Bachelorarbeit

im Studiengang Technische Redaktion

Web-App-Client-Entwicklung für die Technische Dokumentation und Erklärung der mit XML erstellten Sprache XWEB-TR zum Erzeugen von standardisierten Web-Apps mit wiederverwendbaren Inhalten

Verfasser: Aaron Maletz
Niemeyerstraße 15
30449 Hannover
Matrikelnummer: 1227919
Fachsemester: 8
Studiengang: Technische Redaktion

Hochschule Hannover

Prüfer: Prof. Dr. Volkert Brosda

Zweitprüferin: Prof. Dr. Claudia Villiger

Bearbeitungszeitraum: 27. Mai 2016 bis 29. August 2016

Datum: 26.08.16

Inhaltsverzeichnis

1. Einleitung

Genauso wie das Erstellen von Inhalten gehört auch die Wahl des Präsentationsmediums zu den Aufgaben eines Technischen Redakteurs. In der heutigen digitalisierten Welt wird viel Wert auf die Aufbereitung von Informationen gelegt. Da sich Informationen schnell verändern, ist es besser Form und Inhalt getrennt voneinander bearbeiten zu können. Durch eine solche Trennung lassen sich die Informationen auf unterschiedlichen Medien in verschiedensten Situationen verwenden.
Eine Web-App bietet den großen Vorteil, dass sie von jedem Endgerät aufgerufen werden kann. Sie ist unabhängig von Betriebssystemen, da sie über einen Webbrowser aufgerufen wird. Dennoch ist es aufwendig, für jeden Dokumentationsgegenstand eine eigenständige Web-App zu erstellen, ohne Teile wiederzuverwenden. Um eine Web-App als Form der Technischen Dokumentation einzusetzen, sollten Inhalt, Gestaltung und Funktionen modularisiert werden. Dazu kann eine standardisierte, erweiterbare Vorlage für den Erstellungsprozess benutzt werden. Mit der Metasprache XML (Extensible Markup Language) können eigene Auszeichnungssprachen gebildet und in andere Sprachen transformiert werden.

Diese Arbeit beschäftigt sich mit der Konzeption der Auszeichnungssprache XWEB-TR. Sie soll die Entwicklung von standardisierten Web-Apps ohne weitreichende Kenntnisse in der Webentwicklung ermöglichen. Außerdem ist sie speziell auf die Bedürfnisse der Technischen Dokumentation ausgerichtet. Ihre Funktion ist es, die Präsentation übersichtlicher und Inhalte wiederverwendbar zu machen. Durch diese zusätzliche Abstraktionsschicht entstehen viele Vorteile wie die Versionsunabhängigkeit von den verwendeten Frameworks. Außerdem wird durch die standardisierten Bereiche der Sprache XWEB-TR eine systematische Erstellung der Web-App sichergestellt.

2. Web-Apps für die Technische Dokumentation

In diesem Kapitel werde ich die Verwendung von Web-Apps in der Technischen Dokumentation näher beleuchten, ausgehend von den Funktionen einer Web-App über die Wiederverwendung der Inhalte zu den Vor- und Nachteilen im Vergleich zu anderen Formen der Inhaltsaufbereitung.

2.1 Die Web-App

Web-Apps sind mobil angepasste Webseiten, die mit den gängigen Webtechnologien HTML, CSS und JavaScript erstellt werden können. Unter Einbeziehung von speziellen Frameworks wie jQuery Mobile entstehen Programme, die sich in Aussehen und Verhalten kaum von nativen Apps unterscheiden. Die Hardware der Endgeräte kann angesprochen werden und dadurch beispielsweise der Standort ermittelt oder das aktuelle Wetter abgerufen werden.
Web-Apps sind speziell für den Einsatz auf mobilen Endgeräten angepasst, können aber ebenso auf jedem anderen Gerät mit üblichem Web-Browser angezeigt werden. Die Darstellung passt sich automatisch der Displaygröße an und die Inhalte können dadurch übersichtlicher strukturiert werden. (Heitkötter et al. 2012: S. 301-303)

2.2 Wiederverwendung durch Modularisierung

Bei einer strikten Trennung von Inhalt, Gestaltung und Funktionen nach dem Entwurfsmuster des Modell-View-Controllers kann eine hohe Wiederverwendbarkeit einzelner Komponenten der Software erzielt werden. Auf der Client-Seite geschieht dies durch das Auslagern der CSS-Stylesheet-Dateien und der JavaScript-Funktionen. Außerdem kann der Inhalt ausgelagert und dynamisch nachgeladen werden. Dadurch ist eine Anpassung des Inhalts zur Laufzeit möglich. Je kleiner die Module sind, umso einfacher ist ihre Wiederverwendung.

2.3 Andere Formen mobiler Darstellung

Native App:

Native Apps werden speziell für ein Betriebssystem entwickelt und können nur auf diesem verwendet werden. Dadurch unterstützen sie im Gegensatz zu Web-Apps alle Funktionen der Hardware des Endgerätes. Die App kann problemlos auf die Kamera oder das Mikrophon zugreifen. Dennoch entsteht durch die proprietäre Entwicklung ein erhöhter Programmieraufwand. (vgl. Heitkötter et al. 2012: S. 307-309)

Hybrid-App:

Durch die Umwandlung einer Web-App in eine Native App wird die Anbindung an die betriebssysteminternen Funktionen ermöglicht. Diese durch einen Wrapper geschaffene Hybrid-App erscheint den Anwendern wie eine Native App. Durch die Umwandlung verliert die App ihre leichte Aktualisierbarkeit. (vgl. Hellbom, 2013:S. 14)

2.4 Vor und Nachteile einer Web-App

Die Verwendung einer Web-App ermöglicht es, Software unabhängig von dem gewählten Endgerät (Smartphone, Computer, Applewatch u.a.), dem verwendeten Betriebssystem und dem verwendeten Browser einer breiten Masse an Personen zugänglich zu machen. Zudem ist die Entwicklung wesentlich kostengünstiger als die Erstellung von Native Apps für alle gängigen Betriebssysteme. Einen weiteren Vorteil bietet die schnelle und unbemerkte Aktualisierbarkeit der Web-App. (vgl. Heitkötter et al. 2012: S. 302 f.)

Eigenschaft	Web-App	Native App	Hybrid App
Verfügbarkeit	+ Auf jedem Betriebssystem mit aktuellem Browser aufrufbar	- nur für ein Betriebssystem entwickelt	○ Umwandlung von Web-App in Native App
Aktualität	+ Onlineaktualisierung + Keine Aktion vom Anwender notwendig + Immer auf dem neusten Stand	- Aktualisierung nur über den jeweiligen Store möglich - Aktion vom Anwender notwendig	○ Durch die Umwandlung muss die App durch den Store aktualisiert werden
Kosten	+ Kostengünstige Erstellung durch die Verwendung von Web-Technologien	- Aufwendige Erstellung für unterschiedliche Betriebssysteme.	+ Einfache Umwandlung von Web-App in Native App
Versionsabhängigkeit	- Abhängig von der jeweiligen Version des verwendeten Frameworks	○ Anpassung der App nur bei Aktualisierung des Betriebssystems	- Abhängig von beiden Faktoren: Version des Frameworks und Aktualisierung des Betriebssystems
Hardwarezugiff	- Beschränkter Zugriff auf Gerätefunktionen	+ Zugriff auf alle Funktionen des Gerätes	○ Durch nachträgliche Entwicklung Zugriff auf alle Funktionen des Gerätes möglich

(vgl. Gust, 2013: S. 23)

Durch die ständige Weiterentwicklung der Web-Technologien und deren Frameworks können einerseits Probleme mit den Versionen auftreten, die nur durch eine beständige Aktualisierung der Web-App vermieden werden können, andererseits führt dies zu neuen Features und besserer Hardwareanbindung. (vgl. Heitkötter et al. 2012: S. 302 f.)

2.5 Web-Apps in der Technischen Dokumentation

Die Verwendung von elektronischen Medien zur Darstellung von Inhalten ist kein neues Feld der Technischen Dokumentation. Webseitenstrukturen werden häufig

als Onlinehilfen für Computersoftware genutzt. Auch Apps finden schon Verwendung in der Technischen Dokumentation. Mercedes-Benz stellt seit 2012 mit der Guides App eine interaktive Bedienungsanleitung ihrer Fahrzeuge auf Smartphones und Tablets zur Verfügung. (vgl. Neumann, 2012)

Durch Web-Apps können Informationen schnell, effizient und personenbezogen zum Endanwender gelangen. Ein Fortschritt ist die Auswertung nutzerbezogener Daten, welche die ausgegebenen Informationen beeinflussen können.
Durch die Strukturierung der Inhalte in kleine Teilinhalte (Module), die einfach miteinander verknüpft werden können, entsteht eine besondere Übersicht der Inhalte.
Für den Technischen Redakteur soll die Erstellung möglichst einfach und das Ergebnis wiederverwendbar sein.

3. Die Sprache XWEB-TR

In diesem Kapitel werde ich eine kurze Einführung in die grundsätzliche Entwicklung eigener Sprachen mit XML geben. Danach erläutere ich die Komponenten einer beispielhaften Web-App und deren Überführung in die entworfene Sprache XWEB-TR mit der dazugehörigen Transformation in eine Web-App durch einen XSLT-Prozessor. Im Anschluss werde ich die beiden mit XWEB-TR erstellten Web-Apps erklären und Hinweise zur Erweiterung und sinnvollen Nutzung der Sprache geben.

3.1 XML zur Erstellung neuer Sprachen

Die Metasprache XML bietet die Möglichkeit, eigene auf XML basierende Sprachen zu erstellen. Bekannte Beispiele, die für die Technische Dokumentation ausgebaut wurden sind: XHTML, DITA und DocBook. Um eine neue XML-basierte Sprache zu erstellen, wird eine XML-Instanz erstellt, welche die Sprache repräsentiert. Dieser Instanz wird ein Stylesheet zur Überprüfung der Konformität zugewiesen, um zu gewährleisten, dass jede Instanz der neuen Sprache nach demselben Prinzip auf-

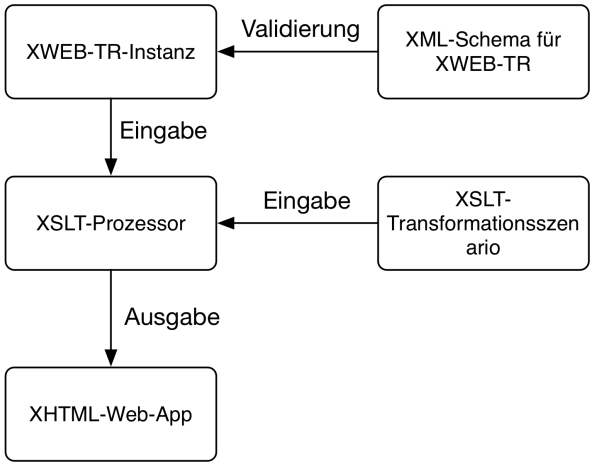
gebaut ist. Um die erstellte Sprache darzustellen, wird ein Prozessor benötigt, der die Sprache in eine andere Form bringt. Ein Beispiel dafür ist der XSLT-Prozessor, der die Instanz sowie das Stylesheet einliest, und sie nach bestimmten Regeln, den Templates, wieder ausgibt. (vgl. Jeckle, 2004)

3.2 Systematische Web-App-Client-Entwicklung

Um eine Web-App aus einer XML-Sprache zu erstellen, ist eine standardisierte Vorgehensweise bei der Konzeption sinnvoll. Die Idee dabei ist, Struktur, Layout und Ereignisbehandlung zu trennen. Dadurch können einzelne Komponenten einfach ausgetauscht werden. So wird die Struktur der Web-App durch ein HTML-Template festgelegt. Das Layout kann durch die Verwendung unterschiedlicher CSS-Layoutklassen geändert werden. Die Zuweisung des Layouts findet in der Ereignisbehandlung statt. Bei der Ereignisbehandlung wird auf eine objektbasierte und ereignisorientierte Programmierung gesetzt. Es werden Anwendungsklassen eingesetzt, die mit einer Reihe an Variablen und Methoden ausgestattet sind. Dazu werden Event Handler und andere Unterprogramme eingesetzt. Um eine Wiederverwendbarkeit der Funktionen zu schaffen, werden Parameter eingesetzt.

3.3 Erläuterungen der Sprache XWEB-TR

XWEB-TR soll Technischen Redakteuren einen einfachen Einstieg in die Entwicklung von Web-Apps als Medium für Dokumentationen bieten. In XWEB-TR werden Elemente definiert, die durch ein XSL-Transformationsszenario in eine Lauffähige Web-App umgewandelt werden. Die Konsistenz der XWEB-TR-Instanz wird mit einem XML Schema sichergestellt. Um die Web-App anzuzeigen, muss die ausgegebene XHTML-Datei über einen Webserver aufgerufen werden.



(Abb. 1: Übersicht der Eingabe- und Ausgabedokumente)

Im Folgenden werden die Struktur der Sprache und alle vorkommenden Elemente erläutert: Das Dokument-Element *web_app* besitzt die Attribute *project* und *title,* sowie die Referenz zur verwendeten Schema-Datei. Die direkten Kindknoten sind *classes, import-data, functions* und *eventhandler.*

In dem Element *classes* werden alle vorkommenden Anwendungsklassen der Web-App definiert. Jede Anwendungsklasse wird mit einem Element *class* definiert. Das Attribut *name* gibt den Namen der Klasse an. Als zweites Attribut kann mit *ref* noch der Name der Superklasse angegeben werden. Dadurch erbt diese Subklasse alle Eigenschaften der Superklasse. Es ist wichtig, als erstes *class*-Element eine Superklasse zu erstellen. Das Element *class* besitzt zwei Kindelemente *inst_var* und *methods*. Hier werden die jeweiligen Variablen und Methoden der Anwendungsklassen erstellt. Die Kindelemente von <inst_var> müssen mit den Namen der Elemente aus der XML-Datei mit den einzulesenden Daten übereinstimmen. Für jede Variable in den Anwendungsklassen wird eine Set-Methode erstellt. Soll diese Methode überladen werden, kann in den Methoden der Anwendungsklassen das Attribut *setter* hinzugefügt werden. Wird dem Attribut der Wert *yes* gegeben, wird diese Methode anstatt der Standardmethode implementiert. Die Methoden der Anwendungsklassen müssen in JavaScript geschrieben werden und

können nicht ausgelagert werden, da sie bei der Transformation den Anwendungsklassen zugeordnet werden müssen.

Das Element *import-data* enthält die Informationen zu dem verwendeten HTML-Template, den Inhalten (Content), den referenzierten JavaScript- und CSS-Dateien. Mit dem Element *layout* wird eine CSS-Datei referenziert, die Informationen zum Design der Web-App enthält. Das HTML-Template muss ebenfalls Bedingungen erfüllen, um eine geeignete Darstellung in der erzeugten HTML-Datei zu bilden. Die Inhalte der Website, aus denen Objekte der Anwendungsklassen erzeugt werden, können aus unterschiedlichen Quellen stammen. Mit dem Attribut *import-type* des Elements *idata* wird die Quelle bestimmt.

Das Element *functions* enthält die Sammlung aller Funktionen. Die Kindelemente *function* referenzieren Funktionen in den eingebundenen Javascript-Dateien. Der Wert des Attributs *name* muss mit dem Namen einer Funktion übereistimmen. Das Attribut *param* gibt die Namen der Parameter der Funktion an. Die Parameter werden über das *data* Element unterhalb der *event* Elemente an die referenzierte Funktion weitergegeben.

Das Element *eventhandler* kann mehrere *event* Elemente enthalten. Diese definieren jeweils ein ausgelöstes jQuery-Event. Jedem *event* Element muss der in *import-data* gewählte context-type zugewiesen werden, um eine Unterscheidung möglich zu machen und Fehler durch falsch eingesetzte Events zu vermeiden. Außerdem muss jedes *event* mit dem Attribut *ref* eine in *functions* definierte Funktion referenzieren. Mit dem Attribut *init* kann angegeben werden, ob das Event bei der Initialisierung des Objektes aufgerufen werden soll. Wenn das der Fall ist, müssen nur die Funktion und die Parameterdaten angegeben werden. Die Kindelemente *selector*, *fire-on*, *data* und *connector* bilden die Syntax eines jQuery –Events ab:

$$\$(selector).on('fire-on','connector',function(){$$
$$ref\_function([data]);$$
$$\});$$

3.4 Erläuterung zur Transformation

Jedes *class* Element in der Eingabedatei erzeugt in der Ausgabe eine JavaScript-Klasse. Die Instanzvariablen dieser Klasse werden durch die Kindelemente *inst_var* erzeugt. Zusätzlich wird für jede Klasse ein JavaScript-Prototyp gebildet, der automatisch Setter-Funktionen für jede Instanzvariable erschafft. Beispiel für die Transformation einer Anwendungsklasse:

Das Codebeispiel der Eingabedatei erzeugt in der Ausgabedatei den dargestellten JavaScript-Code.

Eingabe: Webapp_plants.xml

```
<class name="Plant">
            <inst_var>
                <id/>
                <img/>
                <name_de/>
                <name_lt/>
                <desc_short/>
                <desc_long/>
                <water/>
                <waterneed/>
                <critical/>
            </inst_var>
            <methods>
                <setWater setter="yes">
                    <![CDATA[var diffdays;
                    this.setWaterneed();
                    this.setCritical();
                    if (this.lastwater instanceof Date) {
                        diffdays = getDays(this.lastwater);
                    } else {
                        diffdays = 1;
                    }
                    this.water = Math.round((diffdays *
this.waterneed)*100)/100;
                    ]]>
                </setWater>
      </methods>
```

Ausgabe: webapp_transform_plants.xhtml

```
function Plant(o) {
            this.inst = o;
            this.id;
            this.img;
            this.name_de;
            this.name_lt;
            this.desc_short;
```

```
            this.desc_long;
            this.water;
            this.waterneed;
            this.critical;
            this.methods = ["setId", "setImg", "setName_de", "setName_lt", "setDesc_short",
"setDesc_long", "setWater", "setWaterneed", "setCritical", "giveWater", "overview", "de-
tail"];
        }
Plant.prototype = {
            setId: function () {
               this.id = $(this.inst).attr("id");
            },
            setImg: function () {
               this.img = $(this.inst).find("img")[0].textContent;
            },
            setName_de: function () {
               this.name_de = $(this.inst).find("name_de")[0].textContent;
            },
            setName_lt: function () {
               this.name_lt = $(this.inst).find("name_lt")[0].textContent;
            },
            setDesc_short: function () {
               this.desc_short = $(this.inst).find("desc_short")[0].textContent;
            },
            setDesc_long: function () {
               this.desc_long = $(this.inst).find("desc_long")[0].textContent;
            },
            setWater: function () {
               var diffdays;
               this.setWaterneed();
               this.setCritical();
               if (this.lastwater instanceof Date) {
                  diffdays = getDays(this.lastwater);
               } else {
                  diffdays = 1;
               }
               this.water = Math.round((diffdays * this.waterneed) * 100) / 100;
            },
            setWaterneed: function () {
               this.waterneed = $(this.inst).find("waterneed")[0].textContent;
            },
            setCritical: function () {
               this.critical = $(this.inst).find("critical")[0].textContent;
            }
        };
```

Der Import der referenzierten Inhaltsdaten wird mit AJAX realisiert. Die Daten aus der Quelldatei werden bei der Initialisierung der Web-App in JavaScript-Objekte geparst und in einem Array gesammelt. Die Eventhandler werden in der Transformation zu jQuery-Events.

Eingabedatei Webapp_plants.xml

```
<event context-type="jq-mobile" ref="layoutAssign">
            <selector>document</selector>
            <fire-on>click</fire-on>
            <data>'#page2 .ui-content div','detail'</data>
            <connector>.link</connector>
</event>
```

Ausgabedatei webapp_transform_plants.xhtml

```
$(document).on("click", ".link", function () {
                layoutAssign(['#page2 .ui-content div', 'detail'])
            });
```

3.5 Beispiel Web-App „Meine Pflanzen“

In der Web-App „Meine Pflanzen“, können die eigenen Pflanzen verwaltet werden. Der Benutzer wählt aus einer Liste seine Pflanzen aus. Jede Pflanze hat mehrere Eigenschaften:

- Den Typ der Pflanze
- Eine eindeutige ID, mit der jede Pflanze identifiziert werden kann
- Den Namen in Deutsch und Latein
- Eine Kurzbeschreibung
- Eine ausführliche Beschreibung
- Den Wasserbedarf pro Tag
- Den kritischen Wasserbedarf, der Punkt, ab dem der Benutzer alarmiert wird, dass die Pflanze dringend Wasser benötigt
- Der Pflanzentyp „cactus“ hat zusätzlich die eine Eigenschaft, mit der angegeben wird, wie viel Wasser im Winter benötigt wird.

Beim Starten der App werden die Pflanzendaten geladen. Danach kann der Benutzer mit einem Klick auf die jeweilige Pflanze die detaillierten Informationen der

Pflanze sehen. In dieser Ansicht ist es außerdem möglich, die Pflanze zu „bewässern“. Dadurch wird der Wert des benötigten Wassers auf Null gesetzt und das Datum für die Bewässerung in den Pflanzeneigenschaften vermerkt. Wird die Pflanze am nächsten Tag betrachtet, steigt der Wert der Bewässerung wieder auf den Wasserbedarf pro Tag an. Wenn der Benutzer die Pflanze nun mehrere Tage nicht „gießt“ und den kritischen Wert des Wasserbedarfs überschreitet, erscheint beim Betrachten der Pflanze eine Warnmeldung.

3.6 Beispiel Web-App „Personenverwaltung“

Diese Web-App erstellt eine Ansicht aller Personen aus einer Datenquelle. Jede Person hat die Eigenschaften: eine eindeutige ID zum Identifizieren der Person, einen Vornamen, einen Nachnamen, ein Foto oder ein Bild der Person und, falls die Person ein Student ist, eine Matrikelnummer. Auf der Startseite werden alle Namen der Personen in einer Liste ausgegeben, immer in der Reihenfolge „Vorname“, „Nachname“. Wenn man mit dem Mauszeiger über einen der Namen fährt, erscheint das Bild der dazugehörigen Person.

3.7 Verwendung in der Technischen Dokumentation

XWEB-TR ist vielfältig einsetzbar und nicht nur für die Erstellung von Dokumentationen entwickelt. Für die Technische Dokumentation ist XWEB-TR ein Werkzeug, um einfache Web-Apps zu erstellen, die stetig angepasst und erweitert werden können. Die kurze Einarbeitungszeit und die Modularisierung der Funktionen helfen bei der Erstellung der Web-App im Team. Die Programmierung der Funktionen kann an einen Dienstleister übergeben werden und die Redakteure können sich mit der Erstellung der Inhalte und des Designs befassen. Dafür sind nur Kenntnisse in XML und CSS notwendig. Nach der Fertigstellung können Inhalte und Design ausgetauscht werden. Das Ergebnis ist eine übersichtliche mobile Dokumentation, deren Inhalte immer auf dem neusten Stand sind. Das Design kann für jeden Kun-

den individuell angepasst werden. Durch eine Anpassung der Funktionen kann aus einer Web-App mit wenig Aufwand eine Onlinehilfe oder ein Wiki erstellt werden.

3.8 Probleme und mögliche Weiterentwicklungen

Da XWEB-TR bisher nur ein begrenztes Feld der Einsatzmöglichkeiten einer Web-App produzieren kann, wäre eine Erweiterung des Funktionsumfangs denkbar.
Besonders die Transformation erfüllt noch nicht die Ansprüche einer einsetzbaren Lösung. Der Prototyp der Transformation beherrscht bisher nur die Anbindung an XML-Dateien. Eine sinnvolle Erweiterung wäre es, eine Anbindung an SQL-Datenbanken zu implementieren. Um ein automatisches Nachladen der referenzierten Quelldaten zu garantieren, muss bei der Abfrage des Objekts dessen Inhalt mit dem Inhalt der Quelldaten ausgetauscht werden. Die bestehende Referenz der Objekte über ein Array wird dadurch unnötig. Um die Web-App auch Offline nutzen zu können, müssen die Objekte im Browsercache gespeichert und bei erneuter Verbindung mit dem Internet aktualisiert werden. Die Methoden der Anwendungsklassen werden in der XML-Instanz definiert. Es wäre übersichtlicher, wenn diese auch ausgelagert würden. Um die Web-App als Hybrid-App in den Stores zu vermarkten, wäre eine Umwandlung mittels eines Wrappers denkbar. Dadurch wird der App ihre Aktualität genommen. (Schneider, 2011)
Aus einer Vortransformation der Datenquelle in XML könnte eine XWEB-TR-Instanz mit der Übernahme der Anwendungsklassen ihrer Variablen erzeugt werden. Dadurch wäre die Arbeit noch einfacher und es könnten keine Fehler bei der Übernahme der Daten auftreten.

4. Fazit

Mit XWEB-TR können Technische Redakteure Dokumentationen und andere standardisierte Web-Apps entwickeln. Durch die Trennung von Inhalt, Präsentation und Funktionen kann eine Versionsunabhängigkeit zu den verwendeten Frameworks geschaffen werden. Dadurch ist außerdem eine Weiterentwicklung der Sprache und ihrer Transformation in eine lauffähige Web-App in Teams möglich. Selbst Redakteure ohne besondere Kenntnisse der Webtechnologien können mit der Sprache arbeiten. Die beiden in Kapitel 3.5 und 3.6 beschriebenen Beispielanwendungen haben gezeigt, dass die Verwendung von XWEB-TR mit geringem Aufwand zu unterschiedlichen Ergebnissen führen kann. Um die Sprache sinnvoll einsetzen zu können, sollte sie noch um einige Funktionen, wie in Kapitel 3.7 beschrieben, ergänzt werden. Wesentlich für den massentauglichen Einsatz ist die Einbeziehung unterschiedlicher Datenquellen. So wäre der nächste Schritt die Unterstützung von SQL-Datenbanken als Datenquelle für die auszugebenen Inhalte. Eine Umsetzung als Onlinehilfe oder Wiki kann realisiert werden, dazu sind nur wenige Anpassungen vorzunehmen.

Literaturverzeichnis

Gust, Dieter (2013): Technische Dokumentation auf mobilen Medien: Alles Apps oder was?, [online] http://webforum.tekom.de/fileadmin/user_upload/tekom/berichte/52678FAEED86D_TD_MobileMedia_Gust_20131021.pdf [24.08.2016]

Heitkötter,Henning, Sebastian Hanschke, und Tim Alexander Majchrzak (2012): Evaluating cross-platform development approaches for mobile applications, in *International Conference on Web Information Systems and Technologies*, München Heidelberg: Springer Verlag, S. 120-138.

Hellbom, Kim (2013): Mobile web apps as an alternative to native mobile apps, [online] https://www.theseus.fi/bitstream/handle/10024/56797/Examensarbetet_kimhellbom.pdf [31.07.2016]

Jeckle, Mario (2004): Entwurf von XML-Sprachen, [online] http://www.jeckle.de/entwurfxml/entwurfxml.html [20.08.2016]

Neumann, Markus(2012): Die mobile Bedienungsanleitung von Mercedes Benz: B-Class Guide App, [online] https://www.mobile-zeitgeist.com/die-mobile-bedienungsanleitung-von-mercedes-benz-b-class-guide-app [21.08.2016].

Schneider, Steve (2011): Web-Apps als Zukunft der mobilen Anwendungsentwicklung?, in: Michael Schenk(Hrsg), *Forschungskolloquium am Fraunhofer IFF*, Nr. 11./12., S. 39-42.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig, ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, an denen Quellen verwendet wurden, sind als solche kenntlich gemacht. Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht.

Ort, Datum Unterschrift