

H T
W I
G N

Hochschule Konstanz
Department of Computer Science

Vorgelegt von
Michael Fried
Matrikelnummer 295568

michael.fried@htwg-konstanz.de

B

C



Bachelorarbeit

Entwicklung eines Logbuchs als Progressive Web Application

S



Bachelorarbeit

Entwicklung eines Logbuchs als Progressive Web Application

von

Michael Fried

zur Erlangung des akademischen Grades

Bachelor of Science
in Angewandter Informatik

an der Hochschule Konstanz Technik, Wirtschaft und Gestaltung.

Matrikelnummer: 295568

Abgabedatum: 26.Februar 2020

Erstbetreuer: **Prof. Dr. Marko Boger**
Zweitbetreuer: **Prof. Dr. Markus Eiglsperger**

Quellcode: <https://github.com/friedcode/progressive-sailing>

Ehrenwörtliche Erklärung

Hiermit erkläre ich, Michael Fried, geboren am 01.06.1994 in Waldshut,

- (1) dass ich meine Bachelorarbeit mit dem Titel:

Entwicklung eines Logbuchs als Progressive Web Application

in der Fakultät Informatik unter Anleitung von Professor Dr. Marko Boger
selbstständig und ohne fremde Hilfe angefertigt habe und keine anderen als die
angeführten Hilfen benutzt habe;

- (2) dass ich die Übernahme wörtlicher Zitate, von Tabellen, Zeichnungen, Bildern
und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die
Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen
innerhalb der Arbeit gekennzeichnet habe.
- (3) dass die eingereichten Abgabe-Exemplare in Papierform und im PDF-Format
vollständig übereinstimmen.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Konstanz, 28.Februar 2020

Abstract

Das Ziel dieser Studie ist es, ein elektronisches Logbuch für die Domäne Bootsfahrt, mithilfe von aktuellen Webtechnologien, zu entwickeln. Die Anwendung soll durch responsive Design für die meisten gängigen Bildschirmgrößen optimiert sein, und durch Service Worker Offlinefunktionalitäten bieten. Die Informationen für die Logbucheinträge erhält die Anwendung von einem Server auf dem Boot, welcher das Open Source Protokoll Signal K verwendet.

Um solch eine Anwendung zu implementieren, wurden Nachforschungen betrieben, die den Stand der Technik von Webtechnologien beleuchten und die Stärken und Schwächen von bereits existierenden elektronischen Logbüchern aufzeigen. Daraufhin wurde ein Prototyp mit den, in der Softwareentwicklung gängigen Techniken erstellt. Und letztendlich wurde der Prototyp als Progressive Web Application implementiert.

Diese Abschlussarbeit zeigt, dass man mit den heute verfügbaren Webtechnologien eine Anwendung entwickeln kann, die an keine Plattform gebunden und auch mit nativen Anwendungen mithalten kann.

Inhaltsverzeichnis

Abbildungsverzeichnis	ix
Tabellenverzeichnis	xi
Listingverzeichnis	xiii
Abkürzungsverzeichnis	xvi
1 Einleitung	1
1.1 Problemstellung	1
1.2 Ziel der Arbeit	2
1.3 Vorgehensweise	3
2 Theoretischer Rahmen	5
2.1 Bootsfahrt	5
2.1.1 Logbuchführung	5
2.1.2 Signal K	6
2.2 Usability und User Experience	7
2.3 Prototypentwicklung	8
2.3.1 Scribbles	9
2.3.2 Wireframes	9
2.3.3 Mockups und Prototypen	10
2.4 Webentwicklung	11
2.4.1 Responsive Design	11
2.4.2 CSS-in-JS	12
2.4.3 State Management	13
2.4.4 Progressive Web App	13
2.4.5 Scalable Vector Graphics (SVG)	14
3 Analyse	15
3.1 Nachforschungen über digitale Logücher	15
3.1.1 Logbook App	15
3.1.2 2K Yachting	17

3.2	Technischer Zustand von Webtechnologien	18
3.2.1	Progressive Web Apps	18
3.2.2	Frameworks	19
4	Implementierung	21
4.1	Prototyperstellung	21
4.1.1	Scribbles	21
4.1.2	Wireframes	22
4.1.3	Prototyp	24
4.2	Entwicklung als Progressive Web App	25
4.2.1	Auswahl der Kerntechnologien	25
4.2.2	Initialisierung	26
4.2.3	Navigation und Routing	26
4.2.4	Icon Implementierung	27
4.2.5	CSS und Responsive Design	29
4.2.6	Service Worker und Caching	31
4.2.7	State Management	32
4.2.8	Signal K Implementierung	35
5	Diskussion	37
5.1	Bewertung genutzter Technologien und Methoden	37
5.2	Interpretation von Ergebnissen	38
5.3	Empfehlungen für weiterführende Arbeit	39
6	Fazit	41
	Literatur	43

Abbildungsverzeichnis

2.1	Usability und User Experience	8
2.2	Teilaspekte der User Experience	8
2.3	Verschiedene Versionen von Scribbles	9
2.4	Wireframing Anwendung Balsamiq	10
2.5	Prototyping Anwendung Figma	11
2.6	Umstrukturierung der Seite	11
2.7	Service Worker	14
2.8	Generierte SVG Grafik	14
3.1	Logbook App	17
3.2	2k-sailing Logbook Suite	18
3.3	Wöchentliche npm Downloads vom 18.08.2019-09.02.2020	20
4.1	Scribbles von der Logpage	22
4.2	Wireframes von der Logpage	23
4.3	Prototyp der Logpage	24
4.4	Navigation Drawer	27
4.5	Position SVG Icon	28
4.6	Responsive Design Progressive Sailing	30

Tabellenverzeichnis

3.1 Summe der Downloads vom 12.12.2014-19.12.2019	19
4.1 Genutzte Libraries	25

Listings

2.1	CSS Media Query	12
2.2	SVG Datenstruktur	14
4.1	Router Imports(app.js)	26
4.2	Browser Router(app.js)	26
4.3	Switch Komponente(app.js)	26
4.4	Link Komponente(app.js)	26
4.5	Textfeld in SVG	27
4.6	Position Komponente(position.js)	28
4.7	Nutzung einer Icon Komponente	28
4.8	Styled Component für den Navigation Drawer(app.js)	29
4.9	Implementierung einer styled-component(app.js)	29
4.10	Implementierung der Mobil- und Desktopansicht(LogsOverview.js)	31
4.11	Anpassung der Spaltenzahl(LogDetails.js)	31
4.12	Aktivierung des Service Worker(index.js)	32
4.13	Redux Store(store.js)	32
4.14	Redux Store Einbindung(index.js)	32
4.15	Selector(selectors.js)	33
4.16	Verbindung zum Store(LogsOverview.js)	33
4.17	Action für Hinzufügen von Logeintrag(actions.js)	33
4.18	Reducer für Hinzufügen von Logeintrag(reducers/logs.js)	34
4.19	Funktionen zum Speichern und Laden des States(localStorage.js)	34
4.20	Store Erweiterung(store.js)	35
4.21	Request Pfad	35

Abkürzungsverzeichnis

3G die dritte Generation von Mobilfunkstandards

Abb Abbildung

API Application-Programming-Interface

App Application (Anwendung)

bzw Beziehungsweise

COG Course Over Ground

CRA Create React App

CSS Cascading Style Sheets

DOM Document Object Model

ggf Gegebenfalls

GPS Global Positioning System

HTML Hypertext Markup Language

HTTPS ... Hypertext Transfer Protocol Secure

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

JS JavaScript

Lst Listing

Mmsi Maritime Mobile Service Identity

NMEA National Marine Electronics Association

npm Node package manager

NW Nordwestlich

- PC** Personal Computer
- props** Properties (Eigenschaften)
- PWA** Progressive Web Application
- SchSG** ... Schiffssicherheitsgesetz
- sm** Seemeilen
- SOG** Speed Over Ground
- SVG** Scalable Vector Graphic
- TCP** Transmission Control Protocol
- u.a** und andere
- UDP** User Datagram Protocol
- UI** User Interface
- URL** Uniform Resource Locator
- UTC** Coordinated Universal Time
- UX** User Experience
- W3C** World Wide Web Consortium
- XML** Extensible Markup Language

1

Einleitung

Heutzutage kann es sehr aufwendig sein eine Anwendung für Endnutzer zu entwickeln. Durch die Vielfalt an Endgeräten und deren zahlreiche Unterschiede bleibt Entwicklern oft nichts anderes übrig, als mehrere Anwendungen für unterschiedliche Endgeräteklassen zu entwickeln. Verschiedene Betriebssysteme und variierende Bildschirmgrößen sind dabei die Hauptgründe dafür. Dieser Lösungsansatz hat jedoch einige Nachteile. Für die unterschiedlichen Betriebssysteme sind Kenntnisse in unterschiedlichen Programmiersprachen nötig, was die Komplexität der Implementierung erhöht. Oft werden die verschiedenen Version von mehreren Teams entwickelt. Dies bedeutet für das Unternehmen mehr Personal, und einen höheren zeitlichen und finanziellen Aufwand.

Um diese Probleme zu umgehen, setzen viele Unternehmen auf Progressive Web Apps (PWA). PWAs sind Webanwendungen, die als Webseite im Browser oder als installierte Mobilanwendung auf den meisten Endgeräten genutzt werden können.[\[Rak+\]](#)

1.1. Problemstellung

Diese Studie beschäftigt sich mit der Analyse, Konzeption und Implementierung eines elektronischen Logbuchs für die Domäne Bootsfahrt. Ein Logbuch zu führen ist für die meisten Schiff- und Bootsfahrten Pflicht. Traditionell wird dieses mit Stift und Papier geführt.

Durch den Wunsch nach intelligenten Lösungen für die Bootsfahrt, wurde das Open

Source Datenprotokoll Signal K gegründet. Signal K ist ein modernes und offenes Datenformat für die maritime Nutzung.[[Proa](#)] Mithilfe von Signal K ist es möglich, alle nötigen Informationen für einen Logbucheintrag direkt von den Sensoren und Instrumenten des Bootes auf ein Endgerät zu erhalten.

Daher stellt sich für diese Studie folgende Frage:

Ist es möglich eine plattformunabhängige Anwendung zur Visualisierung einer Vielzahl an Daten (aus der Domäne Bootsfahrt), die für unterschiedliche Bildschirmgrößen optimiert ist und auch Internet-unabhängig operieren kann, allein mit Mitteln der Webtechnologien zu implementieren?

1.2. Ziel der Arbeit

Ziel ist es nun eine Webanwendung zu implementieren, mit welcher der Nutzer Logbucheinträge erstellen und einsehen kann. Das Endgerät ist mit einem Webserver auf dem Boot verbunden, welches die gewünschten Daten mithilfe des Signal K Datenformats an das Endgerät liefert. Diese Daten werden in der Anwendung gespeichert und visualisiert.

Die Anwendung soll alle Bedingungen einer Progressive Web App erfüllen. Dazu zählen[[Devb](#)]:

- **Sicherheit:** Die Seite wird über HTTPS serviert.
- **Responsive Design:** Die Seite passt sich der Bildschirmgröße an und funktioniert dadurch auf Smartphones, Tablets und Desktops.
- **Offlinefunktionalität:** Alle URLs laden auch wenn das Gerät Offline ist.
- **Installierbarkeit:** Durch Metadaten ist die Option verfügbar, die Anwendung zu installieren.
- **Funktionalität bei 3G:** Das erste Laden der App soll bei 3G Verbindung unter 10 Sekunden sein.
- **Cross-Browserfunktionalität:** Die App funktioniert auf allen modernen Browsern.
- **Nichtblockierbarkeit des Netzwerks:** Seitenwechsel sollten sich schnell anfühlen, selbst bei einer langsamen Verbindung.

- **Auffindbarkeit:** Jede Seite sollte eine eigene URL haben.

Außerdem sollte ein Logbucheintrag folgende Informationen enthalten:

- **Position:** Die Position des Bootes in Längen- und Breitengraden.
- **Kurs:** Die Richtung in die das Boot fährt.
- **Geschwindigkeit:** Die Geschwindigkeit des Bootes über Grund und durch das Wasser.
- **Wetterbericht:** Luft- und Wassertemperatur. Windrichtung und -stärke. Wetterzustand(Sonnig, Bewölkt, Regen).
- **Barometerstand:** Luftdruck
- **Notizen:** Ein Feld in dem der Nutzer Kommentare und Ereignisse hinzufügen kann.
- **Schiffsdaten:** Schiffsname, Mmsi ggf. weitere Kennzeichen.

1.3. Vorgehensweise

Diese Studie beginnt mit einer Marktrecherche über elektronische Logbücher. Es werden Vergleichsprodukte gesucht, deren Funktionalitäten recherchiert und deren Vor- bzw. Nachteile beleuchtet. Daraus wird extrahiert, welche Eigenschaften die Zielanwendung benötigt, und wie die Nachteile von Vorgängerprodukten vermieden werden können.

Daraufhin werden Nachforschungen über den Stand der Technik von Webtechnologien durchgeführt. Es wird untersucht, welche Technologien benötigt werden, um die geplanten Eigenschaften der Zielanwendung zu implementieren und eine Auswahl darüber getroffen.

Anschließend wird ein Prototyp erstellt. Dies beginnt mit der Erstellung von ersten Scribbles von einzelnen Elementen der Anwendung auf Papier, dann werden diese Scribbles in ein Wireframe eingebunden um die Positionierung und Navigierbarkeit in der Anwendung zu testen. Dieses Wireframe bietet dann die Grundstruktur für den Prototypen, welcher dann das Design und den Aufbau der Anwendung vorgibt.

Zuletzt wird die Zielanwendung mithilfe der ausgewählten Technologien und des Prototypen als Webanwendung implementiert.

2

Theoretischer Rahmen

In diesem Kapitel wird die Rolle des Logbuchs in der Domäne Bootsfahrt und das offene Datenformat Signal K beleuchtet. Es werden wichtige Aspekte der User Experience und Webentwicklung erklärt.

2.1. Bootsfahrt

2.1.1. Logbuchführung

§ 6 Absatz 3, des Schiffssicherheitsgesetzes besagt:

Der Schiffsführer hat - falls nicht anders vorgeschrieben, im Schiffstagebuch - unverzüglich durch geeignete Eintragungen über alle Vorkommnisse an Bord zu berichten, die für die Sicherheit in der Seefahrt einschließlich des Umweltschutzes auf See und des Arbeitsschutzes von besonderer Bedeutung sind. Bei Schiffsunfällen hat der Schiffsführer, soweit erforderlich und möglich, für die Sicherstellung der Eintragungsunterlagen zu sorgen.[Jus]

Das Logbuch, im Gesetzestext als Schiffstagebuch bezeichnet, ist für die meisten Schiffsführer in Deutschland Pflicht. Meistens werden diese Logbucheinträge traditionell mit Stift und Papier dokumentiert. Dafür muss der Schiffsführer die nötigen

Daten manuell von den Instrumenten ablesen und aufschreiben.

Die Führung eines Logbuchs dient vielen Zwecken, zum Beispiel:[Woe18]

- **Sicherheit in der Navigation:** Regelmäßiges Notieren der Position, Kurs und Geschwindigkeit bringt Sicherheit in der Navigation, sollten die elektronischen Geräte ausfallen.
- **Wetterbeurteilung:** Die Aufzeichnung von Wetterberichten, eigenen Wetterwahrnehmungen und Barometerständen verschaffen eine Basis zur Beurteilung der Wettersituation.
- **Beweismittel:** Bei Unfällen dient ein detaillierter Logbucheintrag als Nachweis zum Unfallhergang.

2.1.2. Signal K

Signal K wurde als offenes Datenformat für die maritime Nutzung entwickelt.[Proa] Es ist auf heutigen Standards für Webtechnologien wie JSON, WebSockets und HTTP aufgebaut. Dadurch stellt es einen Web-freundlichen Weg für den Informationsaustausch auf dem Boot zur Verfügung. Signal K steht frei von Kosten für jeden Nutzer und Entwickler zur Verfügung. Das Datenformat unterstützt mit NMEA 0183 und NMEA 2000 die gängigsten Kommunikationsprotokolle der maritimen Industrie.[Proc] Damit kann es auf meisten Booten angewandt werden.

Die Leitmotive des Signal K Projektes sind:

- **Offen:** Es ist frei nutzbar für jeden. Signal K wird von einer Gemeinschaft von Schiffsgelehrten verwaltet, und Entwickler sind in der Lage Verbesserungen zum Standard vorzuschlagen.
- **Modern:** Es nutzt weit verbreitete, Open Source Technologien.
- **Erweiterbar:** Es kann mit neuen Anforderungen erweitert werden, sobald sie benötigt werden.
- **Flexibel:** Signal K ist nicht an spezifische Hardware gebunden.
- **Respektvoll:** Es ist so konzipiert, damit es mit bereits existierender Ausrüstung und Protokollen arbeiten kann.

Um Signal K auf einem Boot benutzt zu können, benötigt man einen Computer, Raspberry Pi oder ähnliches auf dem Boot, auf welchem ein Signal K Server läuft.

Dieser ist mit der Schnittstelle des NMEA Netzwerks verbunden und kann damit die Sensor- und Instrumentendaten lesen, sie in das Signal K JSON-Format umwandeln und in einer Datenbank speichern, oder an verbundene Geräte weiterschicken.

Signal K bietet seinen Quellcode auf GitHub an.[[Prob](#)] Da findet man bereits nutzbare Server in Nodejs oder Java, die Verbindungen mithilfe von HTTP, WebSockets oder TCP anbieten, eine Client Library welche die Client-seitige Verbindung mit dem Server übernimmt, oder Frontendimplementierungen wie das Instrumentpanel oder Freeboard-sk.

2.2. Usability und User Experience

Der Begriff Usability kann am besten als Gebrauchstauglichkeit oder Nutzerfreundlichkeit ins deutsche übersetzt werden.[[JM18a](#)]

Von der DIN EN ISO 9241-11 wird Usability folgendermaßen definiert:

[...]das Ausmaß, in dem ein Produkt, System oder Dienst durch bestimmte Benutzer in einem bestimmten Anwendungskontext genutzt werden kann, um bestimmte Ziele effizient und zufriedenstellend zu erreichen.[[ISO18](#)]

Der Begriff User Experience(UX) kann am besten als Nutzungserfahrung oder Nutzungserlebnis ins deutsche übersetzt werden.[[JM18a](#)]

Von der DIN EN ISO 9241-210 wird UX folgendermaßen definiert:

[...]alle Aspekte der Erfahrungen eines Nutzers bei der Interaktion mit einem Produkt, Dienst, einer Umgebung oder Einrichtung.[[ISO11](#)]

Usability ist daher ein Teilbereich der User Experience (siehe Abb. 2.1). Es bezieht sich eher auf die Tauglichkeit des User Interface(UI) einer Anwendung.[[JM18a](#)]



Abbildung 2.1: Usability und User Experience

Quelle: [JMb]

Die User Experience bezieht sich auf sämtliche Services, Abläufe und Zusammenhänge zwischen Unternehmen, Produkt, Kommunikation und Markenbildung.[JM18a] In Abb. 2.2 werden die Aspekte der User Experience dargestellt:

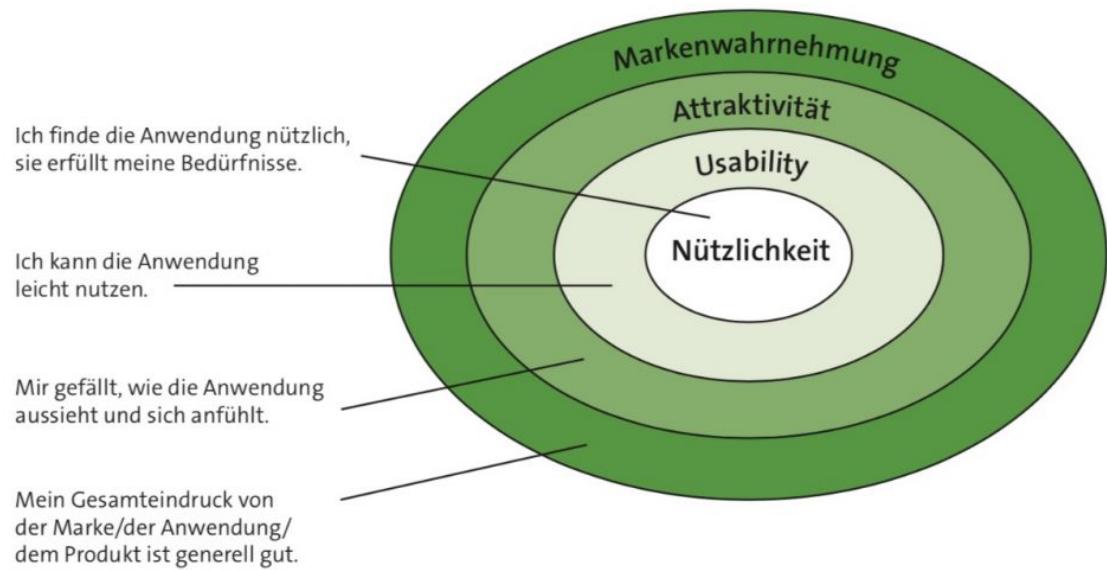


Abbildung 2.2: Teilespektren der User Experience

Quelle: [JMa]

2.3. Prototypentwicklung

Ein wichtiges, in dieser Arbeit benutztes, Werkzeug des UI und UX Design ist die Prototypentwicklung. Damit ist es möglich, schon früh in der Konzeptionsphase

Feedback einzuholen, und anhand dieser Änderungen durchzuführen.

2.3.1. Scribbles

Bei der Prototypentwicklung ist es empfehlenswert, zuerst mit Scribbles zu beginnen. Ein Scribble, auf Deutsch Gekritzeln, ist eine einfache Skizze, die helfen soll Ideen zu entwickeln und Klarheit über den Aufbau und Funktionen einzelner Seiten zu bekommen.[JM18b] Man kann hierbei ganze Seiten, oder auch nur einzelne Elemente skizzieren. Mithilfe von Scribbles ist es möglich viele Varianten einer Idee zu visualisieren (siehe Abb. 2.3). In dieser Phase sind Konzeptänderungen noch sehr günstig und schnell zu machen.

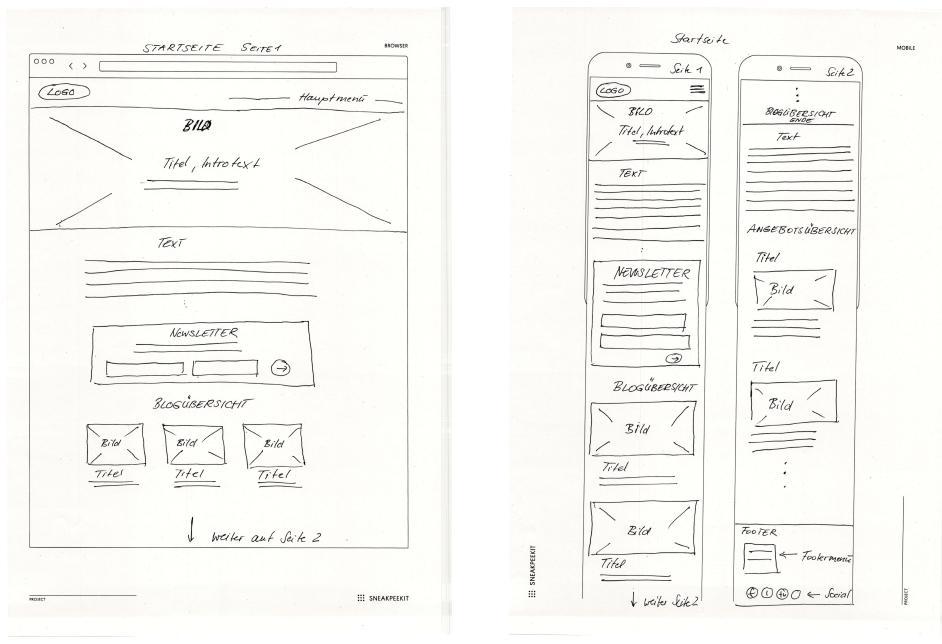


Abbildung 2.3: Verschiedene Versionen von Scribbles
Quelle: <https://wonderwebwoman.de/website-planung/>

2.3.2. Wireframes

Nachdem die Kernideen mithilfe von Scribbles festgehalten sind, ist es möglich diese in Wireframes zu konkretisieren. Wireframes, auf Deutsch Drahtgittermodell, sind Skizzen von ganzen Seiten auf denen die wichtigsten Elemente auf dem Bildschirm zu sehen sind.[JM18c] Für Wireframes gibt es spezielle Programme, die eine Erstellung

erleichtern und beschleunigen, indem sie schon gewisse Beispielkomponenten, wie Knöpfe, Listen oder Navigationselemente zum Einfügen per Drag-and-Drop zur Verfügung haben (siehe Abb. 2.4).

Bei Wireframes spielt der Inhalt noch keine große Rolle und ist oft mit Platzhaltern versehen. Dabei werden bewusst Designaspekte wie Farben, Typografie und Bilder ausgelassen. Hier geht es eher um die Positionierung und die Struktur, und nicht den konkreten Inhalt der Seite. Wireframes können genutzt werden, um in einer frühen Phase Usability-Tests durchzuführen und Feedback einzuholen.[JM18c]

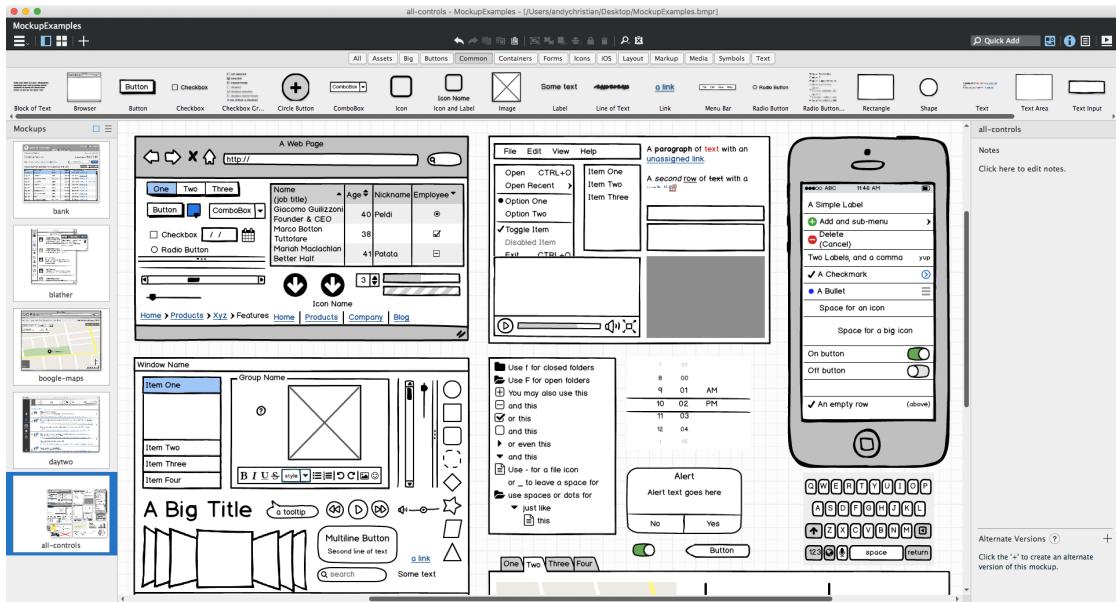


Abbildung 2.4: Wireframing Anwendung Balsamiq

Quelle: <https://swarmonline.com/our-favourite-uxui-design-software/>

2.3.3. Mockups und Prototypen

Hat man ein finales Wireframe, ist es möglich das Gesamtkonzept in einem Mockup oder Prototypen zu visualisieren. Mockups sind statische Bilder die zeigen, wie das fertig implementierte Produkt aussehen kann. Diese werden oft in Zeichenprogrammen erstellt. Prototypen dagegen sind dynamische, interaktive Modelle des Produkts.[JM18d] Prototypen sind dabei auch gut geeignet für Usability-Tests. Für Prototypen gibt es spezielle Programme, mit denen auch Interaktionsmöglichkeiten, wie zum Beispiel der Wechsel zu einer anderen Ansicht bei Knopfdruck (siehe Abb. 2.5), leicht zu implementieren sind. Mockups und Prototypen konkretisieren letztendlich das Aussehen und die Usability der Anwendung. Damit ist es möglich, viele verschiedene Designs auszuprobieren.

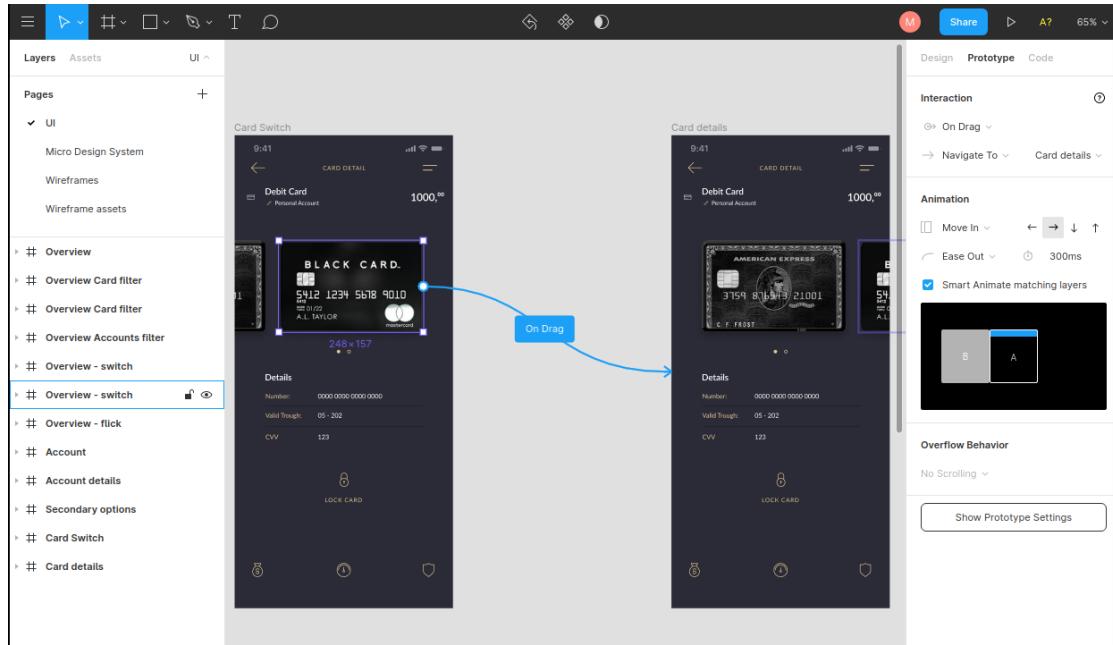


Abbildung 2.5: Prototyping Anwendung Figma
Quelle: Bildschirmabbildung von Figma

2.4. Webentwicklung

2.4.1. Responsive Design

Eine Webseite ist responsive, wenn sie, mithilfe von HTML und CSS, automatisch Teile versteckt, die Größe ändert oder sich umstrukturiert um auf jedem Endgerät gut auszusehen (Desktops, Tablets und Smartphones)(Abb.2.6).[w3sb]

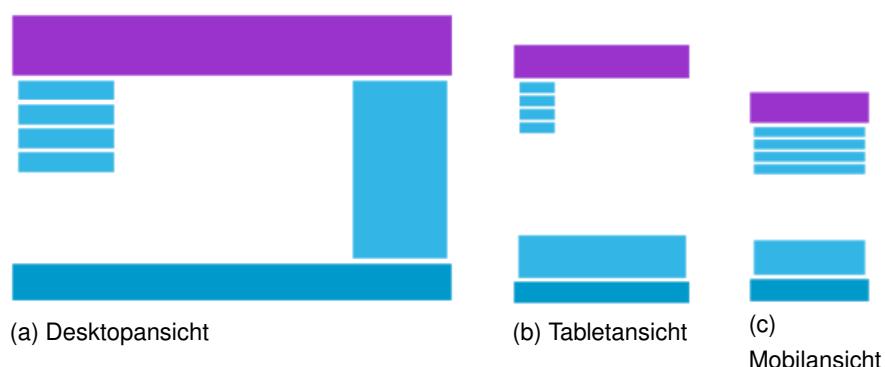


Abbildung 2.6: Umstrukturierung der Seite
Quelle: https://www.w3schools.com/css/css_rwd_intro.asp

Ein wichtiger Denkansatz im Webdesign ist Mobile First. Hierbei wird in der Konzeption einer Webanwendung zuerst die Mobilansicht geplant.[JM18e] Dies hat erstens den Grund, dass mobile Endgeräte heutzutage eine sehr hohe Relevanz für das Web haben, und zweitens damit die relevantesten Anwendungsfälle und Nutzungsmöglichkeiten zum richtigen Zeitpunkt angeboten werden können.

Um eine Webseite responsive zu machen, gibt es verschiedene Hilfsmittel. Flexbox und CSS Grid sind dabei die bekanntesten. Die Grundidee des Flexbox Layouts ist es, dem Container die Fähigkeit zu geben, die Höhe und Breite der Elemente in dem Container anzupassen, um den verfügbaren Raum bestmöglich auszufüllen.[Coy] CSS Grid ermöglicht es, ein grid-basiertes zweidimensionales Layout, mit Reihen und Spalten, zu definieren.[w3sa] Dabei ist zu bemerken, dass Flexbox besser bei eindimensionalen, und CSS Grid besser bei zweidimensionalen Layouts funktioniert. Es ist auch möglich beide Hilfsmittel zusammen zu benutzen.

Ein weiteres Hilfsmittel, welches CSS zur Verfügung stellt, sind Media Queries. Damit ist es möglich die Breite des Bildschirm, oder auch die Ausrichtung (Hochformat, Querformat), abzufragen, um dann für diesen Fall spezielle CSS Regeln anzuwenden. Listing 2.1 würde zum Beispiel bewirken, dass bei allen Geräten deren Bildschirmauflösung in der Breite größer als 600 Pixel ist, das Body Element die Hintergrundfarbe Rot hat.

```
1  @media (min-width: 600px) {  
2      body {  
3          background-color: red;  
4      }  
5  }
```

Listing 2.1: CSS Media Query

2.4.2. CSS-in-JS

CSS-in-JS ist eine Methode, in der CSS Styles in JavaScript definiert sind, anstatt in einer separaten .css Datei.[Inc] Die Probleme, die diese Methode versucht zu lösen, sind unter anderem Globale Namespaces, Abhängigkeiten, Dead Code oder geteilte Konstanten.[Che] Außerdem ermöglicht CSS-in-JS die Verwendung von, im JavaScript definierten Variablen in CSS Style Regeln.

2.4.3. State Management

Die meisten Frameworks kommen mit ihren eigenen State Management Lösungen. Mit immer komplexer werdenden Anwendungen können diese Lösungen jedoch unübersichtlich werden. Unter State Management Libraries ist Redux die bekannteste. Redux kann mithilfe von drei Prinzipien beschrieben werden:[Aab]

- **Einzig wahre Quelle:** Der Zustand (State) der gesamten Anwendung ist an einem Ort gespeichert.
- **Der State ist read-only:** Der einzige Weg den State zu ändern ist durch eine action.
- **Änderungen werden mit pure functions gemacht:** Um zu beschreiben wie sich der State Tree ändert benutzt man eine Funktion, die den vorherigen state und eine action nimmt, und den nächsten State zurückgibt.

Redux wird meistens mit React und React Native genutzt, kann aber auch mit anderen Frameworks genutzt werden. Für Vue gibt es eine von Redux inspirierte, und ähnlich funktionierende State Management Library namens Vuex.

2.4.4. Progressive Web App

Progressive Web Apps sind Webanwendungen, die ein installierbares App-ähnliches Erlebnis auf Desktop und Mobilgeräten bieten.[LeP] Dabei haben PWAs folgende Kernmerkmale:

- **Schnell und Zuverlässig:** Die Zeit, um aussagekräftige Inhalte anzuzeigen und ein interaktives Erlebnis bietet, muss schnell sein. Und dies bei jedem Anwendungsstart.
- **Installierbar:** Die PWA kann wie native Apps installiert werden, und läuft in einem App Fenster ohne Adressleiste oder andere Browser UI.
- **Mobil und Desktop:** Mithilfe von Responsive Design Techniken (siehe Kapitel 2.4.1) passt sich die Anwendung der Bildschirmgröße an.

Diese Erweiterungen werden nicht durch eine einzige Technologie ermöglicht, sondern durch eine Ansammlung an Strategien, Techniken und APIs.[She17]

Um eine Webanwendung installierbar zu machen, benötigt man ein Web App Manifest. Dies ist eine JSON Datei, die dem Browser sagt, wie die Anwendung aussehen soll, wenn sie installiert ist. Ein Manifest definiert unter anderem den Namen, das

Icon und die Start URL.[GK] Damit eine Webanwendung schnell lädt und auch ohne Netzwerkverbindung eine gewisse Basiserfahrung bieten kann, werden Service Worker benötigt. Ein Service Worker ist ein Skript, das der Browser im Hintergrund startet, und Eigenschaften, die keine Nutzerinteraktion benötigen, kontrolliert.[Gau] Dazu zählen u.a Hintergrundsynchronisation und Benachrichtigungen. Die Anwendung bietet dann eine Offline-Erfahrung, indem der Service-Worker die statischen Daten im Cache speichert, und bei Bedarf diese Daten serviert(Abb. 2.7).

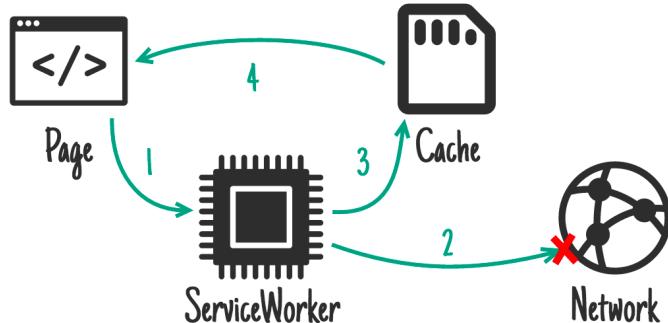


Abbildung 2.7: Service Worker

Quelle: <https://codelabs.developers.google.com/codelabs/your-first-pwapp/#4>

2.4.5. Scalable Vector Graphics (SVG)

SVG ist ein Dateiformat für Vektorgrafiken, entwickelt von W3C.[Ink] Es basiert auf XML, und hat dadurch auch eine ähnliche Struktur (Lst.2.2).

```

1   <svg xmlns="http://www.w3.org/2000/svg" width="300" height="200">
2       <rect x="50" y="50" width="200" height="100" style="fill:teal"/>
3   </svg>
```

Listing 2.2: SVG Datenstruktur

Lst.2.2 ergibt damit die Abbildung 2.8.



Abbildung 2.8: Generierte SVG Grafik

Quelle: <https://inkscape.org/de/entwickeln/das-svg-format/>

Die Vorteile von SVG gegenüber Pixelbasierten Grafiken sind, dass zum einen, das Format Menschenlesbar ist, und zum anderen, dass Grafiken auf- und abskaliert werden können, ohne verpixelt auszusehen.

3

Analyse

Um zu bestimmen, welche Eigenschaften ein elektronisches Logbuch haben sollte, werden in diesem Kapitel Vergleichsprodukte und deren Funktionalitäten untersucht. Anschließend wird der technische Zustand von Webtechnologien beleuchtet, um zu erkennen was die technischen Möglichkeiten dieser sind.

3.1. Nachforschungen über digitale Logücher

Um den Stand der Technik von elektronischen Logbüchern zu ergründen, wurde eine Marktrecherche durchgeführt. Die Erkenntnisse dieser Marktrecherche werden in diesem Kapitel wiedergegeben.

3.1.1. Logbook App

Logbook App ist eine Anwendung für Mobilgeräte mit iOS/iPadOS Betriebssystem. Die Anwendung hat einen Preis von \$15.99.[[Kric](#)] Die wichtigsten Eigenschaften sind folgende:[[Kria](#)]

- **Automatische Einträge:** Logbook App erstellt in einem gesetzten Zeitintervall Logbucheinträge. Es nutzt dazu die GPS Daten des Telefons und sammelt Daten aus den Internet. Folgende Daten werden angegeben:
 - Datum und Uhrzeit

- Seegang
 - Wetter
 - Nautische Standortbestimmung
 - Kurs über Grund
 - Standortgenauigkeit
 - Distanz zum letzten Eintrag
 - Geschwindigkeit über Grund
 - Motorstunden
- **Geocoding:** Damit kann es die GPS Position in eine leichter zu interpretierende Ortsbestimmung umwandeln, wie z.B.: 0,2 sm NW von Leuchtfeuer Celice, Küste Dalmatien, Adriatisches Meer.
 - **Manuelle Einträge für Manöver:** Für Manöver, wie Ablegen, Anlegen, Segel setzen usw., gibt es die Funktionalität manuelle Logbucheinträge zu erstellen. Für die manuellen und automatischen Einträge werden außerdem noch Markierungen auf einer Landkarte gesetzt.
 - **Statistiken:** Mit den gesammelten Daten werden Statistiken erstellt.
 - **Foto und Notizen:** Das Hinzufügen von einem Foto und Notizen in einen Logbucheintrag ist ebenfalls möglich.

Eine Schnittstelle für NMEA Daten ist zwar verfügbar, benötigt jedoch eine manuelle Konfiguration.[\[Krib\]](#) Außerdem, wegen Restriktionen von iOS/iPadOS Geräten, muss das Gerät konstant aktiv mit der Anwendung im Vordergrund sein, damit der TCP oder UDP Stream zu der NMEA Quelle nicht unterbrochen wird.

Die Logbook App bietet gute Visualisierungen für Logbucheinträge, die auf Smartphone und Tablet gut aussehen. Wobei auf größeren Tablets nicht der ganze Platz verwendet wird (Abb.3.1). Die Zusatzfunktionen, wie Geocoding, Kartenmarkierungen, Statistiken, Fotos und Notizen sind hilfreiche Erweiterungen und werten die Anwendung im allgemeinen auf.

Schwachstellen der App sind jedoch, die komplizierte Konfiguration für NMEA, wodurch das Mobilgerät nicht mehr für andere Aufgaben nutzbar ist, und dass in meisten Fällen die Daten vom Mobilgerät selbst, oder aus dem Internet kommen. Sollte das Mobilgerät mal keine ausreichenden Internetverbindung haben, ist es auch nicht möglich die Daten nachträglich anzufordern. Außerdem ist es nur für iOS/iPadOS erhältlich.

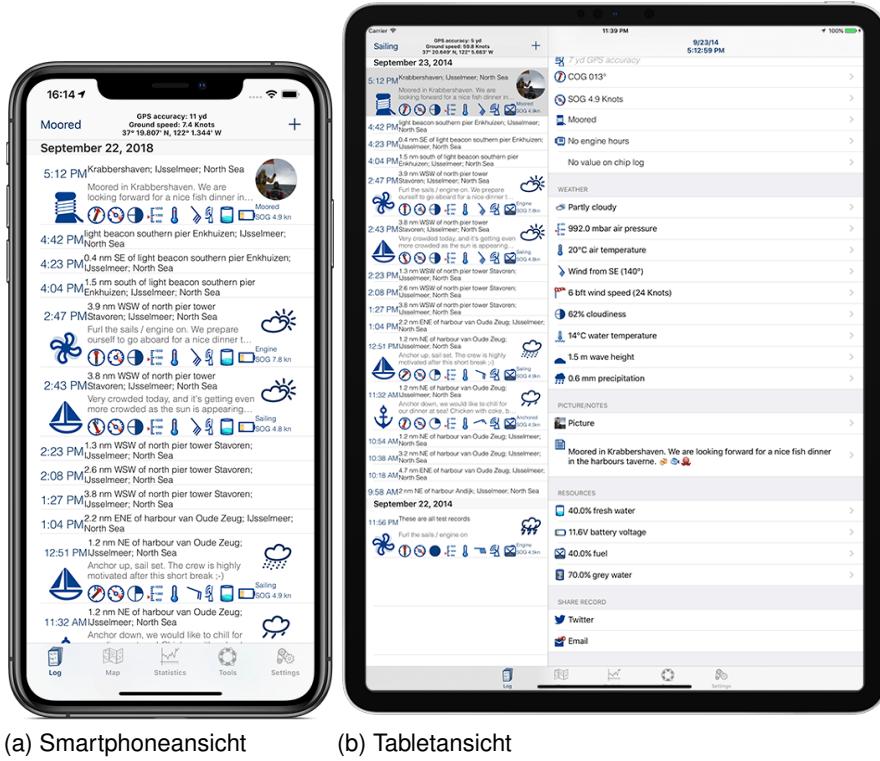


Abbildung 3.1: Logbook App
Quelle: <https://logbook-app.com/>

3.1.2. 2K Yachting

Die Logbook Suite von 2K Yachting gibt es für Windows, Mac oder iPad. Die Basisversion mit manueller Datenerfassung für einen Gerätetyp kostet 69€, mit zusätzlichen 30€ für die automatische Erfassung von NMEA Daten.[[2kyb](#)] Auf PC und Mac werden jedoch noch keine reinen NMEA 2000 Datenquellen unterstützt.

Folgende Daten können dann automatisch ins Logbuch übertragen werden:[[2kya](#)]

- Bordzeit und Zeit in UTC
- Position
- Kurs über Grund (COG)
- Geschwindigkeit über Grund (SOG)
- Windrichtung und Windgeschwindigkeit
- LOG oder TRIP
- Luftdruck, Temperatur und Luftfeuchte

- auf der Karte Wetter: Wassertemperatur

Die Logbook Suite enthält neben dem Logbuch noch weitere Funktionalitäten, wie eine Biographie, einen Planer und ein Wartungsbuch.

Die Logbook Suite bietet viele Zusatzfunktionalitäten in einer Anwendung. Es hat jedoch einige Schwachstellen. Das Design der Anwendung ist veraltet und nicht mehr auf heutigen Standards (Abb. 3.2), wodurch es viel Platz auf dem Bildschirm benötigt. Es ist auch nicht für Smartphones, oder Linux PCs erhältlich. Dass NMEA 2000 nicht unterstützt wird, ist ebenfalls ein großer Nachteil.



Abbildung 3.2: 2k-sailing Logbook Suite

Quelle: <https://www.2k-yachting.de/de/logbook/logbook.html>

3.2. Technischer Zustand von Webtechnologien

3.2.1. Progressive Web Apps

PWAs sind noch sehr neu. Daher ist das Ökosystem rund um PWAs noch nicht ganz ausgereift. Sie sind sehr abhängig von den Browsern in denen sie laufen, und die Fähigkeiten und Berechtigungen die ihnen von diesen erteilt werden. Zum Beispiel muss die Möglichkeit, die Apps zu installieren von Browser gegeben sein. Dies ist von den meisten modernen Browsern gegeben, jedoch nicht von allen. Der Desktop Browser von Firefox ist dazu zum Beispiel nicht in der Lage. Auf

iOS/iPadOS Geräten ist die Installation nur über Safari möglich. Dies bedeutet, dass dieselbe PWA in verschiedenen Browsern, unterschiedliche Eigenschaften haben kann. Außerdem sind Webanwendungen im Allgemeinen in der Navigation und Inputmethoden eingeschränkt. Ein Beispiel dafür ist der Rechtsklick, der im Browser das Kontextmenü öffnet. Dem Rechtsklick eine neue Funktionalität zu geben, ist schwer zu implementieren, und von manchen Browsern sogar verboten. Außerdem empfiehlt sich dies in der Regel nicht, da viele Nutzer bei einem Rechtsklick das Kontextmenü erwarten.

Im Allgemeinen bedeutet das, dass PWAs sehr nah an den technischen Möglichkeiten von nativen Apps sind, vorausgesetzt sie laufen in einem modernen Browser, der die nötigen APIs bereitstellt.

3.2.2. Frameworks

Die bekanntesten Frameworks sind React, Angular und Vue. Das Polymer Project bietet mit Web Components einen eigenen Ansatz für die Webentwicklung. React und Polymer sind genaugenommen keine Frameworks, sondern eine Library/Sammlung von Libraries. Beide werden jedoch oft auch als Framework gezählt, da sie zu großem Teil den gleichen Zweck wie die anderen Frameworks erfüllen. Anhand von Tabelle 3.1 sieht man, dass React mit über 495 Millionen npm Downloads das am weitesten verbreitete Framework ist. Am zweitweitesten verbreitet ist Angular mit über 162 Millionen Downloads. Danach folgen Vue mit über 80 Millionen und Polymer mit über 3 Millionen Downloads.

Package	Downloads
react	495,746,957
@angular/core	162,071,009
vue	83,221,847
@polymer/polymer	3,302,153

Tabelle 3.1: Summe der Downloads vom 12.12.2014-19.12.2019
Quelle: [\[npm\]](#)

Abb. 3.3 zeigt den Wachstum anhand von wöchentlichen Downloads im Zeitraum

vom 18.August 2019 bis zum 09.Februar 2020. Diese Grafik zeigt, dass React nicht nur das am weitesten verbreitete Framework ist, sondern auch das am schnellsten wachsende. Angular und Vue haben einen ähnlich schnellen Wachstum. Polymer wächst am langsamsten.

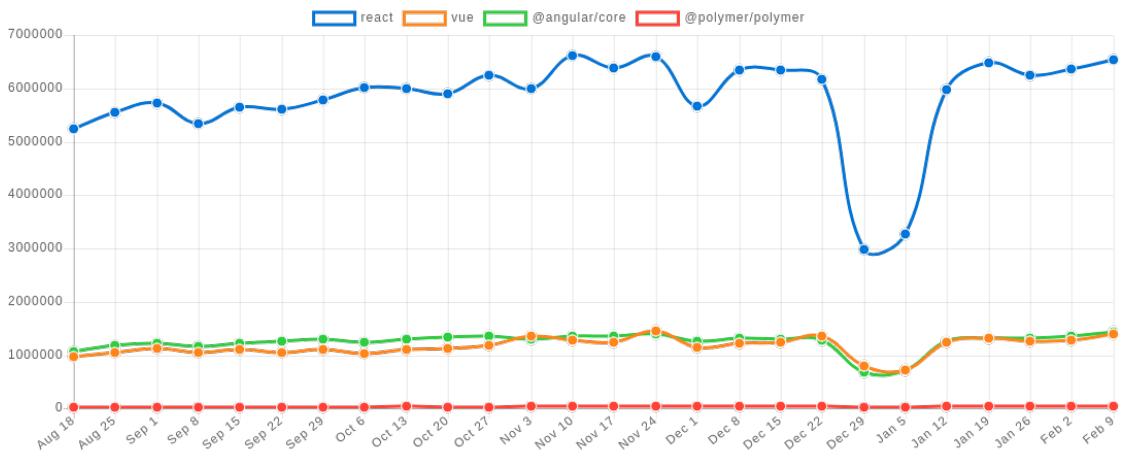


Abbildung 3.3: Wöchentliche npm Downloads vom 18.08.2019-09.02.2020

Quelle: <https://www.npmtrends.com/react-vs-vue-vs-@angular/core-vs-@polymer/polymer>

4

Implementierung

Basierend auf den Stärken und Schwächen von Vergleichsprodukten und den Erkenntnissen über die technologischen Möglichkeiten von PWAs, wurde ein Frontend für das Logbuch implementiert. In diesem Kapitel werden die einzelnen Schritte aufgezeigt und erläutert.

4.1. Prototypstellung

4.1.1. Scribbles

Zu Beginn der Implementierungsphase wurden Überlegungen über die Struktur und das Design der Anwendung getätigt. Hierbei wurde der Fokus auf die Logseite gelegt. Es sollte jedoch mit weiteren Seiten, wie einer Seekarte oder Statistik-Seite, zur Erweiterung bedacht werden. Um unterschiedliche Struktur- und Navigationsansätze zu visualisieren wurden Scribbles erstellt (Abb. 4.1).

Mithilfe von Scribbles wurden mehrere Visualisierungsansätze ausprobiert, und eine engere Auswahl darüber getroffen. Dabei wurde der Mobile First Denkansatz verwendet, und das Design erst für Mobilgeräte erstellt.

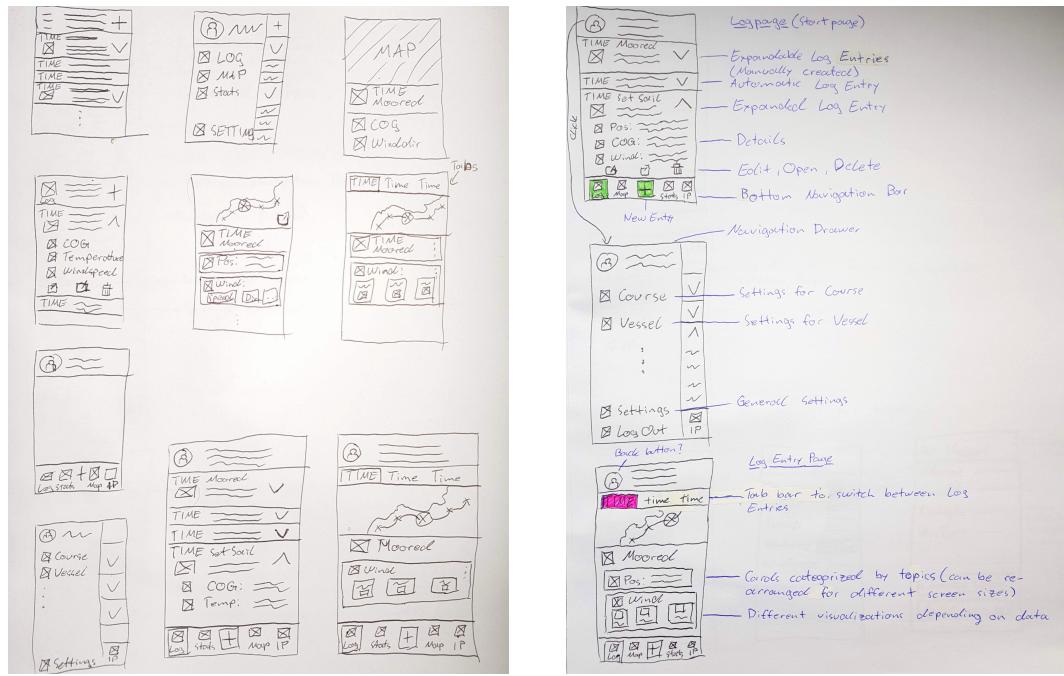


Abbildung 4.1: Scribbles von der Logpage

Quelle: Aufgenommen am 18.02.2020

Als Grundarchitektur wurde folgendes festgelegt:

Die Logübersicht Seite soll zur Übersicht aller Logeinträge dienen, und aus einer Liste mit Kurzinformationen der Einträge bestehen. Durch einen Klick auf eines der Listeneinträge kommt man zu der Logdetails Seite des ausgewählten Eintrags. Dort werden alle Informationen des Logeintrags in passende Kategorien aufgeteilt. Die Kategorisierung wird von der Struktur des Signal K Formats bestimmt.

4.1.2. Wireframes

Folgend zu den Scribbles wurden Wireframes erstellt. Hierbei wurden verschiedene Navigationsmethoden ausprobiert. Zum einen ein Seitenmenü das über einen Knopf oben links ein- und ausgeblendet werden kann (Abb. 4.2b Navigation Drawer). Und zum anderen eine Leiste mit Knöpfen am unteren Rand des Bildschirms (Abb. 4.2c Navigation Bar). Bei Desktops kann diese Leiste dann an den linken Bildschirmrand verlegt werden, da diese Positionierung auf Desktops weniger Platz benötigt (Abb. 4.2e).

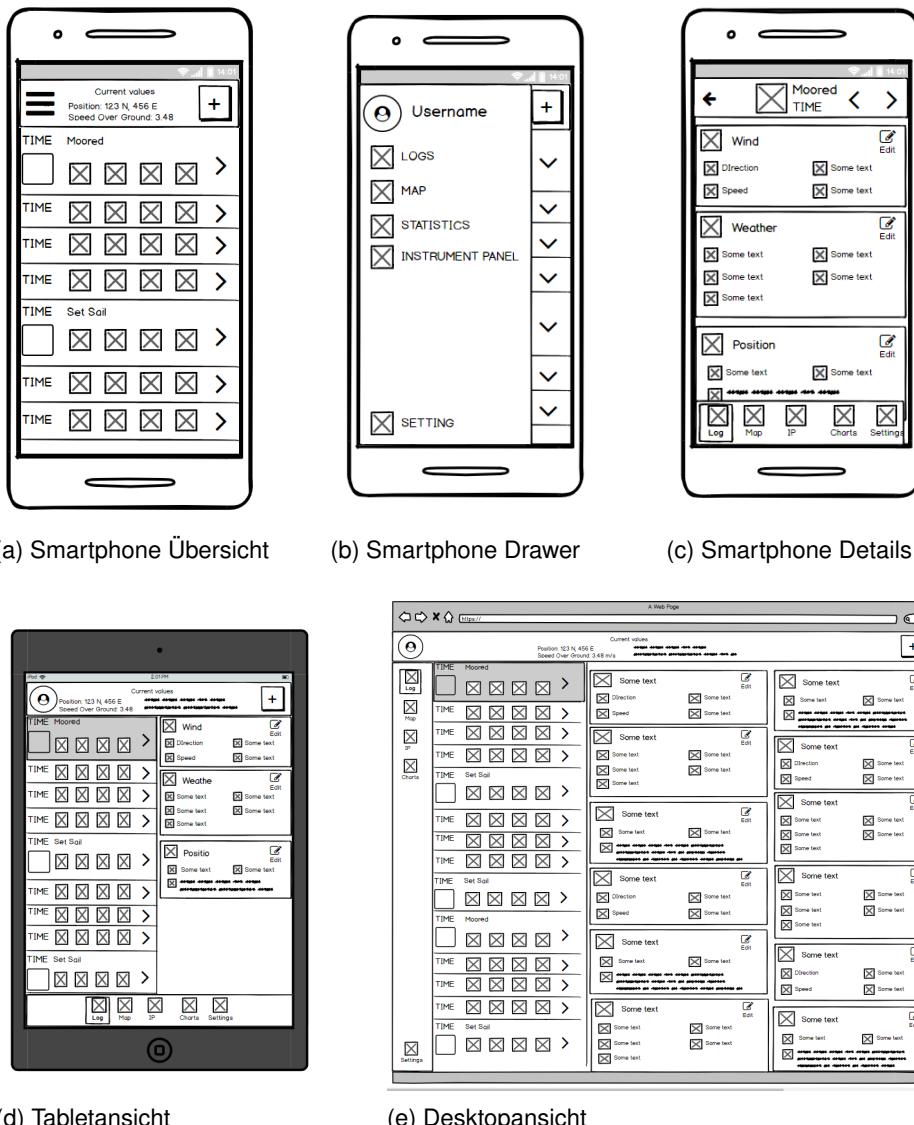


Abbildung 4.2: Wireframes von der Logpage
Quelle: Bildschirmabbildung von Balsamiq

Durch Wireframes wurde die Strukturidee der Anwendung detailreicher definiert.
 Folgendes wurde für die geplante Anwendung definiert:

Die Listenelemente der Übersichtsseite sollen die Uhrzeit, das Manöver und die wichtigsten Eigenschaften mithilfe von eigens dafür erstellte Icons anzeigen. Die Karten der Detailseite sollen einen Header haben, in dem das passende Icon, der Kategorienname und ein Knopf zum manuellen ändern der Daten, stehen. Zusätzlich sollen die Karten zwei Spalten für die einzelnen Werte haben. Die Zeilen sollen sich an die Menge

der Daten für die jeweilige Kategorie anpassen.

4.1.3. Prototyp

Um zu sehen wie das finale Produkt aussehen könnte, wurde ein Prototyp erstellt. Dafür wurden Icons im SVG Format gezeichnet. Auf der Übersichtsseite sollen die Icons die Messwerte des Logeintrags anzeigen. Zum Beispiel soll das Icon für die Position Längen- und Breitengrad anzeigen (siehe Abb. 4.3).

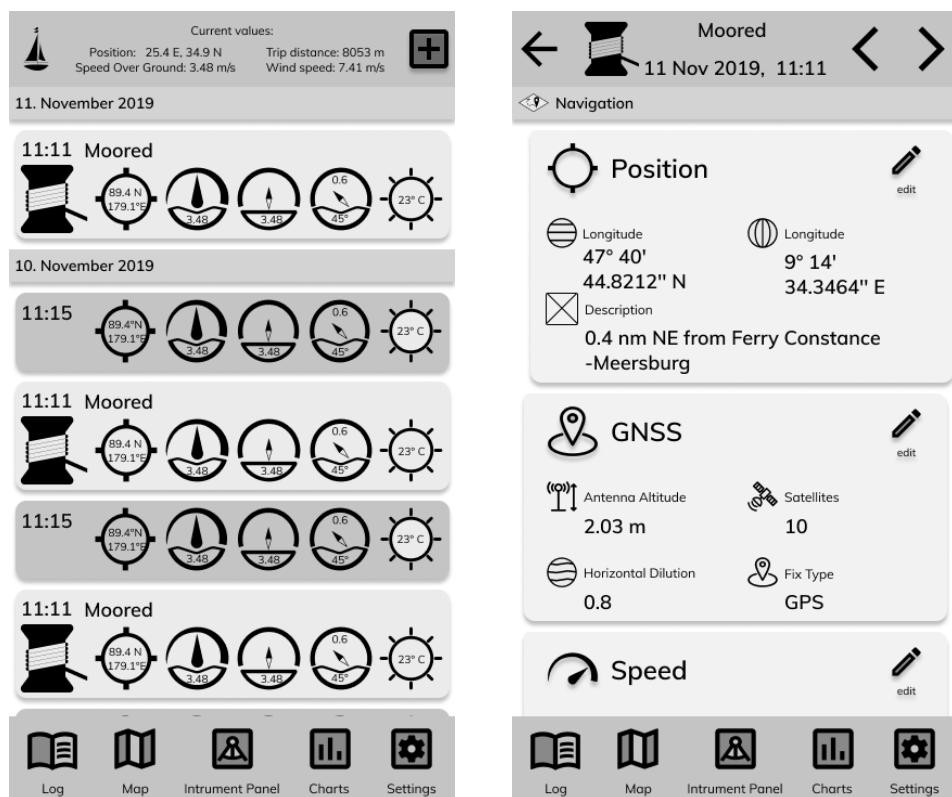


Abbildung 4.3: Prototyp der Logpage
Quelle: Bildschirmabbildung von Figma

Die Erstellung des Prototyps brachte eine noch detailreichere Definition für die geplante Anwendung:

Die Anwendung soll eine Navigation Bar am unteren Rand verwenden. Auf der Übersichtsseite steht im jeweiligen Icon der repräsentierte Wert. Die Einträge in der Detailseite zeigen das jeweilige Icon, den Namen des Eintrags und den Wert selbst.

4.2. Entwicklung als Progressive Web App

4.2.1. Auswahl der Kerntechnologien

Zur Implementierung der Webanwendung wurde das Framework React verwendet. React wurde für die Implementierung ausgewählt, da es ebenfalls von dem Signal K Frontend Instrumentpanel verwendet wurde, und man so eine einheitliche Technologieauswahl hat. Als Package Manager wurde npm verwendet.

Eine Übersicht über alle verwendeten Technologien ist in Tabelle 4.1 zu sehen.

Library	Beschreibung
@material-ui/core	Material Theme UI Library
@signalk/client	Clientseitige Verbindung mit Signal K Server
convert-units	Konvertierung von Werten zwischen verschiedenen Einheiten
pwa-helpers	Funktionen die helfen eine PWA zu bauen
react	React Hauptlibrary
react-dom	DOM Renderer für React
react-redux	React Bindings für Redux
react-router-dom	React Routing Library
react-scripts	Skripts für Create React App
redux	State Management Library
redux-thunk	Thunk Middleware für Redux
styled-components	CSS-in-JS Library für React Komponenten

Tabelle 4.1: Genutzte Libraries

4.2.2. Initialisierung

Zur Initialisierung wurde Create React App(CRA) verwendet. CRA generiert die Projekt Struktur und installiert die nötigen Abhängigkeiten (dependencies).[fac]

4.2.3. Navigation und Routing

Für das Client-seitige Routing wurde react-router-dom verwendet. Zu Beginn wurden folgende Komponenten importiert:

```
1 import {
2     BrowserRouter, Switch, Route, Link
3 } from "react-router-dom";
```

Listing 4.1: Router Imports(app.js)

Mit diesen Komponenten ist es möglich in der render Funktion React Komponenten an Routen zu binden, und Navigationselemente zu erstellen, mit denen man die Route ändern kann.[JF]

Die BrowserRouter Komponente ist dabei an der Wurzel der Element Hierarchie.

```
1 <BrowserRouter>
2     <App/>
3 </BrowserRouter>
```

Listing 4.2: Browser Router(app.js)

Um Routen zu definieren, wird eine Switch Komponente verwendet. In dieser ist für jede Route eine Route Komponente erstellt worden. In diesen kann die React Komponente angegeben werden, die bei dieser Route gerendert werden soll.

```
1 <Switch>
2     <Route path="/logs/details" component={LogDetails}></Route>
3     <Route path="/logs" component={LogsOverview}></Route>
4     <Route path="/settings" component={Settings}></Route>
5 </Switch>
```

Listing 4.3: Switch Komponente(app.js)

Um ein Navigationselement zu erstellen das die Route ändert, wird die Link Komponente verwendet.

```
1 <Link to="/logs">Logs</Link>
```

Listing 4.4: Link Komponente(app.js)

Das Design der Navigationselemente wurde, anders als im Prototyp, als Navigation Drawer implementiert. Dies hat den Grund, dass die Navigation Bar limitierten Platz bietet. Auf Mobilgeräten passen maximal fünf Elemente in die Navigation Bar. Um eine leichte Implementierung von neuen Seiten in der Anwendung zu gewährleisten wurden die Link Elemente in einem Navigation Drawer untergebracht (Abb. 4.4).

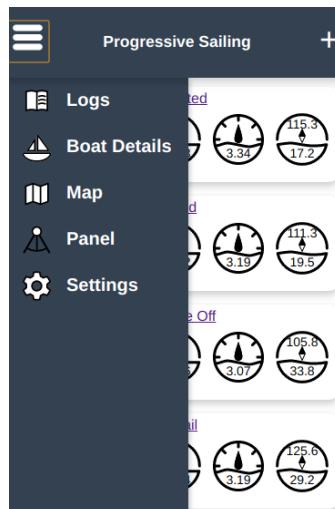


Abbildung 4.4: Navigation Drawer

4.2.4. Icon Implementierung

Die Icons wurden als Vektor Grafiken erstellt. Für die Icons, die auf der Übersichtsseite zu sehen sind, wurde ein bzw. zwei Textfelder hinzugefügt(Lst. 4.5).

```
1 <svg width="100%" height="100%" viewBox="0 0 600 600"
2     ... /* Teil des Codes wurde ausgelassen */
3     <text
4         x="164.206px"
5         y="278.296px"
6         style="font-family:'ArialMT','Arial',
7         sans-serif;font-size:139.605px">
8             0.00
9         </text>
10    </svg>
```

Listing 4.5: Textfeld in SVG

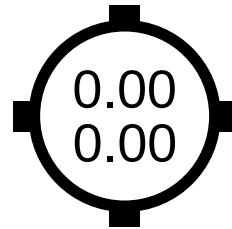


Abbildung 4.5: Position SVG Icon

Um auf Textfelder, Größe und weitere Stilelemente der Icons mithilfe von React zugreifen zu können, wurden die SVG Dateien in React Komponenten eingebunden (Lst. 4.6).

```
1 import React from "react";
2
3 function Position(props) {
4     return (
5         <svg
6             width={props.width || "63px"}
7             height={props.height || "63px"}
8         >
9         ...
10        <text>
11            {props.longitude ? props.longitude.value : null}
12        </text>
13    </svg>
14 );
15 }
```

Listing 4.6: Position Komponente(position.js)

Dieser Komponente werden Eigenschaften (props) übergeben. In diesen Eigenschaften werden die nötigen Daten übergeben. Sind diese nicht verfügbar werden auf vordefinierte Werte zurückgegriffen, oder ein null übergeben.

```
1 <Position longitude={longitudeValue} latitude={latitudeValue}
2             width={"60px"} height={"60px"}>
3     />
```

Listing 4.7: Nutzung einer Icon Komponente

4.2.5. CSS und Responsive Design

Für die CSS Style Regeln der Anwendung wurde die Library styled-components verwendet. Damit ist es möglich Komponenten mit CSS Styles zu erstellen und diese dann in der render Funktion zu benutzen.

```
1 const SideNav = styled.div`  
2     height: 100%;  
3     width: ${props => props.navToggled ? "200px" : "0"};  
4     ...  
5 `;
```

Listing 4.8: Styled Component für den Navigation Drawer(app.js)

In Listing 4.8 ist ein Teil der Definition des Navigation Drawers zu sehen. Darin ist zu sehen, wie props dazu genutzt werden können, den Style einer Komponente zu verändern. Dabei wird der Wert der in der Implementierung (Lst. 4.9) übergeben wird, an die Komponente gebunden. Sollte sich der übergebene Wert ändern, ändert sich auch der Style der Komponente. In diesem Fall wird diese Funktionalität genutzt, um den Navigation Drawer bei Knopfdruck ein- und auszublenden.

```
1 class App extends React.Component {  
2     constructor(props) {  
3         super(props);  
4         this.state = {  
5             navToggled: false,  
6         }  
7     }  
8     ...  
9     render() {  
10         return (  
11             <Button onClick={this.toggleNav}> // Ändert navToggled  
12                 <Icons.Hamburger color="#FFFFFF"/>  
13             </Button>  
14             <SideNav navToggled={this.state.navToggled}>  
15                 ...  
16             </SideNav>  
17         ){}  
18     }  
19 }
```

Listing 4.9: Implementierung einer styled-component(app.js)

Das allgemeine Layout wurde mit CSS Grid definiert. Um die Anwendung Responsive zu machen, wurden zwei Methoden angewandt. Als erstes wurde die Übersichtseite und die Detailseite aufgeteilt. Auf Mobilgeräten ist zu Beginn die Übersicht zu sehen (Abb. 4.6a), und bei der Auswahl eines Logeintrags, wechselt die Ansicht zur Detailseite (Abb. 4.6b). Bei Tablet und Desktop sind beide Ansichten gleichzeitig sichtbar (Abb. 4.6c & d).

(a) Mobil Übersicht

(b) Mobil Details

(c) Tablet Ansicht

(d) Desktop Ansicht

Abbildung 4.6: Responsive Design Progressive Sailing

Diese Eigenschaft wurde wie in Listing 4.10 implementiert:

```
1 render() {
2     // Bildschirmbreite wird abgefragt
3     const {width} = window.innerWidth;
4
5     const isMobile = width <= 500;
6     /* Bei schmalen Bildschirmen wird die Mobilansicht
7      angezeigt sonst die Desktopansicht */
8     if (isMobile) {
9         return <MobileView ... />;
10    } else {
11        return <DesktopView ... />
12    }
13 }
```

Listing 4.10: Implementierung der Mobil- und Desktopansicht(LogsOverview.js)

Die zweite Methode ist in der Tablet- und Desktopansicht der Detailansicht zu sehen. Darin werden die einzelnen Kategoriekarten, mithilfe von Media Queries, je nach Bildschirmbreite auf mehrere Spalten verteilt (Lst. 4.11).

```
1 const DesktopColumns = styled.div`  
2     display: grid;  
3     grid-template-columns: 1fr; // Eine Spalte  
4  
5     @media (min-width: 1020px) {  
6         grid-template-columns: 1fr 1fr; // Zwei Spalten  
7     }  
8  
9     @media (min-width: 1400px) {  
10        grid-template-columns: 1fr 1fr 1fr ; // Drei Spalten  
11    }  
12 `;
```

Listing 4.11: Anpassung der Spaltenzahl(LogDetails.js)

4.2.6. Service Worker und Caching

Die Initialisierung mit CRA übernimmt viel Arbeit um Service Worker in die Webanwendung einzubauen. Bei der Initialisierung wurde bereits eine serviceWorker.js

Datei erstellt, die in der Lage ist alle statischen, von React erstellten Dateien zwischenzuspeichern (cachen). Dieser ist Anfangs noch deaktiviert. Zur Aktivierung wird folgender Befehl verwendet:

```
1 serviceWorker.register();
```

Listing 4.12: Aktivierung des Service Worker(index.js)

Damit ist es möglich die Anwendung auch ohne Verbindung zum Server, der das Frontend serviert, zu nutzen. Um jedoch neue Logeinträge zu erstellen, benötigt die Anwendung immer noch eine Verbindung zu einem Signal K Server, der die nötigen Daten sendet. Zu beachten ist außerdem, dass der Service Worker nur die statischen Daten speichert. In diesem Zustand der Anwendung werden dynamische Daten, wie Logeinträge und Einstellung noch nicht gespeichert, und werden bei einem Schließen der Anwendung gelöscht. Für die Darstellung der Anwendung, wenn sie installiert ist, wurde das manifest.json mit passendem Namen, Icon und Farbe angepasst.

4.2.7. State Management

Das State Management wurde mit Redux implementiert. Dieses speichert die Daten für die Logeinträge, die aktuelle Route, den aktuell ausgewählten Logeintrag, die Einstellungen, die Manöver und die Bootsdetails in einem Objekt, dem Zustand (State). Dieser wird vom Store gehalten und zur Verfügung gestellt(Lst. 4.13).

```
1 import { createStore } from 'redux';
2 import rootReducer from './reducers';
3
4 export default createStore(rootReducer);
```

Listing 4.13: Redux Store(store.js)

Um auf den Store zugreifen zu können, muss dieser der Anwendung erst zur Verfügung gestellt werden (Lst. 4.14). In React wird dafür die App in eine Provider Komponente eingebunden.

```
1 import React from 'react';
2 import {Provider} from 'react-redux';
3 import {store} from './redux/store';
4 ...
5 render(
6     <Provider store={store}>
7         <App />
8     </Provider>,
```

```
9     document.getElementById('root'));
```

Listing 4.14: Redux Store Einbindung(index.js)

Um die Daten des States zu lesen wurden **selectors** verwendet (Lst 4.15).

```
1  export const getSelectedLog = store => store.logs.selectedLog;
```

Listing 4.15: Selector(selectors.js)

Eine React Komponente wird mit der **connect()** Funktion mit dem Store verbunden und kann dann mit den Selector Funktionen auf den State zugreifen (Lst. 4.16).

```
1  import { connect } from 'react-redux';
2  import { getLogs, getSelectedLog, getSettings
3 } from '../../redux/selectors'; // Selector Funktionen
4  import { selectLog, selectPath } from '../../redux/actions';
5  ...
6  /* State von Redux wird zu State von Komponente */
7  const mapStateToProps = state => {
8      return {
9          logs: getLogs(state),
10         selectedLog: getSelectedLog(state),
11         settings: getSettings(state)
12     }
13 };
14 ...
15 export default connect(mapStateToProps,
16     { selectLog, selectPath })(LogsOverview);
```

Listing 4.16: Verbindung zum Store(LogsOverview.js)

Um den State zu ändern, werden **actions** benutzt. Actions sind Datensammlungen, die von der Anwendung zum Store gesendet werden.[Aaa]

```
1  export const addLog = content =>({
2      type: ADD_LOG,
3      payload: {
4          id: nextLogId++,
5          content
6      }
7  });
```

Listing 4.17: Action für Hinzufügen von Logeintrag(actions.js)

Wird eine action versendet, wird sie von einem Reducer verarbeitet. Diese spezifizieren, wie sich der State ändert, wenn eine action ankommt (Lst. 4.18).

```

1  export default function(state = initialState, action) {
2      switch (action.type) {
3          case ADD_LOG: {
4              const { id, content } = action.payload;
5              return {
6                  ...state,
7                  allIds: [...state.allIds, id],
8                  byIds: {
9                      ...state.byId,
10                     [id]: {content}
11                 }
12             ...
13         }
14     }
15 }
```

Listing 4.18: Reducer für Hinzufügen von Logeintrag(reducers/logs.js)

Nun ist es möglich den State im Local Storage des Browsers zu speichern. Damit werden die Daten beim Schließen der Anwendung nicht mehr gelöscht und bleiben für die nächste Nutzung erhalten.

Um dies zu implementieren, wurden zwei neue Funktionen eingebaut (Lst. 4.19).

```

1 // Funktion zum Speichern im Storage
2 export const saveState = (state) => {
3     let json = localStorage.getItem('state') || '{}';
4     let stringifiedNewState = JSON.stringify(state);
5     if (stringifiedNewState !== json
6         && stringifiedNewState !== '{}') {
7         localStorage.setItem('state', stringifiedNewState);
8     }
9 };
10 // Funktion zum Laden des Storage
11 export const loadState = () => {
12     let json = localStorage.getItem('state') || '{}';
13     let state = JSON.parse(json);
14
15     if (state) {
16         return state;
17     } else {
```

```
18     return undefined;
19 }
20 };
```

Listing 4.19: Funktionen zum Speichern und Laden des States(localStorage.js)

Um diese Funktionen zu den richtigen Zeitpunkten anzuwenden wurde der Store entsprechend erweitert (Lst. 4.20).

```
1 import { loadState, saveState } from './localStorage.js';
2 export const store = createStore(
3     rootReducer,
4     loadState(),
5     devCompose(
6         lazyReducerEnhancer(combineReducers),
7         applyMiddleware(thunk)
8     )
9 );
10
11 store.subscribe(() => {
12     saveState(store.getState())
13});
```

Listing 4.20: Store Erweiterung(store.js)

4.2.8. Signal K Implementierung

Die Daten für einen Logeintrag werden vom Signal K Server über einen GET Request erhalten. Über den Pfad (Lst. 4.21)

```
1 <Server-Adresse>/signalk/v1/api/vessels/self
```

Listing 4.21: Request Pfad

ist es möglich die aktuellen Daten des Bootes in Signal K JSON-Format zu erhalten. Bei Erhalten einer Antwort, werden die Daten mithilfe von Redux gespeichert, es wird ein neuer Eintrag in der Logübersicht angezeigt, und bei Auswahl werden die Daten im Logdetailbereich visualisiert.

5

Diskussion

In diesem Kapitel werden die Ergebnisse der Studie bewertet, und Empfehlungen für weiterführende Arbeit ausgesprochen.

5.1. Bewertung genutzter Technologien und Methoden

Die Konzeptionsphase mit Scribbles anzufangen hat viele Vorteile aufgewiesen. Es geht schnell, verschiedene Interfaces auszuprobieren und einen ersten Eindruck dafür bekommen. Man braucht auch kein Programm oder besondere Programmierkenntnisse, wodurch jeder Scribbles erstellen kann. Wireframes sind ein guter Schritt nach Scribbles, um sich ein besseres Bild der geplanten Anwendung zu machen. Durch die Nutzung vordefinierter Elemente, die per Drag-and-Drop eingefügt werden können, bleibt die Erstellungszeit und Lernkurve moderat. Prototypen sind jedoch eine ganz andere Bandbreite. Die Programme die man zur Prototyperstellung verwendet, haben eine sehr steile Lernkurve, und es benötigt viel Zeit und Wissen ein präsentablen Prototypen zu erstellen.

Die Nutzung von React hat sich, nach einer kurzen Einarbeitungszeit, als sehr hilfreich erwiesen. Es bietet eine ausführliche Dokumentation, wodurch der Einstieg erleichtert wird. Die Initialisierung mit CRA hat viele Aufgaben übernommen und die Produktion beschleunigt. Vor allem die Nutzung von React in Zusammenarbeit mit Redux erleichtert viele Aufgaben und verringert damit die Codegröße und Entwicklungszeit. Die drei Prinzipien (Kap. 2.4.3) von Redux beseitigen viele potenzielle Fehlerquellen

und sorgen für Korrektheit der Daten in der Anwendung. Doch Redux hat nicht nur Vorteile. Der Aufbau von Redux ist sehr eigen, was die Einarbeitungszeit etwas erschwert. Außerdem gibt es keine Datenkapselung. Das bedeutet, alle Komponenten können auf die Daten zugreifen, was zu Brüchen der Sicherheit führen kann.

Die Verwendung von styled-components hat zwar die Größe und Übersichtlichkeit der JavaScript Dateien stark beeinflusst, hat dafür aber die Nutzung von JavaScript Variablen im CSS ermöglicht, und dadurch auch dynamische Änderung des Styles vereinfacht.

Das Signal K Protokoll hat den Zugriff auf Bootsdaten vereinheitlicht und günstiger gemacht, und hat bereits viele Anwendungsbereiche gefunden. Das Datenformat ist jedoch inkonsistent aufgebaut, was die Visualisierung der Daten erschwert.

5.2. Interpretation von Ergebnissen

Die Anwendung hat viele der gesetzten Ziele erfüllt, jedoch nicht alle. Folgende Ziele wurden erfüllt:

- **Responsive Design:** Die Anwendung ist responsive, und funktioniert dadurch auf Smartphones, Tablets und Desktops.
- **Offlinefunktionalitäten:** Die Anwendung bietet gute Offlinefunktionalitäten. Es ist möglich die Logbucheinträge einzusehen, jedoch ist es nicht möglich neue zu erstellen.
- **Installierbarkeit:** Installierbar ist die Anwendung in einer Testumgebung auch, jedoch ist die Anforderung in einer Produktionsumgebung dafür, dass die Seite über HTTPS serviert wird.
- **Funktionalität bei 3G:** Bei einer 3G Verbindung dauert es 5,5 Sekunden bis die Seite interaktiv nutzbar ist. Diese Zeit ist unter den vorgesetzten 10 Sekunden, kann jedoch noch verbessert werden. Zum Beispiel durch Server-seitiges Rendern und asynchrones Laden. Getestet wurde dies mithilfe der Google Chrome Erweiterung Lighthouse, mit simulierter 3G Verbindung.[[Deva](#)]
- **Cross-Browserfunktionalitäten:** Die Anwendung wurde auf Google Chrome für Windows, Linux und Android, Mozilla Firefox für Windows, Linux und Android und Microsoft Edge für Windows getestet. Die Anwendung funktioniert auf allen getesteten Browsern größtenteils. Einzig die Installierbarkeit ist nicht bei allen Browsern möglich. Bei Firefox für Linux und Windows, und Microsoft Edge war

es nicht möglich die Anwendung zu installieren. Das Caching und dadurch das schnelle Laden bei wiederholtem Besuchen und Offlinenutzbarkeit im Browser funktioniert jedoch in allen Browsern.

- **Auffindbarkeit:** Jede Seite hat eine eigene URL.

Das Ziel das noch nicht erfüllt wurde, ist dass die Seite über HTTPS serviert wird. Dies liegt daran, dass die Seite noch nicht öffentlich verfügbar ist. An der Implementierung eines Servers der die Seite über HTTPS liefert, gibt es eines zu beachten. Die Kommunikation zum Signal K Server muss auch über HTTPS laufen, sonst blockieren die Browser diese Verbindung.

Der Inhalt von Logeinträgen hängt größtenteils von den Daten, die der Signal K Server liefert, und den Instrumenten auf dem Boot ab. Position, Kurs und Geschwindigkeit waren bei den getesteten Servern immer enthalten. Die Funktionalität, den Wetterbericht aus einer anderen Quelle zu erhalten, ist noch nicht in der Anwendung implementiert. Ein Feld für Notizen ist ebenfalls noch nicht vorhanden. Die Schiffsdaten werden von dem Server überliefert, jedoch müssen diese erst manuell über eine Konfigurationsdatei dem Server übergeben werden.

5.3. Empfehlungen für weiterführende Arbeit

Die wichtigste weiterführende Aufgabe ist die Einbindung von Serverschnittstellen. Beide, Node und Java, Server können mit diesem Frontend verwendet werden. Der Java Server bietet bereits eine implementierte Datenbank und Autorisierung. Für diese Funktionalitäten benötigt das Frontend jedoch eine Schnittstelle und ggf. eine Ansicht (Login-/Registrierungsseite). Eine Synchronisierung der Logbucheinträge die im Local Storage gespeichert sind mit denen in der Datenbank ist ebenfalls eine wichtige Eigenschaft.

Eine weitere nützliche Funktionalität wäre, Notizen und Fotos in Logbucheinträge hinzufügen zu können. Die Werte manuell zu ändern ist ebenfalls noch nützlich.

Der Signal K Server bietet auch die Möglichkeit, über Websockets, kontinuierlich die aktuellen Daten der Instrumente und Sensoren zu erhalten. Diese Funktionalität kann zum Beispiel in einer weiteren Ansicht, einem Instrument Panel, verwendet werden.

Die Einstellungsseite sollte ebenfalls noch erweitert werden, mit Feldern in denen man die Bootsdaten eingeben kann.

6

Fazit

Diese Studie beschäftigte sich mit der Frage, ob es möglich ist mithilfe von heutigen Webtechnologien, eine Anwendung zu schreiben, die sich minimal von einer nativen Anwendung unterscheidet, und dabei auf nahezu jedem Endgerät funktioniert.

Um dies zu beantworten wurde in Kapitel 2 das Vorbild der Anwendung, das Logbuch, untersucht und die technologischen Neuheiten in dieser Domäne, Signal K, beleuchtet. Anschließend wurde ein Einblick in die Prinzipien und Werkzeuge der User Experience gegeben. Daraufhin wurden die genutzten Webtechnologien erklärt.

In Kapitel 3 wurde durch eine Analyse von Vergleichsprodukten bestimmt, welche Eigenschaften ein elektronisches Logbuch benötigt. Und anschließend wurde der technische Zustand von Webtechnologien beleuchtet.

Kapitel 4 dokumentierte die einzelnen Schritte zur Implementierung der Anwendung. Angefangen wurde mit Scribbles, die erste Ansätze über den Aufbau des Frontends aufwiesen. Darauf aufbauend wurden Wireframes erstellt, mit denen mehrere Navigationsmethoden ausprobiert wurden. Anschließend wurde ein Prototyp entwickelt, der einen ersten Eindruck über das Design der Anwendung brachte.

Mit dem fertigen Prototypen, begann die Implementierung der Anwendung als Progressive Web Application. Erster Schritt dabei war die Initialisierung mit CRA, dann folgte die Implementierung des Routing und der Navigation. Die Icons wurden als interaktive Komponenten eingebaut, die Anwendung erhielt ein Responsive Design und anschließend Offlinefunktionalität und Installierbarkeit. Zu guter Letzt wurde das State Management mit Redux und die Datenbeschaffung von Signal K Server implementiert.

In Kapitel 5 wurde überprüft, ob die finale Anwendung die gesetzten Ziele erfüllt, und falls nicht, wie diese in Zukunft erfüllt werden können. Anschließend wurden Empfehlungen für weiterführende Arbeit an diesem Projekt ausgesprochen.

Die Ergebnisse dieser Studie zeigen, dass die Fortschritte im Bereich der Webtechnologien viele neue Möglichkeiten und Anwendungsfälle bieten. Die Standards und Technologien hinter Progressive Web Apps sorgen dafür, dass sich die allgemeine Qualität von Webanwendungen verbessert, und auch über das Web hinauswachsen. Durch die Zusammenarbeit vieler freiwilliger Helfer an Open Source Projekten, ist es möglich, viele Aufgaben nun leichter und günstiger zu lösen. Das Signal K Projekt ist das beste Beispiel dafür.

Literatur

- [ISO11] ISO. *Ergonomie der Mensch-System-Interaktion - Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme*. Beuth Verlag, Berlin, 2011.
- [She17] Dennis Sheppard. "Beginning Progressive Web App Development". In: apress, 2017. Kap. 1.
- [ISO18] ISO. *Ergonomie der Mensch-System-Interaktion - Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte*. Beuth Verlag, Berlin, 2018.
- [JM18a] Jens Jacobsen und Lorena Meyer. "Praxisbuch Usability und UX". In: Rheinwerk Computing, 2018. Kap. 1.
- [JM18b] Jens Jacobsen und Lorena Meyer. "Praxisbuch Usability und UX". In: Rheinwerk Computing, 2018. Kap. 13.
- [JM18c] Jens Jacobsen und Lorena Meyer. "Praxisbuch Usability und UX". In: Rheinwerk Computing, 2018. Kap. 14.
- [JM18d] Jens Jacobsen und Lorena Meyer. "Praxisbuch Usability und UX". In: Rheinwerk Computing, 2018. Kap. 16.
- [JM18e] Jens Jacobsen und Lorena Meyer. "Praxisbuch Usability und UX". In: Rheinwerk Computing, 2018. Kap. 5.
- [Woe18] Nikolas Woeckner. *Gibt es die Logbuchpflicht?* Nov. 2018. URL: <https://sail24.com/praxis/gibt-es-die-logbuchpflicht/> (besucht am 08.02.2020).
- [2kya] 2k-yachting. *Logbook im Detail*. URL: <https://www.2k-yachting.de/de/logbook/logbook.html> (besucht am 16.02.2020).
- [2kyb] 2k-yachting. *Logbook Suite: Lizzenzen, Upgrades und Preise*. URL: https://www.2k-yachting.de/de/logbook/logbook_price.html (besucht am 16.02.2020).
- [Aaa] Dan Abramov und Redux documentation authors. *Redux - Actions*. URL: <https://redux.js.org/basics/actions> (besucht am 21.02.2020).

- [Aab] Dan Abramov und Redux documentation authors. *Redux - Three principles*. URL: <https://redux.js.org/introduction/three-principles/> (besucht am 12.02.2020).
- [Che] Christopher Chedeau. *CSS in JS*. URL: <https://speakerdeck.com/vjeux/react-css-in-js> (besucht am 12.02.2020).
- [Coy] Chris Coyier. *A Guide to Flexbox*. URL: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/> (besucht am 12.02.2020).
- [Deva] Google Developers. *Lighthouse*. URL: <https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammpbjk?hl=de> (besucht am 24.02.2020).
- [Devb] Google Developers. *Progressive Web App Checklist*. URL: <https://developers.google.com/web/progressive-web-apps/checklist> (besucht am 05.02.2020).
- [fac] facebook. *Create React App*. URL: <https://github.com/facebook/create-react-app> (besucht am 19.02.2020).
- [Gau] Matt Gaunt. *Service Workers: an Introduction*. URL: <https://developers.google.com/web/fundamentals/primers/service-workers> (besucht am 14.02.2020).
- [GK] Matt Gaunt und Paul Kinlan. *The Web App Manifest*. URL: <https://developers.google.com/web/fundamentals/web-app-manifest> (besucht am 13.02.2020).
- [Inc] Facebook Inc. *What is CSS-in-JS?* URL: <https://reactjs.org/docs/faq-styling.html> (besucht am 12.02.2020).
- [Ink] Inkscape. *Das SVG-Dateiformat*. URL: <https://inkscape.org/de/entwickeln/das-svg-format/> (besucht am 14.02.2020).
- [JF] Michael Jackson und Ryan Florence. *React Router - Primary Components*. URL: <https://reacttraining.com/react-router/web/guides/primary-components> (besucht am 19.02.2020).
- [JMa] Jens Jacobsen und Lorena Meyer. *Teilaspekte der User Experience*. URL: <https://upload-magazin.de/wp-content/uploads/2018/05/abbildung-1-2-940x485.jpg> (besucht am 09.02.2020).
- [JMb] Jens Jacobsen und Lorena Meyer. *Usability und User Experience*. URL: <https://upload-magazin.de/wp-content/uploads/2018/05/abbildung-1-1-940x306.jpg> (besucht am 09.02.2020).

- [Jus] Bundesministerium für Justiz und Verbraucherschutz. *Schiffssicherheitsgesetz*. URL: <https://www.gesetze-im-internet.de/schsg/BJNR286010998.html> (besucht am 06.02.2020).
- [Kria] Florian Kriesche. *Logbook App Feature Overview*. URL: <https://logbook-app.com/features/> (besucht am 16.02.2020).
- [Krib] Florian Kriesche. *NMEA in der Logbuch App*. URL: <https://logbook-app.com/de/nmea/> (besucht am 16.02.2020).
- [Kric] Florian Kriesche. *Technical data to Logbook App*. URL: <https://logbook-app.com/get-logbook-app/> (besucht am 16.02.2020).
- [LeP] Pete LePage. *Ihre erste Progressive Web App*. URL: <https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/?hl=de> (besucht am 13.02.2020).
- [npm] npm-stat.com. *Download statistics for packages react, vue, @angular/core, @polymer/polymer*. URL: <https://npm-stat.com/charts.html?package=react&package=vue&package=%40angular%2Fcore&package=%40polymer%2Fpolymer&from=2014-12-12&to=2019-12-19> (besucht am 11.02.2020).
- [Proa] The Signal K Project. *Signal K*. URL: <http://signalk.org/index.html> (besucht am 07.02.2020).
- [Prob] The Signal K Project. *Signal K Github Repositories*. URL: <https://github.com/SignalK> (besucht am 08.02.2020).
- [Proc] The Signal K Project. *Signal K Info*. URL: <http://signalk.org/overview.html> (besucht am 08.02.2020).
- [Rak+] Filip Rakowski u. a. *The PWA Book*. URL: <https://divante.com/pwabook/chapter/01-introduction-to-pwa-technology.html#what-are-progressive-web-apps> (besucht am 05.02.2020).
- [w3sa] w3schools. *CSS Grid*. URL: https://www.w3schools.com/css/css_grid.asp (besucht am 12.02.2020).
- [w3sb] w3schools. *HTML Responsive Web Design*. URL: https://www.w3schools.com/html/html_responsive.asp (besucht am 12.02.2020).