Entwicklung einer progressiven Web-Anwendung am Beispiel eines Musiklernprogramms

Bachelorarbeit

Zur Erlangung des ersten akademischen Grades
Bachelor of Science B. Sc.

an der Fakultät Informatik
der Hochschule für Technik und Wirtschaft Dresden
im Bachelorstudiengang **042**

eingereicht von:
Widjaja, Delvin geboren am 29.08.1996 in Jakarta
eingereicht am: 3.3.2020

1. Gutachter: Prof. Dr. Jörg Vogt
2. Gutachter: Prof. Dr. Robert Baumgartl

Abstract

Mobiltelefone sind heute aus unserem Alltag kaum wegzudenken und die rasche Entwicklung der Technologie führt zu einem Wandel, bei dem das Mobiltelefon nicht nur als Kommunikationsgerät benutzt wird, sondern auch als Kameraparat um Fotos zu schießen, als Zahlungsmittel um Geschäfte abzuwickeln bis hin zu reiner Unterhaltung. All diese Funktionen wurden in einem Gerät vereinigt, daher ist die Entwicklung mobiler Anwendungen für viele Unternehmen zu einer Top-Priorität geworden.

Da mobile Anwendungen, oder sogenannte native Anwendungen, für eine bestimmte Plattform gebaut und optimiert werden, bieten sie die beste Leistung und das beste User-Experience. Native Anwendungen haben auch direkten Zugriff auf die Hardware der Geräte wie GPS oder Kamera, so dass Entwickler diese Funktionen in die Anwendung integrieren können. Trotz der Vorteile der nativen Anwendungen haben sie auch einige Nachteile, wie z. B. die Anforderung verschiedener Programmiersprachen, höhere Entwicklungskosten und eine schwierigere Optimierung für die Suchmaschinenoptimierung (SEO).

Als Lösung existieren progressive Webanwendungen (PWA). PWAs bieten alle Funktionen von nativen Anwendungen und decken deren Nachteile in jedem Bereich ab. Sie können plattformübergreifend arbeiten und nur in einer Sprache komplett geschrieben werden, wodurch sich die Entwicklungskosten reduzieren lassen können.

Das Ziel dieser Bachelorarbeit ist die Entwicklung einer progressiven Web-Anwendung am Beispiel eines Musiklernprogramms in JavaScript, das mit jedem Gerät und seiner unterschiedlichen Bildschirmdarstellung kompatibel ist.

Inhaltsverzeichnis

1. Theorie

1.1. Einführung

Vor einigen Jahrzehnten war das Internet nur eine vage Idee. Eine Idee, wo Informationen ortsunabhängig abgerufen werden können. Eine Idee, bei der Informationen sichtbar und für die Öffentlichkeit zugänglich ist. Durch die rasante Entwicklung des Internets haben Webseiten diesen Zweck erfolgreich erfüllt.

Seit die erste Webseite 1991 von Sir Tim Berners-Lee veröffentlicht wurde, ist die Anzahl der weltweit vorhandenen Webseiten im Dezember 2019 auf mehr als eine Milliarde gestiegen[1], angefangen von statischen Webseiten (oder dem so genannten „Read-Only-Web“ von Experten) bis hin zu dynamischen Webseiten, auf denen 1995 erstmals JavaScript vorgestellt wurde. Mit dem zunehmenden Zugang zum Internet haben mobile Geräte und Smartphones die Art und Weise verändert, wie Menschen das Internet nutzen. Die Nutzung des Internets auf einem Mobiltelefon war jedoch oft frustrierend, bis im Juni 2007 das erste iPhone veröffentlicht wurde, welches die neue Ära der Webseiten und Web-Anwendungen einleitete. Darüber hinaus entwickelte sich JavaScript von einer reinen Skriptsprache des Webs zu einer beliebten Sprache für die Entwicklung von Web-Anwendungen. Gegenwärtig gibt es basierend auf JavaScript zahlreiche Frameworks, die speziell für die Entwicklung einer Web-Anwendung verwendet werden.

Diese Bachelorarbeit wird in 2 großen Schwerpunkten unterteilt. Zum einen werden die theoretischen Grundlagen betrachtet und zum anderen wird anhand dieser Kenntnisse eine progressive Web-Anwendung entwickelt. Im ersten Kapitel werden die Begriffe Web-Anwendung und progressive Web-Anwendung konkretisiert und unterschieden. Außerdem werden die erforderlichen Strukturkomponenten und die Funktionsweise einer Web-Anwendung im Allgemeinen erläutert. Das Kapitel 1.3. beschäftigt sich mit den verschiedenen Formen von APIs.

[1] Vgl. *o.V.* (2020).

Kapitel 1.4 enthält Informationen zu der bis heute zwei beliebtesten Entwurfsmuster. Das 2. Kapitel „Projektdurchführung“ enthält eine Erläuterung der Projektentwicklung, bei der nur JavaScript verwendet wurde. Das letzte Kapitel vergleicht die Komplexität der Entwicklung einer Progressive Web-Anwendung in reinem JavaScript und eine in Angular. Am Ende erfolgt eine Zusammenfassung der Bachelorarbeit.

1.2. Web-Anwendungen

Ähnlich wie eine Webseite läuft eine Web-Anwendung in einem Webbrowser wie Google Chrome, Mozilla Firefox oder Safari. Während eine Webseite aus statischen Inhalten besteht, wird eine Web-Anwendung durch ihre Interaktion mit dem Benutzer definiert, daher erfordert sie Benutzereingaben und Datenverarbeitung. Eine Web-Anwendung bietet auch eine Benutzerschnittstelle und kann eine bestimmte Funktion ausführen, um auf Daten zuzugreifen, sie zu erstellen, zu speichern und zu ändern. Da der Zugriff auf Web-Anwendungen nur einen Webbrowser benötigt, müssen Web-Anwendungen nicht heruntergeladen werden.

1.2.1. Progressive Web-Anwendungen

Der Begriff Progressive Web-Anwendung (PWAs) wurde 2015 von Alex Russel und Frances Berriman vom Chrome-Entwicklungsteam geprägt. PWAs sind Web-Anwendungen, die so entwickelt wurden, dass sie sich wie native Anwendungen anfühlen. Jedoch bieten sie mehr Funktionalität als herkömmliche Web-Anwendungen an [2]. Eine native Anwendung ist eine Anwendung, die für Benutzer auf einer bestimmten Plattform oder einem bestimmten Gerät entwickelt wurde und muss im Play Store oder App Store für Android- bzw. iOS-Geräte heruntergeladen werden. Am Beispiel der nativen Anwendung von Yelp und der Web-Anwendung von yelp.com (*ff.* siehe Abbildung 1) kann man feststellen, dass die Web-Anwendung so entwickelt wurde, dass sie das identische Aussehen der nativen Anwendung liefert: Die Browserleiste wird rot und beim Herunterscrollen wird die Suchleiste oben auf der Seite gesperrt.

[2] Vgl. Alter (2017), S.15

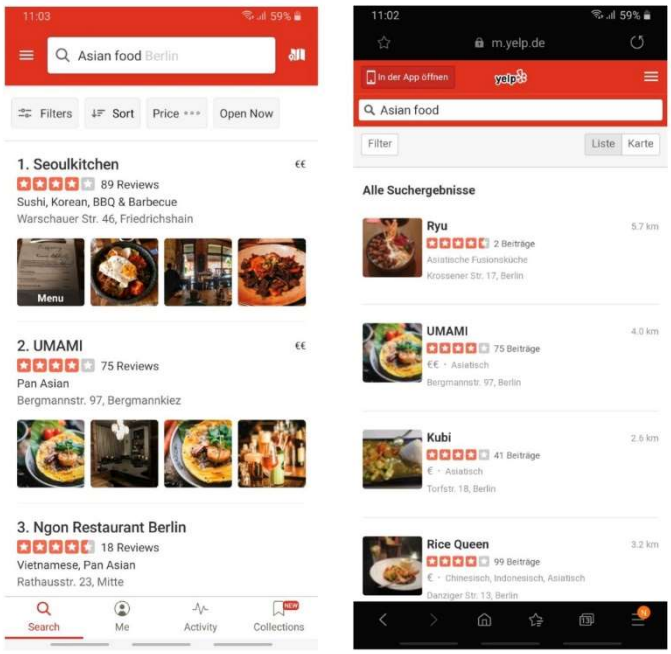

Abbildung 1: (links) native Anwendung, (rechts) Web-Anwendung.

Als Web-Anwendung verfügt PWA über zusätzliche Funktionen, die in normalen Web-Anwendungen nicht zu finden sind, nämlich:

- Unabhängigkeit der Konnektivität.
  Im Gegensatz zu einer normalen Web-Anwendung können PWAs mit einem Netzwerk auch offline und von geringer Qualität arbeiten.

- Einfach zu installieren.
  PWAs können problemlos zum Desktop hinzugefügt werden und erfordern keinen mobilen App-Store.

- Native Anwendung-ähnliches Gefühl.
  PWAs funktionieren wie eine native Anwendung. Benutzer werden die Unterschiede zwischen PWAs und nativen Anwendungen nicht spüren, da PWAs keine URL-Leiste oder Vorwärts- und Rückwärtsbanner haben, die in Webbrowsern zu finden sind, jedoch von Webbrowser ausgeführt werden.

Im darauffolgenden Bild (Abbildung 2) erkennt man alle zusätzlichen Funktionen von PWAs im Vergleich zu nativen und regulären Web-Anwendungen.

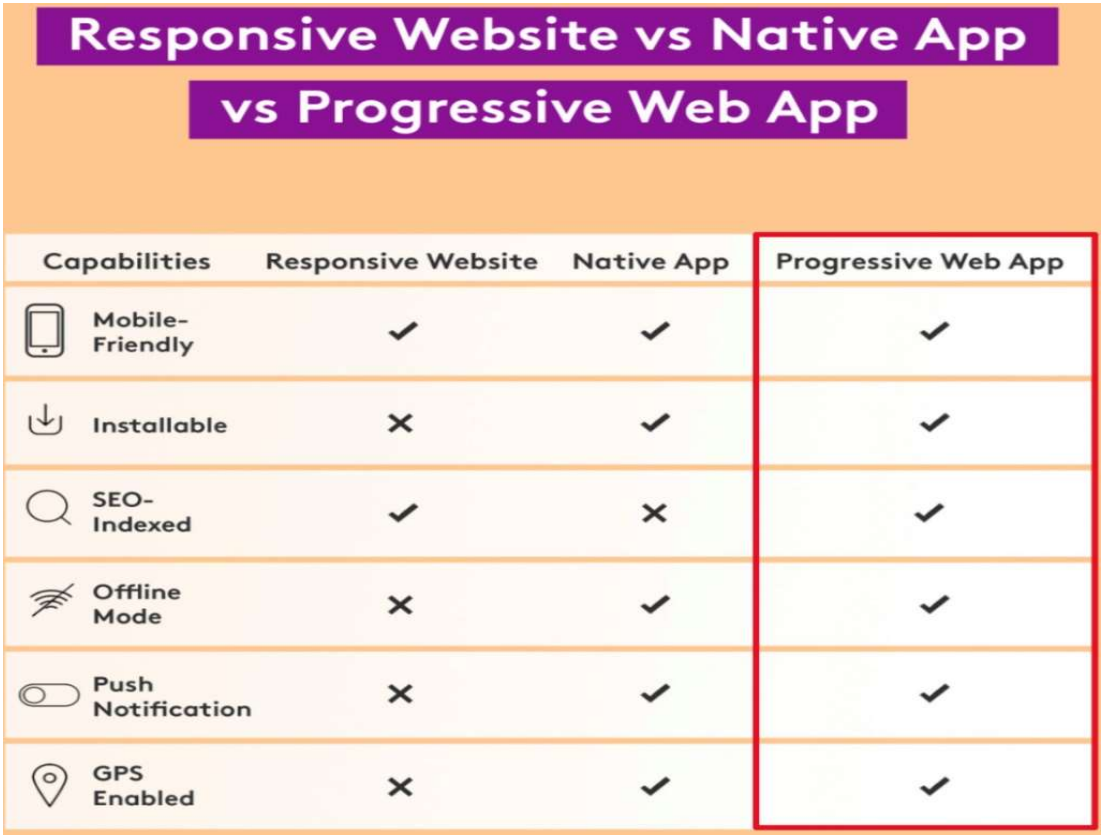

Abbildung 2. Vergleich von nativen, Web-, und progressiven Web-Anwendungen

Um eine PWA mit ihren Funktionen zu entwickeln, müssen einige Anforderungen erfüllt sein:

1. Die PWA muss über HTTPS bereitgestellt werden.
   HTTPS schützt die PWA und ihre Integrität. Es verhindert, dass Eindringlinge ungeschützte Kommunikation ausnutzen, um die Benutzer dazu zu bringen, vertrauliche Informationen preiszugeben oder Malware zu installieren [3].

2. PWA passen sich auf Tablets und Mobilgeräten an.
   Die Bildschirmgröße sollte kein Hindernis für die Anzeige des vollständigen Inhalts der Anwendung sein.

3. Das Web-Anwendungsmanifest sollte verfügbar sein.

[3] Vgl. Basques (*o.J.*)

Dies ist eine JSON-Datei, die Metadaten der Anwendung enthält, z. B. Themenfarbe und Symbole. Das Webmanifest enthält alle Informationen darüber, wie eine PWA aussieht, nachdem der Benutzer die PWA installiert.

4. PWA sollte einen ServiceWorker haben.
   "It's essentially a JavaScript file that runs separately from the main browser thread, intercepting network requests, caching or retrieving resources from the cache, and delivering push messages" [4]. Eine häufige Aufgabe für einen ServiceWorker besteht darin, Offline-Funktionen für die PWA bereitzustellen.

Die Anforderungen an Manifest und ServiceWorker werden im Kapitel „Projektdurchführung" näher erläutert.

Wie bei einer normalen Web-Anwendung gibt es zwei strukturelle Hauptkomponenten zum Erstellen einer PWA: Client- und Serverseite. Technisch wird sie als Frontend bzw. Backend bezeichnet.

1.2.2. Frontend oder Clientseite

Das Frontend oder die Clientseite wird manchmal als „Webdesign" bezeichnet und konvertiert Daten in eine grafische Oberfläche, wobei Hypertext Markup Language (HTML), Cascading Style Sheet (CSS) und JavaScript (JS) verwendet werden. In dieser Hinsicht ist das Frontend all das, was der Benutzer sehen kann. Dies beinhaltet auch die Interaktion mit den Elementen auf der Webseite.

HTML ist die Grundlage einer Webseite, der Standard-Markup-Language zum Erstellen von Webseiten. HTML spielt eine wichtige Rolle bei der Beschreibung der Struktur einer Webseite und besteht aus einer Reihe von Elementen, die dem Browser mitteilen, wie die Inhalte angezeigt werden sollen [5]. Der Rendering Teil bezeichnet man die Interpretation des Inhalts im Browser.

---

[4] Vgl. Introduction to Service Worker (*o.D.*)
[5] Vgl. HTML Introduction (*o.J*)

“CSS is a language for specifying how HTML elements are presented to the users - how they are styled, laid out, etc.” [6] und dies geschieht durch die Benutzung eines Selektors an einem HTML-Element, z. B. ID, Klasse oder Name. “CSS enables Web authors and programmers to finely tune elements for publishing both online and across several different types of media, including print format.” [7]. CSS ist auch dafür verantwortlich, wie die Anwendung auf mobilen Geräten oder größeren Geräten wie Tablets dargestellt wird.

JavaScript ist eine Skript- oder Programmiersprache, die komplexe Funktionen auf Webseiten implementieren können. JavaScript ist der Kern jeder dynamischen und interaktiven Funktionalität auf der Webseite. Es kann sowohl HTML als auch CSS anpassen und ändern. Darüber hinaus hat JavaScript die Möglichkeit auf Benutzeraktionen zu reagieren, einen Request über das Netzwerk zu senden, welche wiederum zur Datenaufnahme führt. Bei letzterem kann JavaScript, wenn es asynchron verwendet wird, alle erforderlichen Daten an einen Server senden und von diesem empfangen, ohne die Webseite zu aktualisieren, was das Kernstück einer PWA darstellt.

1.2.3. Mehr über JavaScript

JavaScript ist so bedeutend geworden, dass es zur Standardprogrammiersprache des Webs wurde. „Stack Overflow“ ist eine Webseite für Entwickler, auf der Entwickler Fragen von anderen Entwicklern stellen und beantworten können. Diese Webseite führt jedes Jahr eine Umfrage durch, welche Sprachen am häufigsten als Programmiersprache verwendet werden. Ab 2019 steht JavaScript das siebte Jahr in Folge ganz oben auf der Liste (*ff.* siehe Abbildung 3). Die einfache Lernkurve und die steigende Nachfrage nach JavaScript-Entwicklern sind ein weiterer Grund, warum JavaScript sehr beliebt ist. Neben seiner Bekanntheit und Einfachheit wird JavaScript von verschiedenen Browsern manchmal unterschiedlich interpretiert, was zu einem großen Nachteil bei der Verwendung dieser Sprache wird. Daher muss der Code vor der Veröffentlichung auf verschiedenen Plattformen getestet werden.

[6] MDN Contributors (2019)
[7] Schafer, Steven M. (2010), S.419

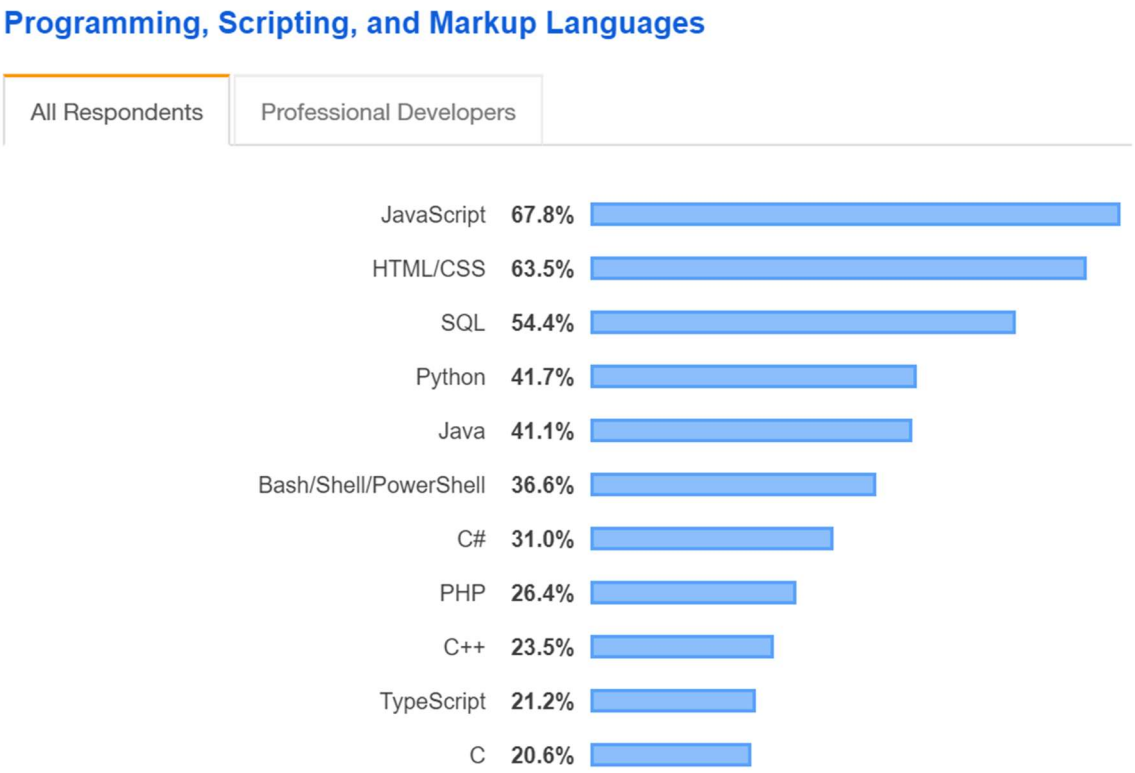

Abbildung 3. Developer Survey Results 2019.

1.2.4. Backend oder Serverseite

Einfachheitshalber ist das Backend all das, was dem Benutzer verborgen bleibt und im Browser nicht sehen können, wie z. B. Datenbank und Server. Typischerweise ist das Backend nicht nur für die Speicherung und Organisation von Daten verantwortlich, sondern auch dafür, dass alles auf der Clientseite funktioniert. Das Backend kommuniziert mit dem Frontend, sendet und empfängt Informationen. Häufig werden die Programmiersprachen Java, Python und C verwendet, aber seit 2009 ist es auch möglich das Backend mit JavaScript zu programmieren.

1.2.5. Wie Web-Anwendungen funktionieren

Eine Web-Anwendung besteht aus einem Frontend und einem Backend. Was dem Benutzer letztendlich angezeigt wird hängt vom Zusammenspiel zwischen diesen beiden Komponenten ab. Im Folgenden ist ein Diagramm zu finden (siehe Abbildung 4), welches die Funktionsweise einer Web-Anwendung erläutert.

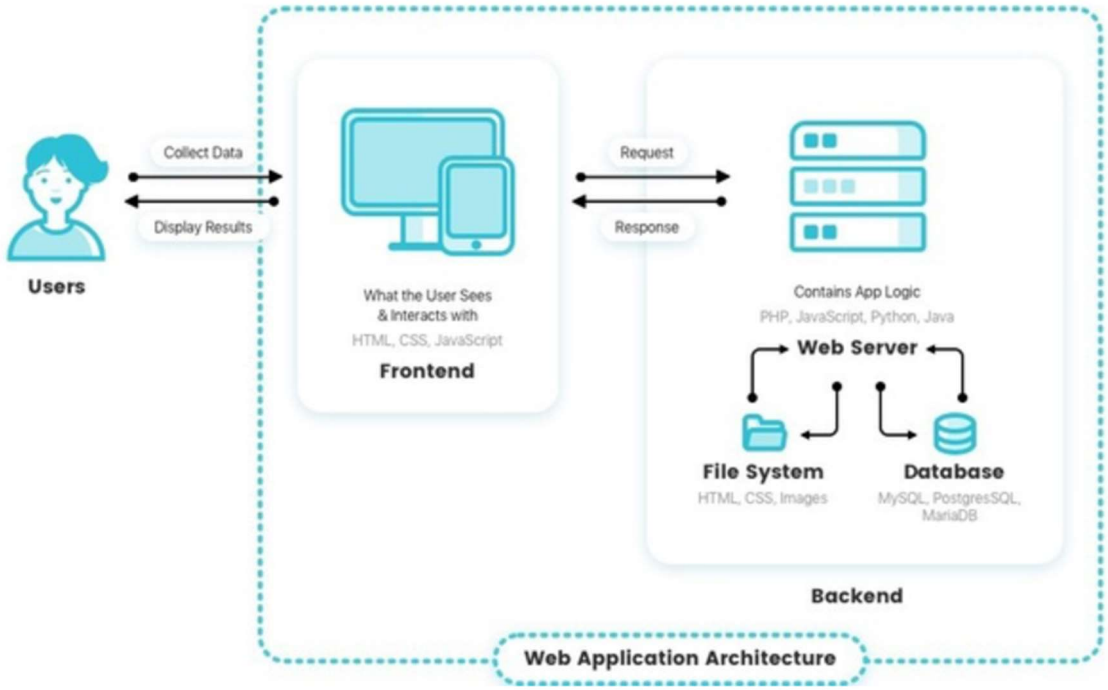

Abbildung 4. Workflow für eine Web-Anwendung.

Die Benutzer greifen über einen Webbrowser auf eine Web-Anwendung zu und lösen einen Request an den Webserver aus. Nachdem der Server den Request von Benutzern erhalten hat, wird der angeforderte Request verarbeitet, z. B. zur Datenbankabfrage oder Datenverarbeitung. Anschließend sendet der Server das Ergebnis zurück. Das Ergebnis wird als eine Response bezeichnet. Schließlich werden die Daten auf dem Display der Benutzer angezeigt.

1.3. Application Programming Interface (API)

“An API is the interface that a software program presents to other programs, to humans, and, in the case of web APIs, to the world via the internet “ [8]. APIs bestehen aus zahlreichen Elementen, z. B. Funktionen und Tools, mit denen Entwickler Anwendungen erstellen können. Mithilfe von APIs kann die Entwicklung einer Anwendung beschleunigt werden, da die erforderlichen Funktionen nicht selbst implementiert werden müssen. Die beiden API-Kategorien in clientseitigem JavaScript sind Browser-APIs und APIs von Drittanbietern.

1.3.1. Browser APIs

Browser-APIs sind in einen Webbrowser integriert und können Daten aus dem Browser und der umgebenden Computerumgebung verfügbar machen und damit nützliche komplexe Dinge tun [9]. Beispiele für die im Browser verfügbaren APIs sind Browser Object Model (BOM) und Document Model Object (DOM). BOM wird zur Interaktion mit dem Browser verwendet, während DOM das gesamte HTML-Dokument darstellt. Ein weiteres Beispiel ist die Web-Audio-API, die später im Kapitel „Projektdurchführung“ erläutert wird.

1.3.2. APIs von Drittanbietern

“Third-party APIs are APIs provided by third parties — generally companies such as Facebook, Twitter, or Google — to allow you to access their functionality via JavaScript and use it on your site “ [10]. Eine von einem Drittanbieter entwickelte API, beispielsweise die Google Maps-API, enthält Vorschriften und Anforderungen für die Implementierung in einem Projekt.

[8] Jin, Brenda *et al.*, 2018, S. 1
[9] MDN Contributors (2016)
[10] MDN Contributors (2017)

1.4. Entwurfsmuster

"In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design" [11]. Mit anderen Worten, es ist eine Beschreibung oder eine Vorlage zur Lösung eines Problems, die in vielen verschiedenen Situationen verwendet werden können. Ein Entwurfsmuster ist wichtig, wenn derselbe Code beibehalten und wiederverwendet werden soll. Ein praktisches Beispiel für Muster ist die Manipulation des Document Object Model (DOM) mit der JavaScript-Methode document.getElementById(). Diese Methode kann verwendet werden, um Zugriff auf das Element mit einer bestimmten ID zu erhalten. Nach dem Zugriff auf das Element können sein Inhalt und Stil in JavaScript bearbeitet werden.

1.4.1. MV* Muster

Die beiden wichtigen Entwurfsmuster sind Model View Controller (MVC) und Model View Presenter (MVP).

Modell - verwaltet die Daten für die Anwendung und definiert die Business-Rules (oder BusinessLogic), wie die Daten geändert oder manipuliert werden können.

View - ist dafür verantwortlich, was dem Benutzer (der Benutzeroberfläche) angezeigt wird.

Das * steht für einen Platzhalter, welches sich entweder um einen Controller oder einen Presenter handeln kann.

1.4.2. MVC

"MVC is an architectural design pattern that encourages improved application organization through a separation of concerns" [12]. Da das Modell und die View auf der vorherigen Seite definiert wurden, kann Controller wie folgt verstanden werden:

[11] Design Patterns (*o.D.*)
[12] Osmani, Addy (2012), S.79

Controller: Im MVC-Pattern ist der Controller für die Bearbeitung der eingehenden Requests verantwortlich. Der Controller fungiert auch als Einstiegspunkt für die Anwendung (siehe Abbildung 5).

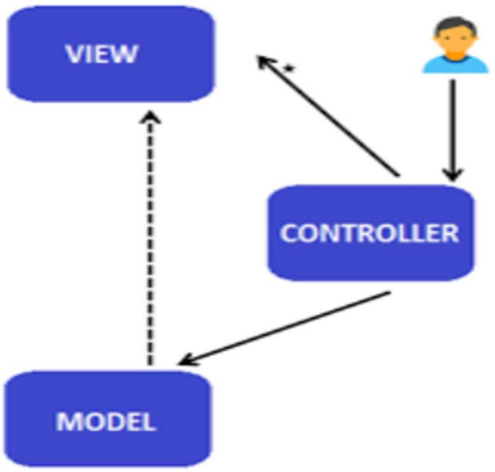

Abbildung 5. Workflow für MVC.

Wenn die Benutzer eine Aktion ausführen, senden sie einen Request an den Controller. Der Controller leitet den Request an das Modell weiter, wo die Daten für eine Response verarbeitet und vorbereitet werden. Zuletzt weist das Modell die View an, ihre Daten entsprechend den neuen Daten zu aktualisieren.

Die Abbildung 6 erläutert die Interaktion zwischen dem Benutzer und dem Entwurfsmuster.

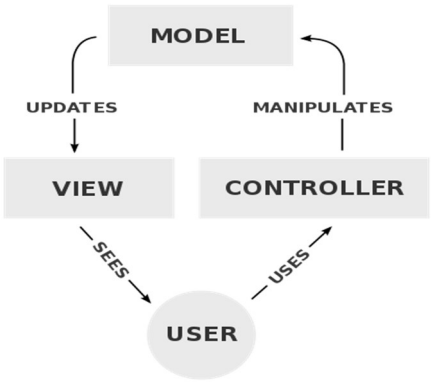

Abbildung 6. Diagramm der Interaktionen innerhalb des MVC-Musters.

Zusammengefasst stellt das Modell die Daten und BusinessLogic einer Webanwendung zur Verfügung. Die View ist eine Benutzeroberfläche und der Controller ist ein Request Handler.

1.4.3. MVP

Das MVP-Entwurfsmuster trennt das Modell von der View durch einen Presenter. “Model-view-presenter (MVP) is a derivative of the MVC design pattern which focuses on improving presentation logic“ [13].

Presenter: Der Presenter empfängt die Benutzereingaben aus der View, verarbeitet die Daten mit Hilfe des Modells und gibt das Ergebnis an die View zurück. Der Presenter manipuliert das Modell und aktualisiert die View. In diesem Pattern ist eine Kommunikation nur über den Presenter möglich (siehe Abbildung 6).

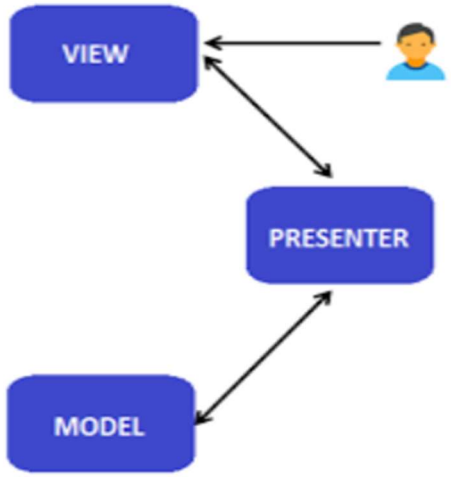

Abbildung 5. Workflow für MVP.

1.4.4. MVP oder MVC?

Der einzige Unterschied zwischen MVP und MVC besteht in der Funktionalität von Controller und Presenter. Da MVP eine Ableitung von MVC ist, müssen bei der Entscheidung, welches Entwurfsmuster verwendet werden soll, zwei Punkte berücksichtigt werden:

1. MVP wird in einer Anwendung auf Unternehmensebene verwendet, in der viel Presenter-logik wie möglich wiederverwendet werden muss.
2. MVP soll verwendet werden, wenn es sich um Anwendungen mit sehr komplexer View und einer großen Benutzeroberfläche handelt, da die gesamte

[13] Osmani, Addy (2012), S.88

Logik in einem Presenter anstatt in mehrere Controller in MVC gekapselt werden kann.

Ein weiterer Gesichtspunkt ist das Unit-Testing. „Unit testing is a piece of code (usually a method) that invokes another piece of code and later checks the correctness of some assumptions" [14]. MVP wäre eine bessere Option, wenn Unit-Testing durchgeführt werden müssen, da der Presenter als ein vollständiges Modell der Benutzeroberfläche gilt und somit unabhängig von anderen Komponenten des Unit-Testing unterzogen werden kann [15].

[14] Saleh, Hazem (2013), S.7
[15] Vgl. Osmani, Addy (2012), S.90

2. Projektdurchführung

2.1. Projektbeschreibung

Das Projekt bestand darin, eine progressive Web-Anwendung am Beispiel eines Musiklernprogramms zu entwickeln, das auf mobilen Geräten kompatibel ist. Dieses Projekt wurde unter Bezugnahme auf die Anwendung „Perfektes Ohr - Musiktheorie, Ohr- und Rhythmus-Training“ erstellt. Die Perfektes Ohr-Anwendung wurde von Aleksandr Osmanov entwickelt und kann sowohl auf Android- als auch auf iOS-Handys heruntergeladen werden. “Perfektes Ohr“ bietet viele Funktionen für Anfänger und Profis in Form von Gehör-, Gesangs- und Rhythmusübungen.

Für dieses Projekt wurde diese progressive Web-Anwendung nur in JavaScript (VanillaJS) geschrieben. Später wird die Anwendungsfunktion „Interval-Identification“ nochmals mit Angular geschrieben, eine von Google entwickeltem JavaScript Framework, und mit der Anwendung in VanillaJS verglichen. Es gibt drei Funktionen, die in VanillaJS geschrieben wurden:

1. Rhythm Input
2. Rhythm Tapping
3. Interval Identification

Diese Web-Anwendung verfügt weder über einen Server noch über eine serverseitige Programmiersprache. Die Daten, die übertragen werden müssen, wurden in einer Array-Variablen deklariert.

2.2. Entwicklungsumgebung

Diese Anwendung wurde auf einem Laptop unter Windows 10 entwickelt und GitHub wurde zur Versionskontrolle verwendet. Laut GitHub wurde das Projekt hauptsächlich in JavaScript geschrieben (siehe Abbildung 7).

Abbildung 7. Anteil der benutzten Programmiersprachen in dem Projekt.

Für das Projekt wurde Visual Studio Code als Integrated Development Environment (IDE) verwendet. Visual Studio Code bietet eine Extension namens Live Server, mit der ein lokaler Entwicklungsserver mit einer Live-Reload-Funktion für statische und dynamische Seiten gestartet wird. Durch die Verwendung des Live-Servers können Entwickler die vorgenommenen Änderungen in Echtzeit anzeigen.

Während der Entwicklung wurde das Projekt auf Google Chrome ausgeführt. Wie alle anderen modernen Webbrowser verfügt Google Chrome über eine Reihe von Webentwickler-Tools, die direkt in den Browser integriert sind. Die heißen „Chrome DevTools“. Da die Web-Anwendung auf Mobilgeräten ausgeführt werden kann, verfügt „Chrome DevTools“ über eine Funktion, mit der Entwickler eine Vorschau des Designs der Web-Anwendung auf einer Vielzahl gängiger Mobilgeräte anzeigen können (siehe Abbildung 8). Alle in diesem Kapitel aufgenommenen Bilder wurden mit „Chrome DevTools“ in der Laptop-Ansicht aufgenommen.

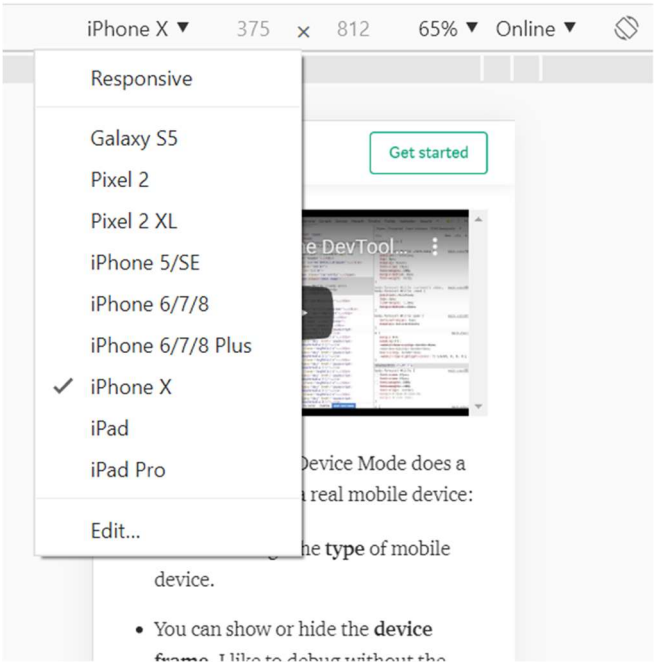

Abbildung 8. Chrome DevTools.

Diese progressive Web-Anwendung ist nur für die Ausführung auf Google Chrome auf Android-Mobilgeräten vorgesehen, um alle Funktionen einer installierbaren PWA zu nutzen, da iOS keine Funktion zur direkten Installation von PWAs unterstützt. Andernfalls kann die Web-Anwendung auf jedem anderen Gerät ausgeführt werden.

2.3. Bibliotheken

Während der Entwicklung einer Anwendung verwenden Entwickler den vorhandenen Code in der Regel erneut, um ein Problem zu lösen, damit sie nicht ihre Zeit verschwenden Code neu zu schreiben, die ein anderer Entwickler bereits geschrieben hat. Programmierer verwenden dazu eine JavaScript-Bibliothek. Die JavaScript-Bibliothek kann als eine Bibliothek mit vorab geschriebenem JavaScript definiert werden, die eine einfachere Entwicklung von JavaScript-basierten Anwendungen ermöglicht [16]. Dieses Projekt verwendet auch eine JavaScript-Bibliothek und eine Web-Audio-API. Nicht zu verwechseln, es gibt Unterschiede zwischen einer API und einer Bibliothek.

Da beide Begriffe definiert wurden, wird im Folgenden der Vergleich zwischen einer Bibliothek und einer API beschrieben (siehe Abbildung 9):

Library	API
A reusable chunk of codes	A point of contact that allows interaction between running codes
It refers to the code itself	It refers to the interface
It is not an API by itself	It can be made of several libraries

Abbildung 9. Unterschied zwischen einer Bibliothek und einer API.

Zum Rendering der Musiknotation verwendet dieses Projekt eine JavaScript-Bibliothek namens „EasyScore“. Diese Bibliothek macht alle „VexFlow“-Elemente über ihre APIs verfügbar. „VexFlow“ selbst ist eine webbasierte Open-Source-API zum Rendering von Musiknotationen, die vollständig in JavaScript geschrieben ist. Zur Vereinfachung der Anzeige des Notenstabs und einiger Notizen bleibt dieses Projekt bei der Verwendung von „EasyScore“. Es erstellt eine Scalable Vector Graphics

[16] Vgl. JavaScript Library (2015)

(SVG) und hängt sie an ein HTML-Element an. Der Code im Projekt (Abbildung 10) sieht so aus:

```
var vf = new VF.Factory({
  renderer: {
    elementId: "noteRT",
    height: 100,
    width: 500
  }
});
```

Abbildung 10. Code zum Rendering ein Musiknotation für Element mit ID *noteRT.*

Ein weiteres wichtiges Merkmal für eine Musiklernanwendung ist der Ton. Für die Wiedergabe der Audiodateien und des Metronom-Soundeffekts verwendet dieses Projekt eine Web-Audio-API. „The Web Audio API provides a powerful and versatile system for controlling audio on the Web, allowing developers to choose audio sources, add effects to audio, create audio visualizations, apply spatial effects (such as panning) and much more" [17]. Die Web-Audio-API verfügt über einen Audio() - Konstruktor. Es erstellt und gibt ein neues HTML-Audioelement zurück, welches man bearbeiten oder zur Audiowiedergabe verwenden kann. Die Syntax für die Verwendung des Konstruktors lautet:

```
audioObj = new Audio(url);
```

Abbildung 11. Code von dem Audio()-Konstruktor.

Im Projekt ist die URL der Pfad zur Musikquelle, die sich im Assets-Ordner befindet. Die Abbildung 12 zeigt wie der Ton von dem Metronom abgespielt wird.

```
const m = new Audio("assets/audio/metronome.wav");
m.play()
```

Abbildung 12. Code für Metronomton-Widergabe.

2.3.1. Assets

Dieses Projekt enthält einen Ordner namens Assets, in dem alle für die Symbole und Audiodateien verwendeten Bilder gespeichert werden. Die Quelle all dieser Elemente befindet sich in der Projektdatei **ResourceList.txt**. Für den Klang des Tones wurde er von einer iPad-Anwendung aufgenommen und mit „Audacity", einem

[17] MDN Contributors (2017)

kostenlosen Open-Source-Digital-Audio-Editor und einer Aufnahmeanwendungs-software, bearbeitet.

2.4. Styling – Responsives Design

Da auf die meisten Web-Anwendungen (und Webseiten) auf Mobilgeräten zugegriffen werden, ist es unabdingbar geworden wie sie auf kleineren Bildschirmen aussehen. Der Begriff responsives Design hat einen großen Einfluss auf die Entwicklung von Web-Anwendungen. „Responsive design, overall, is a way to make websites that can be easily viewed and used on any type of device and size of the screen, all the way from the smallest mobile phones up to the widest desktop monitors “ [18].

Mit dem CSS kann dies durch eine Technik namens „Media Query“ erreicht werden. Mit „Media Query“ können die Entwickler viele Anpassungen vornehmen, z. B. kleinere Ränder zwischen zwei Elementen auf kleineren Geräten oder größere (und besser lesbare) Schriftgrößen auf größeren Geräten.

Angesichts der Tatsache, dass es so viele Geräte mit unterschiedlichen Bildschirmgrößen existieren, werden in diesem Projekt nur drei „Media Queries“ für Telefone mit einer Mindestbreite von 350 Pixeln und maximal 480 Pixeln, Tablets mit einer Mindestbreite von 768 Pixeln und maximal 1024 Pixeln sowie Laptops mit einer Mindestbreite von 1025 Pixeln (siehe Abbildung 13).

```
/* Phones */
@media only screen and (min-width: 350px) and (max-width: 480px) {}

/* Tablets */
@media only screen and (min-device-width: 768px) and (max-device-width: 1024px) {}

/* Laptops */
@media only screen and (min-device-width: 1025px) {}
```

Abbildung 13. Media Queries für Mobiltelefon, Tablets und Laptops.

[18] Peterson, Clarissa (2014), S. 3

“The @media rule is used in media queries to apply different styles for different media types/devices” [19]. “The *only* keyword is used to prevent older browsers that do not support media queries with media features from applying the specified styles.” [20] . Mit der Verwendung von “Media Query“ wird die Anwendung auf verschiedenen Geräten so angezeigt (siehe Abbildung 14).

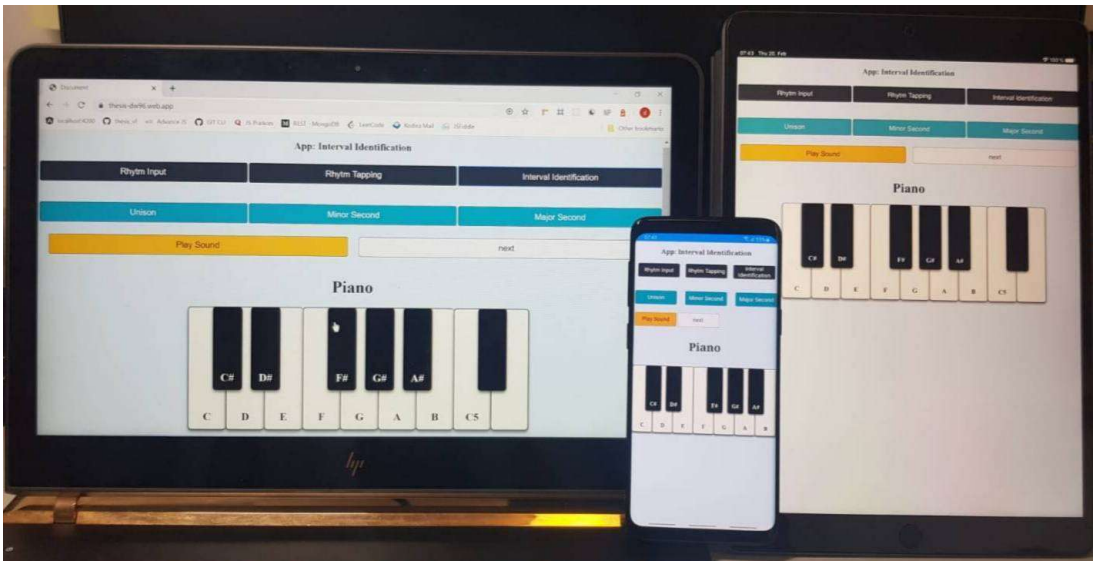

Abbildung 14. Die Anwendung auf einem Laptop, Mobiltelefon, und Tablet.

2.5. Hosting und Deployment

Das Deployment ist ein Prozess, um Endbenutzern eine Anwendung zur Verfügung zu stellen. Das heißt, wenn Benutzer die URL eingeben, können sie die neueste Version der Anwendung sehen. Zu diesem Zweck verwendet dieses Projekt den Hosting-Service von Firebase.

Firebase ist eine Entwicklungsplattform für Mobil- und Webanwendungen von Google, die nützliche Dienste zum Erstellen einer besseren Anwendung wie Authentifizierung, Echtzeitdatenbank, Analyse usw. bereitstellt. Mit Firebase können sich Entwickler auf ihre Anwendungsfunktionen konzentrieren, da Firebase sich um alle Serverangelegenheit kümmert. Zudem bietet es kostenlose Subdomains in den domains web.app und firebase.app.com. Dieses Projekt finden Sie unter der URL thesis-dw96.web.app oder thesis-dw96.firebase.app.com.

[19] CSS @media Rule (*o.J.*)
[20] Srivastava, Tejasvi (*o.J.*)

2.6. Installierbare PWA

Eine der Hauptmerkmale von PWAs ist die Installierbarkeit, mit der die Benutzer die Anwendung zum Startbildschirm hinzufügen können. Sie wird wie eine native Anwendung angezeigt. Zu diesem Zweck können Benutzer es manuell hinzufügen, indem sie auf die Optionen in ihrem Browser klicken und die Option „Add to Homescreen“ auswählen.

Wenn die PWA in Chrome und Firefox die folgenden Kriterien erfüllt [21]:

- Die Anwendung ist noch nicht installiert
- wird über HTTPS bereitgestellt
- hat einen ServiceWorker registriert
- enthält ein Web-App-Manifest, das folgende Attribute umfasst:
    - short_name oder name, der Name von der PWA.
    - icons, bedeutet das anzuzeigende Icon mit einer Größe von 192pixel und 512pixel.
    - start_url, eine Zeichenfolge, die die Start-URL der PWA darstellt.
    - display, muss als Wert fullscreen, standalone oder minimal-ui annehmen. Der Wert standalone ist der Standardwert, der die Anwendung wie eine native Anwendung erscheinen lässt.

wird das Event beforeinstallprompt ausgelöst und zeigt eine Notifikation an, welche angibt, dass die Anwendung als PWA installiert werden kann. Der Name der PWA lautet „Rhythm Pro“. Wird er dem Startbildschirm hinzugefügt, entsteht eine Abkürzung des Namens von „RPro“ (*ff.* siehe Abbildung 15 und 16).

[21] Vgl. LePage, Pete (2020)

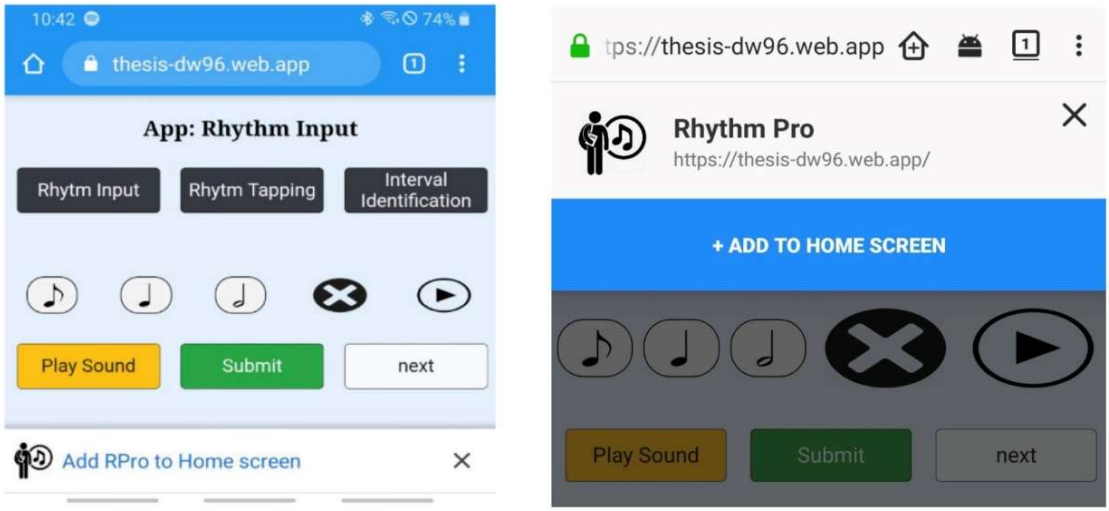

Abbildung 15: “Add to Homescreen” Notifikation für Chrome und Firefox.

Abbildung 16: Icons für die PWA in Chrome und Firefox.

Jedes Betriebssystem behandelt PWA anders. Das Anzeigen einer Notifikation zum Installieren der PWA in Chrome und Firefox ist zwar möglich, in Chrome für iOS jedoch nicht möglich. Auf iOS-Geräten können Benutzer PWAs nur über die URL in Safari installieren (siehe Abbildung 17), da Browser von Drittanbietern PWA nicht installieren dürfen und einige von ihnen keinen ServiceWorker unterstützen [22].

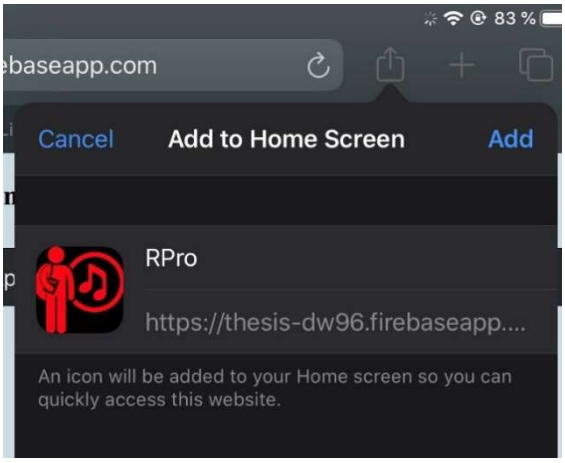

Abbildung 17. “Add to Homescreen” in Safari.

[22] Vgl. Caputa, Maciej (2018)

Obwohl PWAs in Safari gestartet werden können, gibt es bei der Ausführung auf iOS-Geräten noch einige Einschränkungen:

- Die Funktion " Zum Startbildschirm hinzufügen" Notifikation ist in Safari nicht verfügbar, daher sind Benutzer nicht über die Funktionen der Anwendung informiert.
- Beschränkung, wie viele Daten gespeichert werden können.
- Es ist unfähig Web-Push-Benachrichtigung anzuzeigen.
- Icons, die in der Manifest Datei der Webanwendung angegeben sind, werden auf iOS-Geräten nicht verwendet. Die transparenten Teile des Icons sollten berücksichtigt werden, da dieser Teil schwarz eingefärbt wird. Aus diesem Grund hat das Icon dieses Projekts eine andere Farbe als das Icon in Chrome oder Firefox für Android. Um ein Icon für iOS-Geräte hinzuzufügen, sollte ein Link-Tag in der HTML-Datei hinzugefügt werden. Der Wert für das apple-touch-icon im folgenden Beispiel (Abbildung 18) stellt das Symbol für den Startbildschirm auf iOS-Geräten dar:

```
<link rel="apple-touch-icon" href="./icon.png">
```

Abbildung 18. Der Link Tag für das Hinzufügen von Icons auf iOS Geräte.

PWAs werden entwickelt, um den Benutzern eine ähnliche Erfahrung wie bei nativen Anwendungen zu bieten. Während dies auf Android-Geräten sehr gut ausgeführt werden kann, gibt es bei iOS-Geräten noch einige größere Probleme, die behoben werden müssen.

2.7. Anwendung – VanillaJS

Dieses Kapitel gibt genauere Einblicke über die Entwicklung von „RhythmPro“, wie ein Benutzer anhand von Inputs das Aussehen der Benutzeroberfläche verändern kann und wie die Daten in einer Array-Variable zur Wiedergabe eines Tons oder zur Anzeige der Note in der Musiknotation verwendet werden.

2.7.1. Anwendung – Header

Der PWA-Header befindet sich oben in der Anwendung (siehe Abbildung 19). Er hat einen Titel der zur Zeit laufenden Anwendung und eine Navigationsleiste in Form von Knöpfen zum Umschalten von einer Anwendung zur anderen. Jeder Knopf hat eine ID von *RI*, *RT* und *II*. Diese IDs sind die Abkürzungen der Namen der einzelnen Anwendungen. *RI* steht für „Rhythm Input“, *RT* für „Rhythm Tapping“ und *II* für „Interval Identification“.

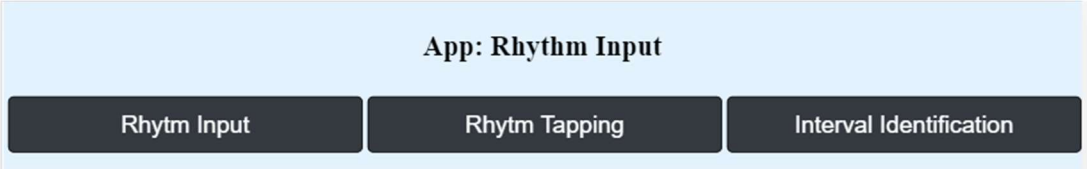

Abbildung 19. Der Header für die Anwendung in VanillaJS.

Wenn eine Button angeklickt wird, wird die folgende Funktion in Abbildung 20 ausgelöst.

```
function showApp(appId) {
  if (appId.id === "RI") {
    appName = "App: Rhythm Input";
    appRI.classList.remove("hideApp");
    appRT.classList.add("hideApp");
    appII.classList.add("hideApp");
  } else if (appId.id === "RT") {
    appName = "App: Rhythm Tapping";
    appRT.classList.remove("hideApp");
    appRI.classList.add("hideApp");
    appII.classList.add("hideApp");
  } else {
    appName = "App: Interval Identification";
    appII.classList.remove("hideApp");
    appRI.classList.add("hideApp");
    appRT.classList.add("hideApp");
  }
}
```

Abbildung 20. Funktion zum Wechseln der Anwendungen.

Der Inhalt der Anwendung wird entsprechend der an die Funktion übergebenen ID angezeigt, indem für jedes Element die Klasse *hideApp* entfernt und hinzugefügt wird. Diese Klasse hat die display-Eigenschaft, die das Layout steuert und angibt, ob und wie ein Element angezeigt wird. Beim Hinzufügen der Klasse *hideApp* hat die Display Eigenschaft den Wert „none“ welches das Element ausblendet. Beim Entfernen dieser Klasse wird das Element angezeigt.

2.7.2. Anwendung – Rhythm Input

Die Benutzer hören ein Rhythmusmuster und müssen diesen Rhythmus in das Musiknotationssystem eingeben mit den vorhandenen Optionen (siehe Abbildung 21).

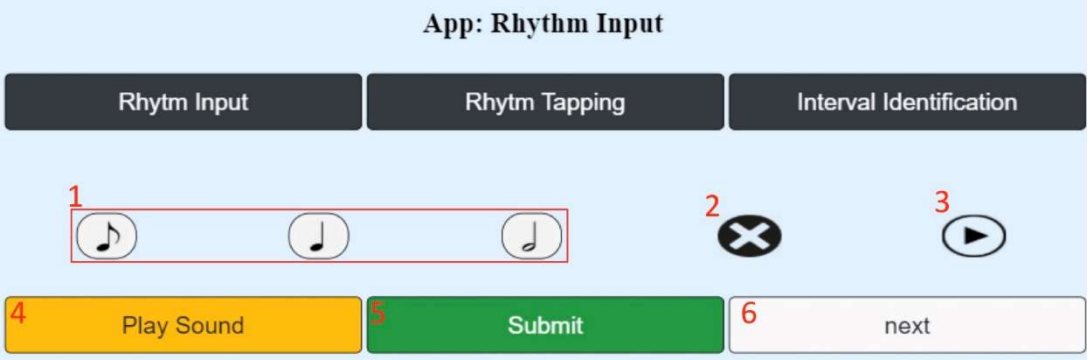

Abbildung 21. „User Interface“ (UI) der Anwendung Rhythm Input.

1. Benutzereingabe: Benutzer haben drei Optionen, nämlich Eine Achtel-, eine Viertel- und eine Halbe-Note.

2. Löschbutton: Beim Anklicken dieses Buttons werden alle Benutzereingaben entfernt.

3. Vorschautonbutton: Beim Klicken auf den Button hören die Benutzer ihre Eingabe als Rhythmusmuster abgespielt.

4. „Play Sound“-Button: Dieser Button dient zum Starten oder Wiederholen der aktuellen Übung.

5. „Submit“-Button: Die Benutzereingaben werden mit der Lösung der Übung verglichen

6. „next“-Button: Nach dem Einreichen können die Benutzer mit der nächsten Übung fortfahren. Andernfalls wird die aktuelle Übung übersprungen, wenn der Benutzer nicht auf den Button "Submit" geklickt hat.

Nachdem die Benutzer ihre Antwort eingereicht haben, vergleicht die Anwendung die Eingabe mit der Lösung der Übung. Die Anwendung teilt den Benutzern umgehend mit, ob ihre Eingabe korrekt oder falsch ist. Die „Correct!“ oder „Wrong!"-

Meldung befindet sich direkt unter dem Button „Play Sound“ (siehe Abbildung 22).

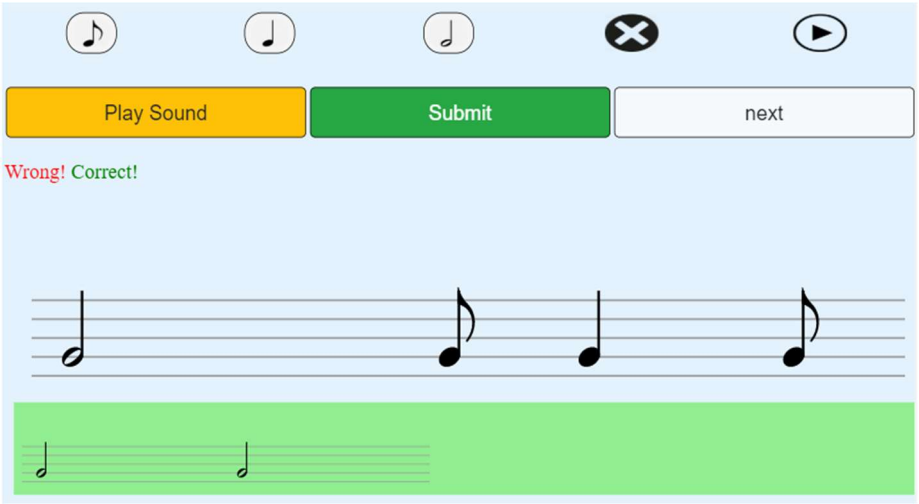

Abbildung 22. UI der Anwendung Rhythm Input nach dem Klicken auf „Submit“.

Wenn die Eingabe nicht mit der Lösung übereinstimmt, wird ein grünes Notensystem mit dem richtigen Rhythmusmuster direkt unter der Eingabe des Benutzers hinzugefügt, andernfalls werden den Benutzern die Meldung „Correct!“ angezeigt.

Die Übung und ihre Lösung werden in einer Array-Variablen deklariert (siehe Abbildung 23).

```
let rhytmInputDb = [
  { q: [4, 4] },
  { q: [1, 1, 4, 2] },
  { q: [2, 4, 1, 1] },
  { q: [4, 2, 2] },
  { q: [1, 1, 2, 4] }
];
```

Abbildung 23. Array Variable *rhytmInputDb.*

Jedes Objekt im Array repräsentiert die Aufgabe. Das Schlüsselwort q steht für „question“ und kann 3 mögliche Werte haben: 1, 2 und 4, die zur Anzeige der Achtel-, Viertel- bzw. Halbnote verwendet werden. Die Lösung wird angezeigt, wenn der Benutzer eine falsche Eingabe macht. So wird der Inhalt des Array q mithilfe der JavaScript-Methode .map() neu angeordnet (siehe Abbildung 24).

```
rhytmInputDb[indexRIQuestion].q
  .map(x => {
    if (x == 1) return "G4/8";
    else if (x == 2) return "G4/q";
    else return "G4/h";
  })
  .join(",")
```

Abbildung 24. Code für die „Bearbeitung“ der Attribute q.

Das verarbeitete Objekt gibt eine Zeichenkette aus, die in der VexFlow-Methode übergeben wird, um die richtigen Noten anzuzeigen. Zum Beispiel die Variable rhytmInputDb[0].q wurde verarbeitet und gibt die Zeichenkette von „G4/h,G4/h“ aus. Diese Zeichenkette bedeutet, dass die Aufgabe aus zwei Halbnoten besteht.

Nachdem alle Aufgaben beantwortet wurden, finden die Benutzer eine Übersichtsseite, die die Punktzahl der Benutzer und eine kurze Nachricht „Exercise done!“ enthält. Die Anwendung wird nach 3 Sekunden erneut gestartet.

2.7.3. Anwendung – Rhythm Tapping

Die Benutzer sehen ein Rhythmusmuster auf dem Notensystem. Die Benutzer müssen diesen Rhythmus so genau wie möglich mit dem Metronom antippen. Die Dauer der Noten in dieser Übung ist dieselbe wie bei der Anwendung „Rhythm Input“, bei der es sich um eine Achtel, eine Viertel und eine halbe Note handelt. Nachdem der Benutzer mit seiner Eingabe fertig ist, kann er die Lösung mit dem „Hint“-Button sehen. Wenn dieser Button geklickt wird, werden grüne Punkte angezeigt, die den Benutzern die richtigen Stellen anzeigen. Die Übung beginnt, nachdem die Benutzer auf den „Rhythm Tapping“-Button geklickt haben. Die Abbildung 25 zeigt die UI von dieser Anwendung.

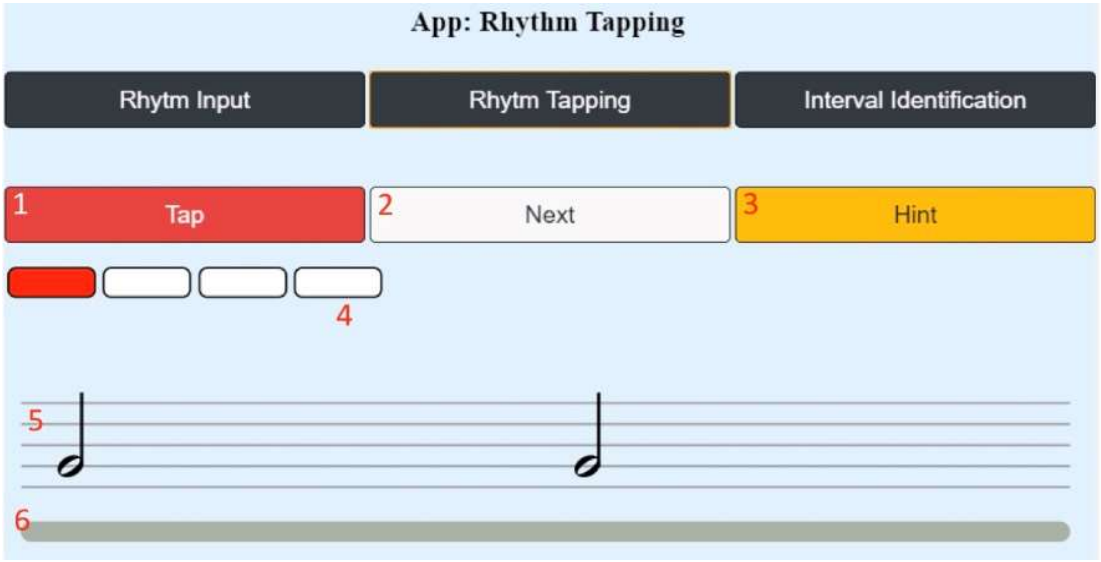

Abbildung 25. UI für die Anwendung „Rhythm Tapping“.

1. „Tap“-Button: Jedes Mal, wenn die Benutzer auf den „Tap“-Button anklicken, wird ein gelber Punkt auf der grauen Linie hinzugefügt. Dieser Punkt stellt die Position der laufenden Übung dar.

2. „Next“-Button: Weiter zur nächsten Übung.

3. „Hint“-Button: Dieser Button zeigt den Benutzern die richtige Lösung mit grünen Punkten an. Nachdem dieser Button angeklickt wurde, ändern alle Benutzereingaben (gelbe Punkte) ihre Deckkraft, sodass die grünen Punkte besser sichtbar sind als die Benutzereingaben.

4. Metronom Balken: Der Takt des Metronoms besteht aus vier kleinen Blöcken, wobei jeder Block einen Takt darstellt.

5. Notensystem für die Aufgabe: Jede Übung hat eine Taktart von 4/4, daher beträgt die Summe der gespielten Schläge vier.

6. Grauer Balken: der Balken wird durch das Anklicken jedes Benutzers mit einem gelben Punkt gefüllt. Ansonsten grüne Punkte, wenn der „Hint“-Button angeklickt wird.

Um die Noten auf dem Notensystem darzustellen, wird die Übung in einer Zeichenfolge definiert. Alle Übungen werden dann in einem Array gespeichert (siehe Abbildung 26). Jedes Element des Arrays wird nacheinander an den Renderern übergeben, wobei ein Index verwendet wird, der der Übung entspricht, in der sich die Benutzer befinden.

```
let rhythmTappingDb = [
  "G4/h,G4/h",
  "G4/8,G4/h,G4/q,G4/8",
  "G4/q,G4/q,G4/h",
  "G4/q,G4/q,G4/8,G4/q,G4/8",
  "G4/q,G4/8,G4/8,G4/h"
];
```

Abbildung 26. Array-Variable, wo die Übungen gespeichert werden.

Diese Anwendung hat zwei wichtige Funktionen, nämlich der Metronom-Balken, die den Benutzern hilft zu entscheiden, wann sie klicken sollen, und der graue

Balken für die Visualisierung, wenn die Benutzer rechtzeitig auf den Button tippen. Wenn die Anwendung startet, sehen die Benutzer, dass der Metronom-Balken ihre Farbe von weiß nach rot ändert und gleichzeitig ein Metronom-Soundeffekt abgespielt wird.

Der Metronom-Balken spielt in einer Endlosschleife und beginnt mit einem roten Block, dann zwei Blöcke, bis alle Blöcke rot sind, dann geht er mit nur einem roten Block wieder an den Anfang zurück. Der Zustand, in dem nur ein Block rot ist, ist der Zeitpunkt, an dem der Benutzer mit seiner Eingabe beginnen muss (siehe Abbildung 27).

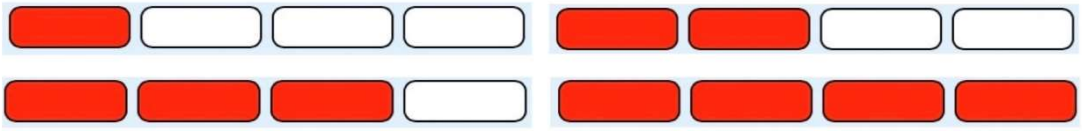

Abbildung 27. Zustände für den Metronom Balken.

Die Metronom-Visualisierung und der Ton werden mit der setInterval-Methode in JavaScript harmonisch abgespielt. Die Funktion innerhalb des Intervalls wird wiederholt ausgeführt, beginnend nach dem Zeitintervall und dann kontinuierlich in diesem Intervall wiederholt (Abbildung 28).

```
const m = new Audio("assets/audio/metronome.wav");
metronomebar = setInterval(() => {
  m.play();
}, 750);
```

Abbildung 28. Intervallmethode für den Metronom Ton.

Für die Metronom-Visualisierung werden die Stile der roten Blöcke mit Hilfe des JavaScript Document Object Model (DOM) in der Intervallmethode bearbeitet. Es ist eine einfache bedingte Anweisung, um zu entscheiden, welche Blöcke rot eingefärbt werden sollen (*ff.* siehe Abbildung 29). Die Variable *progressBar* bezeichnet die Anzahl der roten Blocks.

```
startBar = setInterval(() => {
  if (progressBar < 4) {
    document.getElementById("RT" + progressBar.toString())
            .style.backgroundColor = "red";
    progressBar++;
  } else {
    marginleftRT = 0;
    progressBar = 1;
    let ele = document.getElementsByClassName("met-bar");
    for (let i = 1; i < 4; i++) ele[i].style.backgroundColor = "white";
  }
}, 750);
```

Abbildung 29. Intervallmethode für den Metronom Balken.

Der graue Balken unter dem Notenliniensystem ist der Bereich, in dem sowohl die Eingaben der Benutzer als auch die Lösung angezeigt werden. Jedes Mal, wenn der Benutzer auf den „Tap“-Button klickt, wird ein neues Element erstellt, das eine benutzerdefinierte Klasse hat, und schließlich wird der linke Rand angepasst. Der linke Rand jedes Punktes hat unterschiedliche Werte, so dass die Punkte nebeneinanderstehen. Die Abbildung 30 zeigt die aufgerufene Funktion bei jedem Klick-Event und die Abbildung 31 ist die UI nach der Benutzereingabe.

```
function inputRT() {
  let answer = document.createElement("div");
  answer.setAttribute("id", "answerRT");
  answer.classList.add("answerRtClass");
  answer.style.left = marginleftRT.toString() + "px";
  document.getElementById("sb-parent").appendChild(answer);
}
```

Abbildung 30. Die Funktion zum Erstellen eines gelben Punktes.

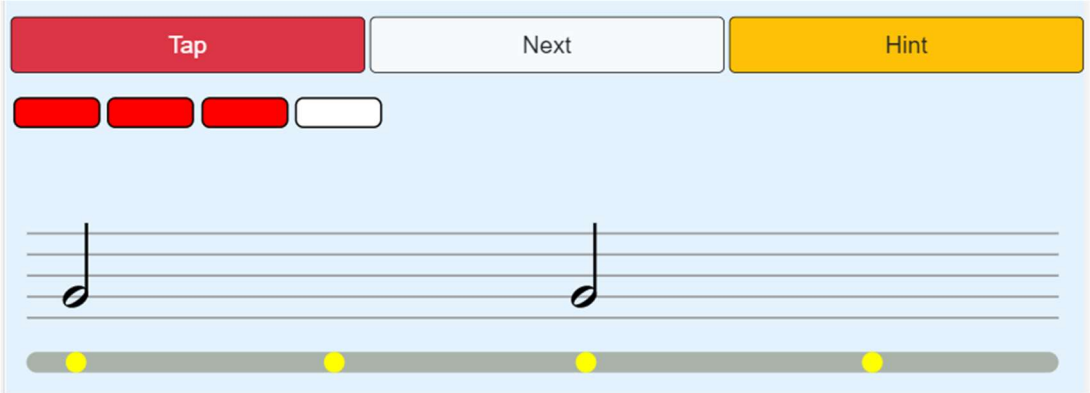

Abbildung 31. UI nach der Benutzereigabe.

Der graue Balken hat die gleiche Breite von 500 Pixeln wie das Notensystem. Die Idee ist, den aktuellen linken Rand des gelben Punkts hinzuzufügen, solange der

Wert des linken Randes aufgrund des Intervalls weiter zunimmt. In der obigen Abbildung wird der graue Balken in 4 Bereiche unterteilt, wobei jeder Bereich eine Länge von 125 Pixel hat. Kurz gesagt,

1 Metronom Block = 125 Pixel = 750 Millisekunden

Aus dieser Gleichung wird das Intervall (Abbildung 32) so eingestellt, dass der linke Rand alle 6 Millisekunden um einen Pixel addiert wird.

```
inputBar = setInterval(() => {
  marginleftRT = marginleftRT + 1;
}, 6);
```

Abbildung 32. Intervallmethode zum Addieren des linken Rand der gelben Punkte.

Die CSS *position* Attribute spielt auch hier eine wichtige Rolle (siehe Abbildung 33 unten).

```
▼<div id="shadow-bar" class>
  ▼<div id="sb-parent" style="display: flex;flex-direction: row;">
     <div id="answerRT" class="answerRtClass" style="left: 19px;"></div>
     <div id="answerRT" class="answerRtClass" style="left: 144px;"></div>
     <div id="answerRT" class="answerRtClass" style="left: 266px;"></div>
     <div id="answerRT" class="answerRtClass" style="left: 405px;"></div>
   </div>
 </div>
```

Abbildung 33. HTML-Struktur von dem grauen Balken.

Der graue Balken ist ein DIV-Element mit dem ID *shadow-bar*, während das DIV-Element mit der ID *sb-parent* ein Container für die gelben Punkte mit der ID *answerRT* ist. Der *shadow-bar* hat das CSS-Positionsattribut mit dem Wert von „relative“, was bedeutet, dass dieses Element relativ zu seiner normalen Position platziert wird. Innerhalb *shadow-bar* ist *sb-parent* Container als „Flexbox“-Container gestaltet, dessen untergeordnete Elemente in einer Zeilenposition angezeigt werden. Die untergeordneten Komponenten *answerRT* sind absolut zu ihrem zuerst positionierten übergeordneten Element, *shadow-bar*, gesetzt. Dadurch wird der linke Rand der gelben Punkte beginnend am linken Rand von *shadow*-bar gezählt. Die Abbildung 34 stellt die UI für den grauen Balken dar.

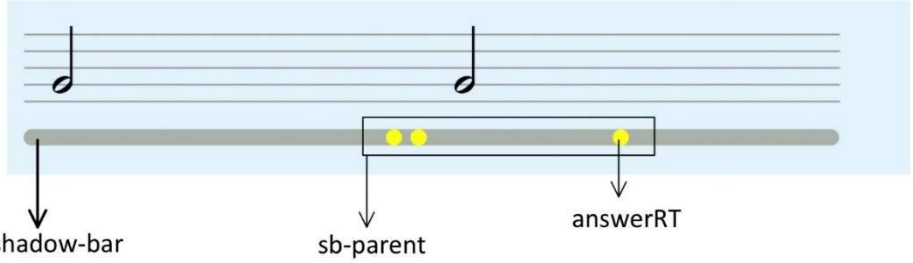

Abbildung 34. Erklärung über den grauen Balken und dessen „Container“.

Mit dem „Hint“-Button wissen die Benutzer, wo genau die richtigen Stellen sind. Der Hinweis ist eine Array-Variable, die zwei bis vier Zahlen in einer Zeichenfolge enthält. Diese Zahlen sind der linke Rand jedes grünen Punktes. Beispielsweise bedeutet die erste Zeichenfolge des Arrays *solutionKeyRT* (in der Abbildung 35), „25.265“, dass diese Übung zwei Noten enthält, wobei der erste Punkt 25 Pixel und der zweite Punkt 265 Pixel von der linken Seite des grauen Balkens entfernt ist.

```
let solutionKeyRT = [
  "25,265",
  "25,105,315,430",
  "25,150,272",
  "25,135,245,320,432",
  "25,140,215,295"
];
```

Abbildung 35. Eine Variable für die Lösung.

Die Abbildung 36 zeigt den Zustand, wenn der Benutzer nach seiner Eingabe auf den „Hint“-Button anklickt.

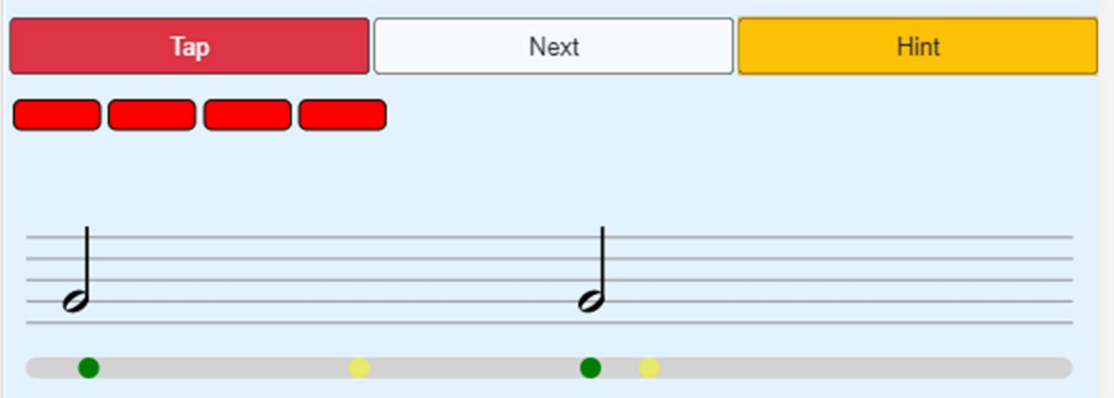

Abbildung 36. Grüne Punkte als die Lösung.

Am Ende der Übung wird die Meldung „Exercise Done.“ angezeigt. Die Benutzer können es erneut versuchen, indem sie auf den Button "Rhythm Tapping" klicken, und die Anwendung wird neu gestartet.

2.7.4. Anwendung - Interval Identification

Benutzer müssen das von der Anwendung gespielte Intervall identifizieren und das richtige Intervall aus den angezeigten Optionen auswählen: „Unison“, „Minor Second“ und „Major Second“.

Die Anwendung „Interval Identification“ verfügt über ein Piano-Element, das als Hinweis für die Anfänger verwendet werden kann. Dieses Feature wurde ebenfalls in Angular entwickelt. Die Entwicklung dieses Features in VanillaJS und Angular wird am Ende des Kapitels Angular verglichen. Die Abbildung 37 ist die UI von dieser Anwendung.

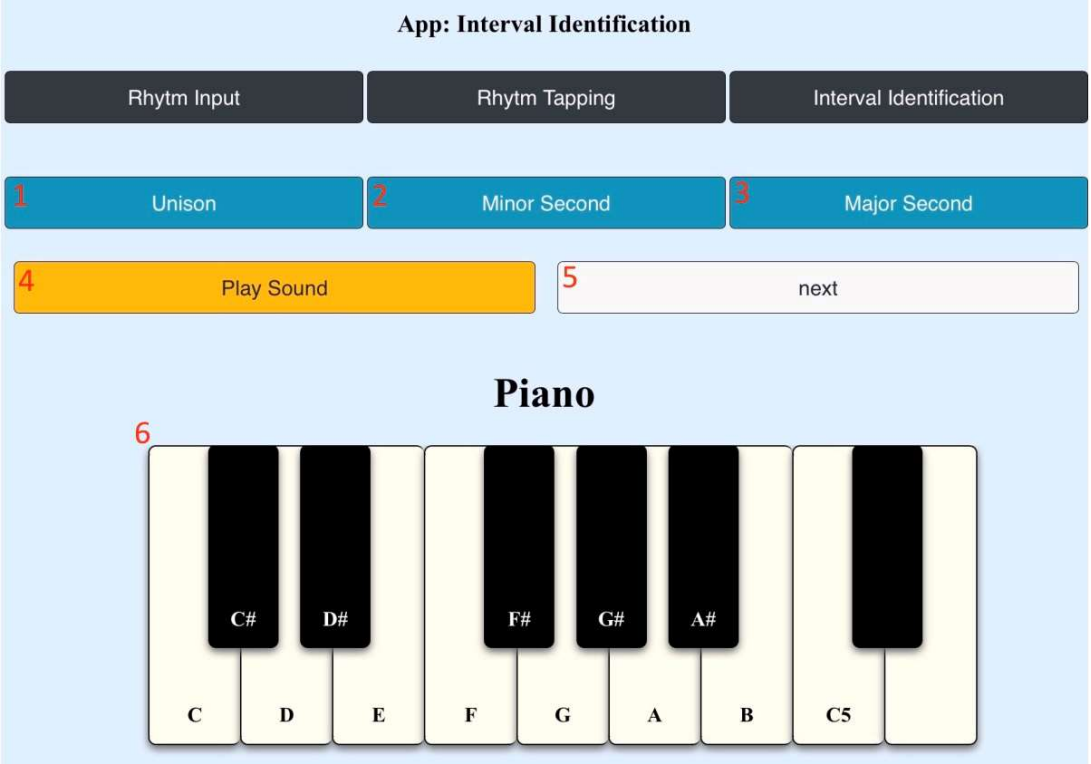

Abbildung 37. „Interval Identification“ UI.

1. „Unison“-Button: Ein Optionsbutton für die Benutzereingabe. Unison bedeutet 2 Töne mit der gleichen Tonhöhe.

2. “Minor Second“-Button: Minor is also called a half step or halftone. It is defined as the interval between two adjacent notes in a 12-tone scale. For

example, C is adjacent to C♯; the interval between them is a semitone or minor second [23].

3. “Major Second”-Button: Major second is also called a whole note. It is a musical interval encompassing two adjacent staff positions. For example, the interval from C to D [24].

4. „Play Sound“-Button: Diese Taste spielt den Wiederholungston ab oder wiederholt ihn.

5. „Next“-Button: Weiter zur nächsten Aufgabe.

6. Klavier: Ist eine Note auf der Taste abgebildet so wird beim Klicken, der entsprechende Ton abgespielt. Ansonsten wird kein Ton gespielt.

Die Übung beginnt, wenn die Benutzer auf den Button „Play Sound“ klicken. Nach dem Klicken hören die Benutzer zwei Musiktöne und wählen dann eine von drei verfügbaren Optionen aus. Über den Optionen-Knopf erscheint ein Hinweis, der anzeigt, was die erste von der Anwendung gespielte Note ist. Das Klavier unten auf der Seite kann ebenfalls verwendet werden, indem man auf die Taste klickt. Die letzten beiden Tasten spielen keinen Ton, da das Intervall nur im Bereich C4 - C5 liegt. Nachdem der Benutzer auf eine der Optionen geklickt hat, wird die Eingabe des Benutzers mit der Lösung verglichen. Wenn die Benutzer die Übung richtig beantworten, ändert die angeklickte Taste ihre Hintergrundfarbe in Grün, andernfalls ändert sie sich in Rot und den Knopf mit der richtigen Antwort wird in Grün gefärbt (siehe Abbildung 38 und *ff.* 39).

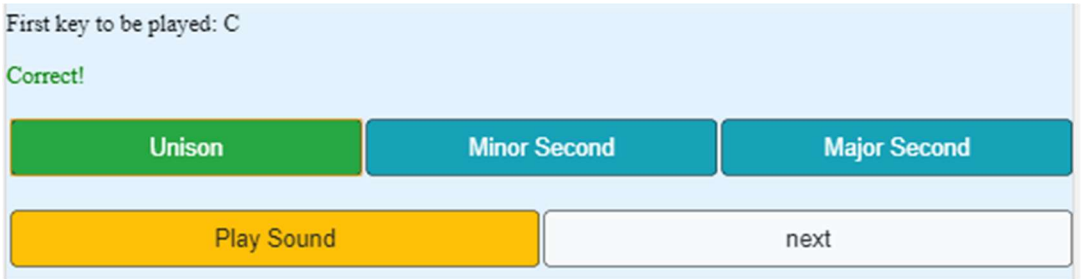

Abbildung 38. UI für richtige Eingabe.

[23] Semitone (*o.J.*)
[24] Major Second (*o.J.*)

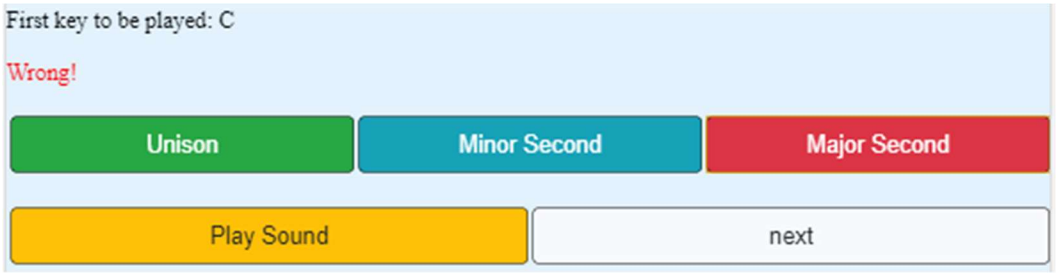

Abbildung 39. UI für falsche Eingabe.

Nachdem die Benutzer auf den „Next“-Button geklickt haben, wird eine neue Aufgabe geladen. Der Hinweis ändert sich entsprechend und alle Eingabeknöpfe werden in ihren Ausgangszustand mit blauer Hintergrundfarbe zurückgesetzt (siehe Abbildung 40).

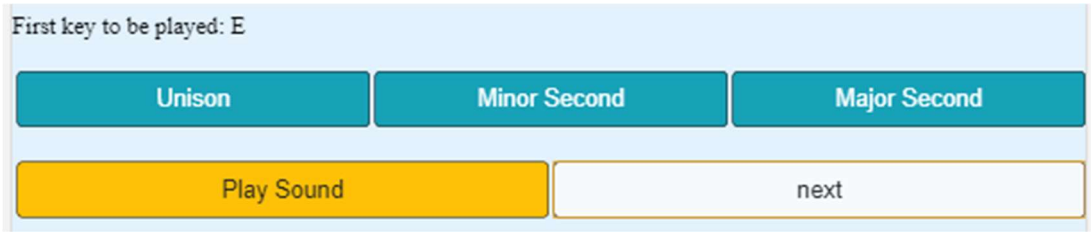

Abbildung 40. Neue Aufgabe nach dem Klicken auf den „Next“-Button.

In dieser Anwendung wird eine Array-Variable verwendet, um die Lösung zu speichern, den Ton für die Übungen abzuspielen und den Benutzern einen Hinweis darauf zu geben, welche Note als erste gespielt wird. Unten ist die Variable mit ihren Eigenschaften s, q und startKey. s steht für die Lösung (engl. solution), q für die Frage (engl. question) und startKey ist der Hinweis (siehe Abbildung 41).

```
let intervalAppDb = [
    { s: "unison", q: [0, 0], startKey: "C" },
    { s: "major", q: [2, 3], startKey: "E" },
    { s: "major", q: [7, 12], startKey: "A#" },
    { s: "minor", q: [9, 1], startKey: "D" },
    { s: "unison", q: [2, 2], startKey: "E" },
    { s: "major", q: [7, 6], startKey: "B" },
    { s: "minor", q: [4, 11], startKey: "G" }
];
```
Abbildung 41. Array Variable *intervalAppDb*.

Die Attribute q enthält ein Array von Zahlen und diese Zahlen sind die Werte jedes *data-note*-Attributs auf jeder Taste des Klaviers. „The dataset read-only property of the HTMLOrForeignElement interface provides read/write access to all the custom

data attributes (data-*) set on the element" [25]. Das bedeutet, die Verwendung des *data*-Attributs gibt Entwicklern die Möglichkeit, ein benutzerdefiniertes *data*-* Attribut zu haben, dessen Wert manipuliert werden kann. Unten ist ein Snippet-Code für die weiße Taste C und die schwarze Taste C# (siehe Abbildung 42).

```
<ul id="piano" data-note="100">
  <li data-note="0" class="key">
    <div data-note="8" class="black-key">C#</div>
    C
  </li>
</ul>
```

Abbildung 42. HTML-Elemente für das Klavier.

Das Element mit dem ID-*piano* ist ein Container für das Klavier, während das *UL*- und *LI*-Element die Tasten des Klaviers sind. Auf der Abbildung 43 ist eine JavaScript addEventListener-Methode an das Klavier-Element angehängt, die die Callback-Funktion jedes Mal aufruft, wenn der Benutzer ein Klick-Event auslöst. (mouse down).

```
let pianoPlayer = document.getElementById("piano");
pianoPlayer.addEventListener("mousedown", e => {
  if (e.target.dataset.note !== "100") {
    let playSound = new Audio(audioFiles[e.target.dataset.note]);
    playSound.play();
  }
});
```

Abbildung 43. Callback-Funktion für das Klavier

Die If-Bedingung prüft zunächst, ob der Benutzer auf die Klaviertasten klickt und dann einen Ton abspielt, der dem Wert des *data-note*-Attributs entspricht. Der Wert des benutzerdefinierten Attributs *data-note* bezieht sich auf einen Index der Array-Variablen audioFiles. Wenn Benutzer beispielsweise auf die Taste E klicken, die den Wert 2 für das Attribut *data-note* hat, spielt das audioFiles[2] eine E-Note ab. Der Index des Arrays audioFiles wird so angepasst, dass er die gleiche Reihenfolge wie der Wert des *data-note*-Attributs hat (*ff.* siehe Abbildung 44).

[25] MDN Contributors (2011)

Abbildung 44. *data-note* Attribute und Index von *audioFiles.*

Nachdem alle Übungen beantwortet wurden, wird eine Meldung „Exercise Done." und die Punktzahl der Benutzer angezeigt. Alle Eingabebuttons werden durch das Nachrichtenelement ersetzt, und die Übung beginnt von neuem, wenn die Benutzer auf den Button „Interval Identification" klicken.

2.8. Anwendung – Angular

Dieses Kapitel behandelt die Entwicklung der Anwendung „Interval Identification" in Angular, die die gleiche Funktionalität, UI und BusinessLogic hat.

2.8.1. Einführung des Frameworks Angular

Angular ist ein von Google entwickeltes Framework für die Frontend-Entwicklung, das zur Erstellung einer dynamischen Webanwendung verwendet wird. „This framework is component-based, which means, Angular allows developers to build web applications out of components that neatly encapsulate all the style and function required for a certain feature to work" [26]. Der Vorteil eines komponentenbasierten Ansatzes liegt in der Wiederverwendbarkeit, wenn die gleiche Funktionalität an mehreren Stellen in der Anwendung bereitgestellt werden muss.

[26] Yoshitaka, Shiotsu (2017)

2.8.2. TypeScript und JavaScript

Angular ist vollständig in TypeScript entwickelt worden, und für Entwickler, die eine Angular-Anwendung erstellen möchten, ist es erforderlich, TypeScript zu verwenden. Eines der beiden Hauptziele von TypeScript ist die Bereitstellung eines optionalen Schriftsystems für JavaScript, da die Typen nachweislich die Qualität und Verständlichkeit des Codes verbessern können [27]. Das bedeutet, dass die Entwicklung in TypeScript den Entwicklern hilft, Fehler durch ein Typensystem abzufangen. Wenn TypeScript-Dateien (Dateien mit der Erweiterung .ts) in Browsern kompiliert werden, werden die Dateien als Ausgabe in Standard-JavaScript-Dateien (mit der Erweiterung .js) konvertiert. Der Compiler wird als „Transpiler“ bezeichnet und dient dazu, eine Programmiersprache in eine andere zu konvertieren. Unten ist ein Beispiel für die Deklaration von vier Variablen in TypeScript (Abbildung 45). Im Unterkapitel DOM-Manipulation in Angular wurden diese Variablen als Referenzen auf die DOM-Elemente verwendet.

```
inputUntouched: boolean = true;
unisonValue: boolean;
minorValue: boolean;
majorValue: boolean;
```

Abbildung 45. Variablen in Angular.

2.8.3. Bibliotheken und Styling

Das Styling dieser Anwendung wird von „Bootstrap 4“ übernommen, einem CSS-Framework zur Entwicklung von Responsive Design- und Mobile-First-Projekten. Um den Ton des Klaviers und der Übungen zu spielen, verwendet diese Anwendung eine leichte JavaScript-Audiobibliothek namens „BuzzJS“. Dieser Code unten erzeugt eine neue Soundinstanz, die in der Variable sound1 deklariert wird. Danach lädt die zweite Zeile den Ton und beginnt mit der Wiedergabe (Abbildung 46)

```
let sound1 = new buzz.sound(URL);
sound1.play();
```

Abbildung 46. Tonwidergabe mit der JavaScript-Bibliothek „BuzzJS“.

[27] Vgl. Syed, Basarat Ali (2017), S.10

2.8.4. Angular CLI

Angular verfügt über eine Command Line Interface (CLI) zum Erstellen von Angular-Anwendungen. Mit Hilfe der CLI müssen Entwickler keine Zeit für die Installation und Konfiguration aller erforderlichen Abhängigkeiten und die Verkabelung aller Komponenten aufwenden. Beispielsweise wird mit dem Befehl „*ng generate component component_name“* eine Komponente erstellt. Entwickler müssen die HTML-Datei nicht mit den CSS- und TypeScript-Dateien referenzieren. Dies wird automatisch von Angular erledigt.

2.8.5. Projektstruktur

Dieses Projekt enthält 4 Komponenten: Header, „Intervall App, „Rhythm Input“ und „Rhythm Tapping“. Diese vier Unterordner sind alle benötigten Komponenten (siehe Abbildung 47). Jede Komponente wurde mit der Angular CLI erstellt.

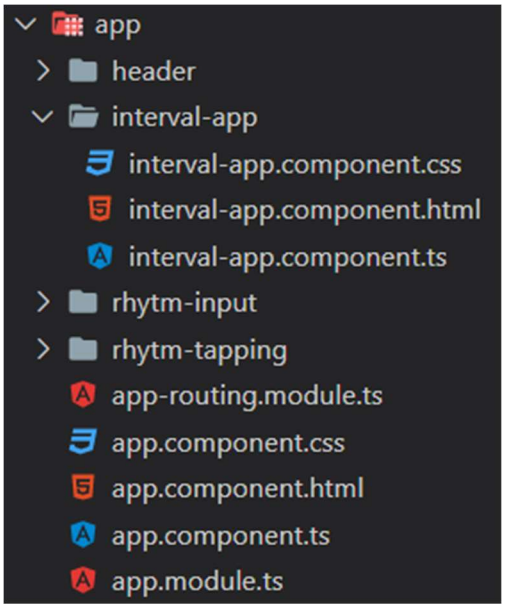

Abbildung 47. Projektstruktur der Anwendung in Angular.

Entwickler können die Komponenten auch manuell generieren. Die manuell erzeugten Komponenten müssen in die Datei **app.module.ts** importiert werden (*ff.* siehe Abbildung 48).

```
import { RhytmInputComponent } from "./rhytm-input/rhytm-input.component";
import { RhytmTappingComponent } from "./rhytm-tapping/rhytm-tapping.component";
import { IntervalAppComponent } from "./interval-app/interval-app.component";
import { HeaderComponent } from "./header/header.component";

@NgModule({
  declarations: [
    AppComponent,
    RhytmInputComponent,
    RhytmTappingComponent,
    IntervalAppComponent,
    HeaderComponent
  ],
  imports: [BrowserModule, AppRoutingModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Abbildung 48. **app.module.ts** Datei.

2.8.6. Angular – Interval Identification

Die Abbildung 49 zeigt die Benutzeroberfläche der Anwendung, die in Angular entwickelt wurde.

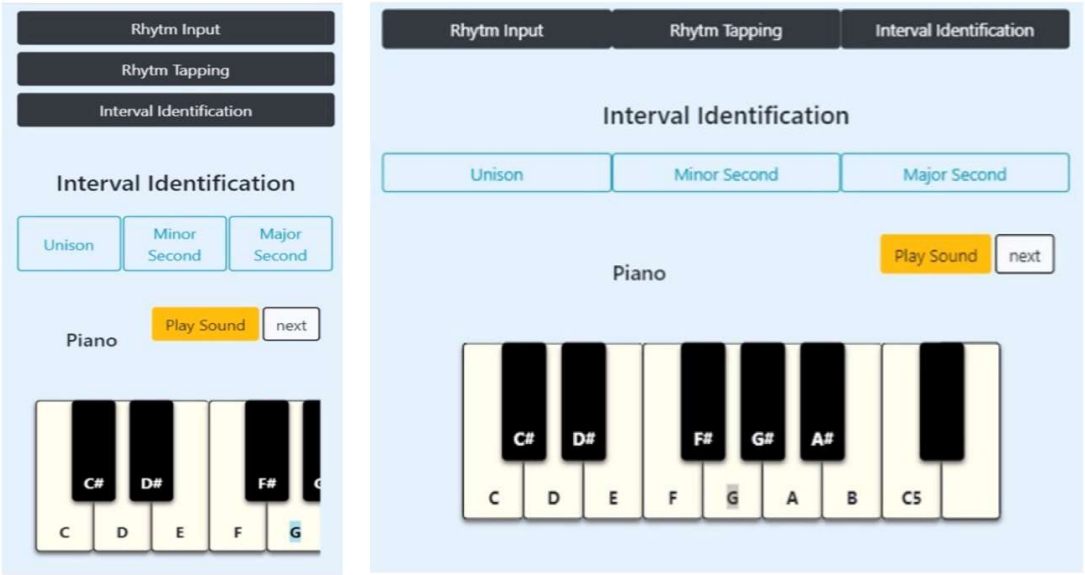

Abbildung 49. UI auf Mobiltelephone (links) und Laptops bzw. Tables (rechts)

2.8.7. Angular - Header

Der Header der Anwendung wird zum Navigieren verwendet. Beim Klicken auf einen Anwendungsoptionsbutton ändert sich die URL und Angular fügt die Komponente ein, die basierend auf der URL des aktuellen Browsers abgeglichen wird. All dies wird in der Datei **app.routing.module.ts** konfiguriert (siehe Abbildung 50).

```
const routes: Routes = [
  { path: "", redirectTo: "rhytm-input", pathMatch: "full" },
  { path: "rhytm-input", component: RhytmInputComponent },
  { path: "rhytm-tapping", component: RhytmTappingComponent },
  { path: "interval", component: IntervalAppComponent }
];
```

Abbildung 50. Routing in Angular-Anwendung.

Das eingebaute *routerLink*-Attribut (Direktiven in Angular genannt) zeigt auf den angegebenen Link der Anwendung. Von der Abbildung unten ist zu erkennen, dass jeder Knopf seinen eigenen Link hat (siehe Abbildung 51).

```
<div class="container">
  <div class="row">
     <button class="btn btn-dark col-4" routerLink="/rhytm-input">Rhytm Input</button>
     <button class="btn btn-dark col-4" routerLink="/rhytm-tapping">Rhytm Tapping</button>
     <button class="btn btn-dark col-4" routerLink="/interval">Interval Identification</button>
  </div>
</div>
```

Abbildung 51. HTML für die Optionsbuttons.

Wenn Benutzer beispielsweise auf den Button „Interval Identification“ klicken, ändert sich die URL in http://localhost:4200/interval und die IntervalAppComponent wird auf die Seite geladen.

2.8.8. Responsives Design mit Bootstrap

Die Komponente der „Interval Identification“ verwendet eine Reihe von Containern, Zeilen und Spalten, um den Inhalt zu gestalten und auszurichten. Die Klasse *container* passt den Rand des Inhalts automatisch an, um ihn in der Mitte der Seite zu platzieren. Die Klasse *row* wird als ein Container für Spalten verwendet. In diesem Projekt wird die Klasse *row* in einem DIV-Element für die Unison-, Minor Second und Major Second-Knopfoptionen verwendet. Jede Zeile hat 12 Spalten und jede Spalte gibt in Kombination mit einer Zahl die Anzahl der zu verwendenden Spalten an (*ff.* siehe Abbildung 52).

```
<div class="container">
  <div class="row">
    <button class="btn col-4">Unison</button>
    <button class="btn col-4">Minor Second</button>
    <button class="btn col-4">Major Second</button>
  </div>
</div>
```
Abbildung 52. HTML Elemente für die Eingabebuttons mit „Boostrap 4“.

Alle Buttons haben die Klasse *col-4*, was bedeutet, dass jeder Knopf bei Geräten mit einer Bildschirmbreite von weniger als 576 Pixeln die Breite von vier Spalten hat. Die Regel für das kleinste Gerät wendet diese Regel für alle anderen größeren Geräte an, da keine andere Regel definiert ist.

2.8.9. DOM Manipulation in Angular

In Angular können Entwickler mit Hilfe einer Attribut-Direktive das Aussehen oder Verhalten eines DOM-Elements ändern. Zum Ändern der Hintergrundfarbe der Eingabeknöpfe, nachdem der Benutzer daraufgeklickt hat, wird eine Attribut-Direktive namens *NgClass* relevant, da sie zum Hinzufügen und Entfernen von CSS-Klassen auf einem HTML-Element verwendet wird (siehe Abbildung 53).

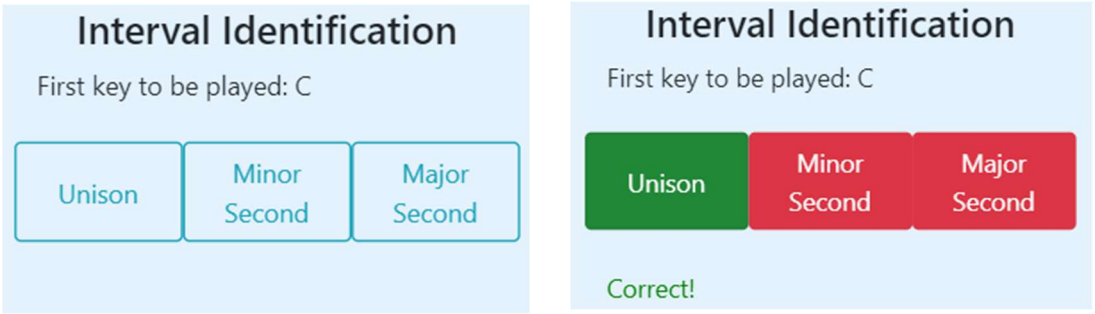

Abbildung 53. Zustand vor und nach der Benutzereingabe.

Unten ist ein vereinfachter Code aus dem Projekt mit der *NgClass* auf der Unison-Optionsknopf zu finden:

```
<button class="btn col-4"
       [ngClass]="{ 'btn-outline-info':inputUntouched==true,
       'btn-danger':unisonValue==false && inputUntouched==false,
       'btn-success':unisonValue==true }">
    Unison
</button>
```
Abbildung 54. *NgClass* für die Optionbuttons.

Der Unison-Button hat den Klassenattributwert *btn*, der für das Styling des Buttons mit „Bootstrap 4“ vordefiniert ist, und das Hinzufügen einer *btn*-*-Klasse definiert die Hintergrundfarbe des Button-Elements. *NgClass* fügt eine Klasse für den Button hinzu, die von dem Wert der booleschen Variablen inputUntouched und/oder unisonValue abhängt. Die beiden Variablen sind in der Datei **interval-app.component.ts** deklariert. Diese Variablen verwalten den Zustand der Benutzerinteraktion, wobei inputUntouched = true bedeutet, dass die Benutzer mit keinen Optionsknöpfen (Klick) interagiert haben. Die Klasse *btn-danger* ändert die Hintergrundfarbe auf rot, wenn die Benutzereingabe nicht mit der Lösung übereinstimmt (unisonValue = false). Andernfalls wird die Hintergrundfarbe auf grün geändert, was signalisiert, dass die Benutzereingabe korrekt ist und die Klasse *btn-success* angewendet wird.

Eine weitere Attribut-Direktive, die verwendet wird, ist *NgIf*. Dieses Attribut wird verwendet, um Teile der Anwendung basierend auf einer Bedingung ein- oder auszublenden. Unten ist die Abbildung für den Zusammenfassungsteil, der den Bereich der Buttons ersetzt, wenn die Benutzer mit den Übungen fertig sind.

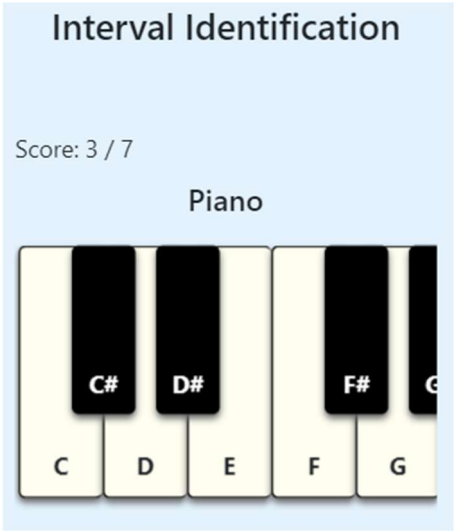

Abbildung 55. Zusammenfassungseite.

Das erste DIV ist ein Element, in dem alle Optionsknöpfe platziert sind, und das zweite DIV umschließt die Aktionsknöpfe für „Play Sound“ und „Next“. Das zweite Element ist so gestaltet, dass es immer auf der rechten Seite der Seite schwebt. Das letzte Element ist ein Paragraph-Element für den Zusammenfassungsteil. Wann immer die Übung beendet wird, ändert sich der Wert von showSummary Variable auf

true und löst Angular aus, um die DIV-Elemente auszublenden und das Absatzelement anzuzeigen (siehe Abbildung 56).

```
<div *ngIf="!showSummary"></div>
<div *ngIf="!showSummary"></div>
<p *ngIf="showSummary">
   Score: {{trueAnswer}} / {{ intervalAppDb.length}}
</p>
```

Abbildung 56. Nutzung von *NgIf.*

3. Schlussfolgerung

3.1. Vergleich zwischen Angular und VanillaJS für PWA

Die erste Anwendung, die nur VanillaJS verwendet, wurde absichtlich so entwickelt, dass keine Bibliothek von Drittanbietern (mit Ausnahme von „VexFlow“ und „EasyScore“) verwendet werden musste, während in Angular jede beliebige Bibliothek verwendet werden konnte. Bei diesem Aspekt gibt es keinen signifikanten Unterschied zwischen Angular und VanillaJS. Das einzige, was Zeit kostete, war die Anwendung einer „Media Query“ für das responsive Design für die VanillaJS Anwendung, während sich „Bootstrap 4“ um das responsives Design der Angular-Anwendung kümmerte.

Die Verwendung von Angular für den Aufbau einer progressiven Web-Anwendung, die viele Zustände hat, ist einfacher. Dies wird deutlicher, wenn es um die DOM-Manipulation geht. Die ausschließliche Verwendung von VanillaJS bedeutete, dass jedes Element nach seiner Verwendung wiederholt gestylt werden musste, und dies konnte durch Hinzufügen und Entfernen der CSS-Regel erreicht werden, die die Anzeige des Elements gestaltete. Angular behielt dies mit seinen eingebauten Attribut-Direktiven *NgIf* bei. Der Ansatz mit *NgIf* erleichterte die Entwicklung, da dieses Attribut nur einmal auf HTML geschrieben wurde. Die Änderung der Hintergrundfarbe der Knöpfe in der Anwendung „Interval Identification“ war ebenfalls einfacher, da *NgClass*, das mit der eingebetteten *NgIf*-Funktion arbeitet, verwendet wurde.

Durch die Routing-Funktion und die komponentenbasierte Idee nahm der Entwicklungsprozess in Angular auch weniger Zeit in Anspruch, so dass sich die Entwickler auf die Logik der Anwendung konzentrieren konnten. In Bezug auf die Anwendungsgröße wäre die Anwendung mit nur VanillaJS schwieriger zu updaten, wenn eine neue Funktion hinzugefügt wird. In Angular konnte dies mühelos erreicht werden, indem eine neue Komponente über die Angular-CLI erstellt und die Route registriert wurde.

Der letzte Punkt ist das Entwurfsmuster für beide Anwendungen. Bei der Anwendung in VanillaJS würde sich die Entwicklungskomplexität erhöhen, wenn ein Entwurfsmuster verwendet würde. Es gibt einige Vor- und Nachteile bei der Verwendung eines Entwurfsmusters. Die Vorteile wie das einfache Unit-Testing und die Wiederverwendbarkeit des Codes können durch seine Nachteile überwunden werden. Diese Anwendung hätte jedes beliebige MV*-Muster verwenden können, aber da keine Kommunikation mit der Model-Schicht erforderlich war (durch API-Aufrufe oder das Abrufen von Daten aus der Datenbank), würde die Verwendung von Entwurfsmustern die Entwicklung erschweren, wobei ein weiterer Punkt zu erwähnen ist, dass das Entwurfsmuster nicht für eine einfache und kleine Anwendung geeignet ist. In der offiziellen Dokumentation von Angular wird nicht explizit erwähnt, welches Entwurfsmuster verwendet wird. Es gibt auch eine Menge Argumente, ob Angular standardmäßig MVC oder MVP verwendet. Die Festlegung, welches Muster mit Angular übereinstimmt, hängt davon ab, wie die Daten und Events behandelt werden. Zum Beispiel bei einem Projekt, bei dem ein Dienst für die Verwaltung der Daten und die Durchführung von „Create – Read – Delete – Update“ (CRUD) Operationen in einer Datenbank verwendet wird, verwendet Angular das MVC-Muster.

3.2. Verbesserungsvorschläge

Obwohl die Anwendung „RPro“ in VanillaJS die gesamte Funktionalität einer Musik-Lernanwendung abdeckt, können bestimmte Verbesserungen dem Benutzer eine bessere UserExperience (UX) verschaffen, insbesondere für Fachleute. Die Implementierung aller Funktionen der Angular-Anwendung wird ebenfalls empfohlen. Im

Folgenden sind einige der Funktionen aufgeführt, die in Zukunft verbessert werden können:

1. Beim „Rhythm Taping“ könnte der visuelle Effekt und der Metronom Ton besser synchronisiert werden. Für Anfänger oder Leute ohne Musikhintergrund ist dies kein Problem. Für Fachleute ist es jedoch leicht zu erkennen, dass zwischen der Farbwechselanimation und dem Metronom Ton eine leichte Verzögerung besteht.

2. Bei der Anwendung „Rhythm Input“, wenn die Achtelnoten in 2er-Sätzen zu einem Schlag zusammengefügt werden, sollte die Note zusammen gestrahlt werden, anstatt die Flaggen der einzelnen Noten zu zeigen.

3. Mehr Unterstützung für andere Geräte, insbesondere in Safari und Samsung Internet, da beide Browser einen großen Marktanteil haben, wenn es um den Marktanteil der mobilen Browser geht.

4. Eine Anmelde- und Registrierungsfunktion. Mit dieser Funktion können alle Benutzerstatistiken in einer Datenbank gespeichert werden, so dass die Benutzer bei jeder Sitzung ihre Verbesserungen verfolgen können. Eine weitere Funktion, die für die Benutzer hilfreich wäre, ist es, den Benutzern die Möglichkeit zu geben, ihre eigene Übung zu erstellen und in der Datenbank zu speichern.

3.3. Schlussfolgerung

Diese Arbeit zielt darauf ab, eine installierbare PWA zu entwickeln, die eine Alternative für native Anwendungen sein kann. Die Entwicklung von PWAs reduziert die Kosten der Entwicklung und bietet gleichzeitig die vollen Funktionen der nativen Anwendungen. Nach einem Vergleich der Entwicklung in VanillaJS und Angular kann man zu dem Schluss kommen, dass die Entwicklung einer PWA in Angular effektiver ist. Von der Verwendung einer CLI zur Generierung aller erforderlichen Komponenten bis hin zu eingebauten Direktiven, die zur Aufrechterhaltung des Anwendungsstatus entwickelt werden, verfügt Angular über alles, was für die

Erstellung einer progressiven Web-Anwendung mit besserer UserExperience erforderlich ist. Um jedoch eine PWA in Angular erstellen zu können, sind gute Kenntnisse von JavaScript erforderlich, und die Verwendung von keinem Framework kann ein gutes Lernwerkzeug sein, um sich in JavaScript zu vertiefen. Nach dem Verständnis des grundlegenden Konzepts von JavaScript tendieren Entwickler dazu, nicht nur hochwertigen Code zu erstellen, sondern sich auch schneller mit Frameworks und Bibliotheken vertraut zu machen. Das Framework wird sich ändern, aber die Haupttechnologie, JavaScript bleibt für immer.

4. Literaturverzeichnis

1. How Many Websites Are There Around The World? (2020) URL: https://www.millforbusiness.com/how-many-websites-are-there, 6.2.2020.
2. Ater, Tal (2017): Building Progressive Web Apps: Bringing The Power of Nativ to The Browser, 1. Auflage, Vereinigte Staaten von Amerika.
3. Basques, Kayce (*o.J.*): Why HTTPS matters, URL: https://developers.google.com/web/fundamentals/security/encrypt-in-transit/why-https, 17.2.2020.
4. Introduction to Service Worker (*o.D.*) URL: https://developers.google.com/web/ilt/pwa/introduction-to-service-worker, 24.2.2020.
5. HTML Introduction (*o.D.*) URL: https://www.w3schools.com/html/html_intro.asp, 8.2.2020.
6. MDN Contributors (2019): What is CSS?, URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS, 8.2.2020.
7. Schafer, Steven M. (2010): HTML, XHTML, and CSS, 5. Auflage, Vereinigte Staaten von Amerika.
8. Jin, Brenda / Sahni, Saurabh / Shevat, Amir (2018): Designing Web APIs, 1. Auflage, Vereinigte Staaten von Amerika.
9. MDN Contributors (2016): Introduction to Web APIs, URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction, 17.2.2020.
10. MDN Contributors (2017): Third-Party APIs, URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Third_party_APIs, 17.2.2020.
11. Design Patterns (*o.D.*) URL: https://sourcemaking.com/design_patterns, 24.2.2020.
12. Osmani, Addy (2012): Learning JavaScript Design Patterns, 1. Auflage. *o.O.*
13. Osmani, Addy (2012): Learning JavaScript Design Patterns, 1. Auflage. *o.O.*
14. Saleh, Hazem (2013): JavaScript Unit Testing, überarbeitete Auflage, *o.Ö.*
15. Osmani, Addy (2012): Learning JavaScript Design Patterns, 1. Auflage. *o.O.*
16. JavaScript Library (2019) URL: https://en.wikipedia.org/wiki/JavaScript_library, 18.2.2020.

17. MDN Contributors (2017): Third-Party APIs, URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API, 18.2.2020.
18. Peterson, Clarissa (2014): Learning Responsive Web Design, 1. Auflage, *o.O.*
19. CSS @media Rule (*o.J.*), URL: https://www.w3schools.com/cssref/css3_pr_mediaquery.asp, 18.2.2020.
20. Srivastava, Tejasvi (*o.J.*): what is the difference between “screen” and “only screen” in media queries?, URL: https://www.geeksforgeeks.org/what-is-the-difference-between-screen-and-only-screen-in-media-queries/, 25.2.2020.
21. LePage, Pete (2020): What does it take to be installable, URL: https://web.dev/install-criteria/, 25.2.2020.
22. Caputa, Maciej(2018): A Few Tips That Will Make Your PWA on iOS Feel Like Native. URL: https://www.netguru.com/codestories/few-tips-that-will-make-your-pwa-on-ios-feel-like-native, 19.2.2020.
23. Semitone (*o.J.*), URL: https://en.wikipedia.org/wiki/Semitone, 20.2.2020
24. Major Second (*o.J.*), URL: https://en.wikipedia.org/wiki/Major_second, 20.2.2020.
25. MDN Contributors (2011), HTMLOrForegeinElement.dataset, URL: https://developer.mozilla.org/de/docs/Web/API/HTMLOrForeignElement/dataset, 20.2.2020.
26. Yoshitaka, Shiotsu (2017): The Future is Angular: An Introduction to Angular 2, URL: https://www.upwork.com/hiring/development/angular-2-framework/, 26.2.2020.
27. Syed, Basarat Ali (2017): TypeScript Deep Dive, *o.O.*

5. Abbildungsverzeichnis

Erklärung über die eigenständige Erstellung der Arbeit

Hiermit erkläre ich, dass ich die vorgelegte Arbeit mit dem Titel

Entwicklung einer progressiven Web-Anwendung am Beispiel eines Musiklernprogramms

selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit als solche und durch Angabe der Quelle gekennzeichnet habe. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Mir ist bewusst, dass die Hochschule für Technik und Wirtschaft Dresden Prüfungsarbeiten stichprobenartig mittels der Verwendung von Software zur Erkennung von Plagiaten überprüft.

Dresden, 2.3.2020 _____

Ort, Datum Unterschrift Student/Studentin