Ostfalia - Hochschule für angewandte Wissenschaften
Fachbereich Wirtschaft
Studiengang Wirtschaftsinformatik

Design von Hypermedia-APIs

Bachelorarbeit

Zur Erlangung des Grades eines Bachelor of Science
der Fakultät Wirtschaft
der Ostfalia - Hochschule für angewandte Wissenschaften

eingereicht bei Prof. Dr. XYZ
Zweitprüfer XYZ

von Marius Brederlow
Pappelweg 1
38536 Meinersen OT Ahnsen
Mat.-Nr. 70047338

Meinersen, den 31. Mai 2013

Zusammenfassung

ca. 12 Zeilen

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Zielsetzung

Wenn wir heutzutage über die Verwendung und Vorteile von Hypermedia sprechen ist dies keine Innovation der letzten Jahre. Bereits im Jahr 1990 beschrieb Tim Berners-Lee in seinem Vorschlag für ein Hypertext-Projekt die wesentlichen Eigenschaften einer Hypertext basierten Anwendung:

"The current incompatibilities of the platforms and tools make it impossible to access existing information through a common interface, leading to waste of time, frustration and obsolete answers to simple data lookup. There is a potential large benefit from the integration of a variety of systems in a way which allows a user to follow links pointing from one piece of information to another one. This forming of a web of information nodes rather than a hierarchical tree or an ordered list is the basic concept behind HyperText."[1]

Die Idee von Berners-Lee ist eine Anwendung nicht in Form von sequentiellen Abläufen und einer hierarchischen Struktur abzubilden, sondern als eine Art Sammlung von lose gekoppelten Ressourcen mit der Möglichkeit sich in dieser Sammlung vor und zurück zu bewegen. Der Vorschlag bewegt sich damit weg von den hierarchischen Strukturen einer verketteten Liste oder der Darstellung als Baum. Diese Idee legte seinerzeit den Grundstein für das, was wir heute als das WorldWideWeb kennen. Das Internet ist wohl das größte und umfassendste, verteilte Informationssystem unserer Zeit. Der Erfolg des Internet beruht sicherlich auf der Tatsache, dass es möglich ist verschiedenste Arten Informationen rund um den Globus miteinander zu vernetzen und abrufbar zu machen.
Im Jahr 2000 hat sich Roy Fielding im Rahmen seiner Dissertation mit dem Thema beschäftigt, welche Architektur das Internet selbst besitzt und wie diese Architektur aussehen sollte. Das Internet ist für ihn per Definition ein verteiltes Hypermedia-System, wobei er Hypermedia als eine Weiterentwicklung von Hypertext betrachtet. Hypermedia umfasst im Gegensatz zu Hypertext nicht nur Verweise auf anderen Hypertext sondern auf Multimediale Inhalte wie Text, Audio und Video. Fielding selbst schlägt als Architektur einer verteilten Hypermedia Anwendung, das von ihm beschriebene "Representational State Transfer"(REST) vor. Ein zentraler Bestandteil dieser Architektur ist die Repräsentation von Ressourcen und die Steuerung der Anwendung durch den Einsatz von Hypermedia.
Dieser Architekturstil gewann in den letzten Jahren zunehmend an Bedeutung bei der Erstellung von verteilten Hypermedia Systemen. Mensch zu Maschine Kommunikation und reine Maschinenkommunikation kann gleichermaßen abgebildet werden. Eine Idee beim Einsatz von Hypermedia in verteilten Informationssystemen besteht darin das Verhalten beim Browsen durch das Internet

[1] Berners-Lee, T. (1990), `http://www.w3.org/Proposal.html`

auch außerhalb des Browsers anwendbar zu machen. Bei der Verwendung eines Browsers kann ein Mensch anhand von Beschreibungen oder dem Kontext der Information einem Verweis folgen. Das Folgen der Verweise ist nicht sequentiell sonder steht immer im Kontext der eingebetteten Informaltion. Mit anderen Worten, ein Mensch kann die Semantik der Information verstehen und dem gewünschten Link folgen. Dies ist möglich ohne den exakten Endpunkt des Verweises zu kennen. An dieser Stelle spiegelt sich auch ein wesentlicher Erfolgsfaktor des Internets bzw. der Nutzung von Hypertext wieder. Niemand kann sich eine große Anzahl von URIs (Uniform Ressource Identifier) merken. Selbst wenn jemand sich alle wichtigen URIs merken könnte ist nicht sichergestellt, dass sie über die Zeit stabil sind. Stabilität ist auch nicht notwendig, da das Folgen eines Links ohne den exakten Endpunkt zu kennen möglich ist.
Betrachtet man eine Client-Server-Architektur als einen oft eigesetzten Stil für verteilte Anwendungen, muss der Client seine verwendeten Endpunkte auf den Servern kennen. Bei vielen Client Applikationen sind diese Endpunkte als ein Bestandteil des Quellcodes festgelegt. Diese Tatsache hat zur Folge, dass im Fall einer Änderung oder Erweiterung des Adressraums die im Client festgelegten URIs invalidiert werden. Demnach besteht schon aufgrund der statischen Benutzung von URIs eine recht enge Kopplung zwischen Client und Server. Um einen lose gekoppelten Client zu entwerfen, sollte er keine festgelegten URIs aus dem Adress-Schema des Servers enthalten. Die einzige Ausnahme ist hier ein zentraler Eintrittspunkt, von dem alle weiterführenden Aktionen ausgehen. Die weiteren Operationsmöglichkeiten und die Navigation durch die Anwendung sollte vom Server dynamisch zur Laufzeit an den Client gesendet werden.
Dieses Vorgehen ist bei einer Mensch-zu-Maschine Kommunikation intuitiv umsetzbar. Es entspricht exakt der Anwendung eines Browsers um durch verschiedene Webseiten als Repräsentation von Informationen im Internet zu navigieren. Intuitive Benutzbarkeit kommt zu stande, weil ein Mensch in der Lage ist den Kontext der Information zu interpretieren und entsprechend zu handeln.
Bei einer Maschine-zu-Maschine Kommunikation ist das nicht der Fall. Eine Maschine kann in der Lage sein einem Verweis zu folgen, sie kennt jedoch nicht die Semantik des Verweises. Hierfür ist zusätzliches Wissen auf der Seite des Clients erforderlich. Bei vielen Anwendungen ist dieses Wissen in einer umfassenden Dokumentation festgelegt. Die Dokumentation beschreibt, welche Verweise und Endpunkte für eine bestimmte Anwendungslogik zur Verfügung stehen. Client Applikationen müssen dieses Wissen implementieren um die Nachrichten der Anwendung zu verstehen und die Anwendung zu steuern. Lose gekoppelte Systeme sind nach diesem Modell nur schwer zu entwerfen. Änderungen an den verwendeten Adressen müssen im Client stets reimplementiert werden.
Hypermedia setzt an dieser Stelle auf ein anderes Prinzip. Um die Implementation von Wissen auf der Clientseite möglichst gering zu halten werden selbstbeschreibende Hypermedianachrichten eingesetzt. Selbstbeschreibende Nachrichten sind angereichert mit semantischen Informationen für die Verarbeitung der Nachricht. Also mit dem Wissen das der Client benötigt um die erhaltenen Informationen darzustellen und die Anwendung zu steuern. Sämtliches, in einer externen Dokumentation festgelegtes Wissen, das nicht als ein Bestandteil der selbstbeschreibenden Nachricht

mitgeliefert wird, ist im Hypermedia Umfeld als Öut of Band Knowledge"bezeichnet. Der Anteil dieser Wissensform sollte im Allgemeinen möglichst gering gehalten werden, um eine zu starke Kopplung zu vermeiden.
Reduziert man eine verteilte Hypermedia Anwendung auf das wesentliche Vorgehen, ergeben sich zwei charakteristische Eigenschaften. Auf der einen Seite ein Client, welcher in der Lage ist Informationen darzustellen und Verweisen zu folgen. Auf der anderen Seite ein Server, der Informationen und Verweise auf seine Ressourcen zur Verfügung stellt. Als ein zentraler Bestandteil der Kommunikation können somit Verweise identifiziert werden. Bestandteile eines Verweises selbst lassen sich anhand der genannten Eigenschaften einer verteilten Hypermedia Anwendung abstrakt ein drei Attribute unterteilen:

- Der URI als aktueller vom Server zur Verfügung gestellter Endpunkt für eine Ressource
- Das Relationsattribut zur Beschreibung der Beziehung zwischen aktuell genutzter und Zielressource
- Das Methoden Attribut um festzulegen mit welcher Protokollmethode die Ressource angesprochen werden kann

Verweise, angereichert mit der beschriebenen Semantik, lösen die Kopplung zwischen Client und Server bis zu einem gewissen Grad. Clients einer verteilten Hypermedia Anwendung benötigen im Idealfall nur Kenntnis über die verwendeten Relationsattribute, wobei unterstellt wird, dass der Client den zentralen Eintrittspunkt der Anwendung kennt und der Server seine Adressen komplett selbständig verwalten kann. Die Steuerung des Applikationsfluss erfolgt ausschließlich über die vom Server zur Verfügung gestellten Verweise. Dieses Vorgehen entspricht dem eines Browsers und einer Hypermedia getriebenen Anwendung gleichermaßen.
Hypermedia birgt viel Potential in Hinblick auf die Erstellung eines verteilten Informationssystems. Durch semantisch angereicherte Nachrichten wird versucht die Kopplung zwischen Client und Server auf ein Minimum zu reduzieren. Ein her geht damit auch eine bessere Wartbarkeit und Erweiterbarkeit der Applikation. Essentieller Grundbestandteil der Kommunikation ist ein Hypermediaformat, das möglichst viele der geforderten Eigenschaften unterstützt und einen lose gekoppelten Nachrichtenaustausch ermöglicht. Aktuell existieren viele verschiedene Formate, die den Erfordernissen in unterschiedlichen Umfängen genügen. Diese Arbeit hat zum Ziel eine Beurteilungsgrundlage für Hypermediaformate zu entwickeln, verschiedene Formate miteinander zu vergleichen und auf ihre Leistungsfähigkeit hin zu untersuchen.

2 Hypermedia

2.1 Definition

Die geläufigste Form von Hypermedia ist HTML in Verbindung mit dem Internet und Millionen von Websites. Hypermedia auf Websites zeichnet sich auf den ersten Blick durch zwei wesentliche Merkmale aus. Websites bestehen zum einen aus multimedialen Inhalten und sind zum anderen untereinander mit Verweisen verknüpft. Außerdem gibt es die Möglichkeit Daten an den Server über Formulare zu senden. Diese Merkmale lassen sich dem bekanntesten Hypermediaformat, (X)HTML ohne weiteres zuschreiben.
Roy Fielding beschreibt im Rahmen seiner Dissertation Representational State Transfer (REST)als einen Architekturstil, auf Grundlage des modernen Internets. Fielding definiert das Internet hierbei als ein hochgradig verteiltes Hypermedia-System, dessen Verteilung über die geographische Ausdehnung hinaus geht und eine grenzenlose Verbindung von Informationen ermöglicht. Dienstanbieter müssen mit einer unabhängigen Entwicklung von Software Komponenten sowie der unregulierten Skalierung des Internet zurecht kommen. Als Schlüsseltechnologie kommt an dieser Stelle Hypermedia zum Einsatz. Hypermedia ist im Rahmen des Internets ein einheitliches Mittel um auf Dienste zuzugreifen. Der Zugriff erfolgt über eingebettete Steuerelemente als ein Teil der erhaltenen Informationen. [2]
Hypermedia bildet zudem eine universell einsetzbare Schnittstelle im Internet. Die gleiche Schnittstelle kann unabhängig von verschiedenen Informationsquellen genutzt werden. Verweise erlauben eine unbegrenzte Strukturierung der Informationen und die Manipulation der Verweise ermöglicht einem Dienstanbieter die Darstellung komplexer Beziehungen zwischen Informationen. Das Führen eines Dienstkonsumenten durch eine Anwendung ist auf diese Weise realisierbar. Außerdem können einfache Anfragen durchgeführt werden, die das Durchsehen von großen Datenbeständen vereinfachen. Einfachheit und Allgemeingültigkeit lassen Hypermedia zu einer leicht zugänglichen Technologie werden. [3]
REST abstrahiert von den implementativen Details, und fokussiert den Einsatz von Hypermedia mit der Interpretation von signifikanten Datenelementen durch selbst beschreibende Nachrichten. Diese Hypermedianachrichten bestehen aus sechs Datenelementen:

- Ressource
- Ressourcenidentifikator
- Repräsentation der Information
- Metadaten der Repräsentation
- Metadaten der Ressource

[2] (Vgl. Fielding, 2000, S. 3)
[3] (Vgl. Fielding, 2000, S. 67f)

- Steuerdaten

Dinge der realen Welt werden durch die Ressource repräsentiert. Die Ressource selbst stellt eine Abstraktion der Information dar und kann einen Dienst, ein Dokument oder eine Sammlung anderen Ressourcen beinhalten.Der Ressourcenidentifikator dient der Adressierung und Referenzierung einer Ressource. Eine stabile und semantisch korrekte Identifikation liegt hierbei in der Verantwortung der Ressourcen verwaltenden Komponente. Die Repräsentation einer Ressource besteht aus Nutzdaten und Metadaten der Repräsentation zu Beschreibungszwecken. Zusätzlich kann eine Antwort auch Metadaten der Ressource enthalten. Diese beinhalten Informationen der Ressource, die nicht spezifisch zu der Repräsentation zugeordnet werden können. Die Steuerdaten dienen der Parametrisierung einer Anfrage. Sie beschreiben die Bedeutung und Verwendung einer bestimmten Aktion innerhalb der Nachricht. [4]

"Hypermedia is defined by the presence of application control information embedded within, or as layer above, the presentation of information."[5]

Während des Entwurfs einer verteilten Hypermedia-Anwendung müssen diese Anforderungen durch den Einsatz eines bestimmten Datenformats abgebildet werden. Die Betrachtung von verschiedenen Hypermediaformaten soll innerhalb dieser Arbeit losgelöst von Programmiersprachen, Technologien und serverseitigen Implementationen erfolgen. Im Mittelpunkt steht eine Betrachtung der Formate entsprechend ihrer Eigenschaften und Fähigkeiten eine Hypermedia-Anwendung zu unterstützen.

2.2 Richardsons Maturity Model

Leonard Richardson beschreibt mit seinem Maturity Model eine Art Stufenplan, wie ein verteiltes Hypermedia System auf Basis von Internet Technologien umgesetzt werden kann. Die Konsequente Umsetzung von REST als Architekturstil und das Nutzen bestehender Technologien steht hierbei im Fokus. Das Zusammenwirken von HTTP, URIs und Hypermedia in Form von HTML ist charakteristisch für das Internet. Diese drei Technologien und dessen Kombination bilden die Grundlage für das erstellen einer Hypermedia Anwendung.[6]

[4] (Vgl. Fielding, 2000, S. 86ff)
[5] Fielding, 2000, S. 68
[6] Richardson, L. (2008), `http://www.crummy.com/writing/speaking/2008-QCon/act3.html`

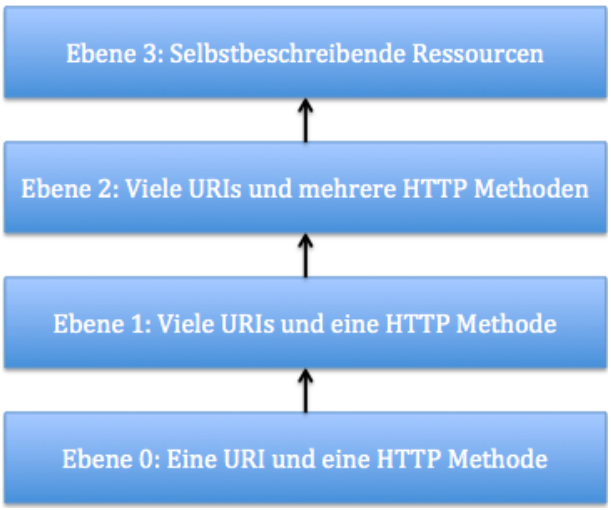

Abbildung 1: Leonard Richardsons Maturity Model

Um die Unterschiede zwischen den einzelnen Ebenen darzustellen wird ein sehr einfaches Warenwirtschaftssystem eingeführt. Dieses System bietet die Möglichkeit eine Liste des gesamten Warenbestands anzufragen, die Details einer bestimmten Ware anzuzeigen und Bestellungen von Waren auszulösen.

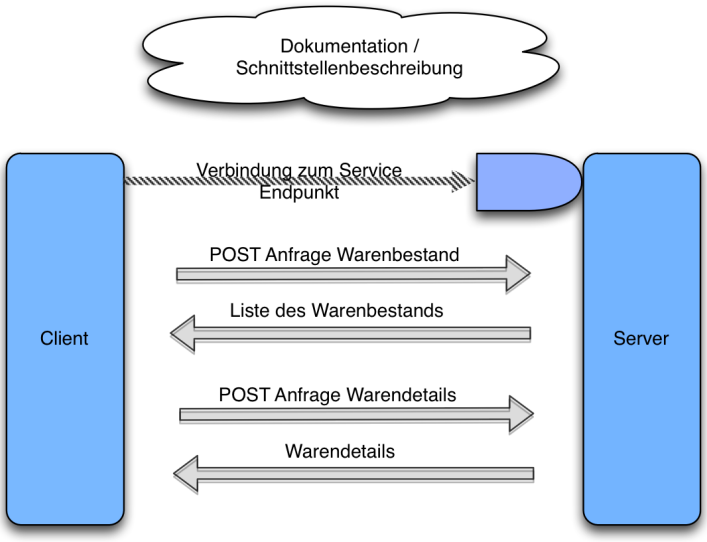

Abbildung 2: Maturity Model Ebene 0

Auf Ebene 0 sind alle Funktionen außerhalb der Nachrichten definiert und werden in der Regel durch ein HTTP-POST getunnelt. HTTP dient auf dieser Ebene als Intermediär und löst selbst implementiere, entfernte Prozeduren oder Methoden aus. Diese Vorgehensweise ähnelt stark dem Konzept der Remote Method Invocation im Fall von JAVA. Die Kommunikation basiert auf dem

serialisieren und deserialisieren von Objekten. Sämtliches Wissen über die Objekte und deren Format ist in einer externen Dokumentation festgelegt. Beispielhaft hierfür ist der Einsatz von "Plain old XML"(POX). Das Verständnis über die ausgetauschten XML-Instanzen liegt innerhalb einer Dokumentation und XML-Schemata werden zur Validation eingesetzt. Dies ist auch der Fall beim SSimple Object Access Protocol"(SOAP), der als ein industrieller Standard für den Austausch von Daten eingesetzt wird. Im Unterschied zu POX wird die XML-Instanz hier jedoch in eine Art Umschlag verpackt. Die Dokumentation erfolgt in einer Beschreibungssprache und liegt somit ebenfalls in externer Form vor.

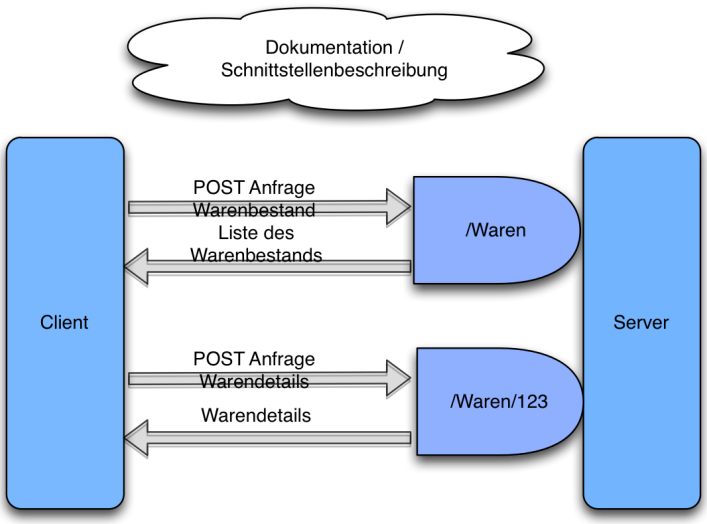

Abbildung 3: Maturity Model Ebene 1

Die Verwendung von Ressourcen kommt auf Ebene 1 zum Tragen. Anstatt alle Anfragen an einen zentraler Endpunkt zu richten, werden auf dieser Ebene Ressourcen direkt angesprochen. Das Konzept der Ressourcen ermöglicht somit eine Art Identifikation von Dingen der Realen Welt. Sollten Details zu einer bestimmten Ware benötigt werden, kann diese nun direkt angesprochen werden. GET ist im Rahmen von HTTP definiert als eine sichere idempotente Operation, die keine signifikanten Änderungen am Status der Anwendung durchführt. Außerdem ist es möglich eine GET Anfrage, im Gegensatz zu einer POST Anfrage, zu cachen. Caching ist einer der wesentlichen Erfolgsfaktoren für eine performante Anwendung. Die Abfrage des Warenbestandes muss aufgrund ihres lesenden Charakters und der Beibehaltung des Anwendungszustands ab dieser Ebene mittels einer GET Anfrage durchgeführt werden.

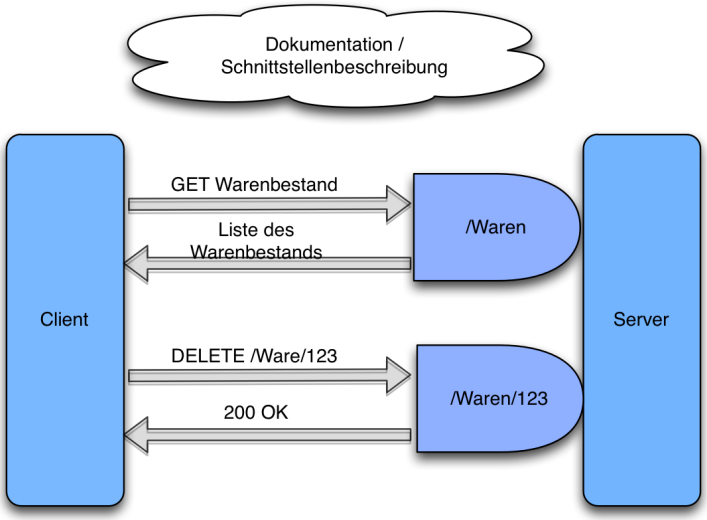

Abbildung 4: Maturity Model Ebene 2

Eine Unterscheidung zwischen lesenden, schreibenden, idempotenten und nicht idempotenten Operationen konnte bis jetzt durch den Einsatz von HTTP als Tunnelmechanismus nicht vorgenommen werden. Eine entfernte Methode wurde stets durch ein POST ausgelöst. Die HTTP konforme Verwendung der Verben wird auf Ebene 2 eingeführt. Anstatt die Liste des Warenbestands über eine POST Anfrage auszuführen wird eine GET Anfrage abgesetzt. PUT, POST und DELETE sind im innerhalb von HTTP definiert als Operationen mit schreibendem Charakter und demzufolge einer Änderung des Anwendungszustands. Wenn eine Ware geändert oder gelöscht werden soll muss eine entsprechende Operation verwendet werden. Neben der korrekten Verwendung der Verben wird auf dieser Ebene ein Mechanismus zur Fehlerindikation eingesetzt. Die Verwendung der HTTP Status Codes entsprechend ihrer Spezifikation kann diese Rolle übernehmen. So muss zum Beispiel die korrekte Abarbeitung einer Anfrage vom Dienstanbieter mit einem "200 OK"quittiert werden. Die Umsetzung eines geeigneten Fehlerindikators liegt in der Verantwortung des Dienstanbieters.

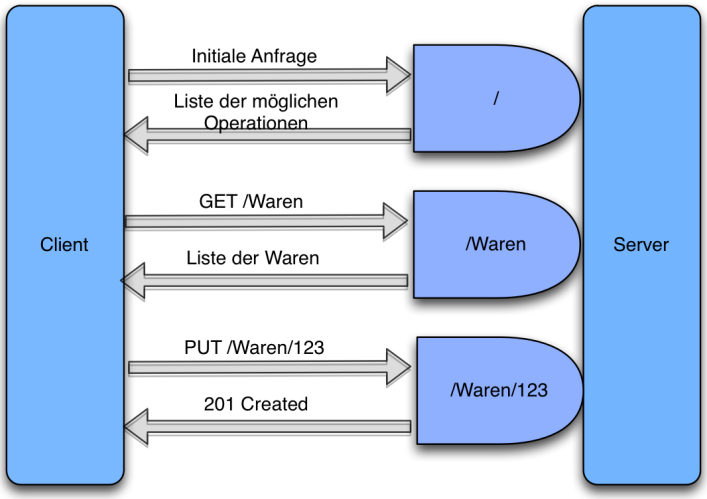

Abbildung 5: Maturity Model Ebene 3

Ebene 3 beginnt mit der Verwendung von Hypermedia Steuerelementen. Eine Nachricht auf dieser Ebene enthält nicht nur reine Nutzdaten sonder auch semantische Informationen zum Verständnis der Daten. Hypermedia Steuerelemente zeigen einem Client welche Operationen er von seiner aktuellen Position aus durchführen kann. Der Client benötigt keine Kenntnis über die exakten Adressen für bestimmte Operationen. Vielmehr wird er vom einem zentralen Einstiegspunkt per Verweis den Anwendungsprozess geführt. Diese selbstbeschreibenden Nachrichten ermöglichen einen variablen Adressraum auf der Seite des Dienstanbieters und reduzieren die Kopplung zwischen Client und Server. Zudem kann der Dienstanbieter neue Verweise und Informationen zu seinen Nachrichten hinzufügen. Solange alle Dienstkonsumenten unbekannte Verweise ignorieren bietet der Einsatz von Hypermedia Steuerelementen eine gefahrlose Erweiterungsmöglichkeit der Nachrichten. Wenn das Warenwirtschaftssystem auf dieser Ebene operiert, benötigt der Client als Information nur den zentralen Einstiegspunkt und Kenntnis über die genutzten Verweise. Im Idealfall ist keine externe Dokumentation mehr nötig. Der Client kennt die benutzten Verweise und weiß anhand der selbstbeschreibenden Nachrichten mit welchen Operationen er auf eine Ressource zugreifen kann.
Das Richardson Maturity Model bietet keinen direkten Ansatz zur Bewertung eines bestimmten Hypermedia Formats. Es beschreibt jedoch Eigenschaften die für das Erstellen einer REST konformen Anwendung und damit einer Hypermedia Anwendung nötig sind. Besonders die Erfordernisse der dritten Ebene beschreiben Eigenschaften, die im Einflussbereich verschiedener Hypermedia Formate liegen. Eine detaillierte Betrachtung des eingesetzten Formats ist nötig um eine gute Anwendung zu erstellen und die gegebenen Vorteile im vollen Umfang zu nutzen. Dieser Anspruch impliziert die hohe Bedeutung für das Erstellen einer Vergleichsgrundlage für verschiedene Formate.

2.3 Hypermedia Faktoren nach Amundsen

Der Hypermediagedanke entstand aus der Fragestellung, wie man ein gutes verteiltes Informationssystem entwerfen kann. Grundsätzlich gibt es verschiedene Ansätze um ein verteiltes System zu entwerfen und die Komponenten miteinander kommunizieren zu lassen. Die zu gründe liegende Fragestellung ist hier, wie kann der Server private Objekte exportieren um sie für den Client sichtbar und benutzbar zu machen. Hierfür gibt es unterschiedliche Plattform- und sprachspezifische Lösungen. Die grundsätzliche Funktionsweise ist hierbei immer ähnlich. Die Objekte werden serialisiert über das Netzwerk geschickt und so an den Client zur Verarbeitung übergeben. Der Client selbst muss also über ein Verständnis der Daten verfügen. Diese Herangehensweise hat den Nachteil, dass eine Änderung der serialisiebaren Objekte auch eine Änderung des Clients zur Folge hat. Somit entsteht immer ein gewisser Grad an Kopplung zwischen Client und Server.

Amundsen beschreibt mit Hypermedia einen anderen Lösungsansatz. Eine Synchronisation von Client und Server, bei der private Datentypen miteinander geteilt werden ist nicht der richtige Ansatz. Stattdessen sollte eine Technik zur Datenbeschreibung unabhängig von Platform, Sprache und privaten Datentypen verwendet werden. Diese Herangehensweise nennt Amundsen die Hypermedia Lösung. Die Probleme des Typemarshalling werden im Fall von Hypermedia durch das Anreichern der Nachrichten mit zwei Arten von Metadaten gelöst. Ein Teil sind Metadaten über die Daten selbst. Der andere Teil sind Metadaten über den Applikationsstatus und die die Möglichkeiten für den Client, den Applikationsstatus zu ändern. Losgelöst von privaten Datentypen ermöglicht dieses nachrichtenorientierte Design, eine spezifischere Anpassung an die Anforderungen der Problemdomäne. Durch Hypermedia Nachrichten werden somit keine privaten Datentypen geteilt sondern vielmehr ein generelles Verständnis der Daten. [7]
Nach Amundsen besteht eine Hypermedia Nachricht somit aus insgesamt drei Teilen:

- Daten
- Metadaten über die Daten
- Metadaten über den Applikationsstatus

Um Hypermedia selbst zu identifizieren, definiert Amundsen eine Sammlung abstrakter Eigenschaften und Funktionen, wie ein Hypermediaformat diese drei charakteristischen Datenteile umsetzt. Diese Sammlung wird Hypermediafaktoren (H-Faktoren) genannt und ist grundsätzlich unabhängig von der verwendeten Plattform und dem Transportprotokoll. In der Regel wird bei Hypermedia-Applikationen HTTP als Transportprotokoll eingesetzt, weil es eine Reihe von Vorteilen wie z.B. Nachhing, Chunking und Kompression bereits implementiert hat. HTTP ist aber keine zwingende Vorraussetzung für den Einsatz von Hypermedia. Diese H-Faktoren unterteilen sich dann noch in

[7] (Vgl. Amundsen, 2012, S. 10)

die zwei Bereiche Links und Control Data. Diese insgesamt neun H-Faktoren bilden das Gerüst für ein beliebiges Hypermediaformat. So kann man anhand dieser Faktoren bestimmet, ob ein bereits existierender Medientyp den Applikationsspezifischen Anforderungen genügt oder aber was bei der Erstellung eines eigenen Medientyps beachtet werden muss. [8]

Im Folgenden werden die einzelnen Faktoren genauer betrachtet. Die Link-Faktoren dienen im Wesentlichen der Bewegung des Clients im Applikationsfluss, wohingegen die Control Faktoren im dadurch definiert sind, dass sie zusätzliche Informationen beim Ausführen eines Verweises zur Verfügung stellen. Hier geht es vor allem um die Beschreibung der beinhalteten Nutzdaten. Die Berücksichtigung dieser Faktoren trägt dazu bei, dass alle Seiten etwas über die Beschaffenheit der Daten wissen oder aber die Daten in einer gewünschten Kodierung anzufordern. Die Illustration der H-Faktoren erfolgt am Beispiel von HTTP mit HTML als eingesetzten Hypermedia Typ. Die nachstehende Abbildung gibt einen Überblick der betrachteten Hypermedia Faktoren.

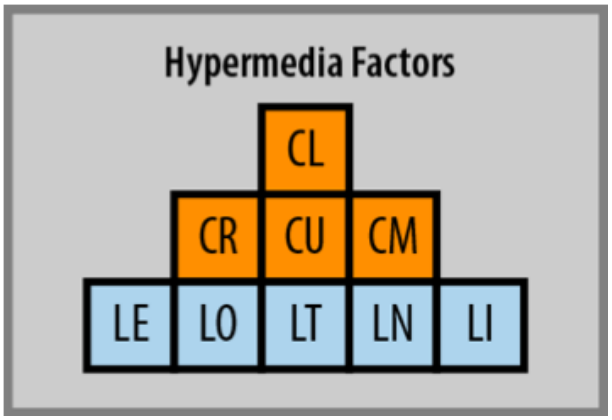

Abbildung 6: Hypermedia Faktoren nach Amundsen

2.3.1 Embedding Links

Embedding Links(H-Faktor LE) dient der Darstellung einer Ressource im aktuell genutzten Ausgabefenster. Ein Embedding Link nutzt die Lesen-Operation des verwendeten Protokolls und stellt die Antwort innerhalb des aktuellen Fensters dar. Ein gebräuchliches Beispiel für Embedding Links sind auf einer Website eingebundene Elemente. [9]

Listing 1: HTML Beispiel LE

```
<iframe src="..."> Einbetten eines Rahmens
```

[8] (Vgl. Amundsen, 2012, S. 13f)
[9] (Vgl. Amundsen, 2012, S. 15)

```
<img src="..."> Einbetten eines Bildes
```

Die im Listig 1 dargestellten Tags sorgen bei der Darstellung im Browser für eine Einbettung von lokalen oder entfernten Ressourcen und sind im Fall von HTML repräsentativ für den H-Faktor der Embedding Links.

2.3.2 Outbound Links

Outbound Links (H-Faktor LO) sind das, was man im Allgemeinen unter Navigationslinks versteht. Beim Folgen eines Outbound Links wird die Lesen-Operation des verwendeten Protokolls ausgeführt und die bestehende Ansicht durch die Antwort ersetzt. [10] Die Antwort von Google auf eine Suchanfrage beinhaltet Outbound Links. Folgt man einem Verweis aus der Antwort wird die darauf folgende Antwort die aktuelle Ansicht ersetzen.

Listing 2: HTML Beispiel LO

```
<a href="http://externe-domain.de">
```

In HTML werden Verweise durch das Anchor-Element (a) kenntlich gemacht. Die Kombination des Anchor- und Hyper-Reference- Elements (a href) sorgt beim Aktivieren für eine Darstellung der refferenzierten Ressource in einer neuen Ansicht.

2.3.3 Templated Links

Templated Links (H-Faktor LT) haben wie Embedding und Outbound Links nur einen lesenden Charakter. Templated Links ermöglichen dem Client das Senden von Daten an einen Server. Beispielsweise können bei einem Suchdienst die Parameter übergeben werden, so dass die Suchbegriffe als ein Teil des Links mit übermittelt werden. Die Zusammensetzung des Links ist abhängig von der Spezifikation des genutzten Medientyps. Ein Verweis der eine Suche durchführt, sollte entsprechend der Spezifikation wie im folgendem Listing zusammengesetzt sein. [11]

Listing 3: HTML Beispiel LT

```
<a href="http://externe-domain.de">
```

Templated Links für Suchanfragen in HTML bestehen nach Spezifikation aus der Zieldomain und einem Platzhalter für die vom Nutzer übergebene Zeichenkette.

[10] (Vgl. Amundsen, 2012, S. 15)
[11] (Vgl. Amundsen, 2012, S. 16)

2.3.4 Idempotent Links

Idempotent Links (H-Faktor LI) haben einen schreibenden Charakter und stellen eine Möglichkeit dar, Ressourcen auf dem Server anzulegen oder zu löschen. Idempotenz hängt hierbei natürlich von dem verwendeten Transportprotokoll und der serverseitigen Implementation ab. Bei der Verwendung von HTTP als Transportprotokoll sind GET,PUT und DELETE idempotente Operationen zur Kommunikation mit dem Server. [12]

Listing 4: HTML Beispiel LI

```
<form action="http://domain.de/ressource" method="put">
  <input type="submit"/>
</form>
```

Das im Listig dargestellte HTML-Formular dient dem Anlegen oder Aktualisieren einer Ressource. Durch das Method-Attribut ist festgelegt, welche Protokoloperation auf die Zielressource ausgeführt wird. Im Zusammenspiel von HTML und HTTP als genutztes Transportprotokoll ist PUT als eine idempotente Operation implementiert und führt auch bei mehrfacher Ausführung immer zum gleichen Ergebnis.

2.3.5 Non-Idempotent Links

Non-Idempotent Links (H-Faktor LN) bietet wie H-Faktor LI eine Möglichkeit Daten zu übertragen. Der Unterschied hier ist die Idempotenz. So liegt es in der Verantwortung des Medientyps zu definieren, wie mit nicht idempotenten Operationen umgegangen wird. Im Fall von HTTP ist die Methode POST als nicht idempotente Operation implementiert. POST wird genutzt um per Formular eine neue Ressource auf dem Server anzulegen. [13]

Listing 5: HTML Beispiel LI

```
<form action="http://domain.de/ressource" method="post">
  <input type="submit"/>
</form>
```

HTML-Formulare, die als Operation POST nutzen um Daten zum Server zu senden, legen eine neue Ressource auf dem Server an. Im Gegensatz zu PUT als idempotente Operation liefert POST nach der Durchführung nicht immer das gleiche Ergebnis. Stattdessen legt POST bei mehr-

[12](Vgl. Amundsen, 2012, S. 16)
[13](Vgl. Amundsen, 2012, S. 17)

facher Ausführung, mehrere neue Ressource an. Der Umgang mit nicht idempotenten liegt in der Verantwortung und Implementation des konkret verwendeten Medientyps.

2.3.6 Read Controls

Read Controls (H-Faktor CR) beschreibt ob und wie die Möglichkeit besteht, eine lesende Operation zu reglementieren. Ein Beispiel hierfür sind die Accept-Header des HTTP Protokolls. Dem Client wird so die Möglichkeit gegeben ein bestimmtes Format oder die Antwort in einer bevorzugten Sprache anzufordern. [14]

Listing 6: HTML Beispiel CR

```
GET /beispiel/ressource HTTP/1.1
Host : domain.de
Accept : text/html
```

Read Control wird an dieser Stelle durch einen HTTP-Header veranschaulicht. Der Header resultiert aus einer lesenden Anfrage auf eine Ressource. Das Accept-Element im Header setzt hierbei die Steuerung der lesenden Operation GET um und schränkt die Annahme auf das Format HTML ein. Ressourcen mit mehreren Repräsentationsformen können auf diese Weise in einem spezifiziertem und vom Client unterstützen Format angefordert werden.

2.3.7 Update Controls

Update Controls (H-Faktor CU) bietet die gleichen reglementierenden Möglichkeiten, die der H-Faktor CR bietet für schreibende Operationen. Die Reglementierung von schreibenden Operationen kann bei HTTP z.B. durch das Content-Typ Feld im Header durchgeführt werden. So hat der Server Kenntnis darüber, in welcher Form ihm die Nutzdaten übergeben werden. [15]

Listing 7: HTML Beispiel CU

```
<form action="ressource/bild">
  <input type="file" accept="image/jpeg">
  <input type="submit">
</form>
```

[14] (Vgl. Amundsen, 2012, S. 18)
[15] (Vgl. Amundsen, 2012, S. 18)

Durch das Accept-Attribut wird der Upload einer Datei beschränkt. Das festlegen auf den Multipurpose Internet Mail Extension Type (MIME-Type) ïmage/jpegßtellt sicher, dass eine neue Ressource, die über dieses Formular erstellt wird, den MIME-Type JPEG haben muss. Ein Anlegen von neuen Ressourcen kann somit exakt reglementiert werden. Elementarer Bestandteil der Reglementierung ist das Verwenden von MIME-Types. Eine Liste aller offiziell registrierten MIME-Types wird durch die Organisation Ïnternet Assigned Numbers Authority"(IANA) gepflegt und veröffentlicht.

2.3.8 Method Controls

Method Controls (H-Faktor CM) stellen Indikatoren dar, die beschreiben um was für eine Art von Operation es sich handelt bzw. welche Operation für einen konkreten Fall gültig sind. Ein HTML FORM-Element kann mit unterschiedlichen Methoden abgesendet werden. Das Attribut METHOD beschreibt an dieser Stelle mit welcher Operation des Transportprotokolls das Formular übertragen wird. [16]

Listing 8: HTML Beispiel CM

```
<form action="ressource/bild" method="put">
<input type="submit"/>
</form>

<form action="ressource/bild" method="post">
<input type="submit"/>
</form>
```

Beide dargestellten Formulare senden ihre Nutzteten an "/ressource/bild". Sie unterscheiden sich jedoch in der verwendeten Protokolloperation. Durch die Method Controls kann am Beispiel von HTML festgelegt werden ob eine Ressource mit einer idempotenten oder mit einer nicht-idempotenten Operation angesprochen wird.

2.3.9 Link Annotation Controls

Zusätzlich zur Reglementierung von Schreib- und Leseoperationen kann ein Hypermedia Typ Link Annotation Controls (CL) unterstützen. Verweise werden dann mit semantischen Informationen angereichert. Clients, die diese Metadaten verstehen, können sich entsprechend der Semantik verhalten.

[16] (Vgl. Amundsen, 2012, S. 18f)

Listing 9: HTML Beispiel CL

```
<div>
  <a href="../seite1.html" rel="first">
  <a href="../seite4.html" rel="next">
  <a href="../seite2.html" rel="prev">
</div>
```

Dieses HTML Beispiel zeigt die Anreicherung der Links mit semantischen Informationen. Ein Client der die verwendeten Relationsattribute kennt kann die Sammlung von Verweisen durchsuchen und anschließend ohne seine eigene Position zu kennen auf die nächste, vorherige oder erste Seite navigieren.

2.3.10 Design Elemente

Zusätzlich zu den 9 H-Faktoren definiert Amundsen vier Design-Elemente. Diese Design-Elemente stellen vor allem vier charakteristische Eigenschaften von Hypermedia Formaten dar. Aus diesem Grund werden sie im Folgenden beschrieben und im weiteren Verlauf zum Vergleich unterschiedlicher Formate verwendet.
Die vier Hypermedia Design-Elemente sind: [17]

- Base Format
- State Transfer
- Domain Style
- Application Flow

Jeder Hypermediatyp basiert auf einem bestimmten zugrundeligenden Format. Bekannte Formate sind XML, JSON, CSV und viele weitere. Ein Hypermediatyp kann auf einem beliebigen Basisformat aufbauen. Erst die Anreicherung des Basisformats mit zusätzlichen Metadaten und der Unterstützung von Hypermedia-Faktoren in einem bestimmten Umfang definieren einen Medientyp. Der Medientyp bestimmt außerdem ob eine Zustandsänderung der Applikation unterstützt wird oder nicht. Man kann hier unterscheiden zwischen der Art und Weise wie ein Zusatndsänderung durchgeführt werden kann. Hierbei gibt es drei Ausprägungen der Unterstützung. Keine Unterstützung bei nur lesenden Formaten. Definierte Unterstützung nach einer externen Dokumentation und Ad-Hoc Unterstützung mittels einer in die Nachricht eingebetteten Hypermedia Steuerung. Der Domain Style beschreibt wie stark ein Medientyp an eine Problemdomäne angepasst ist. Diese Stile sind kategorisiert als spezifisch, allgemein und agnostisch. Als letztes Design

[17] (Vgl. Amundsen, 2012, S. 20)

Element gilt es noch den Application Flow zu etwähnen. Hierbei geht es darum ob und in welchem Maße der Client in der Lage ist den Applikationsfluss zu steuern. Auch hier unterscheidet man drei Ausprägungen. Der Applikationsfluss kann entweder nicht, wesentlich oder hauptsächlich durch den Medientyp gesteuert werden. [18]

2.3.11 Hypermedia Ebenen

Zusätzlich zu den Hypermedia-Faktoren und den Design Elementen definiert Amundsen noch vier Ebenen auf denen sich ein Hypermedia Typ befinden kann. Diese vier Ebenen sind aus der Sicht des Clients beschrieben und unterscheiden sich nach ihrem Möglichkeiten, in wie fern sie im Falle einer Veränderung angepasst werden müssen. Sie sind also gekennzeichnet durch einen bestimmten statischen und einen variablen Anteil. Statisch bezieht sich in diesem Kontext darauf, dass die statischen Anteile bei einer Veränderung oder Erweiterung der Applikation angepasst werden müssen. Die variablen Anteile hingegen können verändert werden ohne Probleme bei der Kommunikation zwischen Client und Server zu verursachen. Entweder dem statischen oder variablen Anteil zugeordnet werden die vier Eigenschaften Content, Address, Read Write Semantics und Appflow. Diese Eigenschaften sind wie folgt definiert:[19]

- Content beschreibt die Fähigkeit Elemente einer Nachricht hinzuzufügen oder zu entfernen.
- Address ist definiert als die Möglichkeit, einen Unified Ressource Identidifier zu ändern, an den der Client seine Anfragen sendet.
- Read Write Semantics ist eine Eigenschaft, bezüglich des Schreib-Leseverhaltens. Ein Hypermediatyp, der diese Eigenschaft besitzt ist in der Lage anzuzeigen, welche Elemente einer Nachricht schreibbar sind und welche Protokolfunktion zu benutzen sind.
- Appflow bezeichnet die Möglichkeit den Applikationsfluss kontextbezogen zu steuern. Der Client benötigt nur die Antwort des Servers und kann die Steuerelemente der Nachricht benutzen um sich im Applikationsfluss zu bewegen.

Als Ebene 0 definiert Amundsen Serialisierte Objekte. Diese Ebene trifft auf Techniken wie SOAP und RPC-XML zu. Die Kommunikation zwischen Client und Server läuft über ein spezielles, Anwendungsbezogenes Nachrichtenformat. Die Definition der Kommunikationsregeln, also welches Format mit welchem Inhalt wo benutzt wird muss durch eine externe Dokumentation erfolgen. Diese Regeln werden dann im Client codiert und sie verfügen über keinerlei variablen Anteil. Die Konsequenz ist, dass der Client im Falle einer Änderung in allen vier Eigenschaften angepasst

[18] (Vgl. Amundsen, 2012, S. 20)
[19] Vgl. Amundsen, M. (2010), `http://amundsen.com/hypermedia/scraps/#hypermedia-levels`

werden muss.

Ebene 1 ist beschrieben als standardisierte Datenformate. Beispielhafte Medientypen sind Comma Seperated Value (CSV) oder Extendible Markup Language (XML). Die Kommunikation zwischen Client und Server läuft über ein standardisiertes Datenformat. Dies hat den Vorteil, dass beide Seiten in der Lage sind den Nachrichten zu validieren. So können Elemente oder Attribute einer Nachricht hinzugefügt oder entfernt werden und die Kommunikation läuft weiterhin fehlerfrei. Der variable Anteil beschränkt sich jedoch auf die Eigenschaft Content. Die weiteren drei Eigenschaften sind nach wie vor statisch und eine Änderung würde zu Fehlern in der Kommunikation führen.[20]

Ebene 2 erweitert die standardisierten Datenformate mit Links. Atom und AtomPub sind Medientypen dieser Ebene. Der wesentliche Unterschied ist, dass ab dieser Ebene URIs in den Nachrichten enthalten sind. URIs werden somit nicht mehr im Client codiert sonder der Client bekommt mit jeder Nachricht URIs geliefert, denen er folgen kann. Dieser Fakt erweitert auch den variablen Anteil aus Ebene 1 um die Eigenschaft Address. Eine Anwendung, die einen Medientyp dieser Ebene benutzt kann ein variables URI-Schema benutzen. Sollte der Server seine URIs ändern, kann er dies tun ohne die Kommunikation zu bestehenden Client zu gefährden. Die Eigenschaften Read Write Semantics und Appflow sind weiter statisch.[21]

Ebene 3 ist genauso definiert wie Ebene 2, jedoch erweitert um die Möglichkeit der Applikationssteuerung. Die Kopplung zwischen Client und Server, die einen Medientyp dieser Ebene benutzen, besteht aus einer Vereinbarung über den verwendeten Datentyp und den Elementen einer Nachricht, die zur Applikationssteuerung benutzt werden. Der Client muss in der Lage sein die Steuerelemente einer Nachricht zu erkennen und dem Benutzer so die derzeit aktuellen Steueroptionen anzuzeigen. Hypermediaformate dieser Ebene sind unter anderem die Hypertext Markup Language (HTML) und VoiceXML.[22]

Betrachtet man die H-Level nach Amundsen etwas abstrakter, sind die Ebenen nach dem Grad der Kopplung zwischen Client und Server definiert. Wobei der Grad an Kopplung als Wissen verstanden werden kann, welches der Client über die Nachrichten haben muss. Dieses Wissen wird auch als Öut of Band Knowledge"bezeichnet. Der Verzicht, dieses Wissen innerhalb des Clients zu codieren, sorgt für eine bessere Flexibilität und Wartbarkeit des Gesamtsystems. Hypermediaformate ab der dritten Ebene betten ihre Nachrichten in einen semantischen Kontext. Die Kopplung besteht hier dann ausschließlich aus den definierten Steuerelementen und der Einigung über das

[20] Vgl. Amundsen, M. (2010), `http://amundsen.com/hypermedia/scraps/#hypermedia-levels`
[21] Vgl. Amundsen, M. (2010), `http://amundsen.com/hypermedia/scraps/#hypermedia-levels`
[22] Vgl. Amundsen, M. (2010), `http://amundsen.com/hypermedia/scraps/#hypermedia-levels`

Datenformat. Innerhalb dieser Arbeit werden die H-Faktoren und die vier Design-Elemente als Grundlage verwendet, um verschiedene Formate untereinander vergleichbar zu machen. Diese Ansicht ist nur als Vergleich zu sehen, da Hypermedia Typen nicht zwangsläufig alle H-Faktoren und Design Elemente unterstützen müssen. So kann ein rein zum lesen ausgelegter Medientyp z.B. auf das Design Element des Applikationsflusses verzichten. Ein gängiges Beispiel hierfür ist ein Atom-Feed Reader. Der Medientyp beinhaltet nur eine kurze Vorschau und einen Link auf den gesamten Artikel. Funktionen, die den Applikationsfluss steuern sind nicht notwendig.

2.4 Design-Prozess

2.5 Anwendungsbeispiel

3 Bewertung

3.1 Bewertungsgrundlage

4 Hypermedia-Formate

4.1 Basisformate

4.2 JSON

Beschreibung des Formats

4.2.1 Beispielentworten aus der Anwendung

4.2.2 Bewertung JSON

5 Fazit und Ausblick

*Literaturverzeichnis

Amundsen, M. (2012). *Building Hypermedia APIs with HTML5 and Node*. OReilly.

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. UNIVERSITY OF CALIFORNIA, IRVINE.

Ehrenwörtliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Bachelorthesis selbstständig und ohne unerlaubte fremde Hilfe angefertigt habe, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Wolfenbüttel, den 31. Mai 2013