	-	**13 %**
Mittel	11 %	-	0 %	**8 %**
Lang	19 %	28 %	100 %	**25 %**
Gesamt	**16 %**	**26 %**	**50 %**	**Gesamt: 21 %**

Abb. 81: Gesamtvorkommen der Kriterien nach Umfang und Medium für Ja

Neben langen PDFs sind es vor allem lange und kurze Onepages, die die Heuristik erfüllen. Bei langen und mittleren Webseiten ist die Übereinstimmung bereits deutlich geringer. Lange Onepages erfüllen die Heuristik zu 17 % und liegen damit deutlich hinter den langen Webseiten.

Insgesamt haben lange Medien mit 25 % am meisten Übereinstimmungen, mittlere Medien haben mit 8 % am wenigsten. Anders als vermutet haben kurze Medien mit 13 % sogar eine höhere Übereinstimmung als mittlere Medien. Nach dem Medium betrachtet haben Webseiten mit 16 % am wenigsten Übereinstimmungen. PDFs verfügen trotz der technischen Einschränkungen zu 50 % über Kriterien der codenahen Platzierung.

4.2.2.8.2. Betrachtung nach Kriterien

Die Betrachtung nach Kriterien zeigt, dass vor allem das Kriterium „Inline-Kommentare“ in den Forschungsobjekten sehr häufig vertreten war.

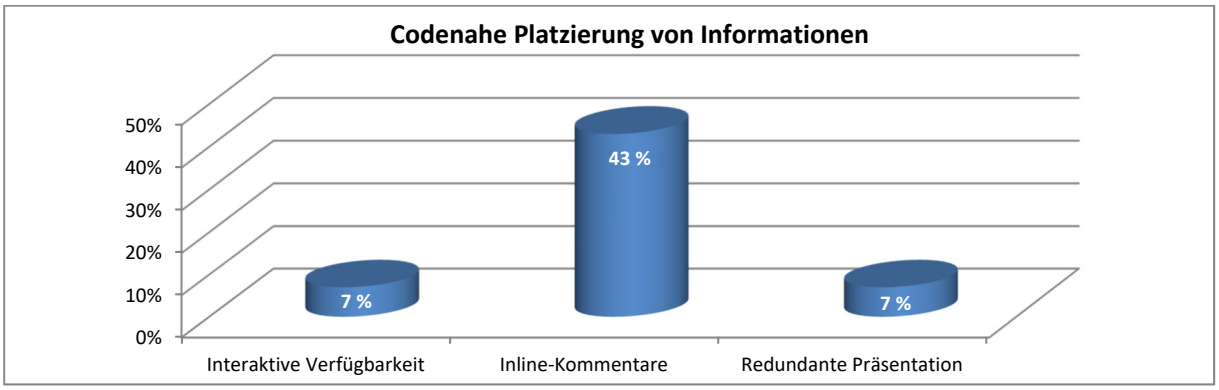
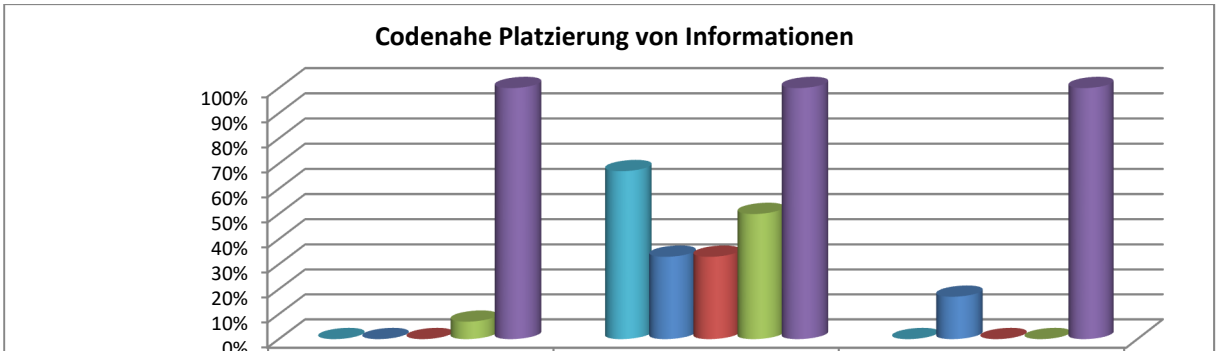


Abb. 82: Gesamtvorkommen der Kriterien für Ja

Inline-Kommentare sind mit 43 % mit Abstand am häufigsten vertreten. Sie sind gängige Praxis in der Software-Entwicklung, so dass ihr häufiges Vorkommen nicht überraschend ist. Die beiden anderen Kriterien „Interaktive Verfügbarkeit von Informationen“ und „Redundante Präsentation wichtiger Informationen“ liegen mit 7 % deutlich darunter.

Die Betrachtung der Kriterien in Abhängigkeit von Umfang und Medium zeigt, dass die beiden weniger vorkommenden Kriterien jeweils in nur zwei Kombinationen vorhanden sind. Zu den Kombinationen gehören lange PDFs, lange Webseiten und lange Onepages.

[82] Obwohl einige Anbieter das Generieren von „Interaktiven PDFs“ ermöglichen, unterscheidet sich diese Interaktivität von der, die in der Heuristik gefordert wird. Die Heuristik bezieht sich vor allem auf das Ein-/Ausblenden von Informationen durch den Nutzer per Hover-Funktion, Klick etc. Die angebotenen interaktiven PDFs von z. B. Adobe beziehen sich eher auf das Einbetten von Video- und Audio-Dateien sowie auf Standardfunktionen wie Lesezeichen, Hyperlinks und Querverweise (vgl. Adobe 2016). Die Umsetzung der geforderten Interaktivität wäre trotzdem durch z. B. Abbildungen möglich, die sich als Hover-Ereignis vergrößern. Für das Öffnen von interaktiven PDFs wird jedoch ein besonderes Programm, der Flash Player, benötigt. Ein PDF Reader ist dafür nicht ausreichend.



	Interaktive Verfügbarkeit	Inline-Kommentare	Redundante Präsentation
Onepage - Kurz	0%	67%	0%
Onepage - Lang	0%	33%	17%
Webseite - Mittel	0%	33%	0%
Webseite - Lang	7%	50%	0%
PDF - Lang	100%	100%	100%

Abb. 83: Gesamtvorkommen der Kriterien nach Umfang und Medium für Ja

Bei den beiden weniger vorhandenen Kriterien ist die Verteilung sehr ungleichmäßig. Lange PDFs haben jeweils mit 100 % den höchsten Anteil an beiden. Für die interaktive Verfügbarkeit folgen mit 7 % lange Webseiten sowie für die redundante Präsentation lange Onepages, die mit 17 % deutlich unter dem Wert des PDFs liegen. Die Verteilung von Inline-Kommentare als insgesamt häufigstes Kriterium ist relativ gleichmäßig. Am wenigsten Übereinstimmungen haben mittlere Webseiten und lange Onepages mit 33 %, der höchste Wert liegt bei langen PDFs mit 100 %. Das Vorkommen der Inline-Kommentare verdeutlicht die Medienneutralität dieses Kriteriums (Webseite, Onepage, PDF).

Die „Interaktive Verfügbarkeit von Informationen“ wurde in einem langen PDF gefunden (TROC). Die Umsetzung dieses Kriteriums in einem PDF ist aufwendig, da entsprechende Textstellen manuell verlinkt werden müssen. Die gefundenen Verlinkungen führen aus einem Code-Beispiel zu der jeweiligen Parameterbeschreibung und wieder zurück.

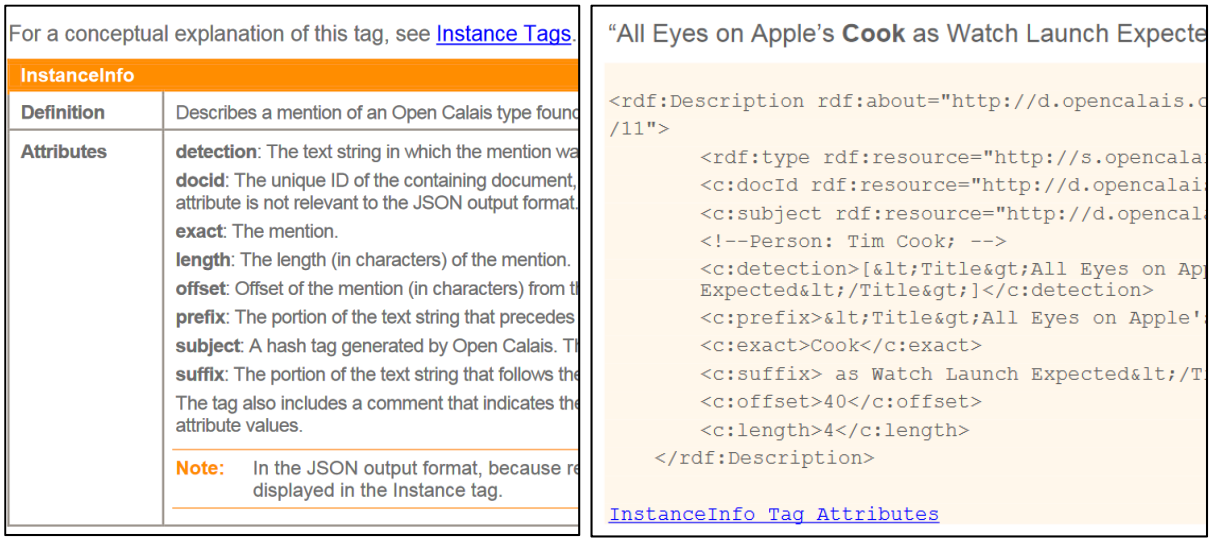


Abb. 84: Interaktivität als Verlinkung zu konzeptuellen Informationen mit Rückweg (v. l. n. r.; TROC)

Auf Webseiten wurde die interaktive Verfügbarkeit ebenfalls über Verlinkungen und in einem Fall mit einem Panel gelöst. Mit dem Panel konnten per Klick Child-Attribute ein-/ausgeblendet werden.

Inline-Kommentare wurden zum größten Teil in codetypischer Typografie und farbiger Auszeichnung umgesetzt (s. a. Abbildung 85). Das Vorkommen von Inline-Kommentaren ist zwar insgesamt sehr hoch, die Qualität der Kommentare variiert jedoch stark. Einige weisen keinen Mehrwert für die Dokumentation auf, da sie eher die Funktion einer Überschrift, einer Quellenangabe o. ä. einnehmen.

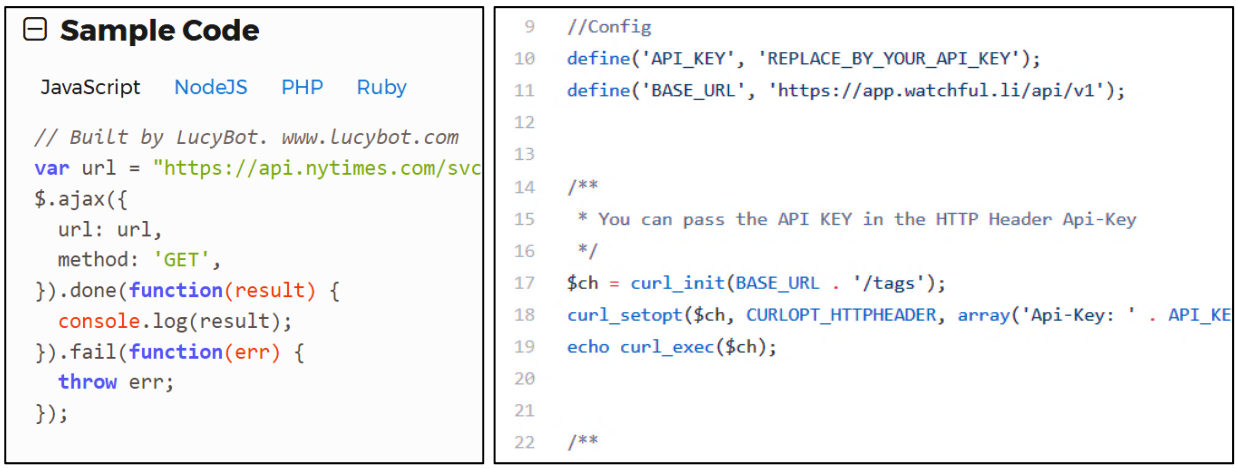


Abb. 85: Inline-Kommentare mit und ohne Mehrwert (v. l. n. r. NYTB, WR)

Das Kriterium „redundante Präsentation wichtiger Informationen“ war insgesamt am wenigsten zu finden. Im PDF wurde es durch Inline-Kommentare umgesetzt, in der langen Onepage durch vorgelagerte Erklärungen der Parameter als alphabetische Liste (OFEC). Die gleichen Erklärungen waren in der tabellarisch umgesetzten Demo-Funktion enthalten.

4.2.2.8.3. Problematiken

In Bezug auf die Interaktivität waren viele Ansätze zu sehen, die oftmals nicht konsequent durchgeführt wurden. Das bezog sich vor allem auf fehlende Rückwege oder ungenaue Verlinkungen, die nicht gewertet werden konnten. In zwei Dokumentationen wurde aus der Referenzdokumentation in die entsprechende Zeile eines Code-Beispiels auf Github verlinkt, leider aber ohne Rückweg (HW, Reddit).

Abb. 86: Verlinkung aus der Referenzdokumentation in den Working Code (v. l. n. r.; Reddit)

Das Vorhandensein von Inline-Kommentaren war zum einen abhängig von der Dokumentation und den Code-Beispielen, zum anderen aber auch von der gewählten Sprache. Teilweise waren Inline-Kommentare für eine Funktion nicht in allen Programmiersprachen vorhanden, obwohl es sich um

allgemeine Informationen handelte. Es kann nur vermutet werden, dass der Inhalt des Kommentars nur für die jeweilige Sprache relevant war.

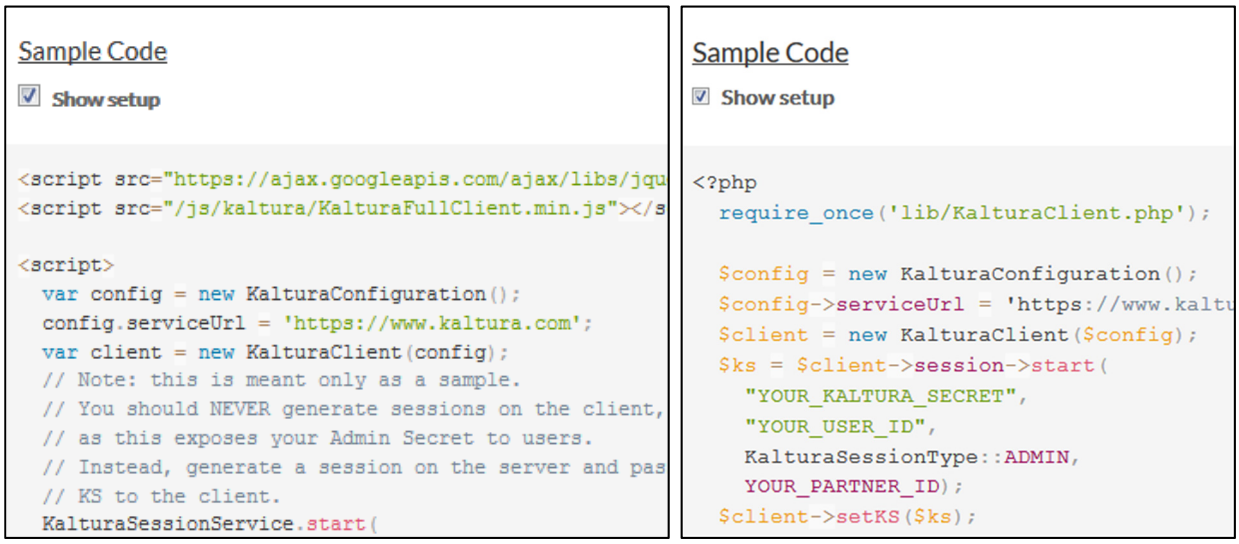


Abb. 87: Gleiches Code-Beispiel in PHP ohne und in JavaScript mit Kommentar (v. l. n. r.; Kaltura)

4.3. Zusammenfassung

Die Auswertung der Untersuchungsergebnisse hat gezeigt, dass alle Heuristiken in den Forschungsobjekten gefunden werden konnten. Die Häufigkeit der Vorkommen schwankte jedoch sehr stark, was sowohl in Bezug auf die Heuristiken als auch in Bezug auf die einzelnen Kriterien auftrat. Innerhalb der Heuristiken gab es zum Teil auffällig häufig oder selten vorkommende Kriterien. Es gab kein Kriterium, das gar nicht von den Forschungsobjekten erfüllt und lediglich ein Kriterium, das in allen Forschungsobjekten gefunden wurde. Dabei handelte es sich um die „konsistente Feinstruktur der Kapitel“. Am wenigsten war die „redundante Präsentation wichtiger Informationen“ mit zwei Vorkommen in absoluten Zahlen vorhanden.

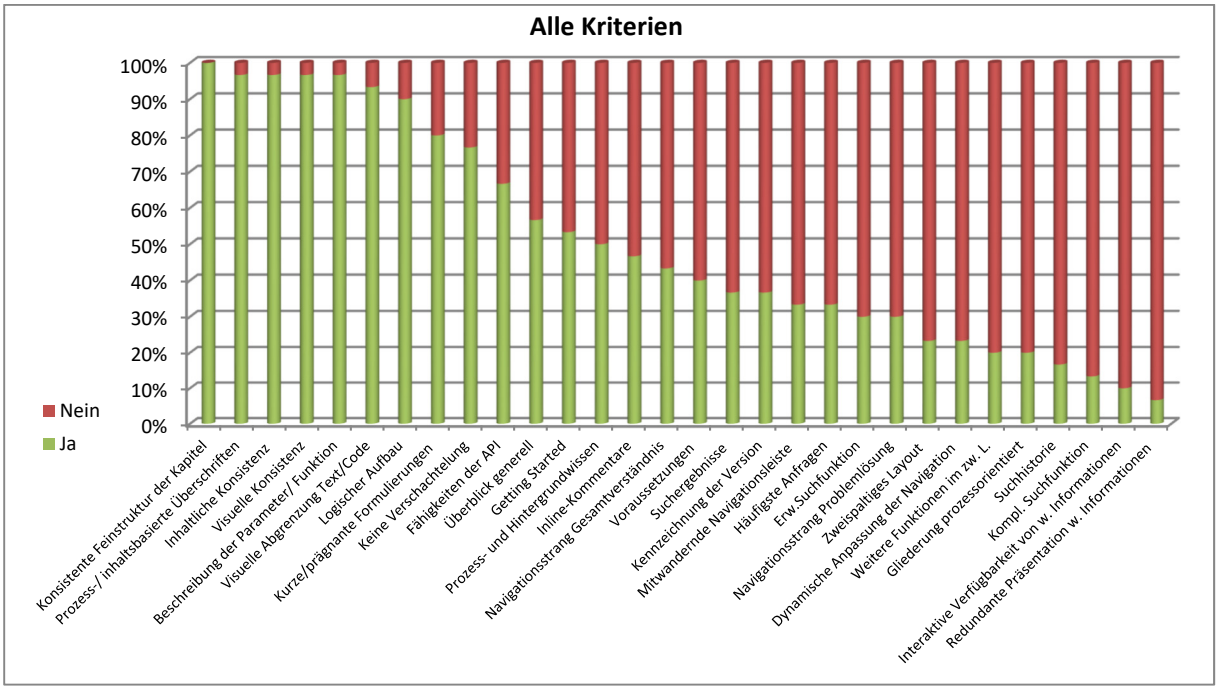


Abb. 88: Gesamtvorkommen aller Kriterien absteigend geordnet nach Übereinstimmungen für „Ja“

Vor allem Kriterien zur visuellen Gestaltung und der generellen Strukturierung von Dokumentation wurden sehr häufig gefunden. Im Mittelfeld der Vorkommen sind eher Kriterien zur inhaltlichen Umsetzung von wichtigen Informationen sowie wichtigen Informationsblöcke situiert. Eher weniger erfüllte Kriterien beziehen sich auf die technisch basierte Umsetzung von informationsspezifischen Zugriffen (konzeptuell vs. codebasiert) sowie Suchfunktionen und zweispaltige Layouts.

Die Ergebnisse bezogen auf die Kriterien zeigen, dass grundsätzliche Anforderungen an API-Dokumentation von den Forschungsobjekten insgesamt gut erfüllt wurden. Ohne Konsistenz und sinnvolle Struktur wäre auch eine aus technischer Sicht sehr aufwendig umgesetzte Dokumentation eher nicht angenehm für den Anwender, da er z. B. die gewünschten Inhalte nur schwer finden könnte. Weniger gut erfüllt wurden Kriterien, die sich auf die Umsetzung der beiden Lernansätze bezogen.

Bei den Forschungsobjekten der Untersuchung handelte es sich durchgehend um externe Web-APIs, die im RESTful-http programmiert wurden. Trotzdem alle APIs technische Gemeinsamkeiten haben, unterscheiden sie sich stark in der Anzahl der Übereinstimmungen. Die Unterschiede bewegen sich in einem Rahmen von 50 %.

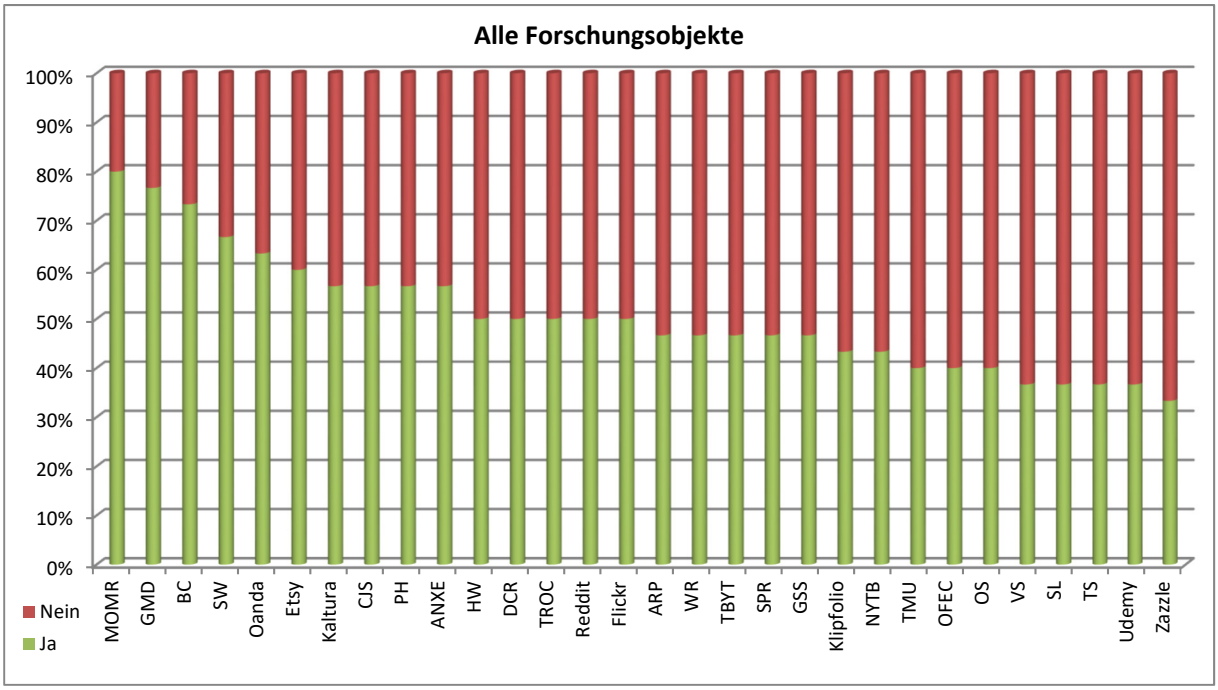


Abb. 89: Gesamtvorkommen der Kriterien geordnet nach Forschungsobjekten für „Ja“

Die Ergebnisse bezogen auf die Forschungsobjekte zeigen, dass genau die Hälfte der untersuchten API-Dokumentationen mindestens die Hälfte der untersuchten Kriterien erfüllt. Die Unterschiede lassen sich weder am Medium noch am Umfang der Forschungsobjekte festmachen. Auch eine bereichsspezifische Einordnung (nach Geschäftsfeld wie z. B. E-Commerce) greift nicht. Insgesamt scheint die Gestaltung von API-Dokumentation stark autorenabhängig zu sein und eher unabhängig von formalen Charakteristika zu stehen.

4.4. Fehlerbetrachtung

Im Laufe der Untersuchung wurden vereinzelt methodische Probleme deutlich. Zu diesen Problemen zählten der generelle Umfang, das Vorgehen in der Kontrolle, die Bewertungsskala, die Vergleichbarkeit der Medien und die ungleiche Verteilung der Forschungsobjekte auf die formalen

Kriterien. Aufgrund dieser Problematiken ist die durchgeführte Untersuchung nur eingeschränkt als repräsentativ zu betrachten.

Der generelle Umfang der Forschungsobjekte und Kriterien war sehr groß. Insgesamt wurden 900 Kriterien geprüft, die mit „Ja“ oder „Nein“ beantwortet werden konnten. Dieser Umfang birgt ein gewisses Risikopotenzial für die Genauigkeit der Untersuchung. Weiterhin steht der Umfang in Beziehung mit einem anderen Problem, dass das Vorgehen in der Kontrolle betraf. Die Untersuchung in dieser Arbeit wurde alleinig von der Autorin durchgeführt, so dass keine zusätzliche Einschätzung der Kriterien vorliegt. Die große Anzahl der Kriterien und die Korrektur durch eine Person könnten ein gewisses Fehlerpotenzial eröffnen.

Während der Untersuchung und ebenfalls in der Auswertung fiel auf, dass eine Bewertungsskala für einige Kriterien besser geeignet gewesen wäre. Durch eine Abstufung in z. B. „Ja“ – „Eher Ja“ – „Eher Nein“ – „Nein“ wäre eine genauere Bewertung der Forschungsobjekte möglich gewesen. In der Auswertung wurden die Abweichungen aufgrund der eingeschränkten Bewertungsskala deshalb zusätzlich beschrieben.

Die Forschungsobjekte wurden für einen Vergleich in unterschiedliche Medien aufgeteilt. In der Auswertung wurde bereits thematisiert, dass einige Heuristiken mit bestimmten Medien leichter umzusetzen waren als mit anderen. Ein Beispiel dafür ist die Suchfunktion, die in PDFs eigentlich nicht vorkommen konnte. Die Vergleichbarkeit der unterschiedlichen Medien ist dementsprechend nur für solche Kriterien repräsentativ, die als medienneutral einzuschätzen sind.

Ein kritischer Punkt für die Auswertung war ebenfalls die ungleiche Verteilung der Forschungsobjekte auf die formalen Kriterien. Die Vergleichbarkeit in der Auswertung konnte aber durch Prozentangaben gewährleistet werden. Die ungleiche Anzahl führte jedoch in einigen Fällen sehr schnell zu sehr hohen oder sehr niedrigen Prozentwerten aufgrund sehr geringer absoluter Zahlen.

5. Gesamtbetrachtung: Gewichtung der Kriterien

In der Auswertung wurden die Übereinstimmungen, deren Beziehung zu den formalen Kriterien sowie die Verteilung auf die Kriterien der Heuristiken dargestellt. Dabei wurde jede Übereinstimmung im gleichen Maße gewertet, ohne Einschätzung der Relevanz der einzelnen Kriterien. An diesem Punkt setzt die Gesamtbetrachtung an, die die Forschungsobjekte in ihrer Gesamtheit beurteilt. Ein Forschungsobjekt, das die Grundregeln guter Dokumentation nicht beachtet, dafür aber über eine intelligente Suchfunktion verfügt, ist trotzdem tendenziell nicht anwenderfreundlich. In diesem Fall müssten die Kriterien des guten Dokumentierens stärker gewertet werden als die der Suchfunktion, um die Qualität der Dokumentation zu beurteilen.

Aufbauend auf der durchgeführten Studie wird durch die Autorin eine qualitative heuristische Bewertung der Forschungsobjekte hinsichtlich des Gesamteindrucks durchgeführt. Sie wird anhand mehrerer Eckpunkte umgesetzt, die allerdings nicht verpflichtend sind. Dazu gehören die Bedienbarkeit, Übersichtlichkeit und Auffindbarkeit von Inhalten sowie eine sinnvolle Struktur. Nachfolgend werden die Ergebnisse dieser Bewertung vorgestellt und mit denen der objektiven Untersuchung verglichen. Die Gesamtbetrachtung kann nur anhand der genannten Aspekte erfolgen, da keine technische Anwendung der API durchgeführt wird. Dementsprechend beziehen sich die folgenden Ausführungen ausschließlich auf die Gestaltung der Dokumentationen und nur begrenzt auf deren Inhalt.

Platz neu	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Name	DCR	ANXE	BC	GMD	MOMR	PH	Oanda	SW	TBYT	NYTB	CJS	Troc	ARP	Etsy	SPR	SL	OFEC	Klipfolio	HW	Kaltura	Reddit	WR	Flickr	Udemy	OS	GSS	VS	TS	TMU	Zazzle
Platz alt	11	7	3	2	1	7	5	4	16	21	7	11	16	6	16	26	23	21	11	7	11	16	11	26	23	16	23	23	23	30

Abb. 90: Ergebnis der Gesamtbetrachtung

Das Ranking der Forschungsobjekte in der Gesamtbetrachtung unterscheidet sich zum Teil sehr stark von dem der kriteriengestützten Untersuchung.[83] Nachfolgend werden repräsentative Forschungsobjekte detailliert vorgestellt.

Den ersten Platz der Gesamtbetrachtung belegt die Docker Cloud Rest API (DCR), die in der kriteriengestützten Untersuchung den Platz 11 innehat. Sie überzeugt durch ein klares Layout und eine eindeutige Strukturierung. Außerdem hat sie eine dynamische Navigation und eine überschaubare Anzahl von Einträgen im Menü. Alle Inhalte können klar zugeordnet werden, die Kapitel sind eindeutig voneinander abgegrenzt und die Code-Beispiele mit einer Textpassage referenziert sowie typografisch hervorgehoben. Auch die visuelle Konsistenz wirkt sich sehr angenehm aus.

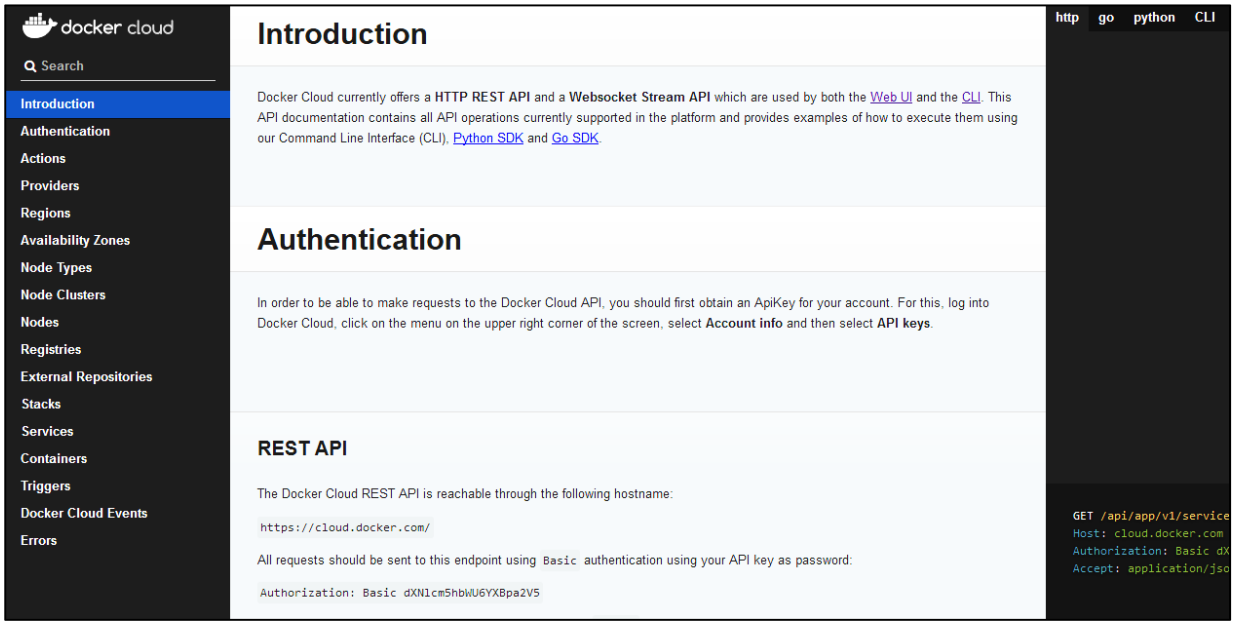


Abb. 91: Erster Platz in der Gesamtbetrachtung für Docker Cloud (DCR)

Das zweispaltige Layout der DCR API kommt ebenfalls bei der ANX Exchange API (ANXE) vor, die Platz 2 im Ranking einnimmt. Insgesamt überzeugen die beiden Forschungsobjekte durch eine ansprechende Optik sowie die Möglichkeit sich schnell zurechtzufinden.

Bei der viertplatzierten Google Maps Directions API (GMD) ist die visuelle Konsistenz im negativen Sinn ausschlaggebend. Zwar ist sie generell gegeben, gleichzeitig gibt es aber derart viele

[83] Die Übersicht der Ergebnisse der Gesamtbetrachtung samt Kommentaren ist im Anhang unter „Gesamtbetrachtung“ zu finden.

verschiedenen Elemente, dass die Seite insgesamt unruhig wirkt. Verschiedene Elemente sind Texte, Videos, Grafiken, Code-Boxen, komplett wechselnde Layouts der Bereiche usw.

Der Erstplatzierte der kriteriengestützten Untersuchung belegt in der Gesamtbetrachtung den fünften Platz (MOMR). Die Dokumentation von MOMR ist technisch sehr gut, optisch ist sie jedoch unruhig und wirkt teilweise zerstückelt. Das könnte daran liegen, dass die API-Dokumentation Teil einer sehr großen Webseite ist. Die Hauptwebseite steht konsequent optisch im Vordergrund und gibt der Dokumentation eine verschachtelte Optik. Außerdem werden viele Farben und fettgedruckte Hervorhebungen eingesetzt.

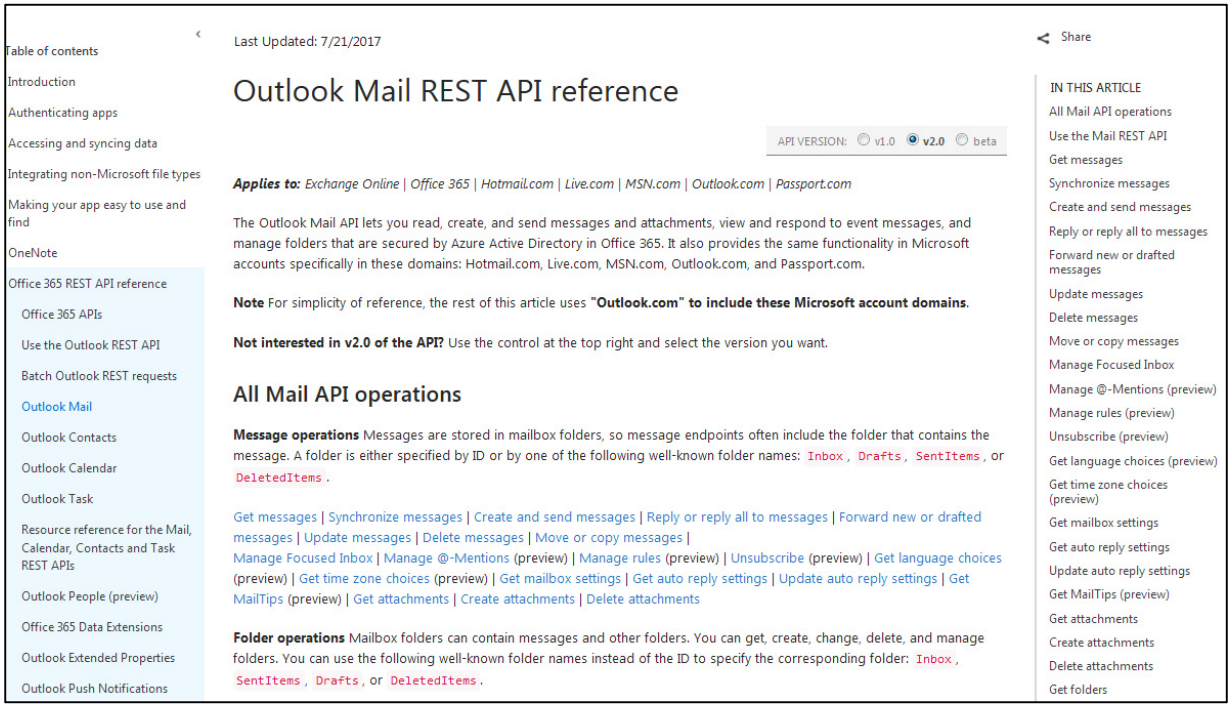


Abb. 92: Optisch unruhige Dokumentation der Microsoft Office Mail Rest API (MOMR)

Auf dem Platz 9 der Gesamtbetrachtung ist die The Budget your Trip API (TBYT), die im kriteriengestützten Ranking lediglich Platz 16 erreicht hat. Aus technischer Sicht hat diese Dokumentation viele Schwächen, gerade die Startseite und das simple Menü überzeugen aber optisch. Die visuelle Inkonsistenz der Kapitel stört, trotzdem ist die Dokumentation übersichtlicher als z. B. die von MOMR. TBYT erfüllt vor allem inhaltlich relevante Aspekte wie eine Intelligent Suchfunktion nicht, trotzdem ist sie aus gestalterischer Sicht gut.

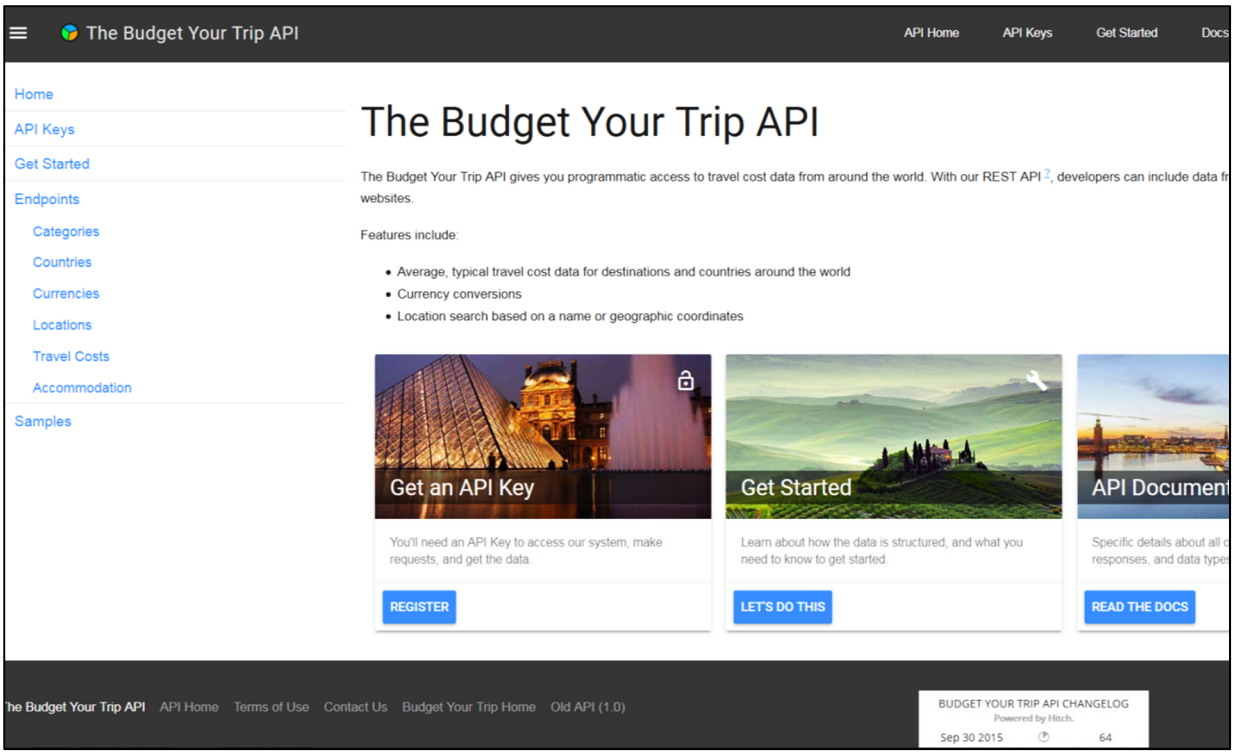


Abb. 93: Gestalterisch gut umgesetzte Dokumentation mit technischen Schwächen (TBYT)

Einen noch größeren Sprung legt die New York Times Books API (NYTB) hin, die von Platz 21 auf Platz 10 in der Gesamtbetrachtung springt. Ausschlaggebend für die gute Bewertung im vorderen Drittel waren die sehr gute Übersichtlichkeit und Bedienbarkeit der codeorientierten Dokumente. Besonders gefallen hat die direkte Weiterleitung in die Konsole aus den Funktionsbeschreibungen. Auf diese Weise werden die Klick-Pfade verkürzt und eine sehr gute Weiterleitung innerhalb der Dokumentation geschaffen.

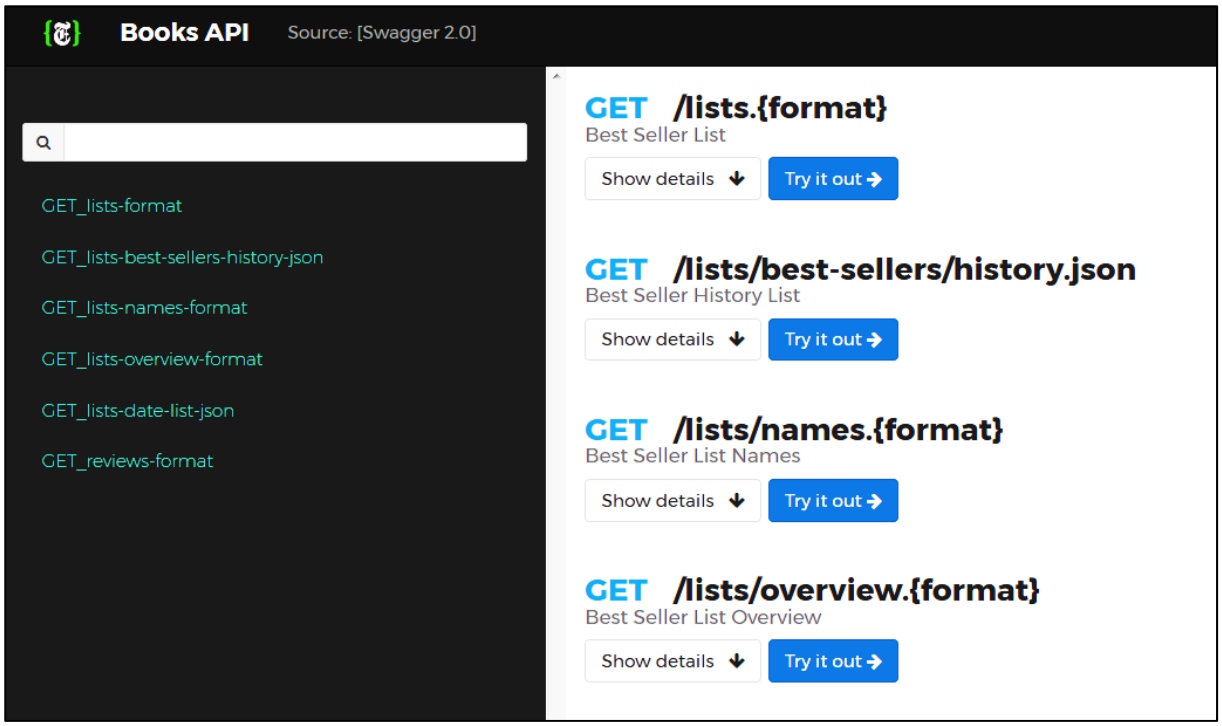


Abb. 94: „Try it out" – Button in den Funktionsbeschreibungen (NYTB)

Deutlich verbessert hat sich auch die Streetlayer API (SL), die mit einer neutralen Wirkung von Platz 26 auf Platz 16 sprang. Sie verfügt über keine besonderen Features, diese konnten somit aber auch nicht ablenken oder die Dokumentation unübersichtlich machen.

Eine API, die sich deutlich verschlechtert hat, ist Etsy. Sie rutscht von Platz 6 auf Platz 14. Das liegt vor allem an langen Scrollwegen und einer nicht mitwandernden Navigation, die mittig über dem Text platziert ist. In der Navigation gibt es zudem sehr viele Einträge, die mit Registerkarten umgeschaltet werden müssen. Das erschwert einen Gesamtüberblick maßgeblich. Die Aufteilung der Navigation ist zwar nachvollziehbar, manche Kapitel wirken aber trotzdem etwas versteckt. Eine Beispiel dafür ist das „Beginner´s Tutorial“, das nicht in der ersten Registerkarte gezeigt wird, sondern unter „Developer Resources“. Die Platzierung dieser Verlinkung ist nicht optimal gewählt, da sie gerade für Einsteiger relevant ist und deswegen direkt auf der Startseite präsentiert werden sollte.

Extrem verschlechtert hat sich ebenfalls die Kaltura API, da Sie von Platz 7 auf Platz 20 in der Gesamtbetrachtung rutscht. Problematisch sind vor allem die sehr verschachtelte Navigation und die unübersichtlichen Kapitel. Die verschachtelte Navigation wird per Klick ausgeklappt, bei einem neuen Klick klappt sich das vorherige Kapitel wieder zu. Durch diese Gestaltung ist es nicht möglich, eine Gesamtübersicht der Funktionen oder zumindest der Funktionsthemen einzusehen. In den Kapiteln müssen viele Inhalte per Klick ausgeklappt werden, obwohl genug Platz für sie vorhanden wäre. Das ist umständlich und unübersichtlich, weil der Nutzer keinen schnellen Überblick gewinnen kann. Durch diese Gestaltung kommen lange Klick-Pfade zustande, die nur schwer nachvollzogen werden können. Außerdem ist die Schrift sehr klein und die Funktionen sind nur bedingt deskriptiv benannt.

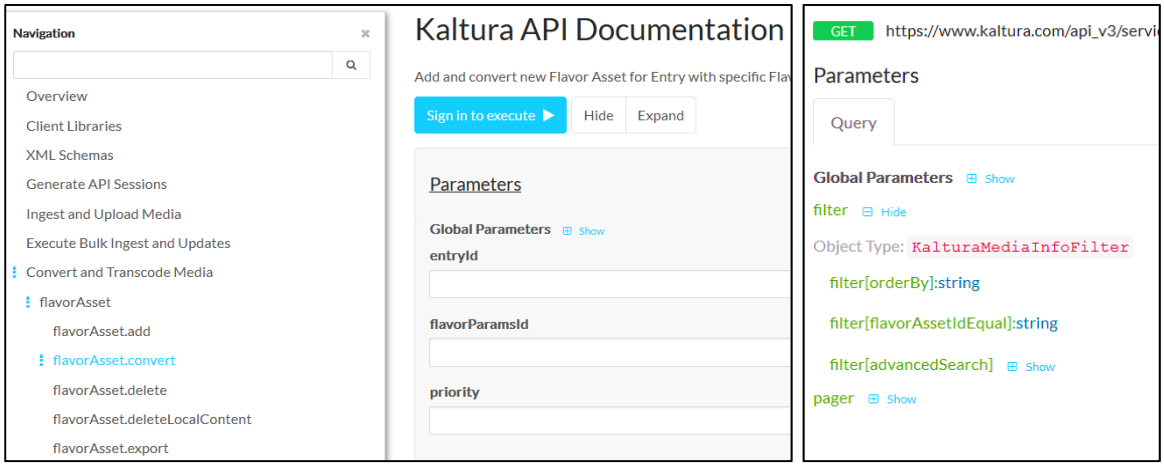


Abb. 95: Aufwendig ausgeklappte Navigation und per Klick ausklappbare Inhalte (v. l. n. r.; Kaltura)

Ebenfalls verschlechtert haben sich die Reddit und die Flickr API, die von Platz 11 auf Platz 21 und 23 gefallen sind. Bei Reddit ist die extrem lange Navigation problematisch, die nicht mitwandert oder dynamisch ist. Die Onepage von Reddit ist insgesamt sehr lang, so dass die Übersichtlichkeit komplett verloren geht. Außerdem waren die Inhalte der Kapitel zwar konsistent, es waren aber nicht in jedem Fall alle Elemente vorhanden. Dadurch entstand z. T. ein unruhiges Bild.

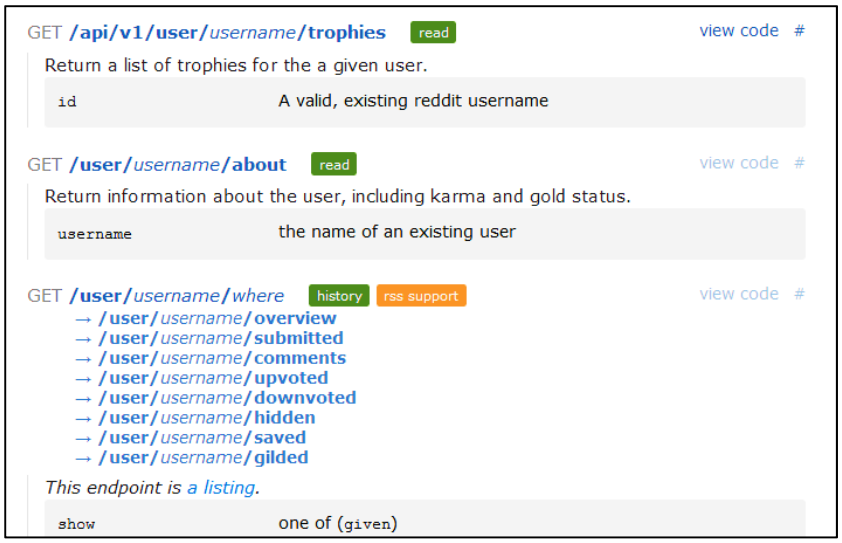


Abb. 96: Konsistente Kapitel, die durch wechselnde Elemente unübersichtlich werden (Reddit)

Die Dokumentation der Flickr API wirkt insgesamt überladen und hat keine nachvollziehbare Struktur. Die Klickpfade sind völlig unklar und werden sowohl über sporadisch vorhandene, klassische Steuerungselemente als auch über Verlinkungen im Text angeboten. Die Aufteilung der Inhalte auf die verschiedenen Seiten wirkt zusätzlich nicht durchdacht.

5.1. Zusammenfassung

Insgesamt hat die Gesamtbetrachtung ergeben, dass eine klare visuelle Gestaltung sehr wichtig ist. Dabei spielen die Aspekte Layout, Farbigkeit, Typografie und Strukturierung eine große Rolle. Die Inhalte müssen klar voneinander getrennt sein, das gilt vor allem für Onepages und PDFs. Die Dokumentationen sollten nicht zu klein geschachtelt werden, das Integrieren in eine große Webseite ist ebenfalls ungünstig.

Die in der Gesamtbetrachtung als gut eingestuften Forschungsobjekte erfüllen vor allem die Heuristiken „Transparente Navigation“, „Konsistente Gestaltung“, „Grundregeln guter Dokumentation“ sowie zum Teil „Selektiver Zugriff auf konzeptuelle Informationen“. Diese Heuristiken beziehen sich zum Großteil auf die Gestaltung der Dokumentationen, so dass eine hohe Übereinstimmung als gut in der Gesamtbetrachtung angesehen werden kann.

Die Gesamtbetrachtung zeigt eine Perspektive für die Gestaltung von API-Dokumentation auf. Tatsächlich muss eine Dokumentation, die kriteriengestützt sehr gut abgeschnitten hat, nicht automatisch auch in der Gesamtbetrachtung gut abschneiden. Diese Divergenz unterstreicht, dass einige Kriterien für die Qualität mehr gewichtet werden müssten als andere. Eine Dokumentation mit Suchfunktion, die gleichzeitig über eine unklare Struktur und nicht deskriptive Überschriften verfügt, steigert ihre Qualität durch eine Suchfunktion nicht.

Die Gewichtung einzelner Kriterien könnte ein mögliches Forschungsziel für zukünftige Studien sein. Durch das Einbeziehen von Entwicklern in der Expertenrolle könnten die durch die Gesamtbetrachtung heuristisch erarbeiteten Hypothesen empirisch eruiert werden.

6. Konklusion

Die durchgeführten Untersuchungen haben gezeigt, dass eine knappe Mehrheit der gesuchten Kriterien in den Forschungsobjekten vorkam. Die Forschungsfrage *„Können die erarbeiteten*

Heuristiken in API-Dokumentationen aus der Praxis nachgewiesen werden?“ konnte somit verifiziert werden.

Einige Heuristiken wurden häufiger, andere weniger häufig erfüllt. Besonders häufig erfüllt wurden solche, die sich auf die Grundlagen der Dokumentationserstellung beziehen. Dazu gehören eine konsistente Gestaltung, die Grundregeln guter Dokumentation, eine transparente Navigation und eine durchdachte Struktur. Die Heuristiken, die über die interaktiven und stark usability-orientierten Kriterien verfügten, wurden tendenziell weniger erfüllt. Dazu gehören der schnelle Einstieg, der selektive Zugriff auf konzeptuelle Informationen, die intelligente Suchfunktion und die codenahe Platzierung von Informationen.

Problematisch in der Bewertung war vor allem die schwankende Qualität der Inhalte, so dass Kriterien nur grenzwertig mit „Ja“ oder „Nein“ bewertet werden konnten. Eine abgestufte Bewertungsskala könnte diese Problematik in zukünftigen Untersuchungen entschärfen.

Die Ergebnisse der Untersuchung verdeutlichen den Handlungsbedarf in der Erstellung von API-Dokumentation. Während grundlegende Anforderungen tendenziell gut bewältigt werden, besteht Bedarf für die Entwicklung von innovativen und arbeitssparenden Konzepten zur Berücksichtigung und Umsetzung unterschiedlicher Lernansätze. Im Rahmen dieser Arbeit wurden mögliche Themen für die zukünftige Forschung an API-Dokumentation heuristisch erarbeitet, die nachfolgend vorgestellt werden. Das Ziel ist es, die Qualität und Usability von API-Dokumentationen langfristig zu verbessern, sowohl für die Autoren als auch für die Anwender.

6.1. Ausblick

Die Untersuchung hat im Rahmen der heuristischen qualitativen Vorgehensweise zahlreiche Hypothesen hervorgebracht, die in zukünftigen Studien untersucht werden könnten. Dazu gehören die Komplexität und Linearität der Forschungsobjekte, die Gewichtung von Kriterien, die Anwendung von Frameworks oder die Auslagerung von Inhalten auf externe Webseiten.

6.1.1. Komplexität und Linearität

In dieser Arbeit wurden die Forschungsobjekte im Hinblick auf Umfang und Medium unterschieden. Neben diesen formalen Aspekten könnte ebenfalls im Hinblick auf Komplexität und Linearität unterschieden werden. Gerade im Fall der Linearität stellt sich die generelle Frage, inwiefern lineare Dokumentationen noch dem Zeitgeist entsprechen. Entwickler arbeiten am PC und sind nicht-lineare Informationsdarstellungen gewöhnt, so dass sie sich des Skimmens und Scannens bedienen. Eine lineare Dokumentation könnte den Entwickler deshalb in seiner gewohnten Arbeitsweise irritieren und sogar behindern.[84] Andererseits können lineare Strukturen als bekannt vorausgesetzt werden, da Bücher und andere klassische Informationsmedien linear aufgebaut sind. Die Frage nach der Usability und Akzeptanz von Linearität unter Entwicklern könnte ein interessanter Forschungsansatz sein.

Im Hinblick auf die Komplexität wäre eine Einstufung bestimmter Kriterien denkbar. Beispielsweise müssten sehr einfache API-Dokumentationen eher keine technischen Besonderheiten erhalten (wie

[84] Grundsätzlich ist jedes Arbeiten oder Denken linear, so lange es sprachlich vollzogen wird. Menschliche Sprache hat einen linearen Charakter, weil alle Wörter chronologisch hintereinander gesprochen/gedacht werden müssen. Die Linearität der Dokumentation bezieht sich in diesem Zusammenhang auf ihre Gliederung, die entweder aufeinander aufbauend oder gleichwertig nebeneinanderstehend vernetzt sein kann.

die codenahe Platzierung von Informationen). Komplexe Dokumentationen wiederum würden von dem Einsatz einiger Kriterien stark profitieren. Das könnte u. a. für die codenahe Platzierung von Informationen oder den selektiven Zugriff auf Informationen gelten. Die Erstellung einer API-Dokumentation könnte daran angelehnt in verschiedene Umsetzungsstufen in Abhängigkeit von der Komplexität gegliedert werden.

1. Sprache, Struktur, Optik und Konsistenz
2. Verweise und unterstützende Bedienelemente
3. Interaktivität

Die erste Stufe könnte Kriterien wie kurze/prägnante Formulierungen, prozessorientierte Gliederung, Gestaltungsgrundlagen (Farbe, Form, Aufteilung) sowie die visuelle und inhaltliche Konsistenz enthalten. Die inhaltliche Konsistenz greift bereits in die zweite Stufe, in der die Dokumentation inhaltlich aufeinander verweist und so ein Netz aus Informationen bildet, das über die lineare Darstellung der Stufe eins hinausgeht. Zusätzlich können Kriterien der codenahen Platzierung von Informationen und des selektiven Zugriffs auf Informationen hier angewendet werden. In der dritten Stufe werden interaktive Kriterien umgesetzt, dazu gehören dynamische Navigationen, Demo-Funktionen für Code-Beispiele usw.

Das vorgeschlagene Stufenmodell würde die Erstellung von Dokumentation in drei Stufen einteilen, die einen unterschiedlichen Aufwand bedeuten. Eine Dokumentation wäre bereits qualitativ gut, wenn die Kriterien der ersten Stufe erfüllt sind. Für kurze oder simple APIs wäre eine Umsetzung in dieser Stufe vermutlich ausreichend. Die zweite und die dritte Stufe bedeuten nicht nur technisch einen erhöhten Aufwand, sondern auch einen Aufwand in Bezug auf die inhaltliche Durchdringung der Thematik. Wie in der Auswertung erwähnt, müssen Kriterien der codenahen Platzierung von Informationen oder des selektiven Zugriffs auf Informationen durch Konzepte detailliert geplant werden, damit der gewünschte Mehrwert abgeschöpft werden kann. Dokumentationen, die die zweite und dritte Stufe erfüllen, wären vor allem für komplexe oder sehr umfangreiche APIs geeignet.

6.1.2. Gewichtung der Kriterien

Eine weitere Hypothese, die zukünftig untersucht werden könnten, ist die Frage nach der Gewichtung der Kriterien. Besonders in der Gesamtbetrachtung fiel auf, dass einige Kriterien für die Anwenderfreundlichkeit von Dokumentation wichtiger sind als andere. Eine interessante Perspektive wäre außerdem das Aufdecken von Abhängigkeiten unter den Kriterien. Es wäre z. B. denkbar, dass eine Suchfunktion ohne prozessorientierte Überschriften oder eine deskriptive Namensgebung eher nicht produktiv nutzbar ist.

6.1.3. Frameworks

Einige der untersuchten Dokumentationen wurden mit Frameworks erstellt, die von Entwicklern generell gut angenommen werden (s. a. Dusold 2016). Während der Untersuchung ist aufgefallen, dass nicht alle Dokumentationen, die beispielsweise mit dem Swagger Framework erstellt wurden, über die gleiche visuelle Umsetzung verfügten. Der Mehrwert im Hinblick auf die Bekanntheit der Strukturen und das damit verbundene bessere Zurechtfinden wird durch die visuellen Abweichungen in Frage gestellt. Ebenfalls problematisch ist der sehr stark codeorientierte Ansatz der Frameworks. Inwiefern in ihnen konzeptuelle Informationen platziert werden können sowie die Frage des Mehrwerts von Framework Dokumentationen könnten interessante Forschungsansätze sein.

6.1.4. Auslagerung von Inhalten auf externe Webseiten

In vielen Fällen wurden die Dokumentationen durch Inhalte auf externen Webseiten ergänzt. Am häufigsten wurden Inhalte auf Github ausgelagert, dabei handelte es sich meist um codebasierte Informationen oder die Möglichkeit, in einem Forum Fragen zu stellen und zu kommunizieren. Die Bewertung der ausgelagerten Inhalte war im Rahmen dieser Arbeit nur begrenzt möglich. Die Systematik der ausgelagerten Inhalte sowie deren generelle Motivation wären ein interessantes Forschungsthema. Außerdem könnte die Akzeptanz der Entwickler gegenüber der Auslagerung der Inhalte sowie der Usability von Github untersucht werden.

7. Quellen

7.1. Literaturquellen

Aho, Alfred; Lam, Monica; Sethi, Ravi; Ullmann, Jeffrey (2008): Compiler. Prinzipien, Techniken und Werkzeuge. München: Pearson Studium.

Antos, Gerd; Hasler, Ursula; Perrin, Daniel (2011): Textoptimierung. In: Habscheid, S. (Hrsg.): Textsorten, Handlungsmuster, Oberflächen: Linguistische Typologien der Kommunikation. Berlin, New York: Walter de Gruyter. S. 638 – 658.

Bengel, Günther (2015): Grundkurs Verteilte Systeme: Grundlagen und Praxis des Client-Server und Distributed Computing. Berlin, Wiesbaden: Springer Verlag.

Bergstrom, Jennifer; Schall, Andrew (2014): Eye Tracking in User Experience Design. Amsterdam: Elsevier.

Borchardt, Andreas; Göthlich, Stephan (2009): Erkenntnisgewinnung durch Fallstudien. In: S. Albers; D. Klapper; U. Konradt; A. Walter; J.Wolf (Hrsg.): Methodik der empirischen Forschung. Wiesbaden: Springer Fachmedien. S. 33 - 48.

Bortz, Jürgen; Döring, Nicola (2007): Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler. Heidelberg: Springer Medizin Verlag.

Diggle, Peter; Heagerty, Patrick; Liang, Kung-Yee; Zeger, Scott (2013): Analysis of Longitudinal Data. Oxford: Oxford University Press.

Digitale Gesellschaft (2012): Wie das Internet funktioniert. Eine Anleitung für Entscheidungsträger und Interessierte. Berlin: Digitale Gesellschaft e.V.

Dusold, Dusold (2016): API-Dokumentation aus der Perspektive von Software-Entwicklern: Was macht sie besonders hilfreich? Masterarbeit im Fach Technische Redaktion und Wissenskommunikation an der Hochschule Merseburg.

Fielding, Roy T. (2000): Architectural Styles and the Design of Network-based Software Architectures. Dissertation im Fach Information and Computer Science an der University of California, Irvine.

Fischer, Peter; Hofer, Peter (2010): Lexikon der Informatik. Berlin, Heidelberg: Springer Verlag.

Friendly, Lisa (1995): The Design of Distributed Hyperlinked Programming Documentation. In: S. Fraisse; F. Garzotto; T. Isakowitz; J. Nanard; M. Nanard (Hrsg.): Hypermedia Design. Proceedings of the International Workshop on Hypermedia Design (IWHD'95) Montpellier. Berlin, New York: Springer Verlag. S. 151 - 173.

Galloway, Matt (2014): Effektiv Objective-C 2.0 programmieren: 52 Profi-Lösungen für bessere iOS- und OS-X-Programmierung. Heidelberg: dpunkt.verlag.

Gellenbeck, Edward M.; Cook, Curtis R. (1991): An Investigation of Procedure and Variable Name as Beacons during Program Comprehension. In: J. Koenemann-Bellinveau; T.G. Mohen; S.P. Robertson (Hrsg.): *Empirical Studies of Programmers: Fourth Workshop.* Norwood NJ: Ablex. S. 65 - 81.

Grünwied, Gertrud (2013): Software-Dokumentation. Grundlagen – Praxis – Lösungen. Renningen: Expert Verlag.

Häder, Michael (2015): Empirische Sozialforschung. Eine Einführung. Wiesbaden: Springer Fachmedien.

Höld, Regina (2009): Zur Transkription von Audiodaten. In: R. Buber; H. Holzmüller (Hrsg.): *Qualitative Marktforschung: Konzepte - Methoden – Analysen.* Wiesbaden: Gabler. S. 655 - 668.

Jacobsen, Jens (2005): Website-Konzeption. Erfolgreiche Web- und Multimedia-Anwendungen entwickeln. Bonn: Addison-Wesley.

Jacobson, Daniel; Brail, Greg; Woods, Dan (2012): APIS: A strategy guide. Sebastopol, Kalifornien: O´Reilly.

Kuckartz, Udo (2005): Einführung in die computergestützte Analyse qualitativer Daten. Wiesbaden: VS-Verlag.

Laudon, Kenneth; Laudon, Jane; Schoder, Detlef (2010): Wirtschaftsinformatik: Eine Einführung. München: Pearson Studium.

Meng, Michael; Steinhardt, Stephanie; Schubert, Andreas (2016): API Documentation: Exploring the information needs of software developers. Forschungsarbeit im Fach Technische Redaktion an der Hochschule Merseburg.

Rocher, Graeme; Brown, Jeff (2010): Grails 1.2: das produktive Web-Framework für die Java-Plattform. Heidelberg, München: mitp.

Rüping, Andreas (2013): Dokumentation in agilen Projekten. Lösungsmuster für ein bedarfsgerechtes Vorgehen. Heidelberg: dpunkt.verlag.

Stylos, Jeffrey (2009): Making APIs More Usable with Improved API Designs, Documentation and Tools. Ann Arbor MI, London, Berlin: Pro Quest LLC.

Tapscot, Don (1997): Die digitale Revolution: Verheißungen einer vernetzten Welt - die Folgen für Wirtschaft, Management und Gesellschaft. Wiesbaden: Gabler Verlag.

Bellem, Birgit; Dreikorn, Johannes; Fleury, Isabelle; Haldimann, Ralf; Klemm, Viktoria; Kurrus, Matthias; Prusseit, Ines; Reuther, Ursula (2013): Leitlinie Regelbasiertes Schreiben. Deutsch für die Technische Kommunikation. Stuttgart: Tekom.

Tergan, Sigmar-Olaf (2004): Realistische Qualitätsevaluation von E-Learning. In: D. Meister; S.O. Tergan; P. Zentel (Hrsg:): Evaluation von E-Learning. Zielrichtungen, methodologische Aspekte, Zukunftsperspektiven. Münster, New York, München: Waxmann. S. 131 - 154.

Thiemann, Petra (2008): Benutzerfreundliche Online-Hilfen. Grundlagen und Umsetzung mit MadCap Flare. Wiesbaden: Vieweg+Teubner Verlag | GWV Fachverlag.

Tritsch, Bernhard (1997): Verteiltes Lernen in Computernetzen. Eine Tele-Media-Trainingsarchitektur. Berlin, Heidelberg: Springer Verlag.

Walsh, Gianfranco; Kilian, Thomas; Hass, Bethold H. (2011): Grundlagen des Web 2.0. In: G. Walsh; T. Kilian; B.H. Hass (Hrsg.): *Web 2.0: Neue Perspektiven für Marketing und Medien.* Heidelberg, Berlin: Springer Verlag. S. 3 - 20.

Wörterbuch für Lexikographie und Wörterbuch-Forschung (2010): Wörterbuch für Lexikographie und Wörterbuch-Forschung. Mit englischen Übersetzungen der Umtexte und Definitionen sowie Äquivalenten in neun Sprachen. Berlin, New York: De Gruyter.

Woolfolk, Anita (2008): Pädagogische Psychologie. München: Pearson Studium.

Zehrer, Christiane (2014): Wissenskommunikation in der technischen Redaktion. Die situierte Gestaltung adäquater Kommunikation. Berlin: Frank & Timme.

7.2. Quellen aus Zeitschriften

Bednarczyk, Marta; Queins, Stefan (2013): Dokumentation in agilen Projekten – so geht's. In: *Projekt Magazin, Das Fachportal für Projektmanagement.* Jg. 2013, Nr. 10, S. 1 - 8.

Blinman, Scott; Cockburn, Andy (2005): Program Comprehension: Investigatin the Effexts of Naming Style and Documentation. In: *AUIC '05 Proceedings of the Sixth Australasian conference on User interface.* Nr. 40, S. 73 – 78.

Buse, Raymond P.L; Weimer, Westley (2012): Synthesizing API Usage Examples. In*: ICSE '12 Proceedings of the 34th International Conference on Software Engineering.* S. 782 - 792.

Brooks, Ruven (1983): Towards a theory of the comprehension of computer programs. In: *International Journal of Man-Machine Studies.* Jg. 8, Nr. 6, S. 543 - 554.

Crosby, Martha E.; Scholtz, Jean; Wiedenbeck, Susan (2002): The Roles Beacons Play in Comprehension for Novice and Expert Programmers. In: *Proceedings of the Fourteenth Annual Workshop of the Psychology of programming Interest Group London, UK.* S. 58 - 73.

Jeong, Sae Young; Xie, Yingyu; Beaton, Jack; Myers, Brad A.; Stylos, Jeff; Ehret, Ralf; Karstens, Jan; Efeoglu, Arkin; Busse, Daniela K. (2009): Improving Documentation for eSOA APIs through User Studies. In*: Pipek, V. et al.: International Symposium on End-User Development IS-EUD 2009, Lecture Notes in Computer Science LNCS 5435.* Berlin, Heidelberg: Springer Verlag. S. 86 - 105.

Ko , Andrew J.; DeLine, Robert; Venolia, Gina (2007): Information Needs in Collocated Software Development Teams. In: *ICSE '07 Proceedings of the 29th international conference on Software Engineering.* S. 344 - 353.

Ko, Andrew J.; Riche, Jane (2011): The Role of Conceptual Knowledge in API Usability. In: *IEEE Symposium on Visual Languages and Human-Centric Computing VL/HCC.* S. 173 - 176.

Lethbridge, Timothy C.; Singer, Janice; Forward, Andrew (2003): How Software Engineers Use Documentation: The State of the Practice. In: *Journal IEEE Software.* Nr. 20, S. 35 - 39.

Marvin, Scott (2014): What is API-Documentation? In: *STC intercom.* September 2014, S. 6 - 8.

Meng, Michael; Steinhardt, Stephanie; Schubert, Andreas (2017): API Documentation: What do Software Developers want? Zum Druck angenommen im Journal of Technical Writing and Communication.

Michalkiewicz, Martha (2015): Wie Heuristiken uns helfen Entscheidungen zu treffen. In: *The inquisitive mind. Kognitionspsychologie im Alltag. Teil 2: Lernen und Gedächtnis.* Nr. 4.

Mühlfeld, Claus ; Windolf, Paul ; Lampert, Norbert ; Krüger, Heidi (1981): Auswertungsprobleme offener Interviews. In: *Soziale Welt*. Jg. 32, Nr. 3, S. 325-352.

Novick, David G.; Ward, Karen (2006[a]): Why don't people read the manual? In: *Departmental Papers (CS).* Nr. 15.

Novick, David G.; Ward, Karen (2006[b]): What users say they want in documentation. In: Departmental Papers (CS). Nr. 13.

Nykaza, Janet; Messinger, Rhonda; Boehme, Fran; Norman, Cherie L.; Mace, Matthew; Gordon, Manuel (2002): What programmers really want: results of a needs assessment for SDK documentation. In: SIGDOC '02 Proceedings of the 20th annual international conference on Computer documentation. S. 133 – 141.

Robillard, Martin P. (2009): What makes apis hard to learn? Answers from developers. In: IEEE Software. Jg. 26, Nr. 6, S. 27 - 34.

Robillard, Martin P; Dagenais, Barthélémy (2010): Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributors. In: *FSE '10 Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering.* S. 127 – 136.

Robillard, Martin P.; DeLine, Robert (2011): A field study of API learning obstacles. In: *Journal of Empirical Software Engineering.* Nr. 6, S. 703 - 732.

Robillard, Martin P.; Maalej, Walid (2013): Patterns of Knowledge in API Reference Documentation. In: *IEEE Transactions on Software Engineering doi:10.1109/TSE.2013.12.* Jg. 39, S. 1264 - 1282

Shneiderman, Ben; Mayer, Richard (1979): Syntactic/semantic interaction in programmer behavior: a model and experimental results. In: *International Journal of Computer and Information Sciences.* Nr. 8, S. 219 - 283.

Shneiderman, Ben (1977): Measuring computer program quality and comprehension. In: *International Journal of Man-Machine Studies.* Nr. 9, S. 465 - 478.

Silito, Jonathan; Begel, Andrew (2013): App-Directed Learning: An Exploratory Study. In: *MOBILESoft 2014 Proceedings of the 1st International Conference on Mobile Software Engineering and Systems.* S. 50 - 53.

Sridhara, Giriprasad; Pollok, Lori; Vijay-Shanker, K. (2011): Generating Parameter Comments and Integrating with Method Summaries. In: *ICPC '11 Proceedings of the 2011 IEEE 19th International Conference on Program Comprehension.* S. 71 - 80.

Stylos, Jeffrey; Faulring, Andrew; Yang, Zizhuang; Myers, Brad A. (2009): Improving API Documentation Using API Usage Information. In: *IEEE Symposium on Visual Languages and Human-Centric Computing VL/HCC.* S. 119 - 126.

Teasley, Barbee E. (1994): The effects of naming style and expertise on program comprehension. In: *International Journal of Human-Computer Studies.* Nr. 40, S. 757 - 770.

Watson, Robert (2009): Improving Software API Usability through Text Analysis: A Case Study. In*: Professional Communication Conference, IEEE International.* S. 1 - 7.

Watson, Robert (2012): Development and Application of a Heuristic to Assess Trends in API Documentation. In: *Proceedings of the 30th ACM international conference on Design of communication.* S. 195 – 203.

Watson, Robert; Stamnes, Mark; Jeannot-Schroeder, Jacob; Spyridakis, Jan H. (2013): API documentation and software community values: a survey of open-source API documentation. In: *SIGDOC '13 Proceedings of the 31st ACM international conference on Design of communication.* S. 165 - 174.

Wiedenbeck, Susan (1986): Beacons in computer program comprehension. In: *International Journal of Man-Machine Studies.* Nr. 25, S. 697 - 709.

7.3. Internetquellen

1&1 (2017): Intelligente Suche: Eine userfreundliche Suchfunktion für Ihren Onlineshop (online). https://hosting.1und1.de/digitalguide/websites/webseiten-erstellen/intelligente-suche-alles-zur-usability-der-suchfunktion/ [10.03.2017].

18F (o. J.): Digital Service Delivery. An official website of the United States Government (online). https://18f.gs a.gov/ [17.03.2017].

Adobe (2016): Dynamische PDF-Dokumente (online). https://helpx.adobe.com/de/indesign/using/dynamic-pdf-documents.html [26.05.2017].

ANX Exchange API v3 (o. J.): Documentation (online). http://docs.anxv3.apiary.io/# [17.03.2017].

ANXPRO (2016): About us (online). https://anxpro.com/pages/about [17.03.2017].

Arlo (2015): REST Public API (online). https://developer.arlo.co/doc/api/2012-02-01/pub/ [17.03.2017].

Becker, Hannah (2015): Was ist ein Widget? Einfach erklärt (online). http://praxistipps.chip.de/was-ist-ein-widget-einfach-erklaert_42003 [08.03.2017].

Berkemeyer, Kim (2017): Google Maps als Navi: Ein Vorteil lässt der Konkurrenz (fast) keine Chance (online). http://www.chip.de/news/Google-Maps-Deshalb-kennt-das-Navi-den-Verkehr-besser-als-alle-anderen_ 99347317.html [29.03.2017]

boxDevelopers (o. J.): API Reference (online). https://docs.box.com/reference#api-docs-directory [15.03.2017].

Budget Your Trip (o. J.): The Budget Your Trip API (online). http://www.budgetyourtrip.com/api/v3/ [17.03.2017].

Carta, David (2013): Park-Miller-Carta Pseudo-Random Number Generator (online). http://www.firstpr.com.au/dsp/rand31/ [30.03.2017].

Chambers, John (2012): Internet of Everything: Fueling an Amazing Future #TomorrowStartsHere (online). http://blogs.cisco.com/news/internet-of-everything-2 [23.06.2017].

CMS made simple (2017): Was ist ein CMS? (online). http://www.cmsmadesimple.de/simples-php-cms-als-open-source/was-ist-ein-cms.html [15.03.2017].

Commerce.js (o. J.): Documentation (online). https://commercejs.com/docs/ [19.03.2017].

Commerce.js (2016): Full-stack eCommerce API for developers & designers (online). https://commercejs.com/ [19.03.2017].

Cremer, Gino (2016): Schnellere Websites dank Caching – Was ist Caching? (online). https://pixelbar.be/ blog/schnellere-websites-dank-caching-was-ist-caching/ [09.03.2017].

D&B Developer (o. J.): Fliptop Person API Documentation (online). https://developer.fliptop.com/documentation [15.03.2017].

Docker (2017): What is Docker? (online). https://www.docker.com/what-docker [17.03.2017].

Docker Cloud (o. J.): API Reference (online). https://docs.docker.com/apidocs/docker-cloud/#introduction [17.03.2017].

dpBestflow (2015): Directory Structure (online). http://www.dpbestflow.org/file-management/directory-structure [04.01.2017].

Elektronik Kompendium (o. J.): WWW – World Wide Web (online). http://www.elektronik-kompendium .de/sites/net/0906091.htm [23.07.2017].

Etsy (2017): API Documentation (online). https://www.etsy.com/developers/documentation/ [17.03.2017].

Facebook (o. J.): Facebook Login für deine Apps & Webseiten (online). https://developers.facebook.com/ products/login [12.10.2016].

flickr (o. J.[a]): Der App Garden (online). https://www.flickr.com/services/api/ [15.03.2017].

flickr (o. J.[b]): Der Entwicklerleitfaden (online). https://www.flickr.com/services/developer/ [17.11.2016].

Frank, Jessica (2016): Hochschule Merseburg Was zieht Schwaben nach Merseburg? (online). http://www.mz-web.de/merseburg/hochschule-merseburg-was-zieht-schwaben-nach-merseburg--24710668 [01.03.2017].

Gerken, Jens (2003): Validität und Aussagekraft von Usability Test Methoden (online). http://hci.uni-konstanz.de/hausarbeiten/praktika/Usability_Test_Methoden.pdf [21.10.2016].

Google (2015): Google Maps, Google Earth und Street View verwenden (online). https://www.google.de/intl/de/permissions/geoguidelines.html [30.03.2017].

Google Books (o. J.): Ergebnis 1 von 48 für Heuristik in diesem Buch (online). https://books.google.de/ books?id=2nlLCgAAQBAJ&printsec=frontcover&dq=heuristik&hl=de&sa=X&sqi=2&pjf=1&ved=0ahUKEwjB2detkczSAhUGahoKHRPyATQQ6AEIGjAA#v=onepage&q=heuristik&f=false [10.03.2017].

Google Maps APIs (o. J.): Nutzern bei der Navigation helfen (online). https://developers.google.com/maps/ documentation/directions/?hl=de [17.03.2017].

Gräsel, Alexander (2012): Softwarearchitektur: Umgebungsabhängigkeit (online). http://blog.axxg.de/software architektur-umgebungsunabhaengigkeit/ [24.03.2017].

Gründerszene (o. J.): Application-Programming-Interface (API). (online). http://www.gruenderszene.de/ lexikon/begriffe/application-programming-interface-api?ref=interstitial [12.10.2016].

Gründerszene (2009): Web APIs. Ein nicht-technischer Erklärungsversuch. (online). http://www.gruenderszene .de/allgemein/web-apis-ein-nicht-technischer-erklarungsversuch?ref=interstitial [12.10.2016].

Guide Star (2015): About Us (online). https://learn.guidestar.org/about-us/ [17.03.2017].

Guide Star (2016): Guide Star Search API Version 1_1. (online). https://community.guidestar.org/docs/DOC-1779-guidestar-apis#jive_content_id_GuideStar_Search_API_Version_1_1 [17.03.2017].

Here (2017[a]): Maps for developers (online). https://developer.here.com/ [17.03.2017].

Here (2017[b]): Weather API Developer's Guide. Overview (online). https://developer.here.com/rest-apis/documentation/weather/topics/overview.html [17.03.2017].

Hery-Mossmann, Nicole (2016[a]): Backend und Frontend - was ist das? Einfach erklärt (online). http://praxis tipps.chip.de/backend-und-frontend-was-ist-das-einfach-erklaert_41384 [04.01.2017].

Hery-Mossmann, Nicole (2016[b]): Was ist ein Framework? - einfach erklärt (online). http://praxistipps .chip.de/was-ist-ein-framework-einfach-erklaert_41348 [09.03.2017].

Highscore (2010): Programmieren in Java: Einführung. Kapitel 4: Funktionen (online). http://www.highscore. de/java/einfuehrung/index.html [03.01.2017].

ITWissen.info (2005): Deployment (online). http://www.itwissen.info/deployment-Softwareverteilung.html [09.03.2017].

ITWissen.info (2013): API (application programming interface) (online). http://www.itwissen.info/API-application-programming-interface-Programmierschnittstelle.html (07.04.2017).

KalturaDeveloper (2016): API Docs. API Overview (online). https://developer.kaltura.com/api-docs/#/Overview [17.03.2017].

Klipfolio (o. J.): Documentation (online). http://apidocs.klipfolio.com/docs [17.03.2017].

Lehrner-Mayer, Karina (2016): Five tips for creating documentation that focuses on the user (online). http://www.tcworld.info/e-magazine/technical-communication/article/five-tips-for-creating-documentation-that-focuses-on-the-user/ [13.10.2016].

McKinsey&Company (2015): Internet der Dinge kann 2025 weltweit bis zu 11 Billionen Dollar Mehrwert schaffen (online). https://www.mckinsey.de/internet-der-dinge-kann-2025-weltweit-bis-zu-11-billionen-dollar-mehrwert-schaffen [19.03.2017].

Microsoft (2017[a]): Verwenden von IntelliSense (online). https://msdn.microsoft.com/de-de/library/hcw1s69b.aspx [17.03.2017].

Microsoft (2017[b]): Outlook Mail REST API reference (online). https://msdn.microsoft.com/en-us/office/office365/api/mail-rest-operations [17.03.2017].

Mihaly, Ferenc (2011): Writing helpful API documentation (online). http://theamiableapi.com/2011/11/01/api-design-best-practice-write-helpful-documentation/ [13.10.2016].

MSXFAQ (2017): REST - Representational State Transfer (online). http://www.msxfaq.de/code/rest.htm [30.03.2017].

Muranko, Beate; Drechsler, Rolf (2006): Technische Dokumentation von Soft- und Hardware-Systemen: Die vergessene Welt (online). http://www.informatik.uni-bremen.de/agra/doc/work/GI_ITG_GMM_tDoku_200 51 .pdf [13.10.2016].

Mutschler, Bela (2015): Suchfunktionen auf Websites (online). https://de.onpage.org/blog/suchfunktionen-auf-websites [15.03.2017].

MySQL (2017): What is MySQL? (online). https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html [15.03.2017].

NYT Developers Network (o. J.): Books API (online). http://developer.nytimes.com/books_api.json#/README [17.03.2017].

Nowak, Harald (2002): Dokumentation in der Software-Entwicklung (online). http://www.infrasoft.at/downloads/Dokumentation_in_der_Softwareentwicklung.pdf [13.10.2016].

Oanda API (2014): Introduction (online). http://developer.oanda.com/rest-live/introduction/ [17.03.2017].

Oanda (2017): Über uns (online). https://www.oanda.com/lang/de/resources/about/ [17.03.2017].

Omkt (o. J.): Was ist eine API? (online). http://www.omkt.de/api/ [12.10.2016].

OnPageWiki (2017): DHTML (online). https://de.onpage.org/wiki/DHTML [08.03.2017].

OpenFEC API (o. J.): OpenFEC API (Beta) Documentation (online). https://api.open.fec.gov/developers/#!/dates/get_calendar_dates [17.03.2017].

Open Secrets (o. J.[a]): Our Vision and Mission: Inform, Empower & Advocate (online). https://www.opensecrets.org/about/ [17.03.2017].

Open Secrets (o. J.[b]): Resource Center: Create (online). https://www.opensecrets.org/resources/create/apis.php [17.03.2017].

Opto 22 (2017): Snap Pac System (online). http://www.opto22.com/site/snap_pac_system.aspx [17.03.2017].

Opto 22 Developer (2017): Getting Started with the SNAP PAC REST API (online). http://developer.opto22.com/rest/pac/ [17.03.2017].

Oracle (o. J.[a]): How to Write Doc Comments for the Javadoc Tool (online). http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html [29.03.2017].

Oracle (o. J.[b]): Javadoc FAQ (online). http://www.oracle.com/technetwork/java/javase/documentation/index-137483.html [24.02.2017].

Parnin (2013): API Documentation (online). http://blog.ninlabs.com/2013/03/api-documentation/ [20.10.2016].

Parson (2013): Entwicklerdokumentation – das notwendige Übel? (online). https://www.parson-europe.com/de/wissensartikel/202-entwicklerdokumentation-wissen.html [13.10.2016].

Paymill (o. J.): API Reference (online). https://developers.paymill.com/API/index#introduction [15.03.2017].

Photobucket (2010): Photobucket API Introduction (online). http://pic.pbsrc.com/dev_help/WebHelpPublic/PhotobucketPublicHelp.htm [17.03.2017].

Pinterest (o. J.): Pinterest Developers. Help millions of people discover, save and do things they love (online). https://developers.pinterest.com/ [12.10.2016].

ProgrammableWeb (o. J.): Search the largest API Directory in the web (online). https://www.programmableweb.com/category/all/apis?page=3 [17.03.2017].

qz-online, Portal für Qualitätsmanagement (2016): Die Rolle der Dokumentation in Software-Projekten (online). https://www.qz-online.de/qualitaets-management/qm-basics/software-qualitaet/dokumentation/artikel/die-rolle-der-dokumentation-in-software-projekten-258004.html?survey_258004.currentstep=1&_req_id=91476438724340:DA0D5B44FF33F811581A6229502FE7BE1673ACD5 [14.10.2016].

reddit (2016): The conversation starts on reddit (online). https://about.reddit.com/ [17.03.2017].

reddit (2017): API DOKUMENTATION (online). https://www.reddit.com/dev/api [17.03.2017].

Rodriquez, Alex (2015): RESTful Web services: The basics (online). http://www.ibm.com/developerworks/webservices/library/ws-restful/ [02.11.2016].

Rouse (2017[a]): application (online). http://searchsoftwarequality.techtarget.com/definition/application [29.03.2017].

Rouse (2017[b]): Programmierschnittstelle (API) (online). http://www.searchenterprisesoftware.de/definition/Programmierschnittstelle-API [29.03.2017].

Schmeling, Roland; Gocke, Andrea (2013): Software-Dokumentation – Anforderungen, Normen, Wirklichkeit (Norm 8) (online). http://tagungen.tekom.de/fileadmin/tx_doccon/slides/460_Software_Dokumentation_Anforderungen_Normen_Wirklichkeit.pdf [13.10.2016].

SelfHTML (2017[a]): CSS/Einbindung (online). https://wiki.selfhtml.org/wiki/CSS/Einbindung [08.03.2017].

SelfHTML (2017[b]): HTML/Regeln/Kommentare (online). https://wiki.selfhtml.org/wiki/HTML/Regeln/Kommentar [27.03.2017].

Shipcloud (o. J.[a]): Shipcloud Documentation (online). http://web.hs-merseburg.de/~schuber2/sc_doku/shipcloud-doku.html [07.04.2017].

Shipcloud (o. J.[b]): Create webhook request JSON schema (online). https://developers.shipcloud.io/reference/webhooks_request_schema.html [12.04.2017].

SoapUI by Smartbear (2017): SOAP vs. REST challenges (online). https://www.soapui.org/testing-dojo/world-of-api-testing/soap-vs--rest-challenges.html [22.02.2017].

Spotify Developers (2016): Spotify Web API (online). https://developer.spotify.com/web-api/ [17.03.2017].

Stack Overflow (2017): Developer Survey Results 2017 (online). https://insights.stackoverflow.com/survey/2017 [23.06.2017].

Steinhardt, Stephanie; Schubert, Andreas; Meng, Michael (2015): Software-Entwicklerdokumentation: Was wollen Entwickler wirklich? (online). http://tagungen.tekom.de/fileadmin/tx_doccon/slides/1119_Software_Entwicklerdokumentation_Was_wollen_Entwickler_wirklich_.pdf [01.03.2017].

streetlayer (2017): API Documentation (online). https://streetlayer.com/documentation [19.03.2017].

Süddeutsche Zeitung (2012): Google verbraucht Strom für eine ganze Stadt (online). http://www.sueddeutsche.de/digital/energiebilanz-des-internet-konzerns-google-verbraucht-strom-fuer-eine-ganze-stadt-1.1141053 [29.03.2017].

Swagger (o. J.): Swagger Editor (online). http://editor.swagger.io/#!/ [09.03.2017].

Swagger (2016): Swagger. The world´s most popular API framework (online). http://swagger.io/ [09.03.2017].

t3n (2013): URL oder URI? Wir zeigen euch den Unterschied (online). http://t3n.de/news/url-uri-unterschiede-516483/ [12.01.2017].

Techslides (2016): Top 10 Free Templates for API Documenation (online). http://techslides.com/top-10-free-templates-for-api-documenation [30.03.2017].

TEIA (2017): Java und Webapplikationen: Ein- und mehrzeilige Kommentare (online). https://www.teialehrbuch.de/Kostenlose-Kurse/JAVA/6660-Ein-und-mehrzeilige-Kommentare.html [27.03.2017].

Thattil, Sascha (2014): Was sind Webservices? (online). http://www.yuhiro.de/was-sind-webservices/# [22.02.2017].

Thomson Reuters Open Calais (2017): Getting Started with Thomson Reuters Open Calais™ API (online). http://www.opencalais.com/opencalais-api/ [17.03.2017].

Tilkov, Stefan (2009): REST – Der bessere Web Service? (online). https://jaxenter.de/rest-der-bessere-web-service-8988 [02.11.2016].

Tinfoil Security (2017[a]): About Tinfoil (online). https://www.tinfoilsecurity.com/about [17.03.2017].

Tinfoil Security (2017[b]): Developer Documentation (online). https://www.tinfoilsecurity.com/developer/api [17.03.2017].

TryMyUI (2015): TryMyUI 2 (online). https://www.trymyui.com/apipie [19.03.2017].

Twilio Docs (2016): REST API: Sending SMS or MMS (online). https://www.twilio.com/docs/api/rest/making-calls [10.03.2017].

Udemy (2016): Udemy API Dokumentation (online). https://www.udemy.com/developers/ [17.03.2017].

Upon Online Marketing (2014): Web 2.0 ist Vergangenheit, jetzt kommt das Web 3.0 (online). https://www.upon-onlinemarketing.de/web-2-0-ist-vergangenheit-jetzt-kommt-das-web-3-0/ [23.06.2017].

UserLandSoftware.inc (1999): XML-RPC.Com (online). http://xmlrpc.scripting.com/ [20.02.2017].

VisualSearchApi (2016): Visual Search API Documentation (online). http://www.soutu360.com/visualsearchapi /apiDoc/apiDoc?apiDoc=V# [17.03.2017].

W3C (2000): XML Schema (online). https://www.w3.org/XML/Schema [22.02.2017].

W3C (2016): HTML & CSS (online). https://www.w3.org/standards/webdesign/htmlcss [22.02.2017].

W3schools (2017[a]): XML Tutorial (online). https://www.w3schools.com/xml/ [30.03.2017].

W3schools (2017[b]): JSON - Introduction (online). https://www.w3schools.com/js/js_json_intro.asp [30.03.2017].

w3schools (2017c): CSS: hover Selector (online). https://www.w3schools.com/cssref/sel_hover.asp [10.03.2017].

Watchful (2017[a]): Manage and secure all your Joomla! & WordPress websites with ease in a single dashboard for only $1/site/month or less (online). https://watchful.li/ [17.03.2017].

Watchful (2017[b]): How do I use the Watchful REST API? (online). https://watchful.li/faqs/148-tools/621-watchful-rest-api [17.03.2017].

Wolff, Eberhard (2017): Was sind eigentlich Microservices? (online). https://jaxenter.de/was-sind-microservices-40571 [23.06.2017].

Zazzle (2014): Zazzle API & Stores (online). https://www.zazzle.de/api [17.03.2017].

Ziemer, Tim (2013): Software Development Kit – Was ist das? (online). http://praxistipps.chip.de/software-development-kit-was-ist-das_12162 [13.12.2016].

7.4. Technische Normen

7.4.1. ISO/IEC

ISO/IEC IEEE 15289 (2015-05): Systems and software engineering - Content of life-cycle information items (documentation).

ISO/IEC 25064 (2013-09): Systems and software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Common Industry Format (CIF) for usability: User needs report.

ISO/IEC IEEE 26514 (2010): Standard for adoption of ISO/IEC 26514:2008 systems and software engineering-requirements for designers and developers of user documentation.

ISO/IEC IEEE 26515 (2011-12): Systems and software engineering - Developing user documentation in an agile environment.

7.4.2. DIN/EN

DIN EN 82079-1 (2013-06): Preparation of instructions for use - Structuring, content and presentation - Part 1: General principles and detailed requirements (IEC 82079-1:2012); German version EN 82079-1:2012.

DIN EN 82079-1 (2012): Erstellen von Gebrauchsanleitungen - Gliederung, Inhalt und Darstellung - Teil 1: Allgemeine Grundsätze und ausführliche Anforderungen.

8. Abbildungsverzeichnis

9. Anhang

9.1. API-Beispielanfrage

9.1.1. Anfrage

https://maps.googleapis.com/maps/api/directions/json?origin=Chicago,IL&destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City,OK&key=*YOUR_API_KEY*
(Google Maps API o. J.)

9.1.2. Antwort in JSON


```
{
  "status": "OK",
  "geocoded_waypoints" : [
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJ7cv00DwsDogRAMDACa2m4K8",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJ69Pk6jdlyIcRDqM1KDY3Fpg",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJgdL4flSKrYcRnTpP0XQSojM",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJE9on3F3HwoAR9AhGJW_fL-I",
        "types" : [ "locality", "political" ]
     }
  ],
  "routes": [ {
    "summary": "I-40 W",
    "legs": [ {
      "steps": [ {
        "travel_mode": "DRIVING",
        "start_location": {
          "lat": 41.8507300,
          "lng": -87.6512600
        },
        "end_location": {
          "lat": 41.8525800,
          "lng": -87.6514100
        },
        "polyline": {
          "points": "a~l~Fjk~uOwHJy@P"
        },
        "duration": {
          "value": 19,
          "text": "1 min"
        },
        "html_instructions": "Head \u003cb\u003enorth\u003c/b\u003e on \u003cb\u003eS Morgan
St\u003c/b\u003e toward \u003cb\u003eW Cermak Rd\u003c/b\u003e",
        "distance": {
          "value": 207,
          "text": "0.1 mi"
        }
      },
      ...
      ... additional steps of this leg
    ...
    ... additional legs of this route
      "duration": {
        "value": 74384,
        "text": "20 hours 40 mins"
      },
      "distance": {
        "value": 2137146,
        "text": "1,328 mi"
```

```
      },
      "start_location": {
        "lat": 35.4675602,
        "lng": -97.5164276
      },
      "end_location": {
        "lat": 34.0522342,
        "lng": -118.2436849
      },
      "start_address": "Oklahoma City, OK, USA",
      "end_address": "Los Angeles, CA, USA"
    } ],
    "copyrights": "Map data ©2010 Google, Sanborn",
    "overview_polyline": {
      "points":
"a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\\|tNtqf@l{Yd_Fblh@rxo@b}@xxSfytAblk@xxaBeJxlcBb~t@zbh
@jc|Bx}C`rv@rw|@rlhA~dVzeo@vrSnc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo@`cFp~xBc`Hk@nurDznmFfwMb
wz@bbl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\\~mbDzeVh_Wr|Efc\\x`Ij{kE}mAb~uF{cNd}xBjp]fulBiwJpgg
@|kHntyArpb@bijCk_Kv~eGyqTj_|@`uV`k|DcsNdwxAott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiAp|gEicy@`omFvaE
rfo@igQxnlApqGze~AsyRzrjAb__@ftyB}pIlo_BflmA~yQftNboWzoAlzp@mz`@|}_@fda@jakEitAn{fB_a]lexClshB
tmqAdmY_hLxiZd~XtaBndgC"
    },
    "warnings": [ ],
    "waypoint_order": [ 0, 1 ],
    "bounds": {
      "southwest": {
        "lat": 34.0523600,
        "lng": -118.2435600
      },
      "northeast": {
        "lat": 41.8781100,
        "lng": -87.6297900
      }
    }
  } ]
}
```


(Google Maps API o. J.)

9.1.3. Antwort in XML


```
<DirectionsResponse>
 <status>OK</status>
 <geocoded_waypoint>
  <geocoder_status>OK</geocoder_status>
  <type>locality</type>
  <type>political</type>
  <place_id>ChIJ7cv00DwsDogRAMDACa2m4K8</place_id>
 </geocoded_waypoint>
 <geocoded_waypoint>
  <geocoder_status>OK</geocoder_status>
  <type>locality</type>
  <type>political</type>
  <place_id>ChIJ69Pk6jdlyIcRDqM1KDY3Fpg</place_id>
 </geocoded_waypoint>
 <geocoded_waypoint>
  <geocoder_status>OK</geocoder_status>
  <type>locality</type>
  <type>political</type>
  <place_id>ChIJgdL4flSKrYcRnTpP0XQSojM</place_id>
 </geocoded_waypoint>
 <geocoded_waypoint>
  <geocoder_status>OK</geocoder_status>
  <type>locality</type>
  <type>political</type>
  <place_id>ChIJE9on3F3HwoAR9AhGJW_fL-I</place_id>
 </geocoded_waypoint>
 <route>
  <summary>I-40 W</summary>
  <leg>
   <step>
    <travel_mode>DRIVING</travel_mode>
    <start_location>
     <lat>41.8507300</lat>
     <lng>-87.6512600</lng>
    </start_location>
```

```
    <end_location>
     <lat>41.8525800</lat>
     <lng>-87.6514100</lng>
    </end_location>
    <polyline>
     <points>a~l~Fjk~uOwHJy@P</points>
    </polyline>
    <duration>
     <value>19</value>
     <text>1 min</text>
    </duration>
    <html_instructions>Head <b>north</b> on <b>S Morgan St</b> toward <b>W Cermak
Rd</b></html_instructions>
    <distance>
     <value>207</value>
     <text>0.1 mi</text>
    </distance>
   </step>
   ...
   ... additional steps of this leg
  ...
  ... additional legs of this route
   <duration>
    <value>74384</value>
    <text>20 hours 40 mins</text>
   </duration>
   <distance>
    <value>2137146</value>
    <text>1,328 mi</text>
   </distance>
   <start_location>
    <lat>35.4675602</lat>
    <lng>-97.5164276</lng>
   </start_location>
   <end_location>
    <lat>34.0522342</lat>
    <lng>-118.2436849</lng>
   </end_location>
   <start_address>Oklahoma City, OK, USA</start_address>
   <end_address>Los Angeles, CA, USA</end_address>
  <copyrights>Map data ©2010 Google, Sanborn</copyrights>
  <overview_polyline>
   <points>a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\|tNtqf@l{Yd_Fblh@rxo@b}@xxSfytAblk@xxaBeJxl
cBb~t@zbh@jc|Bx}C`rv@rw|@rlhA~dVzeo@vrSnc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo@`cFp~xBc`Hk@nur
DznmFfwMbwz@bbl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\~mbDzeVh_Wr|Efc\x`Ij{kE}mAb~uF{cNd}xBjp]ful
BiwJpgg@|kHntyArpb@bijCk_Kv~eGyqTj_|@`uV`k|DcsNdwxAott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiAp|gEicy@
`omFvaErfo@igQxnlApqGze~AsyRzrjAb__@ftyB}pIlo_BflmA~yQftNboWzoAlzp@mz`@|}_@fda@jakEitAn{fB_a]l
exClshBtmqAdmY_hLxiZd~XtaBndgC</points>
  </overview_polyline>
  <waypoint_index>0</waypoint_index>
  <waypoint_index>1</waypoint_index>
  <bounds>
   <southwest>
    <lat>34.0523600</lat>
    <lng>-118.2435600</lng>
   </southwest>
   <northeast>
    <lat>41.8781100</lat>
    <lng>-87.6297900</lng>
   </northeast>
  </bounds>
 </route>
</DirectionsResponse>
```

(Google Maps API o. J.)

9.2. Übersicht der Forschungsobjekte

Nr.	Name	URL der API-Dokumentation	Abkürzung
1	OpenFEC API	https://api.open.fec.gov/developers/#!/dates/get_calendar_dates	OFEC API
2	Udemy API	https://www.udemy.com/developers/	
3	Guide Star Search API	https://community.guidestar.org/community/support/developer	GSS API
4	ANX Exchange API v3	http://docs.anxv3.apiary.io/#	ANX E API
5	Google Maps Directions API	https://developers.google.com/maps/documentation/directions/?hl=de	GMD API
6	Flickr API	https://www.flickr.com/services/api/	
7	Box Content API	https://docs.box.com/reference#api-docs-directory	BC API
8	Klipfolio API	http://apidocs.klipfolio.com/docs	
9	Etsy API	https://www.etsy.com/developers/documentation/	
10	Zazzle API	https://www.zazzle.de/api	
11	Thomson Reuter Open Calais API	http://www.opencalais.com/opencalais-api/	TROC API
12	Reddit API	https://www.reddit.com/dev/api	
13	Open Secrets API	https://www.opensecrets.org/resources/create/apis.php	OS API
14	Spotify Web API	https://developer.spotify.com/web-api/	SW API
15	Photobucket API	http://pic.pbsrc.com/dev_help/WebHelpPublic/PhotobucketPublicHelp.htm	PH API
16	NYT Book API	http://developer.nytimes.com/books_api.json#/README	NYT B API
17	Visual Search API	http://www.soutu360.com/visualsearchapi/apiDoc/apiDoc?apiDoc=V#	VS API
18	Tinfoil Security API	https://www.tinfoilsecurity.com/developer/api	TS API
19	Arlo REST Public API	https://developer.arlo.co/doc/api/2012-02-01/pub/	ARP API
20	Watchful REST API	https://watchful.li/faqs/148-tools/621-watchful-rest-api	WR API
21	The Budget Your Trip API	http://www.budgetyourtrip.com/api/v3/	TBYT API
22	Kaltura API	https://developer.kaltura.com/api-docs/#/Overview	
23	Oanda API	http://developer.oanda.com/rest-live/introduction/	
24	Microsoft Outlook Mail REST API	https://msdn.microsoft.com/en-us/office/office365/api/mail-rest-operations	MOMR API
25	Docker Cloud REST API	https://docs.docker.com/apidocs/docker-cloud/#introduction	DCR API
26	Here Weather API	https://developer.here.com/rest-apis/documentation/weather/topics/overview.html	HW API
27	SNAP PAC REST API	http://developer.opto22.com/rest/pac/	SPR API
28	Commerce.js API	https://commercejs.com/docs/	CJS API
29	TryMyUI 2 API	https://www.trymyui.com/apipie	TMU API
30	Streetlayer API	https://streetlayer.com/documentation	SL API

9.3. Gesamtbetrachtung

Neuer Platz	Forschungsobjekt	Positive Aspekte	Negative Aspekte	Alter Platz
1	DCR	Übersichtlich, zweispaltiges Layout sehr klar, dynamische Navigation, prozessbasierte Überschriften, Gesamtzahl der Überschriften überschaubar, Farbigkeit angenehm	Umschaltung der Programmiersprachen etwas unauffällig	**11**
2	ANXE	Zweispaltiger Aufbau, klare Trennung, gute Farbigkeit, sehr übersichtlich, Code-Beispiele dynamisch	Man muss extra immer auf den Button klicken, damit ein Code-Beispiel generiert wird, ist Vor- und Nachteil zugleich (Code unabhängig vom Kapitel, kann man aber auch übersehen und irritiert sein), sehr viele Menüpunkte	**7**
3	GMD	Schönes Design, sehr übersichtlich, bekannte Bedienung, keine überraschenden Elemente o.ä., Vielfalt der Elemente also Text/Bild/Video	Etwas unübersichtlich, da Elemente sich teilweise sehr ändern und das nicht vorhersehbar ist, komplett anderer Aufbau und anderes Layout im Entwickler-Leitfaden, Menü links und rechts, rechts ist sehr unauffällig	**2**
4	MOMR	Sehr übersichtlich, unaufgeregt, klare Schrift, gute Einteilung der Inhalte, Navigation rechts gut sichtbar, da die Hauptnavigation links beim Scrollen verschwindet, geordnete/prozessbasierte Überschriften	Startseite ausgelagert, da für alle APIs gültig,	**1**
5	PH	Sehr übersichtlich, gute/eindeutige Überschriften, Unterhierarchien sichtbar, Verlinkungen in Kapiteln, Startseite gut	Altbacken weil typische Online-Hilfe	**7**
6	Oanda	Navigation gut, Überschriften gut, Optik sehr gut, geteilt in konzeptuell und Referenzen,	insgesamt etwas viele Menüpunkte	**5**
7	SW	Gute Optik, klare Einteilung	Sehr viele Menüpunkte, API ist nur als Untermenü da, hat keine eigenen Steuerung, Menüpunkte können teilweise weiter geöffnet werden, ist nicht ersichtlich welche (keine Zeichen dran o.ä.)	**4**
8	TBYT	guter Startbereich, gute Optik, ansprechend	Text wirkt irgendwie unübersichtlich und zerstückelt, sehr viele Absätze, bei Referenzdokumentation ähnlich, sehr viele Boxen mit wenig Info, dadurch wird das unübersichtlich, Optik ändert sich stark, keine mitwandernde Navigation, wenig Kapitel,	**16**

Abb. 97: Ergebnisse der Gesamtbetrachtung mit Kommentaren

Neuer Platz	Forschungsobjekt	Positive Aspekte	Negative Aspekte	Alter Platz
9	NYTB	Documentation sehr übersichtlich, Bedienung gut, direkte Weiterleitung in die Console	Readme ist nicht schön, Optik nicht eindeutig durch viele gestückelte Absätze, Verlinkungen usw.,	**21**
10	CJS	Schöne Optik, dynamische Navigation, übersichtlicher Inhalt	Keine Abgrenzung zwischen den Kapiteln im zweispaltigen Layout, Code-Beispiele häufig viel länger als Textspalte, muss dann umständlich alles gescrollt werden	**7**
11	Troc	Lesezeichen vorhanden, gute Texte, gute Optik, übersichtlich	Sehr lang, sehr viel Text, wenig Code-Beispiele, lange Scrollwege	**11**
12	ARP	übersichtlicher Aufbau, gute Überschriften, kurz und prägnant	Menü rechts verschoben, das ist nicht wirklich sichtbar, sondern muss gesucht werden	**16**
13	Etsy	Gute Optik, klare Kapiteleinteilung, klare Abgrenzung der Inhalte	lange Scrollwege, keine mitwandernde Navigation, Navigation ist seltsam platziert und umschaltbar, eigentlich ganz gute Aufteilung, man muss trotzdem etwas suchen, teilweise Kapitel versteckt, Zwischenüberschriften nicht nochmal extra aufgeführt	**6**
14	SPR	Einteilung nachvollziehbar, kurze Texte, sehr übersichtlich, Referenzdokumentation als Swagger, dadurch auch sehr übersichtlich	keine schöne Optik	**16**
15	SL	eigentlich alles ok, fetzt aber auch nicht, übersichtlich, Optik ok	Navigation wandert nicht mit, sieht alles gleich aus, prozessbasierte Überschriften	**26**
16	OFEC	vorgelagerter Text gut zur Einleitung, gute Einteilung der Funktionen	Optik ok, etwas langweilig und eintönig in der Farbigkeit, keine Navigation, man muss scrollen um alle Funktionen zu sehen, die sind dann aber gut eingeteilt	**23**
17	Klipfolio	Kapitel an sich ok, Gestaltung übersichtlich	Navigation unübersichtlich, Funktionen sind nicht abgeteilt, sondern einfach untereinander geschrieben ohne Gruppierung	**21**
18	HW	Guter Einstieg, Optik ok, gut abgegrenzte Inhalte	keine schöne Optik, Navigation lädt sich andauernd neu, das stört, teilweise lang scrollbare Texte, umständliche Gliederung, teilweise sehr verschachtelt	**11**

Abb. 97: Fortsetzung Ergebnisse der Gesamtbetrachtung mit Kommentaren

Neuer Platz	Forschungsobjekt	Positive Aspekte	Negative Aspekte	Alter Platz
19	Kaltura	Optik ok	sehr verschachtelte Navigation, Unterpunkte öffnen sich nur durch Klick, da Webseite keine Dynamik möglich, Funktionskapitel + Parameterbeschreibungen sehr unübersichtlich und kleinteilig verlinkt/interaktiv, nicht nachvollziehbar, zu viele Pfade, kleine Schrift, keine Gesamtliste der Funktionen, man muss sich durch alle Kapitel klicken, man muss aus den nur teilweise deskriptiven Benennungen der F Funktionen auf den Inhalt schließen, sind Überschriften	**7**
20	BC	Übersichtlich, Inhalte sind da dann problematisch	ist wie eine Onepage gemacht, aber nur bedingt scrollbar; sehr viele Menüpunkte, zu große Schrift, Codebeispiele sehr groß im Text	**3**
21	Reddit	Gute Darstellung der Funktionen, sehr konsistent	Sehr viele Überschriften in der Navigation, Navigation ganz ausgeklappt so sind alle Überschriften sichtbar und deshalb so lang, Navigation wandert nicht mit, sehr langer Scrollweg, sehr unterschiedliche Inhalte in den Kapiteln, die nicht immer alle vorhanden sind, dadurch unübersichtlich (Text, Parameterbeschreibung, Code-Beispiel)	**11**
22	WR	gut finde ich die kompakt gestalteten Infos	Startseite unübersichtlich, interaktive Doku mit Swagger, die ist gut, aber Text ist nicht gut abgesetzt in einleitender, Sinn des Videos nicht ganz klar, kein Rückweg aus der interaktiven	**16**
23	Flickr	Überladen, viele Farben, keine nachvollziehbare Struktur, versteckte und nicht nachvollziehbare Verlinkungen	Referenzdokumentation ok, viele gute Verlinkungen	**11**
24	Udemy	optik eigentlich ganz gut,	Überschriften in der Navigation nicht eindeutig, Überschriften und Texte gehen irgendwie ineinander über, sehr kurz, keine Code-Beispiele	**26**

Abb. 97: Fortsetzung Ergebnisse der Gesamtbetrachtung mit Kommentaren

Neuer Platz	Forschungsobjekt	Positive Aspekte	Negative Aspekte	Alter Platz
25	OS	Klare Optik, gute Webseite, konsistentes Webdesign	Umständlich gemacht, Funktionen als verlinkte Liste, öffnet sich dann ein Fenster ohne Rückweg mit ganz anderer Optik, Doku ist in eigentlich Webseite integriert, funktioniert nur einigermaßen, weil die Doku so kurz ist	**23**
26	GSS	Optik der Seite ok, immer Zwischennavigationen	Redundante Inhalte unter den verschiedenen Zugängen, sehr kurze Texte oder nur Links unter den Überschriften, dadurch unübersichtlich und gestückelt, Inhalte an sich ok, halt sehr kurz, Zwischennavigationen wandern nicht mit und Texte gehen dann irgendwie ineinander über	**16**
27	VS	Tabelle mit Eigenschaften oben gut für Übersicht	API-Block mit Funktionen erschließt sich mir nicht ganz - anscheinend sind alle APIs in einem Block dargestellt, man muss sich die richtige raussuchen bzw. die richtigen Funktionen, Optik ist so lala, wirkt altbacken	**23**
28	TS	Farbigkeit gut, Mosaik Aufteilung gut	Schriften teilweise nicht auf passender Höhe, große Textblöcke für die kurze Doku, ungünstige Parameterbeschreibungen im Text, keine Navigation, also kein Überblick bei Beginn, was noch alles kommt	**23**
29	TMU	Optik auf den ersten Blick sehr klar	Völlig unklare Struktur, Pfade sind nicht eindeutig, teilweise kann man Seiten überspringen (Hauptseite Funktion) und kommt direkt zu der Funktion, Rückwege über Pfadanzeige oben, Boxen werden beim Hovern markiert, gibt aber keine Funktion, keine extra Navigation, nur Auflistung der Funktionen, ein Satz einleitender Text	**23**
30	Zazzle	Inhaltsverzeichnis	Alles durcheinander, keine klaren Abgrenzungen, Texte gehen ineinander über, kein klares Layout	**30**

Abb. 97: Fortsetzung Ergebnisse der Gesamtbetrachtung mit Kommentaren

Eigenständigkeitserklärung

Ich versichere eidesstattlich durch eigenhändige Unterschrift, dass ich die vorliegende Masterarbeit

Zur Nutzerfreundlichkeit von API-Dokumentation in der Praxis:

Überprüfung von Heuristiken zur Erstellung von API-Dokumentation

selbstständig und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe. Alle Stellen dieser Arbeit, die dem Wortlaut, dem Sinn oder der Argumentation nach anderen Werken entnommen sind (einschließlich des World Wide Web und anderer elektronischer Text- und Datensammlungen), habe ich unter Angabe der Quellen vollständig kenntlich gemacht.

Ort, Datum

Unterschrift