Technology
Arts Sciences
TH Köln

25knots – Ein Tool zur Verbesserung der gestalterischen Qualität von Artefakten im Hochschulkontext

Umsetzung vom Proof of Concept zur marktfähigen Webanwendung

BACHELORARBEIT

vorgelegt an der Technischen Hochschule Köln

Campus Gummersbach

Im Studiengang Medieninformatik

ausgearbeitet von

Christian Alexander Poplawski

MATRIKELNUMMER 11088931

Erster Prüfer: Prof. Dipl. Des. Christian Noss
Technische Hochschule Köln

Zweiter Prüfer: Dipl. Des. Liane Kirschner
Railslove GmbH

Gummersbach, im August 2017

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit beschäftigt sich mit der Umsetzung der Webanwendung *25knots*, die eine Verbesserung der gestalterischen Qualität von Artefakten im Hochschulkontext sichern soll. Das Konzept für die Anwendung wurde vorhergehend bereits im Praxisprojekt erarbeitet und beispielhaft in Form eines Proof of Concept umgesetzt. Im Rahmen der Abschlussarbeit soll nun, aufbauend auf den erarbeiteten Konzepten und dem Proof of Concept, ein marktfähiges Produkt erstellt werden. Dieses Produkt soll weiterhin von der Community[1]nicht nur verwendet, sondern auch aktiv weiterentwickelt werden können.

Generell sollen in dieser Arbeit alle Aspekte behandelt werden, die auf dem Weg von einem Prototypen zu einem marktfähigen Produkt eine Rolle spielen. Diese umfassen zum Beispiel die Struktur und Gestaltung der Anwendung, technische Entscheidungen die bei der Umsetzung getroffen werden müssen, aber auch Bereiche wie Hosting oder Möglichkeiten zur Weiterentwicklung. Im Folgenden sollen zunächst einige grundlegende Ziele und das generelle Vorgehen bei der Umsetzung definiert werden.

1.1 Motivation

Wie bereits erwähnt wurde das Konzept für die Anwendung bereits im Praxisprojekt erstellt. Dabei wurde viel Zeit und Energie investiert um sicher zu stellen, dass diese Konzepte auch Umsetzbar sind. So wurden zum Beispiel bereits viele (grobe) Algorithmen entworfen, die in der Anwendung verwendet werden können. Des weiteren konnte in persönlichen Gesprächen

[1] Als Teil der *Community* wird hier jede Person gesehen, die ein Interessen an der Anwendung besitzt.

mit verscheidenen Personen innerhalb und auch außerhalb des Hochscholkontextes festgestellt werden dass eine Umsetzung der erarbeiteten Konzepte durchaus eine Relevanz besitzt und von Personen verwendet werden würde.

Die Motivation für diese Arbeit ergibt sich daher aus dem Willen, aus diesem erarbeiteten Konzept einen Mehrwert schaffen zu wollen, der über das verdienen von Credit Points hinaus geht. Die generelle Motivation für eine Anwendung wie *25knots* lässt sich Kapitel *1.3: Relevanz des Themas* entnehmen

1.2 Zielsetzung

Ziel der vorliegenden Arbeit soll es sein, ein von der Community verwendbares und erweiterbares Produkt zu erstellen. Zunächst sei hier ein *verwendbares Produkt* im Rahmen dieser Arbeit definiert. Ein verwendbares Produkt:

1. bietet eine befriedigende Nutzererfahrung
2. ist öffentlich zugänglich
3. ist bei potentiellen Nutzern als Hilfsmittel zum Erreichen eines Zieles bekannt

Eine befriedigende Nutzererfahrung bedeutet für die Anwendung konkret, dass diese gut Strukturiert, ansprechend Gestaltet und ohne Probleme verwendbar sein muss. *Ohne Probleme verwendbar* impliziert hierbei eine hohe Qualität des geschriebenen Codes, um Fehler zu vermeiden. Der Großteil der Anwendung wird sich mit den Umsetzungen dieser Bereiche beschäftigen.

Der einfachste Weg, um eine Webanwendung öffentlich zugänglich zu machen ist in der Regel, diese auf einem Server zu hosten. Weiter details dieser Entscheidung werden im Kapitel *Hosting* erläutert.

Damit diese genutzt werden kann, müssen potentielle Nutzer der Anwendung von dessen Existenz wissen. Mit Blick auf die Zielgruppe bietet sich zunächst eine Bewerbung direkt an der Hochschule, beispielsweise durch die Teilnahme am *Medieninfromatik Showcase* an. Aber auch eine Bewerbung im größeren Rahmen, beispielsweise durch einen Talk beim *Webmontag Köln* ist denkbar.

Zuletzt sei auch die erweiterbarkeit der Anwendung konkret definiert. Eine mögliche Erweiterbarkeit bedeutet zunächst, dass der Code der Anwendung öffentlich verfügbar sein muss, beispielsweise auf der Plattform *Github*. Weiterhin muss der Code gut dokumentiert und verständlich geschrieben sein, um einen Einstieg in das bestehende Projekt zu einfach wie möglich zu halten. Eine erweiterbarkeit bezieht sich aber nicht nur auf geschriebenen Code, sondern kann auch auf konzeptioneller Ebene erfolgen. Auch hier muss die Möglichkeit zur Beteiligung so simpel wie möglich gehalten werden. Eine ausführlichere Diskussion findet sich im Kapitel *Release*

Aus diesen Punkten kann eine zentrale Forschungsfrage für die Arbeit formuliert werden: *Wie kann der Community ein nutzbares und erweiterbares Produkt auf Basis eines Konzeptes und Prototypen bereitgestellt werden?*

Diese lässt sich in zwei Unterfragen unterteilen, die es zu beantworten gilt:

1. Durch welche Maßnahmen kann eine aktive Nutzung und Weiterentwicklung durch die Community gewährleistet werden?

2. Welche technischen Entscheidungen müssen während der Entwicklung getroffen werden?

1.3 Relevanz des Themas

Die Relevanz der Anwendung an sich wurde bereits im Praxisprojekt erläutert, daher soll hier nur eine Kurzfassung der Erläuterung folgen. Der Grundgedanke der Relevanz ist dabei folgender:

> Menschen bilden sehr schnell ein Urteil über die Gestaltung eines Artefaktes und dieses lässt sich nur schwer wieder ändern. Weiterhin wird dieser schlechte erste Eindruck auf andere Bereiche des Artefaktes übertragen und wirkt sich somit unter Umständen auch auf die Gesamtbewertung eines Artefaktes aus.

Gestützt wird dieser Gedanke durch Studien von [**?**], [**?**], und [**?**].

Unterstützend seien hier noch zwei weitere Quellen aufgeführt, die die Relevanz weiter unterstreichen: [**?**] zeigen, dass die Ergebnisse der Studie von Lindgaard et al. auch mit anderen Parametern bestand haben und unterstreichen weiterhin die Wichtigkeit von guter Gestaltung für eine gute Nutzererfahrung [**?**].

Neben der Relevanz des Produktes, das in Rahmen der Arbeit entstehen soll ist aber auch das eigentliche Thema der Arbeit zu rechtfertigen: Die Entwicklung von einem Konzept zu einem fertigen Produkt. Als abschließende Arbeit für den Studiengang Medieninformatik ist dieses ein passendes Thema, da in diesem viele Aspekte aus verschiedenen Modulen des gesamten Studiums vereint werden. Daher bietet das Thema eine gute Verbindung zwischen den verschiedenen Disziplinen innerhalb des Studiums, verbunden mit einer wissenschaftlichen Diskussion verschiedener Vorgehensweisen und Abläufe.

1.4 Abgrenzung

Auch wenn es Teil der Zielsetzung ist, ein marktfähiges Produkt zu erstellen kann hier nicht davon ausgegangen werden, dass das Endergebnis der Arbeit ein Produkt ist, dass als fertig angesehen werden kann. Auch mit Blick auf die spätere Weiterentwicklung durch die Community muss es viel mehr das Ziel sein, eine hochwertige erste Version des Produktes, die als Grundlage für weitere Features dient, also ein *Minimum Viable Product* zu erstellen.

1.5 Zielgruppe

Während der Konzeption im Praxisprojekt wurden Studenten der Technischen Hochschule Köln im Studiengang Medieninformatik als Zielgruppe festgelegt. Es wurde aber bereits im Praxisprojekt deutlich, dass diese Zielgruppe leicht erweiterbar ist. Somit kann jede Person einen Mehrwert aus diesem Tool ziehen, der ein Artefakt erstellen muss, dessen Hauptaugenmerk eigentlich nicht auf der Gestaltung, sondern auf einer bestimmten Funktion liegt. Das kann im Studium an einer mangelnden Bewertung oder in der Wirtschaft an einem mangelndem Budget liegen, jedoch entstehen häufig Situationen, in denen eine grundlegend solide Gestaltung nicht als wichtig erachtet wird, jedoch durchaus Vorteile mit sich bringt.

Für diese Arbeit werden aber zunächst weiterhin die Studenten des Studienganges Medieninformatik als Zielgruppe definiert, um eine Disparität zwischen dem Konzept und der Umsetzung auszuschließen.

1.6 Struktur der vorliegenden Arbeit

Außerhalb dieser Einleitung Teilt sich die Arbeit in drei weitere Kapitel. Im zweiten Kapitel werden zunächst einige Theoretische Grundlagen erläutert und Entscheidungen auf einer taktischen Ebene erläutert. Im dritten Kapitel werden einige Aspekte der Umsetzung der Anwendung erläutert, während im vierten Kapitel einige Fragen bezüglich der Veröffentlichung der Anwendung beantwortet werden.

NOTIZ: Hier wird am Ende noch einmal drüber geschaut, wenn die tatsächliche Struktur etwas deutlicher ist.

Kapitel 2

Theoretische Grundlagen

Hier fehlt noch eine generelle Einleitung zum Kapitel

2.1 Struktur der Anwendung

In diesem Abschnitt soll zunächst die generelle Struktur der Anwendung definiert werden. Weite Teile der Struktur können dabei aus dem Praxisprojekt übernommen werden. Im Praxisprojekt wurden die folgenden Themengebiete behandelt:

- Typographie
- Layout & Struktur
- Whitespace
- Farben
- Bilder
- Interaktive Elemente

Nach einer erneuten evaluation der Ergebnisse des Praxisprojektes konnten die in der Abschlussarbeit zu behandelnden Themengebiete auf zunächst drei eingegrenzt werden (der Bereich *Layout & Struktur* wurde dabei in *Layout & Grids* umbenannt):

- Typographie
- Layout & Grids

- Farben

Diese Abgrenzung begründet sich auf verschiedene Weisen. Im Fall des Bereiches *Whitespace* konnte im Rahmen des Praxisprojektes kein zufriedenstellendes Konzept erarbeitet werden, wodurch sich dieser Bereich per se nicht für eine Umsetzung eignet. Weiterhin muss es Ziel dieser Arbeit sein, am Schluss eine vollständige Anwendung zu erhalten. Um dieses Ziel im zeitlichen Rahmen erreichen zu können mussten weitere Themengebiete vernachlässigt werden. Hier boten sich die Bereiche *Bilder* und *Interaktive Elemente* an, da diese für die gestalterische Grundqualität eine vergleichsweise niedrige Rolle spielen. **NOTE: Vielleicht muss das hier noch belegt werden, wie ein Brötchen**. Diese Bereiche wurden aber ausreichend konzeptioniert und bieten sich als erste Erweiterungen für die Anwendung nach beenden der Arbeit an.

Außerdem müssen für die Anwendung jeweils ein nutzerfreundlicher Einstieg und Ausstieg gefunden werden. Diese wurden im Praxisprojekt nicht explizit ausgearbeitet und fallen somit auch Konzeptionell in den Bereich der Abschlussarbeit und werden später in diesem Kapitel behandelt.

Die finale Struktur der Anwendung für den Rahmen dieser Arbeit sieht also wie folgt aus:

- Einstieg
- Typographie
- Layout & Grids
- Farben
- Ausstieg

2.1.1 Einstieg in die Anwendung

Bereits im Praxisprojekt wurde festgestellt, dass es sinnvoll ist, das Zielmedium des Nutzers zu kennen. Mit Blick auf die Zielgruppe wurden hier drei mögliche Bereiche definiert: Native App, Website und Textdokument. **NOTE: Hier verweis auf Untersuchung im PP**. Diese Bereiche können jedoch auch in sich verschiedene Eigenarten aufweisen, so kann ein Textdokument beispielsweise für das Lesen an einem Bildschirm oder das Lesen in gedruckter Form entworfen werden. Eine komplette Auflistung der möglichen Bereiche oder *scopes* der Anwendung findet sich in Abbildung 2.1 auf Seite 10.

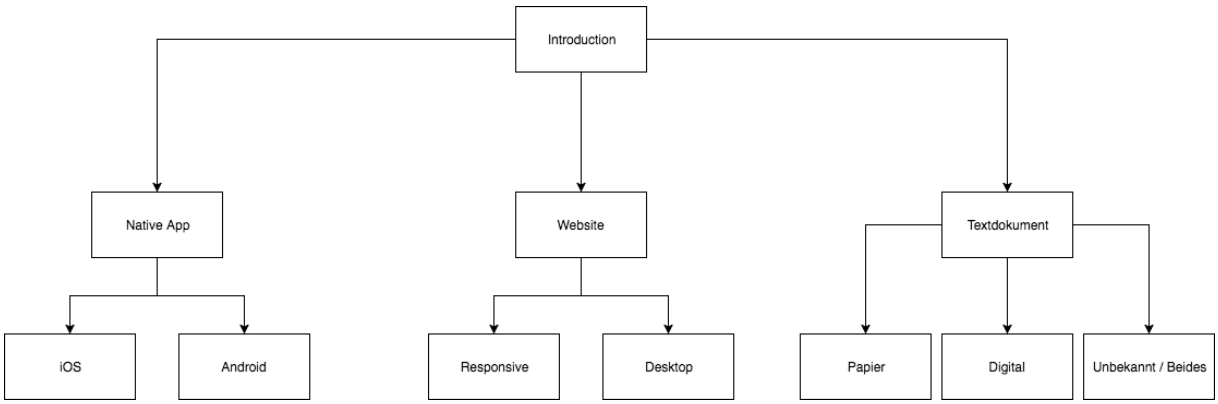

Abbildung 2.1: Mögliche Entscheidungen im Einstieg der Anwendung

Obwohl die Abgrenzung der Bereiche für die hier definierte Zielgruppe ausreichend ist, lassen sich bereits jetzt einige Stellen erkennen, die bei einer möglichen späteren Erweiterung der Zielgruppe überarbeitet werden müsste. Vorrangig betrifft das den Bereich *Website*. Hier ist die vorhandene Unterteilung in *Responnsive* und *Desktop* für ein Echtwelt-Szenario unter Umständen zu allgemein gehalten.

Um den kognitiven Aufwand **Beleg** für den Nutzer möglichst gering zu halten, bietet es sich an, ihn Schrittweise durch das Festlegen des für ihn passenden Bereiches zu führen.

2.1.2 Ergebnisse der Benutzung

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

2.2 Diskussion verfügbarer Technologien

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

2.2.1 Vue.js

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

2.2.2 Angular.js

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

2.2.3 React.js

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

2.3 Technologie-Stack

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

2.4 Gestaltung

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

Kapitel 3

Entwicklung der Anwendung

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

3.1 Gestaltung

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

3.1.1 Wireframes & Struktur

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

3.1.2 High Fidelity Mockups

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

3.2 Besonderhieten im Technologie-Stack

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

3.2.1 Redux

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

3.2.2 React Storybooks

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

3.2.3 CSS-Architektur

Für den Umgang mit CSS in React.js bieten sich verschiedene Möglichkeiten an. Nativ sind zwei Vorgehensweisen möglich: Anwenden von CSS über ausgelagerte Stylesheets (wie es in der Regel bei allen Webseiten gemacht wird) oder das verwenden von Inline-Styles.

Das verwenden von klassischen Stylesheets unterscheidet sich nicht sehr von der Verwendung bei einer statischen Webseite. Auch in React werden Klassennamen oder IDs festgelegt, über die dann im Stylesheet das Aussehen definiert wird (natürlich sind auch alle anderen validen CSS Selektoren anwendbar, Klassen und IDs seine hier nur als die populärsten Varianten beispielhaft genannt). Auch die Verwendung von CSS-Präsporzessoren wie zum Beispiel SCSS stellt kein Problem dar, die kompilierung dieser Dateien muss lediglich in den Build-Prozess von React.js mit eingebunden werden. Eine simplifizierte Anwendung sähe beispielsweise wie folgt aus:

```
<ReactComponent className='myClass'>
  Content
</ReactComponent>
```

Die kompilierte Dom-Node wäre dabei

```
<ReactComponent className='myClass'>
  Content
</ReactComponent>
```

Und könnte im Stylesheet über

```
.myClass {
  color: red
}
```

Angesprochen werden. Auch die Verwendung von Methodiken wie BEM oder SMACCS ist mit diesem Ansatz ohne Probleme möglich. Für den Rahmen dieses Projektes wurde sich jedoch gegen diese Vorgehensweise entschieden, da bewusst ein Fokus auf eine komponentenbasierte Architektur gelegt werden sollte. Auch mit einer komponentenbasierten CSS-Architektur ist eine Komponente immer noch auf zwei Orte aufgeteilt: Die Funktion und das Markup, und das Styling.

[HIER LIESSE SICH AUCH DIE DISKUSSION NOCH AUSFÜHREN, DASS EIGENTLICH LEUTE LANGE DAFÜR GESPROCHEN HABEN, AUSSEHEN UND MARKUP ZU

TRENNEN UND WARUM DAS HIER OKAY IST]

Um das Styling und die Funktion einer Komponente an einem Ort zu halten, bieten sich inline-styles an. Wie auch bei statischen Webseiten werden inline-styles direkt im `style`-Attribut eines Elementes definiert. In React.js werden diese als JavaScript-Objekt übergeben. Eine Beispielhafte Anwendung findet sich in Quellcode XYZ

```
const styles = {
  backgroundColor: 'red',
  fontSize: '12px'
}

export default function myComponent(props) {
  return (
    <div styles={styles}>
            {props.children}
        </div>
  )
}
```

Auf den ersten Blick wirkt die Verwendung von inline-styles problematisch, vielleicht weil diese auf statischen Seiten viele Nachteile mit sich bringen. In einem Komponentenbasierten System wie React.js sind diese Nachteile jedoch nicht present. Durch die Komponentenbasierte Struktur und das damit einhergehende Ziel der Wiederverwendung von Komponenten, müssen Stylingänderungen auch hier nur an einer Stelle vorgenommen werden. Da jede Komponente nur für ihr eigenes Styling verantwortlich ist und nicht für das von Kindern, und auch kein CSS außerhalb der inline-styles verwendet wird (abgesehen von CSS-Reset und einigen globalen Regeln wie der Schriftart), kann es zu keinen Problemen mit der Spezifität von CSS-Regeln kommen.

Allerdings weisen inline-styles einige Limitierungen auf. So können beispielsweise keine Pseudo-Elemente wie `:after` oder `:before` verwendet werden. Auch das definieren von hover-states via `:hover` wird nicht unterstützt.

[HIER NOCH VERWEIS AUF SPEZIFIKATION]

Zwar sind diese Limitierung zum aktuellen Stand des Projektes noch nicht von allzu großer Bedeutung, mit Blick auf eine Weiterentwicklung der Anwendung nach der Abschlussarbeit

können diese in Zukunft jedoch eine größere Rolle spielen. Um diese Limitierungen zu umgehen, wurde das Framework Aphrodite verwendet. Das Framework erlaubt das Festlegen von styles innerhalb der Komponente ähnlich wie inline-styles, erzeugt aber für jedes neue styleobjekt eine einzigartige CSS-Klasse, die via `className` angewendet wird. Die Einzigartigkeit der Klasse wird durch das hinzufügen eines Hauses am Ende des Klassennamens gewährleistet. Listing XYZ zeigt ein Beispiel

```
import React from 'react'
import { StyleSheet, css } from 'aphrodite'

function myComponent(props) {
      <div className={css(styles.componentStyles)} >
            {props.children}
      </div>

      const styles = StyleSheet.create({
            componentStyles: {
                  color: 'blue'
            }
      })
}
```

Die Struktur der style-objekte ist dabei der von inline-style-objekten gleich.

Mit der Verwendung von Aphrodite können also Aussehen und Funktion von Komponenten in der gleichen Datei gehalten werden, ohne dabei den Limitierungen von inline-styles zu unterliegen.

3.3 Algorythmen

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

3.4 Tests

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

Kapitel 4

Veröffentlichung der Anwendung

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

4.1 Hosting

Um den Zugang zur Anwendung für die Community möglichst einfach zu gestalten, bietet es sich an, diese zu hosten (einer andere, weniger geeignete Möglichkeit, wäre beispielsweise nur die Veröffentlichung des Quellcodes und das lokale hosten durch den Nutzer selbst). Da es sich bei der Anwendung um eine rein Clientseitige handelt, stehen eine Vielzahl von Möglichkeiten für das hosting zu Verfügung, da der Server lediglich statische html- und JavaScript-Dateien ausliefern muss.

In die nähere Auswahl kamen in diesem Projekt Github Pages und Heroku. Ein hosting auf einem privaten Server wurde schnell ausgeschlossen, um die mögliche zusätzliche Arbeit möglichst gering zu halten. Da der Quellcode der Anwendung bereits auf Github veröffentlich wurde, scheint das Hosting über Github Pages zunächst die naheliegendste und einfachste Lösung. Ein Deployment auf Github Pages erfolgt nach entsprechenden Einstellungen im Repository einfach durch einen push auf den gh-pages branch. Die Anwenudung ist danach über die Domain username.github.io/rpository erreichbar. Größter Vorteil ist dabei, dass kein externer Service benötigt wird. Es treten aber auch einige Limitierungen auf. Beim testen der Hosting-

Möglichkeiten wurde deutlich, dass das Modul React-Router den zusätzlichen Path nicht ohne weiteres unterstützt. Zwar bieten sich hier relative einfach Lösungen wie das verwenden von hashes anstatt von Pfaden im React-Router an, jedoch wurde mit blick auf die bereits bei simplen dingen auftretenden Probleme diese Möglichkeit zunächst als weniger geeignet eingestuft.

Für das Hosting wurde sich für den Service Heroku entschieden. Heroku ist ein Saas, das ein Deployment und hosting ohne Setup und Konfiguration erlaubt. Ein Deployment auf Heroku läuft dabei über einen remote branch im git repository, kann also mit dem Befehl `git push heroku master` gestartet werden. Heroku erkennt die verwendete Sprache automatisch und stellt alles nötige automatisch ein. Jedoch unterstützt Heroku lediglich Serverseitige Progrmmiersprachen und beitet keinen Support für das Hosting von statischen files. Um diese Anwendung zu hosten, muss also ein Server geschreiben werden. Da dieser nur statische Files hosten muss, ist dieser sehr simple in Node.js und dem Framework express geschrieben:

```
const express = require('express')
const app = express()

app.use(express.static(__dirname + '/dist'))
app.listen(process.env.PORT || 8080)
```

Der Server hostet dabei den ordner `dist`, in dem sich die gebuildeten Files befinden (unter anderem auch die Datei `index.html`, auf die ein Browser automatisch zurück greift). Mit der Datei `Procfile` wird Heroku dann mitgeteilt, welche Befehle beim Deployment der Anwendung ausgeführt werden soll. Im Falles dieser Anwendung ist das die Datei `server.js`. Das `Procfile` besteht alsolediglich aus der Zeile

```
web: node server.js
```

Da sowohl Github Pages, als auch Heroku von sich aus eine recht kryptische URL verwenden, wurde weiterhin die Domain `25knots.de` gekauft, um auch hier den Einstieg für potentielle Nutzer so einfach wie möglich zu gestalten.

4.2 Weiterentwicklung

Nachfolgend soll außerdem auf eine mögliche Weiterentwicklung der Anwendung nach dem Abschluss dieser Arbeit eingegangen werden. In diesem Zusammenhang stellen sich vor allem drei Fragen:

1. Wie können Personen aus der Community dazu gebracht werden, an einer Weiterentwicklung mitzuarbeiten?

2. Wie kann eine durchgängig hohe Qualität des Quellcodes garantiert werden, auch wenn viele verschiedene Menschen daran arbeiten?

3. Wie kann das Mitwirken für Interessenten möglichst einfach gestaltet werden?

Eine konkrete Beantwortung der ersten Frage gestaltet sich recht schwierig. Es lässt sich jedoch ein logischer Zusammenhang zwischen der Anzahl der Nutzer und der Anzahl der Mitwirkenden eines Projektes feststellen. Am Beispiel der Plattform Github können *stars* als relativ sichere Mindestzahl der Nutzer gesehen werden. Laut [**?**] besteht ein Zusammenhang zwischen *stars* und der anzahl der Mitwirkenden an einem Projekt:

> In git-based systems, forks are used to either propose changes to an application or as a starting point for a new project. In both cases, the number of forks can be seen as a proxy for the importance of a project in GitHub. [...] Two facts can be observed in this figure. First, there is a strong positive correlation between stars and forks (Spearman rank correlation coefficient = 0.55). Second, only a few systems have more forks than stars.

Somit ist eine Steigerung der Mitwirkenden also automatisch mit einer Steigerung der Nutzer der Anwendung verbunden. Der Inhalt diese Abschnittes soll also vorrangig mit der Beantwortung der zweiten und dritten Frage beschäftigen.

4.2.1 Sicherung der Code-Qualität

Grober Aufbau für diese Sektion:

1. Travis-CI

2. es-lint

4.2.2 Dokumentation

Grober Aufbau für diese Sektion:

1. Was zeichnet eine gute Dokumentation aus?

2. Dokumentation innerhalb des Codes

3. Dokumentation außerhalb des Codes

4.2.3 Contribution Guidelines

zahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

4.2.4 Tests

zahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

4.3 Vermarktung

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

Kapitel 5

Schluss

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

5.1 Fazit

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln

5.2 Ausblick

Bei der Textgestaltung und automatischen Änderung von Abbildungsnummern, Querverweisen, Seitenzahlen, Gliederungen, Literaturhinweisen etc. bietet sich der Rückgriff auf moderne Textverarbeitungsprogramme an. Nutzen Sie diese zur besseren Lesbarkeit und Strukturierung des Textes, aber vermeiden Sie überflüssige Spielereien. Da besonders bei Textdokumenten mit eingebundenen Objekten wie Bildern, Formeln