

Hochschule Karlsruhe für Wirtschaft und Technik

Fakultät	Informations- und Medienmanagement
Studiengang	Kommunikation und Medienmanagement Master
Veranstaltung	Media Engineering
Dozent	Professor Martin Schober
Semester	SS 2013



# musicate

## Die App zum Musiklernen

### Projektbericht

<b>Maximilian Kistner</b>	<b>42183</b>	<b>Kistner.Maximilian@gmail.com</b>
<b>Felicita Krupka</b>	<b>42244</b>	<b>felicita.krupka@gmail.com</b>

# Inhaltsverzeichnis

1 Einleitung.....	1
2 Projektbeginn.....	2
2.1 Gruppenfindung.....	2
2.2 Projektfindung.....	2
2.3 Themenfindung.....	2
2.4 Vergleich Projektstand.....	3
3 Zielgruppe.....	5
3.1 Soziodemografische Angaben.....	5
3.1.1 Größe der Zielgruppe.....	5
3.1.2 Alter und Geschlecht.....	5
3.2 Vorwissen.....	6
3.3 Lernmotivation.....	6
3.4 Lerndauer.....	6
3.5 Folgen für die Umsetzung.....	6
4 Der Vermittlungsgegenstand Musik.....	7
5 Didaktisches Konzept.....	8
5.1 Rahmenmodell der Didaktik.....	8
5.2 Didaktische Methode.....	9
5.2.1 Zielgruppenanpassung.....	9
5.2.2 Wissensanpassung.....	10
5.2.3 Anpassung an Medien.....	10
5.2.4 Bestimmung der didaktischen Methode.....	10
5.3 Strukturierung der Inhalte mithilfe einer Informationsarchitektur.....	11
5.3.1 Aufbau deklaratives Wissen.....	13
5.3.2 Aufbau prozeduralen Wissens.....	14
5.3.3 Motivation.....	14
5.3.4 Verschiedene Lernwege.....	14
5.4 Lerntheoretische Position.....	16
6 Übersicht Lerninhalte.....	17
7 Konzeption.....	19
7.1 Plattform der App.....	19
7.2 Funktionsumfang.....	20
7.3 Erfassung und Bereitstellung der Inhalte.....	20
7.4 Design.....	21
7.4.1 Analyse bestehender Apps.....	21
7.4.2 Erarbeitung des Designs.....	22
7.4.3 Cognitive Load Theorie.....	23
7.4.4 Entwicklung des Logos.....	24
7.5 Bedienkonzept.....	24
8 Implementierung.....	25
8.1 Inhaltserstellung.....	25
8.1.1 XML-Struktur.....	25
8.1.2 Transformation.....	29
8.1.3 Struktur eines musicate-Pakets.....	29
8.1.4 Aktueller Workflow zur Erstellung, Bereitstellung und Import einer Lektion.....	30
8.1.5 Weiterentwicklung.....	31

8.2 Umsetzung des Designs.....	31
8.3 Umsetzung der Funktionen.....	31
8.3.1 Generelles .....	31
8.3.2 Animationen.....	32
8.3.3 Dynamisches Speichern.....	33
8.3.4 Gesten.....	33
8.3.5 Download und Import .....	33
8.3.6 Inhaltsdarstellung.....	35
8.3.7 Navigation.....	35
8.3.8 Rhythmus-Übung.....	36
8.3.9 Multiple-Choice-Test.....	39
8.3.10 Hörbeispiele & Metronom.....	39
9 Schwierigkeiten.....	41
9.1 HTML-basierter Sampler & Metronom.....	41
9.2 HTML-basierte Darstellung der Notation.....	41
9.3 Debugging.....	41
10 Weiterentwicklung & Zukunftsaussichten.....	43
10.1 Tests auf weiteren Geräten.....	43
10.2 Usability-Tests.....	43
10.3 Umsetzung auf anderen Plattformen.....	43
10.4 Zwischenspeichern von Fortschritten.....	44
10.5 Ausbau der Übungen.....	44
10.6 Ausbau Test.....	44
11 Fazit.....	45
12 Literaturverzeichnis.....	46
Anhang.....	I
UI-Sketches.....	I
Screenshots.....	V

## Tabellenverzeichnis

Tabelle 1: Vergleich des Projektstandes.....	3
Tabelle 2: Entscheidungsstruktur von Interaktionsräumen, nach Kerres (2012:336).....	11
Tabelle 3: Topic Typen der Informationsarchitektur.....	12
Tabelle 4: Expositorische und explorative Elemente.....	15
Tabelle 5: Inhalte der Lektionen.....	18

## Abbildungsverzeichnis

Abbildung 1: Rahmenmodell der Didaktik, Abbildung nach Kerres (2012:195).....	8
Abbildung 2: Struktur der App.....	13
Abbildung 3: Design der oberen Leiste.....	21
Abbildung 4: Menü-Button und Zurück-Button.....	21
Abbildung 5: Seitenleiste.....	22
Abbildung 6: UI-Sketches für musicate.....	22
Abbildung 7: Musicate-Logo.....	24
Abbildung 8: Ordner-Struktur des musicate-Pakets.....	30
Abbildung 9: Import von Lektionen.....	33
Abbildung 10: Ein einfacher Rhythmus.....	37
Abbildung 11: Ein einfacher Rhythmus.....	38

Abbildung 12: Lektion laden.....	V
Abbildung 13: Startscreen.....	V
Abbildung 14: Lektionsübersicht.....	V
Abbildung 15: Lernziele.....	V
Abbildung 16: Überblick.....	VI
Abbildung 17: Menü.....	VI
Abbildung 18: Übung 1/3.....	VI
Abbildung 19: Übung 2/3.....	VI
Abbildung 20: Übung 3/3.....	VII
Abbildung 21: Testfrage.....	VII
Abbildung 22: Testantwort.....	VII
Abbildung 23: Auswertung.....	VII

# 1 Einleitung

Apps für Musik existieren viele – egal ob verschiedene Player mit Songtextanzeige und Webradio oder Tools, die das Frequenzspektrum von aufgenommenen Sounds analysieren. Zum Thema *Musiklernen* jedoch ist die Auswahl sehr gering. Momentan gibt es solche nur für das Iphone.

Deswegen war unser Ziel eine App zu diesem Thema für das Betriebssystem Android zu konzipieren und umzusetzen. In verschiedenen Phasen des Projektes beschäftigten wir uns nacheinander mit der gestalterischen und didaktischen Planung, der Ausarbeitung der Inhalte und der Implementierung der App.

Im Folgenden werden wir die Ergebnisse unserer Arbeit vorstellen. Dabei gehen wir sowohl auf recherchierte Methoden und Konzepte als auch auf die Umsetzung des Projektes ein.

Zuerst wird in Kapitel 2 die Entstehung unseres Projektes erläutert und die zu Beginn feststehenden Ziele und Eigenschaften des Projektes genannt. Dabei werden die im Exposé festgehaltenen Fakten mit den am Ende umgesetzten Projekt verglichen.

In Kapitel 3 wird die von uns adressierte Zielgruppe näher bestimmt und danach in Kapitel 4 werden die Eigenschaften unseres Lehrthemas Musik definiert.

In Kapitel 5 wird das didaktische Konzept erläutert. Aufbauend auf dem Rahmenmodell der Didaktik, wird die zu verwendende didaktische Methode bestimmt. Weiterhin wird die Umsetzung des didaktischen Konzeptes geschildert. Danach folgt in Kapitel 6 eine Übersicht der erstellten Lerninhalte.

In Kapitel 7 wird die Konzeption der für die technische Umsetzung wichtigen Aspekte erläutert. Danach wird in Kapitel 8 die Implementierung beschrieben. Dabei wird auf die Inhaltserstellung, die Umsetzung des Designs und die verschiedenen Funktionen eingegangen. In Kapitel 9 werden Schwierigkeiten, die während der Arbeit am Projekt auftraten, erläutert und die jeweiligen Lösungsansätze beschrieben. Kapitel 10 beschäftigt sich mit der Weiterentwicklung und Zukunftsaussichten der App.

## **2 Projektbeginn**

Im folgenden Kapitel werden die Entscheidungen, die zu dem von uns gewählten Projektthema geführt haben erläutert. Des Weiteren werden verschiedene Eigenschaften des Projektes zu Beginn und zu Abschluss des Projektes verglichen.

### **2.1 Gruppenfindung**

Wir nutzten die Möglichkeit in einer Gruppe zu arbeiten, da somit insgesamt ein größeres Projekt bearbeitet werden konnte. Zudem versprachen wir uns Vorteile daraus, unsere unterschiedlichen Kompetenzen und Interessen zu verbinden. Weiterhin schätzen wir die zusätzliche Motivation, die durch das gemeinschaftliche Arbeiten an einem Projekt entsteht.

### **2.2 Projektfindung**

Es war schnell klar, dass wir uns in mit dem Thema E-Learning beschäftigen wollten, da dieses unmittelbar an das Kompetenzfeld des technischen Redakteurs anknüpft. Beim Lernen mithilfe von elektronischen Medien, steht auf der Erstellungsseite jedoch mehr das didaktische Konzept des Angebots im Vordergrund als bei der Erstellung von Bedienungsanleitung. Da das Themengebiet E-Learning in unseren bisherigen Studien noch nie bzw. nur ansatzweise behandelt wurde, wollten wir uns dieses durch die Durchführung des entsprechenden Projektes selbst erarbeiten und vertiefen.

Weiterhin legten wir fest, dass das E-Learning-Angebot im Rahmen einer Mobile-App stattfinden sollte, da auch in diesem Bereich Interesse vorhanden war und wir eine selbstständige Erarbeitung und Umsetzung als Herausforderung empfanden.

### **2.3 Themenfindung**

Als nächstes mussten wir uns auf ein Thema bzw. Inhalte einigen, die wir in unserem E-Learning-Angebot vermitteln wollten. Unser gemeinsames Interesse) fiel auf den Themenbereich Musik und vor allem das Musizieren mit Instrumenten.

Nach anfänglichen Überlegungen entschieden wir uns dafür, Musiktheorie, vor allem das Lesen von Noten, zum Hauptthema unseres E-Learning-Angebots zu machen. Denn diese Kompetenz sollte jeder beherrschen, der ein Instrument richtig erlernen möchte. Weitere Spezifizierungen und Ideen kamen ab da sehr schnell hinzu.

## 2.4 Vergleich Projektstand

Sowohl der inhaltliche Umfang als auch die technische Umsetzung änderten sich im Verlauf des Projektes. Im folgenden Abschnitt werden die verschiedenen Projekteigenschaften zu Beginn und zum Abschluss des Projektes gegenübergestellt.

	<b>Beginn (Exposé)</b>	<b>Ende</b>
Lehrziele	musikalisches Grundwissen Spezielles Wissen für Basisinstrumente (Gitarre, Schlagzeug, Klavier)	Musikalisches Grundwissen
Inhalt	Lektionen zur Musiktheorie Basislektionen und spezielle Übungen für Gitarre, Klavier und Schlagzeug	Basislektionen zur Musiktheorie
Extras	einfaches Stimmgerät, Akkorddatenbank	Nicht umgesetzt
Inhaltliche Strukturierung	Aufteilung in verschiedene Lektionen, aufeinander aufbauend auf Theorieblock folgt Übung	Aufteilung in verschiedene Lektionen, aufeinander aufbauend Informationsarchitektur mit topicstrukturierten Inhalten einzelne Informationseinheiten nach didaktischen Konzept gegliedert. Auf Lektion folgen Übungen und abschließender Test.
Name	Mapp (Music App)	Musicate (Music+Educate)
Design	Minimalistisches Interface → Aufmerksamkeit auf Inhalte Flatdesign	Minimalistisches Interface innerhalb der Lektionen. Menü/Navigation ansprechend gestaltet.
Umsetzung als	Web-App, Mobile-App	Mobile-App
Erfassung der Inhalte	Keine Überlegung	(Erfassung über Web-Oberfläche) in Xml-Struktur, Umwandlung per Xslt in Html
Erweiterbarkeit	Keine Überlegung	(Über Web-Oberfläche können von anderen Nutzern Lessons bereitgestellt werden.)

*Tabelle 1: Vergleich des Projektstandes*

Wie in der Tabelle zu sehen ist, hat sich das Projekt einiger Wandlung unterzogen. Die Grundidee Musiktheorie zu vermitteln blieb jedoch bestehen. Das Projekt musste leider im Umfang gekürzt werden, da die Aufbereitung der Inhalte, was sowohl die Erstellung der Texte, Midi-Dateien, Notationsbilder, Übungen und Tests umfasst, sehr viel Zeit in Anspruch nahm. Um am Ende ein qualitativ gutes Endprodukt zu erhalten, wurde auf die erweiterten Lektionen zu den Basisinstrumenten verzichtet.

Andere Projektpunkte entwickelten wir hingegen weiter. Die inhaltliche Strukturierung wurde durch das didaktische Konzept konkretisiert. Hinzu kam die Idee auch anderen Nutzern die Möglichkeit zu geben, Lessons zu erstellen und mithilfe unsere App zu verbreiten. Um sicherzustellen, dass die Lessons nach unserem didaktischen Konzept erstellt werden, legten wir eine Informationsarchitektur fest, die mithilfe verschiedener Topic-Typen eine genaue Struktur für das Erstellen einer Lesson vorgibt. Die Erfassung der Inhalte sollte über eine Weboberfläche erfolgen. Diese konnte jedoch leider am Ende des Projektes nicht mehr umgesetzt werden.

### **3 Zielgruppe**

Die Bestimmung der Zielgruppe ist vor allen Dingen in Bezug auf das zu entwickelnde didaktische Konzept des E-Learning-Angebots wichtig. Kerres (2012, 258ff) nennt dazu verschiedene Merkmale. Im folgenden Abschnitt werden alle relevanten Merkmale der Zielgruppe für die musicate-App bestimmt.

Leider konnte im Rahmen des Projektes keine genaue Zielgruppenanalyse mit Befragung von potentiellen Nutzern durchgeführt werden. Die folgenden Angaben setzen sich somit aus eigenen Erfahrungen und Annahmen sowie Statistiken zusammen.

#### **3.1 Soziodemografische Angaben**

##### **3.1.1 Größe der Zielgruppe**

Die Größe der Zielgruppe wird durch verschiedene Faktoren begrenzt. Da das E-Learning-Angebot durch eine App umgesetzt wird, beschränkt sich die Zielgruppe auf Smartphonennutzer. Laut statistia.de (statistia GmbH 1 2012) nutzen in Deutschland 31 Millionen Menschen ein Smartphone. Diese Anzahl reduziert sich jedoch, da die App momentan nur für Android entwickelt wird. Der Marktanteil an der Smartphone-Nutzung von Android im März 2013 beläuft sich ebenfalls laut statistia.de (statistia GmbH 2 2012) auf 58%. Unsere App könnte somit von 17980000 Menschen verwendet werden. Diese Anzahl wird jedoch weiter durch die Interessen der Smartphonennutzer begrenzt, da nur ein Nutzer mit Interesse an Musik bzw. dem Wunsch Musiktheorie zu erlernen, diese App nutzen würde, wobei die Entscheidung, dies mithilfe der App zu tun, wiederum beim ihm liegt.

Eine genaue Anzahl der potenziellen Nutzer kann somit nicht angegeben werden.

##### **3.1.2 Alter und Geschlecht**

Auch das Alter der Zielgruppe wird durch die Smartphonennutzer begrenzt/vorgegeben. Laut GfK (2012) liegen zwei Drittel der Smartphonennutzer in der Altersgruppe 20–49 Jahre. Jedoch wollen wir auch die mit einem geringeren Prozentanteil vertretende jüngere Altersgruppe 13–19 Jahre ansprechen. Weiterhin sollen auch ältere Altersgruppen nicht ausgeschlossen werden.

Es zählen Männer und Frauen zur Zielgruppe.

### **3.2 Vorwissen**

Die Zielgruppe hat hauptsächlich kein oder geringes musikalisches Vorwissen. Es handelt sich nach unserer Auffassung um drei verschiedene Typen.

1. kompletter Anfänger → Kein Vorwissen
2. Anfänger, der jedoch zur Musikschule geht → Geringes Vorwissen
3. fortgeschrittener Anfänger (Hat in der Vergangenheit begonnen ein Instrument zu erlernen, Wissen muss wieder aktiviert und ausgebaut werden)  
→ schlummerndes Vorwissen

### **3.3 Lernmotivation**

Kerres unterscheidet zwischen intrinsischer und extrinsischer Motivation. Die intrinsische Motivation ist durch das „Interesse am Lerngegenstand selbst oder [...] Spaß an der Beschäftigung mit dem Lerngegenstand“ (Kerres 2012:262) gekennzeichnet. Bei extrinsischer Motivation hingegen „stehen die Lernergebnisse im Vordergrund“ (Kerres 2012:262), der Lernende verfolgt also ein Ziel.

Auf die Zielgruppen trifft keine der Motivationen eindeutig zu. Vielmehr wird es Nutzer geben, die entweder intrinsisch oder extrinsisch motiviert sind sowie Nutzer, die aus beiden Motivationen profitieren.

### **3.4 Lerndauer**

Die App soll immer dann zum Einsatz kommen, wenn das Instrument nicht zur Hand ist, der Nutzer jedoch Zeit zum üben hat. Im Alltag sind das die Zeiten in denen der Nutzer nicht zu Hause bei seinem Instrument ist, sondern unterwegs. Typische Nutzungszeiten/szenarien sind Bahnfahrten oder auch Wartezeiten beim Arzt, Amt, etc. 10–15 Minuten stellen somit die minimale Lerndauer dar, die ein Nutzer mindestens zur Verfügung hat.

### **3.5 Folgen für die Umsetzung**

Die beschriebene Zielgruppe ist ziemlich weit. Sowohl in Alter, Vorwissen und Motivation gibt es eine hohe Bandbreite. Das E-Learning-Angebot muss daher sehr vielseitig gestaltet werden, um jeder Nutzergruppe gerecht zu werden und eine optimale Lernumgebung bereitzustellen. Diese wird vor allem durch das didaktische

Konzept bestimmt.

## **4 Der Vermittlungsgegenstand Musik**

Das zu vermittelnde Thema Musiklehre klingt relativ trocken, vor allem der von uns gewählte Abschnitt der Musiktheorie. Die Grundlage bildet *deklaratives Wissen*. Es werden Kenntnisse über die verschiedenen Notenwerte, Tonhöhen, das Notensystem etc. vermittelt. Oft wird dies als die Sprache der Musik bezeichnet. Um Noten lesen zu können und auf einem Instrument zu spielen, müssen diese Kenntnisse jedoch auch angewendet werden. Somit wird auch *prozedurales Wissen* benötigt. In der Musiklehre hängen diese beiden Wissensarten von einander ab. Erst wird das deklarative Wissen vermittelt, danach kann darauf aufbauend das prozedurale Wissen weitergegeben werden. (Vgl. Kerres 2012:119)

Um Musiklehre zu vermitteln, bedarf es somit nicht nur einer Anhäufung an Wissen, sondern auch einer Umsetzung dessen. Diese unterschiedlichen Wissensarten wirken sich ebenfalls auf das didaktische Konzept aus, da je nach Wissensart unterschiedliche Vermittlungsstrategien angewendet werden müssen. (ebd.)

## 5 Didaktisches Konzept

Im folgenden Kapitel wird die Erarbeitung des didaktischen Konzepts beschrieben. Ausgehend vom Rahmenmodell der Didaktik wird zuerst die passende didaktische Methode bestimmt. Danach wird die Strukturierung und der Aufbau App beschrieben und die spezifischen didaktischen Merkmale hervorgehoben. Zum Schluss wird das didaktische Konzept einer lerntheoretischen Position zugeordnet.

### 5.1 Rahmenmodell der Didaktik

Kerres (2012: 194) behandelt im Zuge der Planbarkeit von Lernangeboten das Rahmenmodell der Didaktik, das von Paul Heimann in den 1960er Jahren entwickelt wurde. Dieses Modell, das ursprünglich für die Konzeption von Schulunterricht entwickelt wurde, nennt „die zentralen Entscheidungsfelder der didaktischen Planung“ (Kerres 2012:192) und kann auch auf andere Lernangebote angewendet werden.

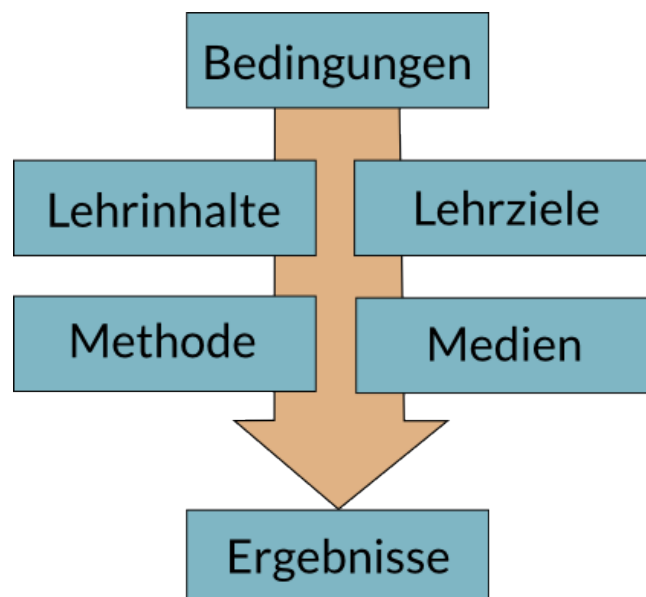


Abbildung 1: Rahmenmodell der Didaktik, Abbildung nach Kerres (2012:195)

Heimann unterscheidet in seinem Modell zwischen *Bedingungsfaktoren* und *Entscheidungsfaktoren*. Zu den *Bedingungsfaktoren* gehören Fakten über die Teilnehmer eines Lernangebots, wie Vorwissen, Interessen und Motivation, und andere Rahmenbedingungen, wie Ort und Zeit. (Vgl Kerres:194)

Die Bedingungsfaktoren und die Entscheidungsfaktoren beeinflussen sich gegenseitig. Eine Änderung eines Entscheidungsfaktors wirkt sich auch auf die anderen Entscheidungsfaktoren aus.

Im Rahmen des von uns zu konzipierenden E-Learning-Angebot sind einige Entscheidungsfaktoren und Bedingungsfaktoren bereits gesetzt:

Bedingungsfaktoren: Zielgruppe (Vorwissen, Motivation, Lerndauer)

Entscheidungsfaktoren: Lerninhalt, Lehrziele, Medien

Der einzige Entscheidungsfaktor, der noch nicht bestimmt wurde, ist die *Methode*. Diese bildet den zentralen Punkt im didaktischen Gesamtkonzept und kann nun in Abhängigkeit von den anderen Faktoren erarbeitet werden.

## **5.2 Didaktische Methode**

Im folgenden Abschnitt werden zuerst die Anpassungen an die verschiedenen Faktoren beschrieben. Auf Basis dieser Feststellungen wird dann die didaktische Methode bestimmt.

### **5.2.1 Zielgruppenanpassung**

Kerres (2012: 260) nennt als entscheidendes Kriterium für die Wahl der didaktischen Methode das Vorwissen der Zielgruppe. Bei wenig bis gar keinem Vorwissen

„bietet sich eine stärker strukturierte Variante der Darstellung und eine sequentielle Organisation der Lerninhalte entlang eines festgelegten Lernwegs an.“ (Kerres 2012:260)

Für Lernende, die ein größeres Vorwissen mitbringen, ist hingegen eine freie und offene Lernumgebung vorteilhaft, da bereits gelernte Inhalte übersprungen und in einer beliebigen Reihenfolge bearbeitet werden können. (Vgl Kerres 2012:260)

Auch die Motivation der Zielgruppe ist ausschlaggebend für die didaktische Methode. Vorwiegend intrinsisch motivierte Personen müssen ihre Lernwege selbst bestimmen können, wozu auch das Ablegen von Tests gehört. Extrinsisch motivierte Personen jedoch benötigen Motivation zu Beginn eines Lernangebots und müssen sich auf die Ziele konzentrieren. Da das Lernen als schwer empfunden wird, sollte der Inhalt gut gegliedert werden und Pausen ermöglichen. Weiterhin ist die Anzeige des Lernfortschritts sehr wichtig für extrinsisch motivierte Personen, da dies die Motivation steigert. (Vgl. Kerres 2012:262f)

### 5.2.2 Wissensanpassung

Auch die Entscheidungsfaktoren Lehrinhalte und Lehrziele sind ausschlaggebend für die didaktische Methode. Beide Faktoren dürfen jedoch nicht gleich gesetzt werden. Der Lehrinhalt, in unserem Projekt die Musiklehre, kann nicht direkt auf die Lehrziele übertragen werden, denn dies würde ein bloßes auswendig Lernen der Inhalte (deklaratives Wissen) verlangen. Wie bereits beschrieben, wollen wir jedoch auch prozedurales Wissen vermitteln und somit den Lernenden die Musiklehre anwenden lassen.

Bei der Vermittlung von deklarativen Wissen werden die Inhalte präsentiert und erläutert. Weiterhin sind auch Übungen wichtig, damit das Wissen gefestigt wird. Um prozedurales Wissen zu erwerben, müssen dann weitere Übungen absolviert werden. Diese müssen variantenreich gestaltet sein und wiederholt durchgeführt werden können. (Vgl. Kerres 2012: 280)

### 5.2.3 Anpassung an Medien

Das von uns gewählte Medium Smartphone verfügt über verschiedene Anzeige/Ausgabe- und Eingabemöglichkeiten. Die Anzeige der Inhalte erfolgt visuell über den Bildschirm in Form von Text und Bildern. Weiterhin wird die Ausgabe von Sound und somit ein auditives Medium ermöglicht. Durch das Smartphone steht weiterhin die Option eine taktilen Eingabe zur Verfügung.

### 5.2.4 Bestimmung der didaktischen Methode

Aufgrund der unterschiedlichen Eigenschaften der Zielgruppe haben wir uns entschieden, eine Mischung aus zwei didaktischen Methoden umzusetzen: der *expositorischen Methode* und der *explorativen Methode*. Beide werden im Folgenden kurz vorgestellt. Anschließend wird der Aufbau unserer App beschrieben und die jeweiligen didaktischen Elemente hervorgehoben.

Bei der *expositorischen Methode* steht „die Präsentation von Inhalten im Vordergrund“ (Kerres 2012:305) Die Inhalte werden durch Medien präsentiert und werden vom Lernenden nacheinander bearbeitet. Weiterhin wird das vermittelte Wissen aktiv in Übungen angewendet. Als besonders lernförderlich gelten motivierende Maßnahmen zu Beginn eines Lernangebots, wie die Nennung von Lehrzielen. Des weiteren sollte

sich die Präsentation von Wissen mit der Anwendung dessen durch den Lernenden abwechseln und die gemachten Fortschritte bzw. der aktuelle Wissensstand rückgemeldet werden. (Vgl. Kerres 2012:305ff)

Bei der explorativen Methode werden Lernangebote so strukturiert, dass der Lernende zwischen den Inhalten springen kann. Die Reihenfolge des Lernen ist somit freier und wird nicht von außen vorgegeben. Der Lernende kann die Thematik selbst erkunden und tastet sich selbstständig vor.

Die folgende Tabelle fasst nochmals die Entscheidungskriterien für die expositorische und explorative Methode zusammen.

<b>Kriterien</b>	<b>Lineare Lernwege Exposition</b>	<b>Offene Lernwege Exploration</b>
Lehrstoff	Hierarchisch gegliedert	Flach gegliedert
Lernsituation	formell	informell
Zielgruppe	homogen	Inhomogen, dispers
Lernstil	unselbstständig	selbstständig
Motivation	extrinsisch	intrinsisch
Vorwissen	niedrig	hoch

*Tabelle 2: Entscheidungsstruktur von Interaktionsräumen, nach Kerres (2012:336)*

Wie bereits dargelegt, trifft anhand der Kriterien keine Methode eindeutig zu. Deshalb wurde eine Mischform gewählt, die einen hauptsächlich expositorischen Charakter hat, jedoch auch explorative Elemente beinhaltet.

### **5.3 Strukturierung der Inhalte mithilfe einer Informationsarchitektur**

Der von uns gewählte Lerninhalt, grundlegende Musiklehre, gliedert sich zunächst in verschiedene Lektionen. Dabei orientierten wir uns an dem Buch „Musiktheorie für Dummies“ (Pillhofer/Day 2012) Eine Lektion umfasst jeweils ein Teilgebiet der Musiklehre, z. B. Notenwerte. Diese einzelnen Teilgebiete haben einen sehr großen Umfang und mussten deshalb in kleinere Einheiten unterteilt werden. Weiterhin mussten Übungen und Tests umgesetzt werden.

Um jede Lektion gleich aufzubauen, haben wir eine Informationsarchitektur entworfen. 5 Topic-Typen dienen dazu die Informationen regelmäßig aufzuteilen. Die folgende Tabelle zeigt die Topic-Typen mit den verschiedenen Merkmalen.

Topic	Name	Inhalt	Medien/Methode /Darstellung	Position in Lektion
1	Lernziele	Lernziele werden stichpunktartig genannt	Text	Am Anfang
2	Überblick	Überblick über gesamte Lektion wird gegeben, Grundbegriffen werden erklärt	Text, Bild	Nach Lernziel
3	Informationseinheit	Einzelne Inhalte der Lektionen werden erläutert, Veranschaulichung durch Hörbeispiele	Text, Bild, Notenbild, Audio	nach Überblick
4	Übung	Elerntes wird durch Übung gefestigt, verschiedene Schwierigkeitsstufen, Prozedurales Wissen wird aufgebaut	Text, Notenbild, Audio, Touch	Nach allen Informationseinheiten
5	Test	Deklaratives Wissen wird getestet	Multiple Choice Drag'n'Drop	am Ende

*Tabelle 3: Topic Typen der Informationsarchitektur*

Die Lektionen sind immer aus der selben Reihenfolge von Topics aufgebaut. Lediglich die Anzahl der Informationseinheiten variiert.

Die Topics wurden als jeweilige Elternelemente in die XML-Struktur übernommen. Auch der Aufbau innerhalb der Topics ist standardisiert.

Die folgende Abbildung zeigt die Topicaufteilung innerhalb einer Lektion sowie die grobe Strukturierung der Informationseinheiten.

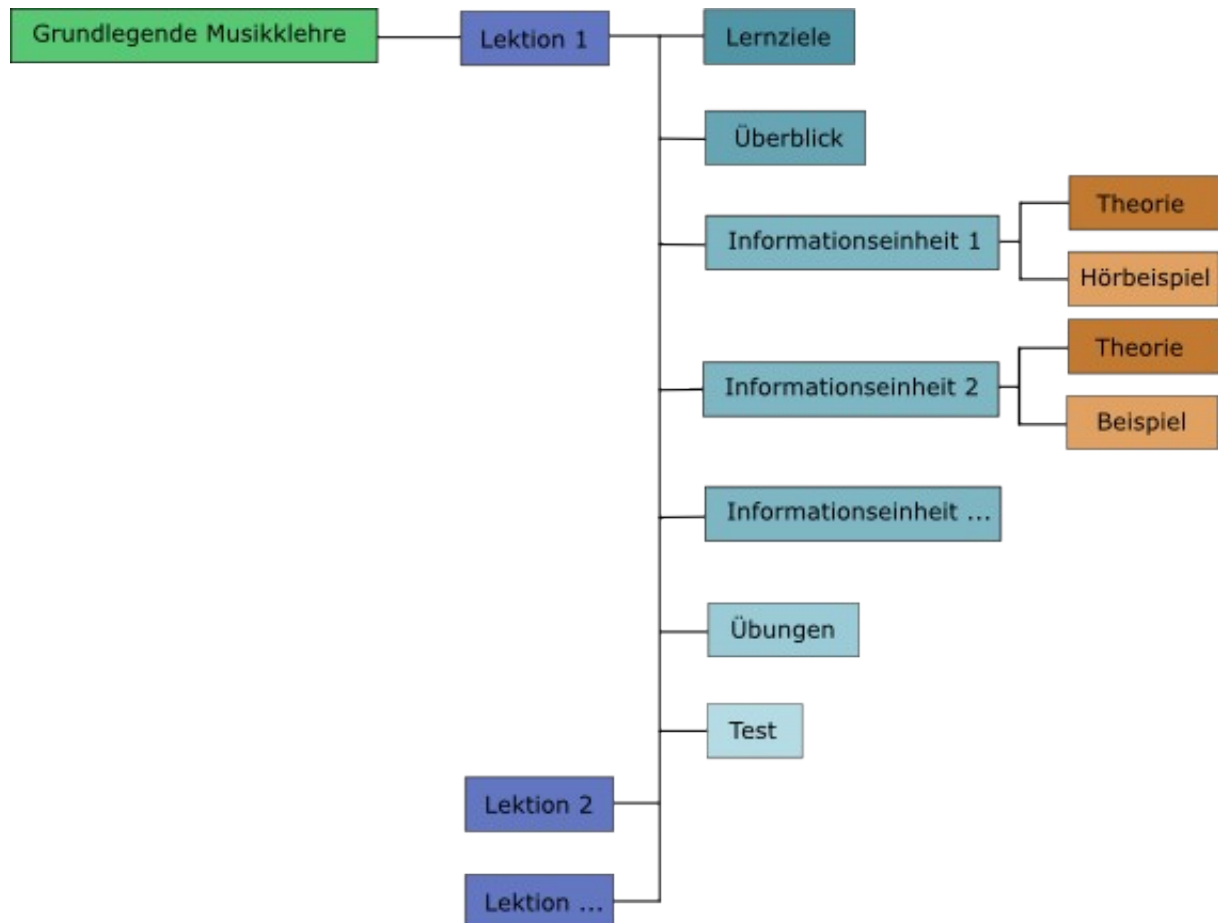


Abbildung 2: Struktur der App

### 5.3.1 Aufbau deklaratives Wissen

Das deklarative Wissen wird in den Topics 2 und 3 aufgebaut. Jede Lektion startet mit grundlegenden Informationen über die nachfolgenden Inhalte. Danach folgen die Informationseinheiten, die meist aufeinander aufbauen und ein sequenzielles Bearbeiten fordern.

In den Informationseinheiten wird zuerst jeweils die Theorie mithilfe von erklärenden Texten und Bildern vermittelt. Danach folgt je nach Lektion ein Hörbeispiel oder normales Beispiel, wodurch der Inhalt gefestigt wird. Der Lernende wird an dieser Stelle bereits aufgefordert sein Wissen anzuwenden, wird jedoch noch nicht auf die Richtigkeit seiner Eingaben überprüft. Die Bearbeitungszeit einer Informationseinheit wird je nach Vorwissen und Motivation des Lernenden sowie je nach Länge der Informationseinheit zwischen 5–15 Minuten eingeschätzt.

Um das deklarative Wissen zu überprüfen, muss der Lernende am Ende jeder Lektion

einen Test, der aus Multiple Choice Fragen und Drag'n'Drop Aufgaben besteht, absolvieren.

### **5.3.2 Aufbau prozeduralen Wissens**

Nachdem der Lernende bereits in den einzelnen Informationseinheiten im Beispielpart sein Wissen versuchen konnte umzusetzen, wird er am Ende einer Lektion zu einer Reihe von Übungen aufgefordert, deren Eingabe geprüft wird. Ihm wird rückgemeldet, ob seine Eingabe schlecht oder gut war. Je nach Übung wird versucht möglichst genau darzustellen, was falsch gemacht wurde.

Wenn eine Übung 5-mal hintereinander nicht richtig eingegeben wurde, kann bei den Tippübungen der entsprechende Takt auch angehört werden. Dies schult zusätzlich das Gehör, was sehr wichtig beim Lernen eines Instruments ist. Weiterhin wird dadurch die Übung für den Lernenden erleichtert und Frustrationsgrenze wird nicht so schnell erreicht.

Standardmäßig müssen nacheinander nach unsere Einstufung zwei leichte, zwei mittlere und zwei schwere Übungen absolviert werden.

### **5.3.3 Motivation**

Vor Beginn einer Lektion werden dem Lernenden die Lernziele genannt. Diese können einerseits motivieren, die Lektion zu beginnen, und geben andererseits einen Überblick über die Lektion. Lernende mit mehr Vorwissen können somit an diesem Punkt der Bedienung entscheiden, ob sie diese Lektion bearbeiten oder überspringen.

Zur weiteren Motivation wird nach der Bearbeitung der Übungen und Tests eine Art Highscore angezeigt. Dieser gibt dem Lernenden eine Rückmeldung über seine Leistung und animiert ihn gleichzeitig, sich zu verbessern.

### **5.3.4 Verschiedene Lernwege**

Wie eine Lektion bearbeitet wird, kann der Lernende selbst entscheiden. Nach Auswahl der entsprechenden Lektion können verschiedene Möglichkeiten gewählt werden.

1. Komplette Lektion bearbeiten (Theorie + Übungen + Test)
2. Theorie (nur Theorie)
3. Übungen (verschiedene Übungen zur Lektion)

Für Anfänger ohne Vorwissen empfiehlt sich die erste Möglichkeit, da der Lernweg vorgegeben wird und die Inhalte schrittweise aufeinander aufbauend eingeführt werden. Die Übungen und der Test werden im Anschluss durchgeführt. Weiterhin kann nur der Theorieteil bearbeitet werden. Dies eignet sich für die Wiederholung von Faktenwissen. Als letzte Möglichkeit können auch nur die jeweiligen Übungen zur Lektion absolviert werden. Hier werden dem Lernenden jedoch mehr Übungen zur Verfügung gestellt als in der eigentlichen Lektion Pflicht sind. Der Lernende kann zwischen leichten, mittleren und schweren Übungen wählen und somit selbstständig seine Fähigkeiten verbessern.

Innerhalb der Lektion gibt es zusätzlich die Möglichkeit zwischen den einzelnen Informationseinheiten zu springen. Davon profitieren vor allem die Lernenden mit mehr Vorwissen, da bereits gelernte Inhalte nicht nochmals bearbeitet werden müssen. Über eine Menü kann die gewünschte Informationseinheit gewählt werden. Damit ist ein freieres Arbeiten möglich.

Abschließend werden die Merkmale unseres didaktischen Konzepts in der folgenden Tabelle zusammengefasst und nach expositorischen und explorativen Elementen unterschieden.

<b>Expositroische Elemente</b>	<b>Explorative Elemente</b>
Hierarchisch gegliederter Inhalte	Freie Wahl des Lernwegs innerhalb der Lektion möglich (Menü)
Hauptsächlich sequenzielle Bearbeitung der Inhalte vorgesehen	Selbstständiges Üben
Nennung der Lernziele zu Beginn einer Lektion	
Rückmeldung über Wissenstand	
Abwechselnd Präsentation von Inhalten abwechselnd und Übungen	

*Tabelle 4: Expoitorische und explorative Elemente*

## 5.4 Lerntheoretische Position

Über den Vorgang des Lernens existieren verschiedene Vorstellungen. In den letzten 50 Jahren gab es hauptsächlich 3 Ansätze, die Lernen unterschiedlich definieren: behavioristischer, kognitivistischer und konstruktivistischer Ansatz. (Kerres 2012: 112)

In Rückblick auf unser Projekt lässt sich unser didaktisches Konzept als kognitivistisch einstufen. Im Kognitivismus nimmt die menschliche Wahrnehmung eine zentrale Position ein, da von ihr abhängt, wie Informationen verarbeitet werden. Lernen, das „als Informationsaufnahme und -speicherung betrachtet“ (Kerres 2012:112) wird, hängt somit von der Darstellung der Lerninhalte ab, jedoch auch von den kognitiven Aktivitäten des Lernenden.

Laut kognitivistischer Ansätze muss das Lernmaterial an die Voraussetzungen der Zielgruppe angepasst werden (Vorwissen). Die didaktischen Methoden Exposition und Exploration werden bevorzugt. Weiterhin wird auch die Unterscheidung von deklarativen und prozeduralem Wissen und die Wichtigkeit verschiedener Lernmethoden um dieses Wissen aufzunehmen betont. (Kerres 2012:119ff)

## 6 Übersicht Lerninhalte

Die folgende Tabelle enthält eine Übersicht über die Inhalte der einzelnen Lektionen, sowie der verwendeten Übungen.

Lektion	Abschnitt	Inhalt	Übung
1 Notenwerte	Überblick	Was ist der Takt? Aufbau einer Note	Tippen der Takte
	IE 1	Ganze Note	
	IE 2	Halbe Note	
	IE 3	Viertelnote	
	IE 4	Achtelnote	
	IE 5	Punktierte Note	
	IE 6	Haltebogen	
2 Pausen	Überblick	Arten von Pausen Pausenbaum	Tippen der Takte
	IE 1	Ganze Pause	
	IE 2	Halbe Pause	
	IE 3	Viertelpause	
	IE 4	Achtelpause	
	IE 5	Punktierte Pause	
3 Takt	Überblick	Taktangabe (Bruch)	Tippen der Takte
	IE 1	Einfache Taktarten	
	IE 2	Der 4/4-Takt	
	IE 3	Der 3/4-Takt	
	IE 4	Der 3/8-Takt	
	IE 5	Der 2/2-Takt	
	IE 6	Zusammengesetzte ungerade Taktarten	
	IE 7	Der 6/8-Takt	
4 Beat	Überblick	Betonung	Tippen der Takte
	IE 1	Die Synkope	
	IE 2	Die Auftaktnote	
	IE 3	Die Triolen	
	IE 4	Die Duolen	
5 Notenschrift	Überblick	Notenhöhe, Notenliniensystem	Noten einzeichnen
	IE 1	Der Violinschlüssel	
	IE 2	Der Bassschlüssel	

<b>Lektion</b>	<b>Abschnitt</b>	<b>Inhalt</b>	<b>Übung</b>
6 Tonschritte	Überblick	Halbtöne und Ganztöne auf der Klaviertastatur	Noten einzeichnen
	IE 1	Halbtonschritte	
	IE 2	Ganztonschritte	
	IE 3	Das Erhöhungszeichen	
	IE 4	Das Erniedrigungszeichen	
	IE 5	Das Auflösungszeichen	
7 Tonleitern	Überblick	Unterscheidung der Tonleitern	Gehörschulung
	IE 1	Die Durtonleiter	
	IE 2	Die natürliche Molltonleiter	
	IE 3	Die harmonische Molltonleiter	
	IE 4	Die melodische Molltonleiter	
8 Der Quintenzirkel	Überblick	Allgemeines zum Quintenzirkel	Vorzeichen bestimmen
	IE 1	Erhöhungszeichen ablesen	
	IE 2	Erniedrigungszeichen ablesen	

*Tabelle 5: Inhalte der Lektionen*

## 7 Konzeption

Vor Beginn der Implementierung von *musicate* wurde versucht, alle wichtigen Aspekte der App zu identifizieren und diese entsprechend zu spezifizieren. Als relevant wurden folgende Bereiche identifiziert:

- Plattform der App
- Funktionsumfang
- Inhaltserfassung und Bereitstellung
- Design
- Bedienkonzept

Auf diese Bereiche soll in diesem Kapitel näher eingegangen werden.

### 7.1 Plattform der App

Die primäre Ziel-Plattform von *musicate* war von Anfang an das mobile Betriebssystem **Android**. Grund hierfür war unter anderem der ständig steigende Marktanteil **Androids** und die Verfügbarkeit entsprechender Testgeräte. Um die Entwicklung auf anderen Plattformen wie iOS besonders einfach zu machen, sollte *musicate* nicht als native App, d.h. in Java, geschrieben werden. Sie sollte stattdessen mithilfe des Frameworks **PhoneGap** umgesetzt werden. **PhoneGap** ist ein Framework, welches es ermöglicht, aus Web-Apps Anwendungen für mobile Geräte zu machen. Mit **PhoneGap** erstellte Apps können auch auf APIs des Betriebssystems zugreifen, die normalen Web-Apps nicht zur Verfügung stehen.

Es war außerdem angedacht, *musicate* auch als normale Web-App für Desktop- und Laptop-Geräte zur Verfügung zu stellen. Bei anfänglichen Tests des Bedienkonzepts fiel jedoch auf, dass die sehr stark an direkter Nutzerinteraktion ausgerichteten Übungen, wie zum Beispiel das „Trommeln“ eines Rhythmus, mit Eingabegeräten wie Maus und Tastatur nicht optimal umgesetzt werden können. Prinzipiell können jedoch alle Funktionen von *musicate* mit entsprechenden Anpassungen auch für moderne Desktop-Browser umgesetzt werden.

## 7.2 Funktionsumfang

*musicate* soll dem Nutzer musikalisches Grundwissen vermitteln. Dieses Wissen soll durch praktische Übungen und Tests gefestigt werden. Somit muss zunächst ein System implementiert werden, das dem Nutzer Inhalte auf dem Endgerät präsentiert. Hierzu gehören sowohl Text-Beiträge als auch Audio-Beispiele

Theoretisches Wissen soll mithilfe von Tests gefestigt werden. Diese Tests sind als Multiple-Choice Tests umgesetzt. Die App muss dem Nutzer die Testfragen präsentieren und seine Eingaben auf Richtigkeit überprüfen können. Des Weiteren muss die erreichte Punktzahl des Nutzers zumindest temporär gespeichert werden, um die Gesamtleistung eines Tests zu ermitteln.

Die praktischen Übungen zur Festigung des erworbenen Wissens sind so gestaltet, dass der Nutzer einen Rhythmus über das Touchdisplay des Geräts „eintippen“ muss. Das System muss also in der Lage sein, solche Eingaben zu registrieren und auf ihre korrektes Timing zu evaluieren. Die Ergebnisse der Übungen müssen ebenfalls gespeichert werden, um die Gesamtleistung zu ermitteln.

Der Nutzer der App soll in der Lage sein, die Inhalte der App zu erweitern bzw. festzulegen. Die App muss also eine Schnittstelle zur Verfügung stellen, die es Nutzern erlaubt, nachträglich neue Inhalte hinzuzufügen.

## 7.3 Erfassung und Bereitstellung der Inhalte

PhoneGap stellt Inhalte mithilfe von HTML dar. Somit ist die einfachste Art der Inhaltsbereitstellung das harte Codieren in HTML. Der Nachteil dieses Ansatzes ist jedoch, dass neue Inhalte nur durch die Entwickler zur App hinzugefügt werden können.

Um die Erstellung von User Generated Content zu ermöglichen und den späteren Nutzern der App die Möglichkeit zu geben, nur die Lektionen herunterzuladen, die sie auch benötigen, wurde ein anderer Ansatz gewählt.

Die Inhalte von *musicate* werden extern mit einem speziellen XML-Schema erfasst, welches auf dem eingesetzten didaktischen Konzept basiert. Dies hat unter anderem den Vorteil, dass die Inhalte per XSLT in beliebig viele Zielformate publiziert werden können. So wäre es z.B. möglich, aus den *musicate*-Lektionen später PDFs zu generieren.

Nach der Erfassung der Inhalte werden sie per XSLT in HTML-Dateien umgewandelt. Diese Dateien werden dann dynamisch in die App geladen. Dieser Vorgang und Informationen zur XML-Struktur werden im Kapitel Implementierung erläutert.

## 7.4 Design

Um neuen Nutzern den Umgang mit der App zu erleichtern, sollte sich das Design an bestehenden Apps orientieren. Als Beispiele wurden hier vor allem Apps der Firma Google verwendet, da diese den Styleguide der Android-Plattform am stärksten befolgen. Zu den untersuchten Apps gehören unter anderem Google Mail und Google Currents.

Außerdem wurden generell Apps untersucht, die dem Nutzer in irgendeiner Art Content präsentieren. Hierzu gehören z.B. Feed-Reader wie Feedly. Die Vorgehensweise bei der Analyse sowie die analysierten Aspekte werden im folgenden Unterkapitel kurz erläutert.

### 7.4.1 Analyse bestehender Apps

Folgende Aspekte wurden bei der Analyse näher betrachtet:

- Design
- Navigation
- Bedienkonzept

Beim Design fiel bei allen analysierten Apps die Leiste am oberen Bildschirmrand auf. Dort befindet sich meistens auf der linken Seite das Logo der App sowie der Titel der App oder der Titel des aktuellen Screens. Die Farbe dieser Leiste war bei allen analysierten Apps hellgrau. Generell setzten die analysierten Apps Farben nur zur Hervorhebung ein.



Beim Wechseln in einen hierarchisch tieferen Bereich der App, erschien meist links des Logos ein Pfeil, der das Navigieren zum vorherigen Screen erlaubt. Vor allem bei Google-Apps fungiert das Logo auch als Link zum Öffnen einer Seitenleiste. Die Leiste am oberen Bildschirmrand enthält meistens Buttons zu häufig genutzten Funktionen in der App.



Abbildung 4: Menü-Button und Zurück-Button

Bei seitenbasierten Inhalten kann der Nutzer meistens mit einer horizontalen Wisch-Geste vor und zurück navigieren. Bei Google-Apps kann durch eine rechtsgerichtete Wisch-Geste, die vom äußersten Rand des Bildschirms gestartet wird, auch die Seitenleiste geöffnet werden.

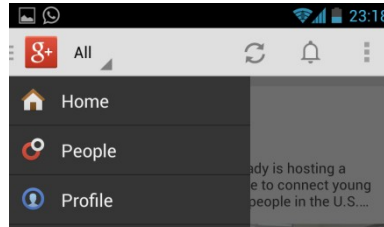
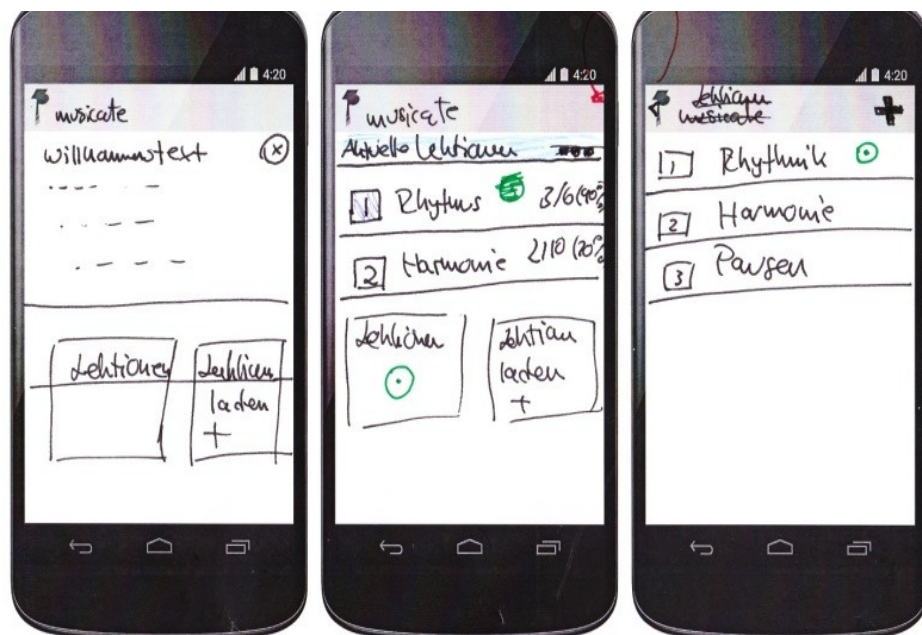


Abbildung 5: Seitenleiste

## 7.4.2 Erarbeitung des Designs

Das Design von *musicate* wurde mithilfe von UI-Sketches erstellt. Nach der Analyse von bestehenden Apps und der Festlegung des Funktionsumfangs, wurden die möglichen Wege der Nutzer anhand von Sketches illustriert. Ergebnis dieser Arbeit waren 25 Sketches, auf welchen das Screen-Design von *musicate* aufbaut (siehe Anhang).



- ① Startseite  
↳ Beim ersten Öffnen
- ① Startseite  
↳ beim erneuten Öffnen
- ② Lektionen-Übersicht

Abbildung 6: UI-Sketches für *musicate*

### 7.4.3 Cognitive Load Theorie

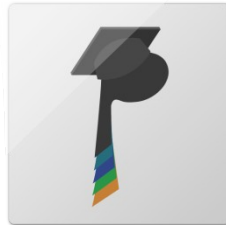
Nachhaltiges Lernen kann nur dann erfolgen, wenn die vermittelten Informationen ins Arbeitsgedächtnis gelangen und von dort aus in Langzeitgedächtnis überführt werden. Das Arbeitsgedächtnis ist jedoch nur begrenzt aufnahmefähig. Je nach dem wie Informationen verarbeitet werden, können diese länger gespeichert werden. (Vgl. Kerres 2012:146)

In Bezug auf die App ist es daher wichtig, das Arbeitsgedächtnis nicht zu überlasten, damit die wichtigen Informationen vom Lernenden aufgenommen werden können. Nach der Theorie der cognitive load von Sweller (2010) muss das Arbeitsgedächtnis beim Lernen aber auch angemessen belastet werden. Nicht relevante Aspekte müssen jedoch trotzdem reduziert werden, da diese beim Lernen stören und von Lernenden mit weniger Vorwissen erst gefiltert werden müssen, was eine zusätzliche kognitive Belastung darstellt. (Vgl. Kerres 2012:146ff)

Beim Design der App wurde daher darauf geachtet, dass innerhalb der Lektionen die kognitive Beanspruchung im richtigen Maß gehalten wird. Während in den Startbildschirmen und der Navigation oft gestalterische Elemente wie Ikonen und farbige Hervorhebung zum Einsatz kommen, sind die Lektionen das Design betreffend sehr schlicht gehalten. Die Aufmerksamkeit des Lernenden soll somit auf den Inhalt beschränkt werden. Weiterhin ist so genügend Kapazität im Arbeitsgedächtnis vorhanden, um die vermittelten Inhalte zu verarbeiten und ins Langzeitgedächtnis zu überführen.

#### 7.4.4 Entwicklung des Logos

Zur besseren Identifikation der App und um den Wiedererkennungswert zu steigern, wurde für *musicate* auch ein Logo entworfen. Das Logo wurde aus verschiedenen Entwürfen herausgearbeitet. Es stellt eine Viertelnote mit einem Doktorhut dar. Das Logo setzt das für die App entwickelte Farbkonzept um.



*Abbildung 7:  
Musicate-Logo*

#### 7.5 Bedienkonzept

*musicate* unterstützt sowohl normale Touch-Eingaben als auch Gesten, wie das Wischen („swipe“). Die Wisch-Geste wird zum Beispiel verwendet, um die Informationseinheiten innerhalb einer Lektion zu ändern.

## 8 Implementierung

In diesem Kapitel werden die verschiedenen Aspekte der Implementierung der finalen App beschrieben. Zunächst wird auf die Erfassung und Ausgabe der Inhaltsdateien auf Basis von XML eingegangen.

Danach werden die Implementierungen des Designs und der Funktionen der App erläutert. Abschließend werden noch einige Schwierigkeiten und Besonderheiten beschrieben, die bei der Implementierung aufgetreten sind.

### 8.1 Inhaltserstellung

Die Lektionen in *musicate* werden in einem speziellen XML-Schema erfasst, welches auf dem zugrunde liegenden didaktischen Konzept aufbaut. Hieraus werden die Inhaltsdateien für die App und die Manifest-Datei erstellt. Auf den genauen Prozess bei der Transformation und die Struktur einer Lektion wird in diesem Kapitel näher eingegangen.

#### 8.1.1 XML-Struktur

Das verwendete XML-Schema wurde speziell für *musicate* entwickelt. In diesem Schema werden alle Inhaltsdaten eingepflegt. Die einzelnen Elemente der Struktur sowie deren Bedeutung sollen nun näher erläutert werden.

Das Wurzel-Element der XML-Datei ist das sog. `<lesson>`-Element.

```
<lesson>
  <title>Rhythmik</title>
  <goals></goals>
  <overview></overview>
  <informationUnit></informationUnit>
  <exercise></exercise>
  <test></test>
</lesson>
```

Das `<title>`-Element enthält lediglich den Titel der Lektion.

Als nächstes folgt das `<goals>`-Element, welches die Lernziele der Lektion enthält.

```
<goals>
  <description>...</description>
```

```
<goalList>
  <listItem...</listItem>
</goalList>
</goals>
```

Das `<goals>`-Element kann ein oder mehrere `<description>`-Elemente enthalten. Jedes `<description>`-Element enthält Informationen zu einem zusammengehörigen Thema.

Es kann ein `<goalList>`-Element mit mehreren `<listItem>`-Elemente geben. Hieraus wird eine Aufzählungsliste generiert, die die Lernziele der Lektion auflistet.

Das `<overview>`-Element gibt einen Überblick über das besprochene Lerngebiet.

```
<overview>
  <description>...</description>
  <picture>
    <filename>...</filename>
    <caption>...</caption>
  </picture>
  <notation>
    <filename>...</filename>
    <caption>...caption>
  </notation>
</overview>
```

Das `<overview>`-Element kann ein oder mehrere `<description>`-Elemente enthalten. Es können auch ein oder mehrere `<picture>`-Elemente verwendet werden, die zum Einbinden von Grafiken verwendet werden können. Ein `<picture>`-Element muss ein `<filename>`-Kindelement besitzen, welches den Dateinamen der Grafik enthält. Es kann außerdem ein `<caption>`-Element zur Darstellung einer Bildunterschrift enthalten.

Das `<informationUnit>`-Element besteht immer aus einem `<theory>`-Element und aus einem `<listeningSample>`- oder `<example>`-Element.

```
<informationUnit>
  <theory>...</theory>
  <listeningSample>...</listeningSample>
  <example>...</example>
</informationUnit>
```

Das `<theory>`-Element vermittelt dem Nutzer Wissen zur Theorie. Es muss ein `<intro>`-Element besitzen, mit dem der Wissensblock eingeleitet wird. Danach können beliebig oft `<description>`-, `<picture>`- oder `<notation>`-Elemente verwendet werden.

```
<theory>
  <title></title>
  <intro></intro>
  <notation>...</notation>
  <description>...</description>
</theory>
```

Das `<listeningSample>`-Element bietet dem Nutzer ein Hörbeispiel und fordert ihn zu

einer kleinen Übung auf, die jedoch nicht bewertet wird. Es muss ein <title>-Element beinhalten. Danach muss ein <sample>-Element folgen. Dieses Element muss ein <filename>-Element beinhalten, das den Dateinamen des Hörbeispiels enthält. Außerdem muss das <sample>-Element ein <notationImage>-Element beinhalten. Dieses Element enthält das Notenbild des Hörbeispiels. Das <notationImage>-Element muss auch ein <filename>-Element beinhalten, welches den Dateinamen des Bilds beinhaltet. Als letztes folgt ein <task>-Element, das dem Nutzer seine Aufgabe im Hörbeispiel vermittelt. Es kann ein oder mehrere <item>-Kindelemente beinhalten.

```
<listeningSample>
  <title>...</title>
  <sample>
    <filename>...</filename>
    <notationImage>
      <filename>...</filename>
    </notationImage>
  </sample>
  <tasks>
    <item>...</item>
  </tasks>
</listeningSample>
```

Das <example>-Element fragt beim Nutzer Theoriewissen ab, bietet jedoch direkt die Lösung an und die Leistung wird nicht gewertet. Es beginnt mit einem <title>-Element. Darauf folgt ein <question>-Element, das die Frage beinhaltet. Das <answer>-Element beinhaltet die Antwort auf die Frage.

```
<example>
  <title>...</title>
  <question>...</question>
  <answer>...</answer>
</example>
```

Als nächstes Element folgt das <exercise>-Element. Hier muss der Nutzer praktische Übungen absolvieren. Es besteht aus einem <intro>- und einem <tapTask>-Element.

```
<exercise>
  <intro>...</intro>
  <tapTask>...</tapTask>
</exercise>
```

Das <intro>-Element beginnt mit einem <title>-Element. Danach folgt ein <task>-Element, das dem Nutzer seine Aufgabe bei der Übung erläutert. Danach folgt ein <notation>-Element, welches das Notenbild des zu klopfenden Rhythmus enthält. Abschließen folgt ein <info>-Element, das dem Nutzer wichtige Informationen zum

Ablauf der Übung gibt.

```
<intro>
  <title>...</title>
  <task>...</task>
  <notation>...</notation>
  <info>...</info>
</intro>
```

Nach dem `<intro>`-Element folgt das `<tapTask>`-Element. Dieses beinhaltet die eigentliche Übung. Es startet mit einem `<title>`-Element. Danach folgt das `<timeSignature_nPM>`-Element, welches Teil der Taktart ist; es ist sozusagen der obere Teil des Bruchs einer Taktangabe.

Danach folgt das `<timeSignature_base>`-Element welches den unteren Teil der Taktangabe enthält; erlaubte Werte sind „q“ für Viertel und „8“ für Achtel. Das `<tempo>`-Element enthält das Tempo der Übung in BPM. Das `<metronome>`-Element gibt den Dateinamen der Metronom-Datei an. Der einzugebende Rhythmus der Übung steht im `<rhythm>`-Element. Näheres zur Notation wird im Kapitel Implementierung erläutert.

Abschließend muss noch ein `<notation>`-Element vorhanden sein, welches das Notenbild noch einmal zeigt.

```
<tapTask>
  <title>...</title>
  <timeSignature_nPM>...</timeSignature_nPM>
  <timeSignature_base>...</timeSignature_base>
  <tempo>...</tempo>
  <metronome>...</metronome>
  <rhythm>...</rhythm>
  <notation>...</notation>
</tapTask>
```

Das letzte Element der Lektion ist das `<test>`-Element.

```
<test>
  <maxPoints>...</maxPoints>
  <task type="multiplechoice">... </task>
</test>
```

Das `<test>`-Element beginnt mit einem `<maxPoints>`-Element. Dieses Element beinhaltet die maximal zu erreichende Punktzahl des Tests. Danach folgen ein oder mehrere `<task>`-Elemente.

```
<task type="multiplechoice">
  <question>...</question>
  <picture>...</picture>
  <answers>
```

```
        <answer type="correct">...</answer>
        <answer type="wrong">...</answer>
    </answers>
</task>
```

Das `type`-Attribut legt die Art des Tests fest. Als erstes Element folgt das `<question>`-Element, welches die Frage des Tests enthält. Danach kann ein `<picture>`- oder ein `<notation>`-Element eingebunden werden. Als letztes folgt ein `<answers>`-Element, das 2 oder mehr `<answer>`-Elemente enthält. Die `<answer>`-Elemente verfügen jeweils über einen `type`-Attribut. Dieser definiert, ob eine Antwort korrekt (`type="correct"`) oder falsch (`type="wrong"`) ist.

### 8.1.2 Transformation

Die Umwandlung der XML-Datei erfolgt mithilfe von Saxon und einem XSLT-Skript. Die resultierenden Dateien sind zum einen HTML-Dateien. Diese bestehen meistens nur aus einem `<article>`- oder einem `<div>`-Element mit einer Klasse, die je nach XML-Element, variiert.

Folgende XML-Elemente werden in eigene HTML-Dateien ausgespielt:

- `<theory>`
- `<listeningSample>`
- `<example>`
- `<intro>` (`<exercise>`)
- `<tapTask>` (`<exercise>`)
- `<task>` (`<test>`)
- `<goals>`
- `<overview>`

Neben den HTML-Dateien wird auch eine Datei mit dem Namen `manifest.xml` erstellt. Auf diese Datei und die Ordner-Struktur einer Lektion wird im nächsten Unterkapitel näher eingegangen.

### 8.1.3 Struktur eines musicate-Pakets

Ein fertiger Ordner einer Lektion beinhaltet alle dazugehörigen Dateien im Wurzelverzeichnis; es gibt also keine Unterordner. Zu den Dateien gehören beispielsweise die HTML-Dateien, die verwendeten Grafiken und die MIDI-Dateien. Bei der Transformation der XML-Datei entsteht eine weitere XML-Datei mit dem Namen `manifest.xml`. Diese Datei wird für den Import der Lektion in `musicate` benötigt.

Die Manifest-Datei hat folgende Struktur:

```
<manifest>
  <title></title>

  <goals></goals>

  <testPoints></testPoints>

  <playOrder>
    <file type="" title=""></file>
  </playOrder>

  <downloadList>
    <filename></filename>
  </downloadList>
</manifest>
```

Das `<title>`-Element enthält den Titel der Lektion. Das `<goals>`-Element enthält den Dateinamen der HTML-Datei, welche die Lernziele enthält. Das `<testPoints>`-Element gibt an wie viele Punkte im Test der Lektion maximal erreicht werden können. Das `<playOrder>`-Element legt die Reihenfolge fest, in welcher die Inhalte in der App präsentiert werden. Die dazu gehörigen `<file>`-Elemente enthalten den Namen der jeweiligen Datei. Das `<downloadList>`-Element enthält in den Kind-Elementen `<filename>` die Dateinamen aller Dateien, die vom Server heruntergeladen werden müssen.



manifest.xml
xyz.html
pic.jpg
song.midi
...

Abbildung 8: Ordner-Struktur des musicate-Pakets

#### 8.1.4 Aktueller Workflow zur Erstellung, Bereitstellung und Import einer Lektion

Zunächst muss die XML-Datei entsprechend des oben erläuterten Schemas erstellt werden. Danach muss die XML-Datei per XSLT in die HTML-Dateien und die Manifest-Datei transformiert werden. Dies geschieht mithilfe eines Batch-Skripts und erfordert keine besonderen Kenntnisse. Anschließend muss sichergestellt werden, dass sich alle Inhaltsdateien und die Manifest-Datei in einem Ordner, der den Titel der Lektion trägt, befinden. Dieser Ordner muss nun auf einem Webserver platziert werden.

Abschließend muss der Link zum Ordner an die Nutzer weitergegeben werden. Die Nutzer tragen den Link und den Titel der Lektion in *musicate* ein und der Download startet.

### 8.1.5 Weiterentwicklung

Im Moment müssen die XML-Dateien der Lektionen noch manuell erstellt werden. Somit ist es für Laien nicht möglich eine eigene Lektion zu erstellen. Es befindet sich jedoch bereits eine Web-App in Entwicklung, welche die Erfassung einer *musicate*-Lektion über den Web-Browser ermöglicht. Nach der erfolgreichen Erstellung der Lektion kann der Nutzer eine XML-Datei herunterladen, die dann per XSLT transformiert werden kann.

## 8.2 Umsetzung des Designs

Das erarbeitete Design wurde per CSS umgesetzt. Da es vor allem bei Android-Geräten eine große Fülle an verschiedenen Display-Größen gibt, wurde das Design „responsive“ angelegt. Beim „Responsive Design“ wird das CSS so angelegt, dass sich der Inhalt in seiner Darstellung optimal der Fensterbreite anpasst. Aus diesem Grund wurde fast ausschließlich mit relativen Positions- und Größenangaben gearbeitet. Um die Inhalte auf die Maße des Displays zu skalieren und das Zoomen zu verhindern, muss im HTML folgendes Meta-Tag hinzugefügt werden:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

## 8.3 Umsetzung der Funktionen

In diesem Kapitel werden die Implementierungen der in der Konzeption festgelegten Funktionen näher erläutert. Es werden die zugrundeliegenden Technologien, die Art der Implementierung und Code-Auszüge dargestellt.

### 8.3.1 Generelles

*musicate* verwendet kein UI-Framework wie JQuery-Mobile. Das UI wurde speziell für dieses Projekt programmiert. Dies liegt zum einen daran, dass JQuery-Mobile sehr viele UI-Elemente zur Verfügung stellt, *musicate* jedoch nur sehr wenige Elemente benötigt. Die Verwendung von JQuery-Mobile hätte also den Entwicklungsprozess erschwert. Des Weiteren ist das Design von JQuery-Mobile in vielen Aspekten sehr stark an iOS orientiert. *Musicate* ist jedoch hauptsächlich für Android bestellt und

sollte deshalb vom Design besser auf dieses System zugeschnitten sein. Es wird das JavaScript-Framework JQuery verwendet.

Alle Unterseiten der App werden dynamisch in die Hauptseite der App geladen. Diese Seite hat folgende Struktur:

```
<body>
  <div id="main">
    <section id="contentContainer">
      <!-- Leiste oben -->
      <header class="app">
        <nav>
          <div class="nav_overview">...</div>
          <div class="nav_backButton">...</div>
          <div id="logo">
            
            <span id="headerText"></span>
          </div>
        </nav>
      </header>
      <!-- Hauptinhalt -->
      <section id="appContent" class="app"></section>
      <!-- Navigation unten -->
      <footer>
        <div id="nav_prevContent">...</div>
        <div id="nav_nextContent">...</div>
      </footer>
    </section>
  </div>
</body>
```

### 8.3.2 Animationen

Um das User Interface dynamischer zu machen und näher an den Look&Feel einer nativen App heran zu kommen, wurden diverse Animationen eingebaut. Diese Animationen wurden mit Hilfe des JavaScript-Frameworks **transit** realisiert. **transit** setzt bei Animationen auf CSS3 animations.

Diese funktionieren sehr viel flüssiger als Animationen, die mit der JQuery-Funktion `animate` durchgeführt wurden. Ein beispielhafter Aufruf von `transit`:

```
$("#div1").transition({display: 'block'}).transition({opacity: 1});
```

In diesem Aufruf wird das Element mit der ID „div1“ eingeblendet und die Sichtbarkeit auf 100% eingestellt. Transit kann nicht nur Werte animieren sondern alle CSS-Attribute ändern.

### 8.3.3 Dynamisches Speichern

An mehreren Stellen müssen in *musicate* Daten dynamisch gespeichert werden. Hierzu zählen z.B. die im Test erreichten Punkte oder Informationen darüber, welche Lektionen installiert sind. Dies wird in *musicate* über HTML5 Web-Storage realisiert. HTML5 Web-Storage verfügt über 2 Varianten: `localStorage` und `sessionStorage`. In *musicate* wird `localStorage` verwendet. Mithilfe von `localStorage` können dauerhaft Informationen in Schlüssel-Wert-Paaren auf dem Gerät gespeichert werden. Im Fall von *musicate* reicht eine Speicherung von Schlüssel-Wert-Paaren nicht aus, denn es müssten komplexe JavaScript-Objekte gespeichert werden. Dies ist jedoch auch möglich, indem das zu speichernde Objekt mithilfe der Methode `JSON.stringify` in einen JSON-codierten String konvertiert wird und anschließend in einem Schlüssel-Wert-Paar-gespeichert wird. Beim Start von *musicate* wird der String geladen und mithilfe der Methode `JSON.parse` wieder in ein Objekt gewandelt.

### 8.3.4 Gesten

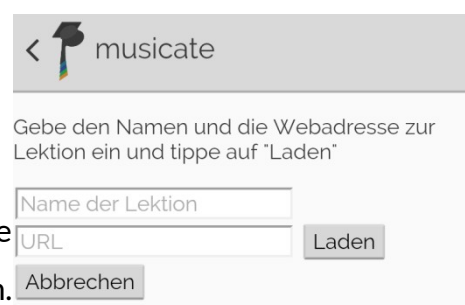
*musicate* unterstützt Eingaben in Form von Gesten. Innerhalb einer Lektion kann der Nutzer zwischen den verschiedenen Inhalten mittels horizontalen Wisch-Gesten navigieren. Diese Funktionalität wird mithilfe des JavaScript-Frameworks **hammer.js** realisiert. Die Integration ist sehr simpel: Zunächst muss die Skript-Datei heruntergeladen und in das Projekt integriert werden. Anschließend muss `hammer.js` für das gewünschte Objekt und die gewünschte Geste initialisiert werden.

```
$(document).hammer().on("swipeleft", "body", function(event) {  
    //Code, der ausgeführt werden soll  
});
```

Im oben gezeigten Beispiel wird **hammer.js** für eine links gerichtete Wisch-Geste (`swipeleft`) auf dem `Body`-Element initialisiert.

### 8.3.5 Download und Import

*musicate* erlaubt es, neue Inhalte hinzuzufügen. Neue Lektionen müssen zuerst heruntergeladen werden.



The screenshot shows the top part of the *musicate* app. At the top, there is a header with a back arrow, a graduation cap icon, and the text "musicate". Below the header, there is a text prompt: "Gebe den Namen und die Webadresse zur Lektion ein und tippe auf 'Laden'". Underneath the prompt, there are three input fields: "Name der Lektion", "URL", and "Abbrechen". To the right of the "URL" field is a "Laden" button.

Hierfür benötigt der Nutzer den Namen der Lektion

*Abbildung 9: Import von Lektionen*

und die URL.

Zunächst werden auf der SD-Karte des Geräts einige Ordner angelegt, in die die Dateien geladen werden. Die Ordnerstruktur ist hier:

```
SD-Wurzelverzeichnis > musicate > lessons > nameDerLektion
```

Hierfür wird die File API von PhoneGap verwendet und die aufgerufene Funktion lautet `nestedDirs(rootFolder,lessonsFolder,lesson)`.

Anschließend wird die Funktion `loadManifest(urlBase)` aufgerufen. In dieser Funktion wird per AJAX die Manifest-Datei des *musicate*-Pakets geladen.

Hier werden zum einen alle `<filename>`-Elemente der Datei in das Array `fileList` geschoben. Dieses Array enthält dann die Namen der Dateien, die in der Funktion `downloader()` heruntergeladen werden müssen. Die Funktion `downloader()` erstellt für jede Datei des Arrays `fileList` die Quell-URL auf dem Server und die URI auf dem Gerät und übergibt diese Daten an die Funktion `downloadFile()` die mithilfe der PhoneGap API File die Datei herunterlädt.

Außerdem werden die Elemente `<title>`, `<goals>` und `<testPoints>` in Variablen geschrieben. Des Weiteren werden alle `<file>`-Elemente des Elternelements `<playOrder>` mit dem Inhalt des Elements und den Werten der Attribute `type` und `title` in das Objekt `contents_` geschrieben. Dieses Objekt wird dann zusammen mit den Werten der Elemente `<title>`, `<goals>` und `<testPoints>` sowie dem Array `fileList` in das Objekt `musicateObject` geschrieben:

```
musicateObject.lessons.push({
  name: lessonTitle,
  goals: lessonGoal,
  files: fileList,
  contents: contents_,
  test: {
    maxPoints: lessonPoints,
    reachedPoints: 0
  }
});
```

Nachdem der Download abgeschlossen wurde, kann die Lektion gestartet werden.

Die bis jetzt erfassten Übungen können über folgende Daten heruntergeladen werden:

1. Name d. Lektion: Takt
  - a. URL: [musicate.digiaulogy.de/Takt](http://musicate.digiaulogy.de/Takt)
2. Name d. Lektion: Notenwerte
  - a. URL: [musicate.digiaulogy.de/Notenwerte](http://musicate.digiaulogy.de/Notenwerte)
3. Name d. Lektion: Pausen
  - a. URL: [musicate.digiaulogy.de/Pausen](http://musicate.digiaulogy.de/Pausen)

### 8.3.6 Inhaltsdarstellung

Die Inhalte der Lektionen werden per AJAX dynamisch in die App geladen. Grundlage hierfür sind zum einen die HTML-Dateien, die mithilfe der Download-Funktion auf das Gerät heruntergeladen wurden und statische HTML-Dateien, die sich bereits in der App befinden. Die Dateinamen der HTML-Dateien für die Lektionsinhalte sowie deren Titel, Typ und Reihenfolge sind im Array `contents` des Objekts `musicateObject.lessons` gespeichert. Bei einem Klick auf die Lektion in der Übersicht, wird anhand des Titels die Position der Lektion im Objekt `musicateObject` ermittelt. Nun kann gezielt auf die Inhalte der Lektion zugegriffen werden. Soll zum Beispiel die erste Datei der zweiten Lektion geladen werden, sähe der Aufruf folgendermaßen aus:

```
musicateObject.lessons[1].contents[0].file
```

Dieser Aufruf würde den Namen einer HTML-Datei zurückliefern. Um die Datei laden zu können wird jedoch der absolute Pfad benötigt. Dieser wird in der Funktion `loadLessonContents` konstruiert. Ein möglicher Pfad wäre beispielsweise:

```
file:///storage/emulated/0/musicate/lessons/Rhythmik/test.html
```

Dieser Pfad ist betriebssystemspezifisch und funktioniert in dieser Form nur für Android. Für andere Plattformen wurde dieses Vorgehen noch nicht getestet.

Dieser Pfad wird dann an die Funktion `loadPages(file)` übergeben, die mithilfe der **jQuery**-Funktion `load` den Inhalt in das `<div>`-Element „appContent“ hineinlädt.

### 8.3.7 Navigation

Wie bereits oben erläutert, werden in *musicate* neue Inhalte per AJAX in die Hauptseite geladen. Das bedeutet, dass nicht der Browserverlauf verwendet werden kann, um zur zuvor besuchten Seite zu wechseln. Dies würde nur funktionieren, wenn die HTML-Seiten über `<a>`-Elemente aufgerufen werden würden.

Aus diesem Grund musste in *musicate* eine Funktion implementiert werden, welche die zuletzt besuchten Seiten im Speicher behält, um später wieder zu diesen navigieren zu können. Immer wenn eine neue Seite besucht wird, wird über die Funktion `updateHistory(target)` ein neuer Eintrag in das Objekt `navigationHistory` geschrieben, der das Ziel des Aufrufs enthält. Das Objekt `navigationHistory` ist Teil des Objekts `musicateObject`. Der folgende Code zeigt die Funktion `updateHistory`:

```
function updateHistory(target) {
    musicateObject.current.navigationHistory.push({
        page: target
    });
}
```

Möchte ein Nutzer nun beispielsweise über den Zurück-Button in der oberen Leiste zum vorherigen Fenster zurückkehren, wird die Funktion `goBackinHistory()` aufgerufen. Diese Funktion löscht den letzten Eintrag aus dem Objekt `navigationHistory` und entscheidet dann basierend auf dem nun aktuellsten Eintrag im Objekt, welche Seite aufgerufen werden muss.

### 8.3.8 Rhythmus-Übung

In der Rhythmus-Übung wird dem Nutzer zunächst ein Notenbild gezeigt. Er muss dann, um die Übung korrekt abzuschließen, den im Notenbild gezeigten Rhythmus korrekt über den Touchscreen eingeben. Der Nutzer hat nach dem Abschluss der Übung die Möglichkeit die Übung zu wiederholen und sich ein Hörbeispiel des korrekten Rhythmus anzuhören. Gibt der Nutzer zu viele Noten ein, werden Punkte abgezogen.

Der App muss zunächst der korrekte Rhythmus bekannt sein. Der Rhythmus wird ebenfalls in der XML-Datei notiert. Die Notation wird in das `<rhythm>`-Element geschrieben.

Die Notenwerte werden folgendermaßen codiert.

Notenwert	Code
Ganze Note	w
Halbe Note	h
Punktierte Halbe Note	dh
Viertelnote	q
Punktierte Viertelnote	dq
Achtelnote	e
Punktierte Achtelnote	de
Sechzehntelnote	sixteen

Vor diesen Wert wird entweder die Tonhöhe, z.B. e4, oder eine Pause(r) gesetzt. Ein kurzes Beispiel soll die Notation näher erläutern:



Abbildung 10: Ein einfacher Rhythmus

Der oben gezeigte Rhythmus würde wie folgt notiert werden:

```
<rhythm>g4_q,g4_q,g4_e,g4_e,g4_q</rhythm>
```

Bei der Transformation der XML-Datei in HTML wird der Rhythmus in ein Attribut der Notationsgrafik gespeichert.

Um die Länge der Notenwerte in Millisekunden berechnen zu können, muss der App auch das Tempo der Übung bekannt sein. Dieses wird ebenfalls als Attribut an die Notationsgrafik angehängt.

Zur weiteren Berechnung wird auch die die Taktart benötigt. Auch diese wird als Attribut an die Notationsgrafik angehängt.

Es muss auch ein Metronom zu hören sein, damit der Nutzer im richtigen Tempo mitklopft. Das Metronom wird über MIDI-Dateien umgesetzt und die Datei muss auch in der XML-Datei angegeben werden. Der Dateiname dieser Datei ist ebenfalls in einem Attribut gespeichert.

Nachfolgend eine beispielhafte Notationsgrafik, notiert in HTML:

```
<figure class="exercise_notation" data-timeSignature_nPM="4" data-timeSignature_base="q" data-bpm="90" data-metronome="metronom_90.mid" data-notationData="e4_dq,e4_dq,e4_q">
  
  <figcaption>Die erste Hälfte einer Bossa-Nova-Clave</figcaption>
</figure>
```

Startet der Nutzer nun eine Übung, so startet im Hintergrund mit kurzer Verzögerung das Metronom (Näheres zur Implementierung des Sounds im Kapitel Hörbeispiele). Im Hintergrund wurde über die Funktion `getNotationData()` der Rhythmus aus dem Attribut der Notationsgrafik extrahiert. Anschließend wird der String, der momentan noch den Rhythmus enthält, mithilfe der JavaScript-Methode `split` an den Kommata zerteilt und in Form einzelner Werte in das Array `notationData` geschrieben. Danach werden mithilfe der JavaScript-Methode `replace` die Tonhöhen von den Notenwerten getrennt und jeweils in verschiedene Arrays geschrieben. Als nächstes wird das Array `notationRhythm`, das die einzelnen Notenwerte enthält, mithilfe einer `for`-Schleife

abgearbeitet und jeder Notenwert in eine Dauer in Millisekunden umgerechnet. Die Dauer errechnet sich aus dem Tempo der Übung und dem jeweiligen Notenwert. Eine Viertelnote dauert, bei einem Tempo von 60 BPM, 1000 Millisekunden; eine Achtelnote dauert 500 ms und eine Sechzehntelnote 250 ms usw. Die Tonhöhen mit ihrem Notenwert und der Dauer in Millisekunden werden abschließend in eine Tabelle geschrieben.

Möchte der Nutzer mit dem Klopfen des Rhythmus beginnen, muss er immer 2 Takte, auf allen Zählzeiten vorzählen, d.h. er muss bei einem 4/4-Takt 8 Schläge vorzählen. Die Länge dieses Einzählens ist zunächst willkürlich gewählt; es würde auch ein Takt ausreichen, jedoch wird ein Musik-Stück i.d.R. über 2 Takte hinweg eingezählt. Das Einzählen ist jedoch essenziell, da der erste Schlag des Einzählens bei der Evaluation des Rhythmus als Startpunkt des 1 Taktes verwendet wird. Das Einzählen ermöglicht also das Festlegen eines Initialwerts, von dem aus die Richtigkeit der Nutzereingaben evaluiert wird.

Bei jedem Schlag des Nutzers auf die Eingabefläche wird ein Zeitstempel erstellt, der dann in das Array `userInputNotes` gespeichert wird. Besteht der Rhythmus wie oben aus 5 Noten, enthält das Array bei Abschluss der Übung also 13 Zeitstempel; 8 Schläge durch das Einzählen + 5 Schläge des Rhythmus.

Ist der Nutzer fertig, muss er die Übung durch einen entsprechenden Button beenden. Dieser Button ruft die Funktion `evaluateUserInput()` auf. In dieser Funktion wird der vom Nutzer eingegebene Rhythmus evaluiert. Hierzu wird der Abstand der Nutzereingaben zum Nullpunkt berechnet. Ein Beispiel soll dies näher erläutern:



*Abbildung 11: Ein einfacher Rhythmus*

Im gezeigten Rhythmus erfolgt ein Schlag auf der ersten Zählzeit, d.h. bei einem Tempo von 60 BPM, bei dem eine Viertelnote 1 Sekunde dauert, würde der erste Schlag des Rhythmus nach 8 Sekunden abgegeben werden, da der Nutzer 8 Viertelschläge vorzählen muss. Diese Vorzählzeit wird jedoch berechnet und immer von den Nutzereingaben abgezogen, d.h. der erste Schlag des Nutzers, der zum Rhythmus gehört, steht in diesem Fall im Array an der 9. Stelle (8.) und hat, nach Abzug der

Vorzählzeit, eine Differenz von 0 ms zum Referenzwert der Berechnung (erste Zählzeit des Takts). Natürlich ist es nahezu unmöglich, jede Note auf die Millisekunde genau zu treffen, deshalb wurde eine Toleranz von +/- 60 ms eingebaut.

Stimmt die Differenz der Nutzereingabe mit der Differenz in der Tabellenzeile der jeweiligen Note überein, so wird diese Zeile grün hinterlegt. Stimmt sie nicht überein oder wurde eine Eingabe trotz einer Pause in der Notation registriert, wird die Tabellenzeile rot hinterlegt.

### 8.3.9 Multiple-Choice-Test

Der Multiple-Choice-Test soll das gelernte Wissen des Nutzers festigen. Der Test wurde mithilfe von Radio-Buttons umgesetzt. Die Fragen und Antwortmöglichkeiten sind ebenfalls in der XML-Datei festgehalten.

Nachfolgend eine beispielhafte Frage in XML notiert:

```
<task type="multiplechoice">
  <question>Wie lang ist diese Note?</question>
  <picture>
    <filename>lesson_1_ganze.PNG</filename>
  </picture>
  <answers>
    <answer type="correct">4 Taktschläge</answer>
    <answer type="wrong">3 Taktschläge</answer>
    <answer type="wrong">2 Taktschläge</answer>
  </answers>
</task>
```

Die maximale Punktzahl eines Tests wird ebenfalls im XML notiert und bereits beim Import der Lektion im Objekt `musicateObject` gespeichert. Hat der Nutzer seine Antwort ausgewählt, muss er die Eingabe noch über einen Button bestätigen. Hierdurch wird die Funktion `validateTestAnswer()` aufgerufen. Diese Funktion überprüft, ob der Nutzer die richtige Antwort gegeben hat und gibt dem Nutzer unmittelbares Feedback. Hat der Nutzer die Frage richtig beantwortet, werden die erreichten Punkte erhöht.

### 8.3.10 Hörbeispiele & Metronom

*musicate* verwendet MIDI-Dateien um Hörbeispiele und das Metronom in der Übung umzusetzen. Hierfür wird die Media API von PhoneGap verwendet. Der Link zur Datei wird als Attribut in das `<div>`-Element des Hörbeispiels geschrieben. Alle Hörbeispiele

haben die HTML-Klasse `soundSample`.

```
<div class="soundSample" data-midiFile="lesson_1_ganze.mid">  
  </div>
```

Bei einem Klick auf das Soundsample wird die Funktion `playAudio(src)` aufgerufen und der Dateiname der Midi-Datei wird übergeben.

Zu Beginn einer Übung wird die gleiche Funktion aufgerufen, nur wird hier der Dateiname der Metronom-MIDI-Datei übergeben. Bei Beendigung der Übung soll auch das Metronom wieder verstummen. Hierfür wird die Funktion `stopAudio()` verwendet.

## 9 Schwierigkeiten

In diesem Kapitel werden einige der Probleme erläutert die bei der Implementierung aufgetreten sind.

### 9.1 HTML-basierter Sampler & Metronom

Zu Beginn des Projekts war *musicate* sowohl als Web-App als auch als mobile App geplant. Aus diesem Grund und wegen der besseren Entwicklungsbedingungen, wurden viele Funktionen für normale Desktop-Browser geschrieben. Entsprechend sollte auch die Wiedergabe von Sounds über den Browser erfolgen. Das HTML-Element `<audio>` schien hierfür gut geeignet. In Chrome (Desktop) funktionierte die Wiedergabe von Rhythmen einwandfrei, auf dem Android-Testgerät kam es jedoch zu Aussetzern und zu Ungenauigkeiten bei den Rhythmen. Aus diesem Grund musste für die mobile Version auf die Media API von PhoneGap zurückgegriffen werden.

### 9.2 HTML-basierte Darstellung der Notation

Zu Beginn des Projekts war angedacht die Notationsbeispiele direkt im Browser zu generieren und nicht auf Grafiken zurückzugreifen. Hierfür gibt es entsprechende JavaScript-Libraries. Ein Beispiel hierfür ist VexFlow. VexFlow unterstützt die Generierung von Notenbildern in SVG und über HTML5 Canvas. Erste Tests waren erfolgreich, jedoch erwies sich die Umsetzung komplexerer Notenbilder als sehr schwierig und zeitintensiv. Aus diesem Grund wurde auf die Generierung der Notenbilder per JavaScript verzichtet und stattdessen Bilder in die Lektionen integriert.

### 9.3 Debugging

Selten macht eine Funktion von Anfang an das, was der Entwickler sich wünscht. Die Identifikation und Beseitigung von Fehlern im JavaScript ist durch Hilfswerkzeuge wie die „Chrome Developer Tools“ oder „JSHint“ relativ effektiv.

Sobald eine Funktion jedoch auf dem mobilen Endgerät ausgeführt wird, können viele hilfreiche Funktionen dieser Tools nicht mehr eingesetzt werden. Zwar liefert die Konsole der Android-Debugging-Bridge auch Nachrichten, die über `console.log` ausgegeben werden, jedoch sind sie nur als Strings verfügbar. Soll zum Beispiel ein Objekt und dessen Inhalte ausgegeben werden, zeigt die Konsole nur den Namen des Objekts an, jedoch nicht die Inhalte. Aus diesem Grund sollte die Entwicklung von nicht-gerätespezifischen Code auf dem Rechner vorgenommen werden, da so lange von den Annehmlichkeiten der oben genannten Tools profitiert werden kann.

Soll jedoch auf APIs von Android zugegriffen werden, muss direkt auf dem Gerät getestet werden. Kommt es hier zu Fehlern, liefert die Konsole meistens eine aussagekräftige Fehlermeldung oder einen Fehlercode, der nach einer kurzen Google-Recherche meist auch eine Lösung liefert.

## 10 Weiterentwicklung & Zukunftsaussichten

In diesem Kapitel werden einige Aspekte der App aufgezeigt, die in der Zukunft noch weiter ausgebaut und getestet werden könnten.

### 10.1 Tests auf weiteren Geräten

Die Entwicklung von *musicate* fand ausschließlich auf einem LG Nexus 4 statt. Es liegen also keine Informationen bzgl. der Kompatibilität mit anderen Endgeräten vor. Dies wäre in der Zukunft noch zu erledigen. Prinzipiell verwendet *musicate* jedoch keine besonders ausgefallenen Funktionen des Geräts, wodurch eine grundsätzliche Kompatibilität gegeben sein sollte. Bei performanceschwachen Geräten könnte es jedoch wegen Effekten und Animationen zu Problemen kommen.

*musicate* speichert Daten auf dem Gerät. Hierzu werden absolute Pfade verwendet. Diese Adressierung funktionierte auf dem Testgerät einwandfrei. Auf anderen Geräten, könnte es jedoch zu Problemen kommen. Dies muss vor der weiteren Entwicklung unbedingt getestet werden.

### 10.2 Usability-Tests

*musicate* wurde bis jetzt nur in geringem Ausmaß während der Entwicklung getestet. Die Tests wurden außerdem ausschließlich vom Entwicklungsteam durchgeführt. Deshalb sollten in Zukunft noch Usability-Tests durchgeführt werden. In diesen Tests sollte zum einen die generelle Usability der App untersucht werden. Hier sollte getestet werden, ob die Nutzer mit der Navigation und der Terminologie zu Recht kommen.

Noch wichtiger wäre es jedoch, die Rhythmus-Übungen auf deren Realisierbarkeit zu überprüfen. In der Entwicklung waren die Übungen, jedoch sind beide Mitglieder des Entwicklungsteams Musiker und gehören deshalb nicht zur intendierten Zielgruppe der App. Vor allem die Toleranz beim Eintippen von Rhythmen ist zu überprüfen. Diese liegt momentan bei +/- 60 ms. In Tests zeigte sich, dass die Abweichung, abhängig vom Tempo, durchaus hoch sein kann. +/- 60ms ist wahrscheinlich der maximale Wert, der verwendet werden kann. Ob auch ein kleinerer Wert möglich ist, müsste durch entsprechende Tests in Erfahrung gebracht werden.

### 10.3 Umsetzung auf anderen Plattformen

Schon zu Beginn der Entwicklung war Android die primäre Plattform der App. Dies lag hauptsächlich an der Verfügbarkeit eines Testgeräts. Da *musicate* mithilfe von

PhoneGap umgesetzt ist, ist es prinzipiell möglich, die App auch auf anderen Plattformen wie iOS zu erstellen, da das JavaScript auf allen Geräten funktioniert und die APIs von PhoneGap meisten Geräte übergreifend vorhanden sind.

Wie bereits erwähnt speichert *musicate* die Daten der Lektionen auf dem internen Speicher des Geräts. Hierzu wird ein absoluter Pfad konstruiert, der Android-spezifisch ist. Um also *musicate* auf anderen Plattformen installieren zu können, müsste in Erfahrung gebracht werden, ob es möglich ist, auf diesen Plattformen Daten lokal zu sichern und wie sich die Pfade zu diesen Dateien darstellen.

#### **10.4 Zwischenspeichern von Fortschritten**

Momentan werden die aktuellen Fortschritte bei der Bearbeitung der Lektionen nicht festgehalten. Wenn also ein Nutzer bereits 50% der Tests abgeschlossen hat, und die App verlässt, werden diese Fortschritte nicht gespeichert. Grundsätzlich sind jedoch alle Grundlagen zur Speicherung der Daten vorhanden. Hierfür wird das Objekt *musicateObject* verwendet. Es wird über HTML5 *localStorage* in den Speicher des Geräts geschrieben und ist dauerhaft verfügbar.

#### **10.5 Ausbau der Übungen**

Momentan sind zwei verschiedene Übungstypen umgesetzt: Das Tippen der Takte und das Bestimmen der Vorzeichen. Für einige Lektionen sind diese Übungen jedoch nicht geeignet. In Lektion 5 und 6 werden die verschiedenen Tonhöhen behandelt. Als Übung zum Aufbau prozeduralen Wissens eignen sich Aufgaben, wo Noten im Notensystem eingezeichnet werden müssen.

In Lektion 7 werden Dur- und Moll-Tonleitern behandelt. Hier wäre es sinnvoll Hörübungen zum Erkennen der Tonarten zu realisieren.

#### **10.6 Ausbau Test**

Die Tests am Ende einer Lektion sind momentan als Multiple Choice-Aufgaben umgesetzt. Um eine Auflockerung dieser Struktur zu erreichen, wäre Aufgabentypen in Form von Drag'n'Drop passend. Verschiedene Notenbilder könnten z. B. Der richtigen Benennung zugeordnet werden.

## 11 Fazit

Im Rahmen der Veranstaltung Media Engineering bearbeiteten wir ein Projekt, das die Konzeption und Umsetzung einer App zum Lernen von Musiktheorie umfasste. Die gemeinsame Arbeit am Projekt schätzen wir als sehr positiv ein, da wir uns im Laufe des Semesters immer wieder gegenseitig motivieren konnten und sich unsere Kompetenzen und Interessen für das gewählte Thema sehr gut ergänzten. Die Aufgabenverteilung verlief sehr gut und war ausgeglichen.

Zu Beginn des Semesters wuchs der Umfang unseres Projektes schnell an. Aus dem mit dem Projektthema verbundenen Interesse und unserem Vorwissen auf diesem Gebiet resultierten immer wieder neue Ideen, die uns bei der Umsetzung halfen, jedoch auch den Umfang sehr ansteigen ließen. Im Laufe des Projektes mussten wir daher einsehen, dass eine Kürzung der Inhalte der Qualität des Gesamtprojektes zugute kam.

Trotz alledem investierten wir viel Zeit in die Konzeption der App, die Ausarbeitung der Inhalte und die Implementierung, sodass andere vorgesehene Projektphasen wie die Evaluierung der App leider auf nach die Projektabgabe verschoben werden mussten. (Siehe Projektplan, Anhang)

Die Bearbeitung dieses Projektes war jedoch nicht nur eine zeitliche Herausforderung. Die selbstständige Erarbeitung und Konzeption der App ließ Raum für kreative Lösungsansätze und verschaffte Einblicke sowohl in das spannende Gebiet der Mediendidaktik als auch in die Programmierung von Apps. Das erarbeitete Wissen auf diesen Gebieten schätzen wir nicht nur als interessant sondern auch sehr nützlich für das spätere Studium und Beruf ein.

In Zukunft werden wir versuchen, uns auch außerhalb des Studiums mit der App zu beschäftigen und die genannten Weiterentwicklungen umzusetzen.

## 12 Literaturverzeichnis

GfK (2012): „Fast jeder zweite Jugendliche nutzt Smartphone. Studie der GfK zu mobilen Endgeräten“

<<http://www.gfk.com/de/news-und-events/presse/pressemitteilungen/Seiten/Fast-jeder-zweite-Jugendliche-nutzt-Smartphone.aspx>>

Kerres, Michael (2012): Mediendidaktik. Konzeption und Entwicklung mediengestützter Lernangebote. 3. Auflage. München: Oldenbourg Verlag.

Statista GmbH 1 (2012): „Anzahl der Smartphone-Nutzer in Deutschland in den Jahren 2009 bis 2012 (in Millionen)“. <<http://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenuutzer-in-deutschland-seit-2010/>> [Stand: Juli 2013. Zugriff: 01.07.2013]

Statista GmbH 2 (2012): „Marktanteile der führenden Betriebssysteme an der Smartphone-Nutzung in Deutschland von Dezember 2011 bis März 2013)“. <<http://de.statista.com/statistik/daten/studie/170408/umfrage/marktanteile-der-betriebssysteme-fuer-smartphones-in-deutschland/>> [Stand: Juli 2013. Zugriff: 01.07.2013]

# Anhang

## UI-Sketches

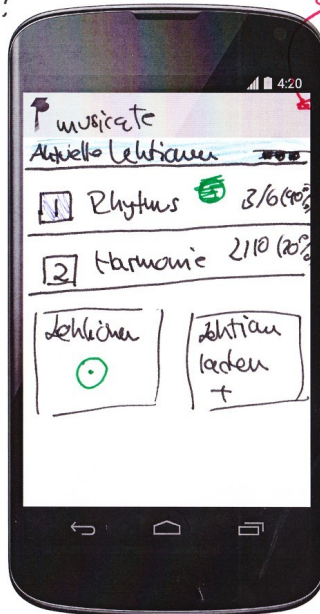
①

sequenz: lehrplananswahl

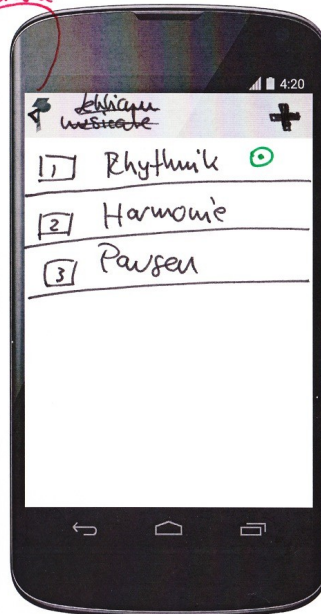
zurück



① Startseite  
↳ beim ersten Öffnen



① Startseite  
↳ beim erneuten Öffnen

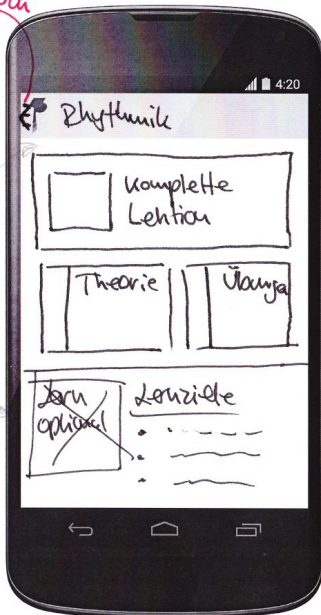


② Lehrplan's Übersicht

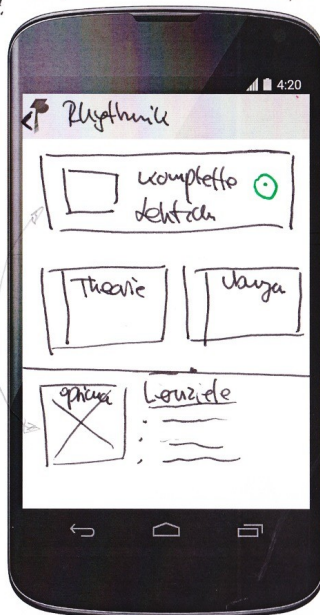
②

zurück

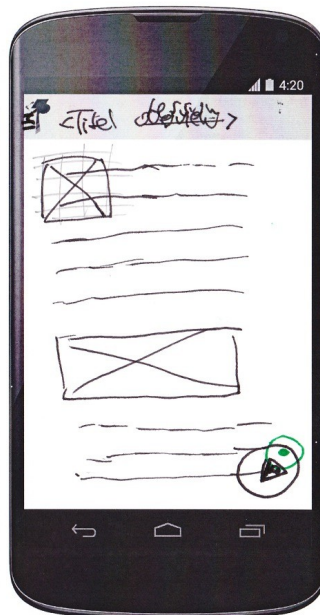
sequenz: komplette Lehrplan durchlaufen



③ Startseite Lehrplan



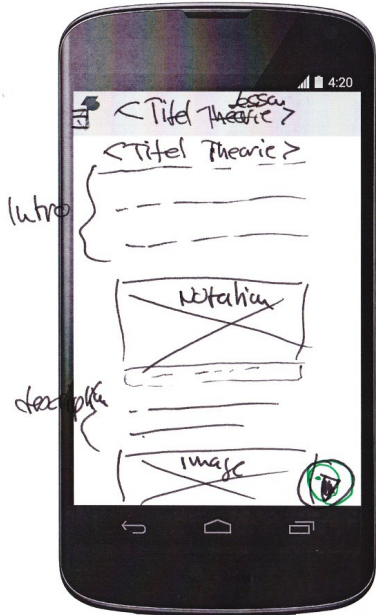
④ Startseite Lehrplan



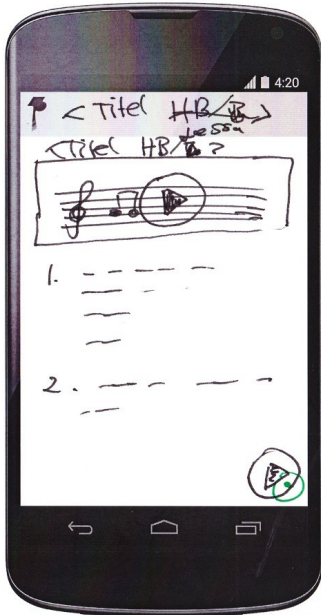
④ Overview

overview:  
→ wip einer Seite, scrollen

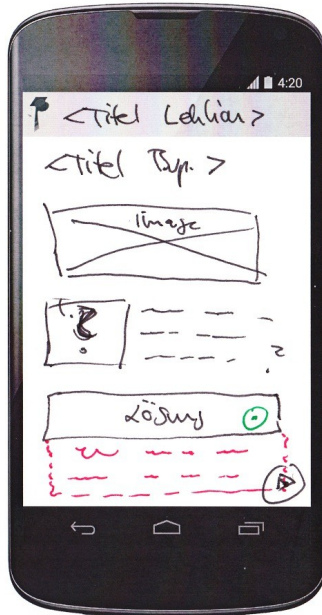
3



③ Information Unit



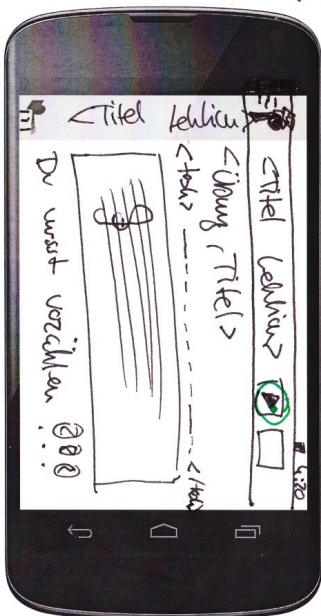
④ a) Fundament mit HB/B



④b

4

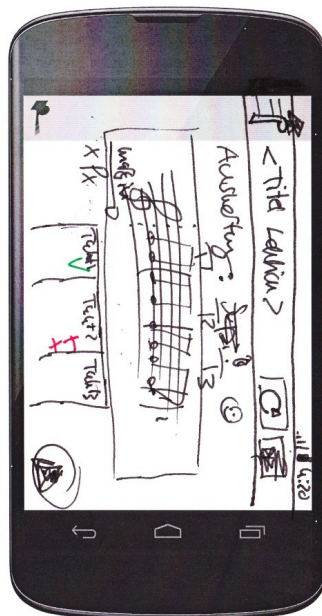
Problem lösen?



⑤ Übung Einführung



⑥ Übung deutsche

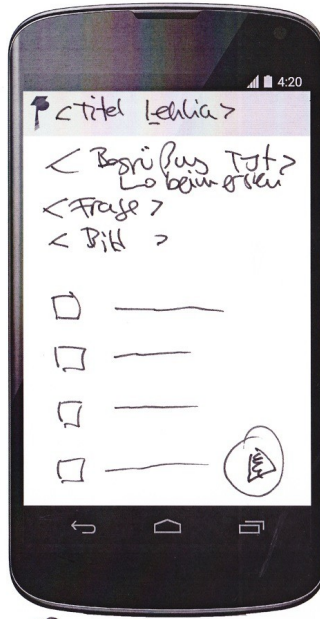


⑦ Übung Antwort

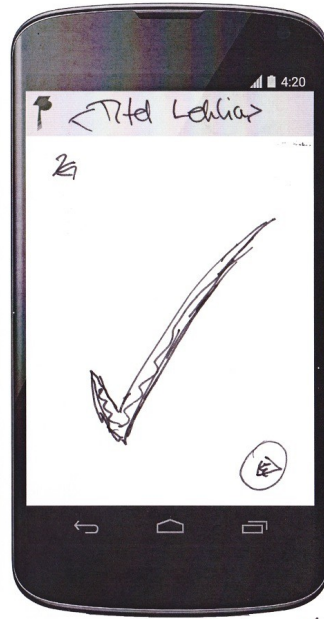
5



8) Test

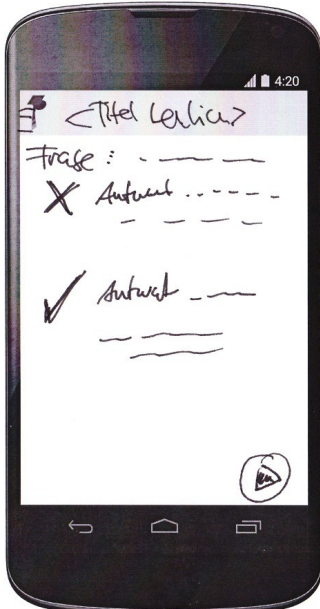


8) Test  
a) multiple choice



8) 9) Test, Antwort  
a) richtig

6



9) Test - Antwort  
b) falsch

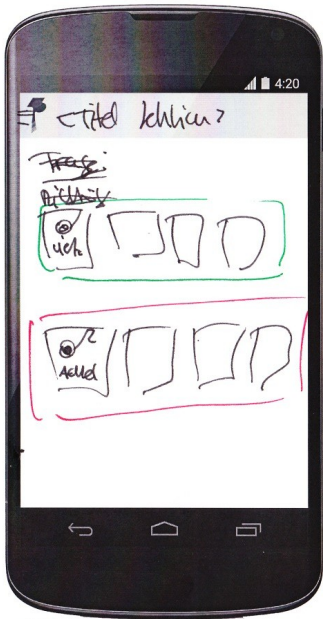


8) b) Das ist die

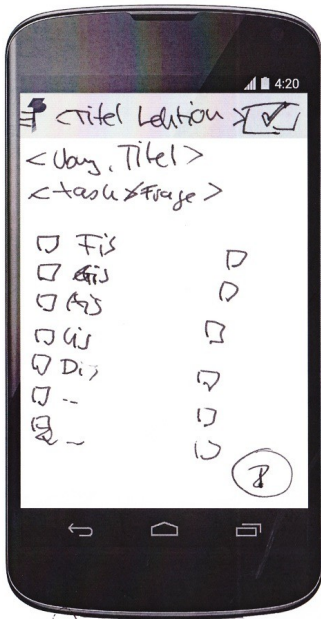


9) b) 1. Richtig

7



9 b) 1.

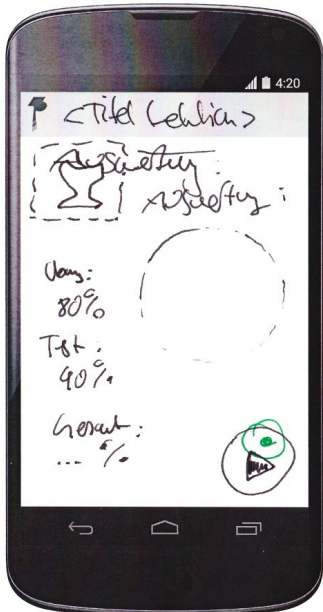


9 c) Lang Text  
eigene

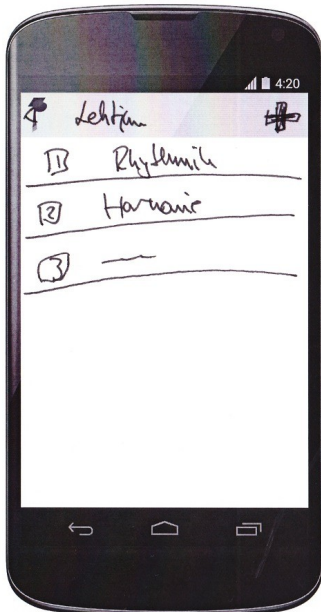


7b) Prüfung Was Aus wert

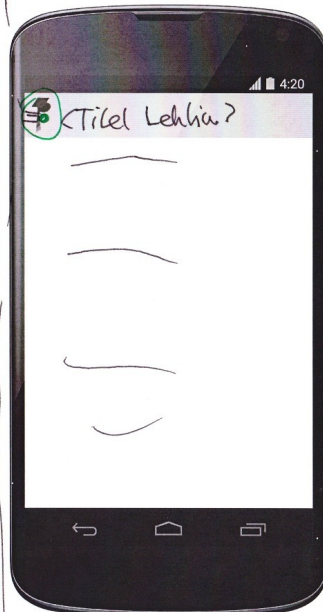
8



10) Auswertung Lehm



11) Lehm-Übersicht



1) Lehm

Sequenz: Sidebar Lehm

## Screenshots

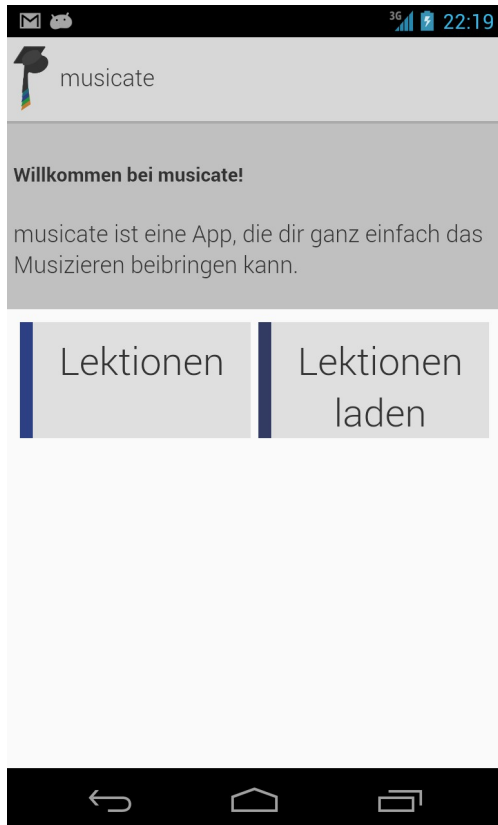


Abbildung 13: Startscreen

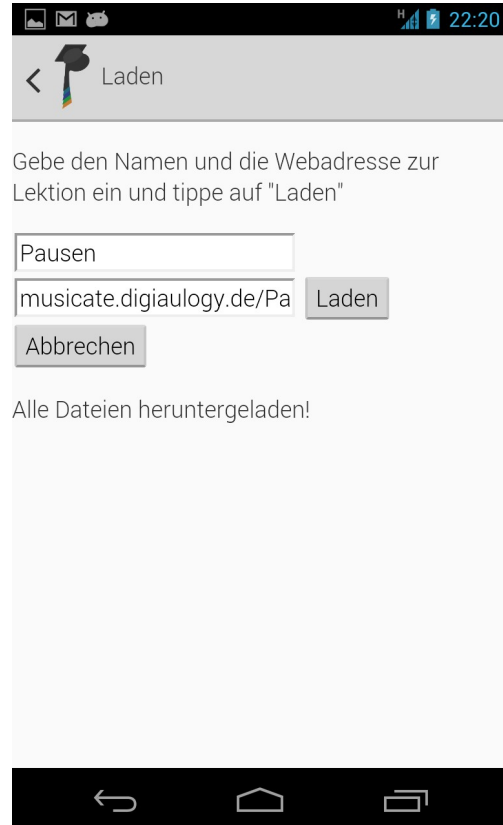


Abbildung 12: Lektion laden

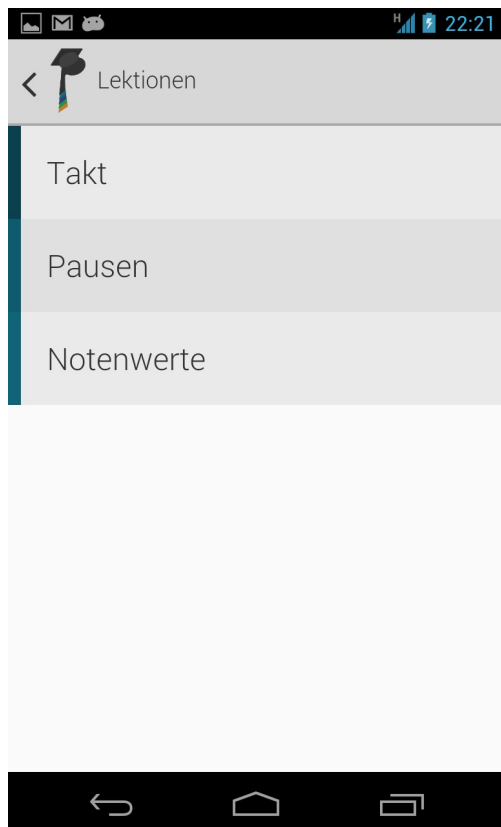


Abbildung 14: Lektionsübersicht

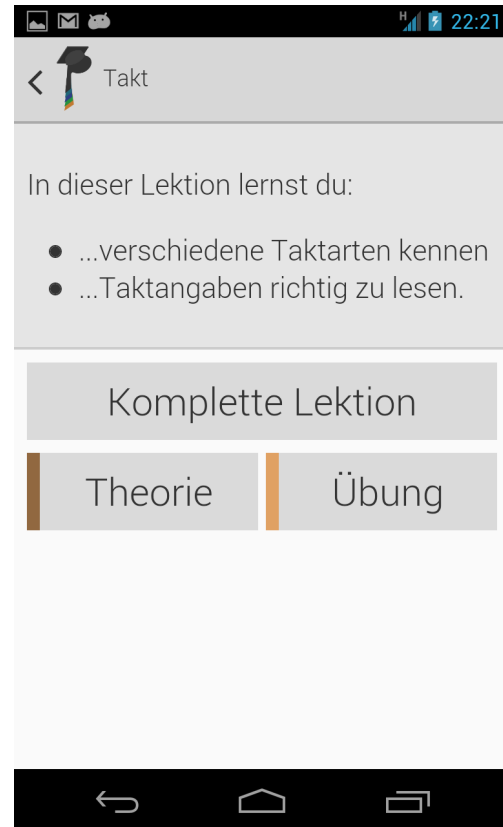


Abbildung 15: Lernziele

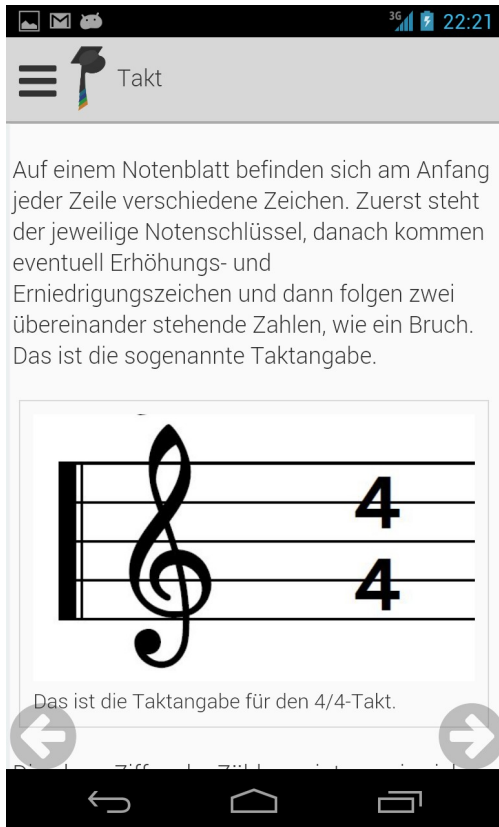


Abbildung 16: Überblick

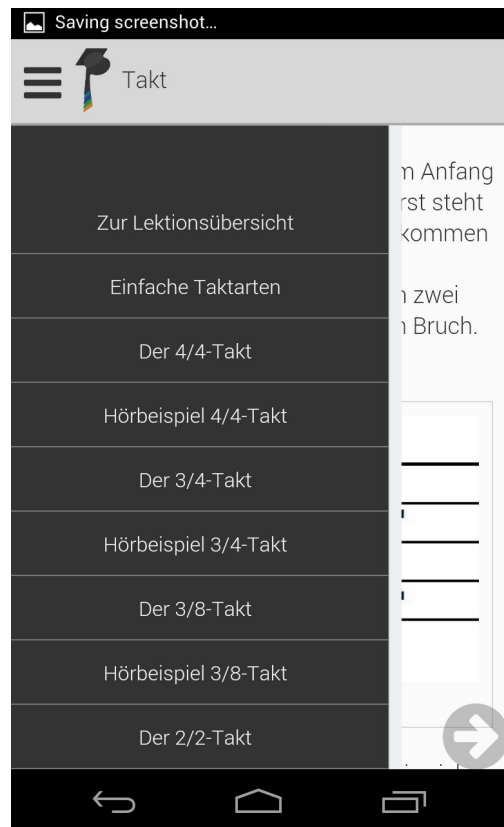


Abbildung 17: Menü

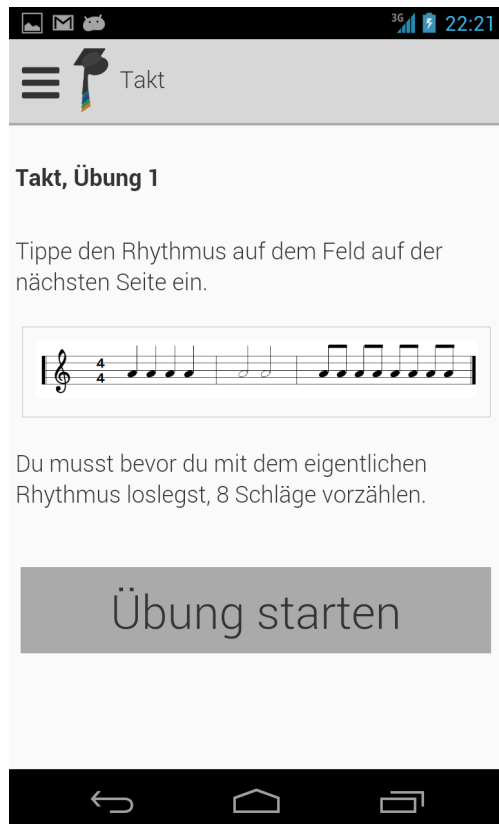


Abbildung 18: Übung 1/3

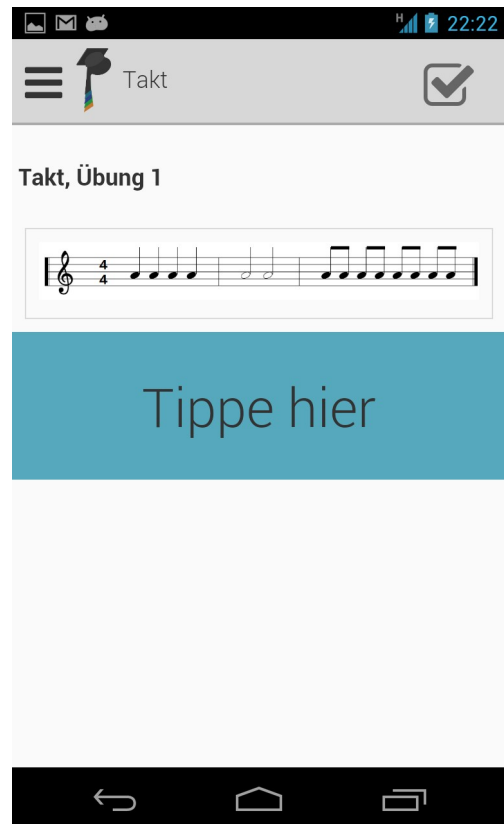


Abbildung 19: Übung 2/3

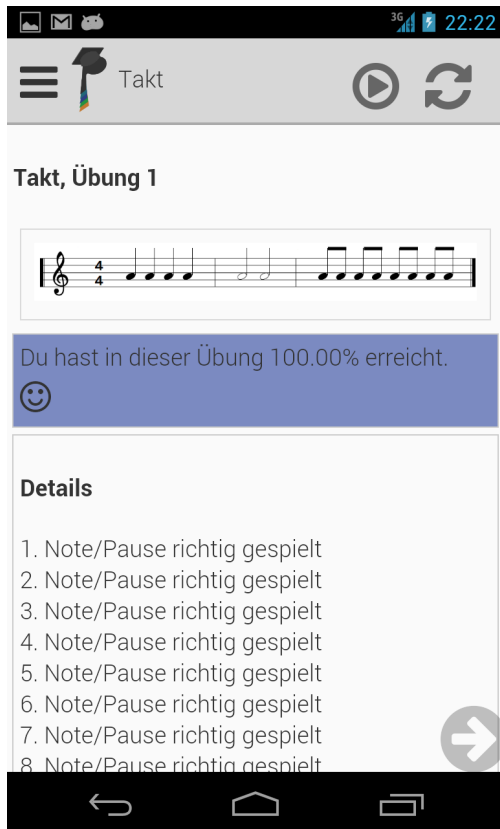


Abbildung 20: Übung 3/3

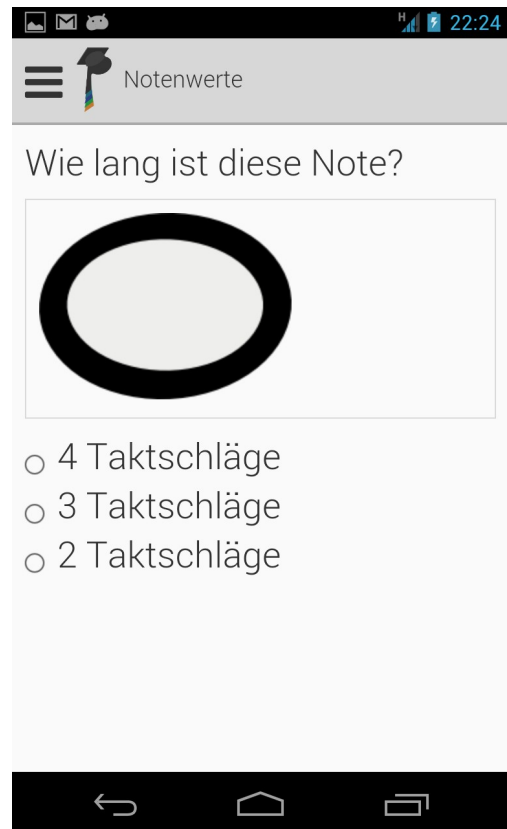


Abbildung 21: Testfrage

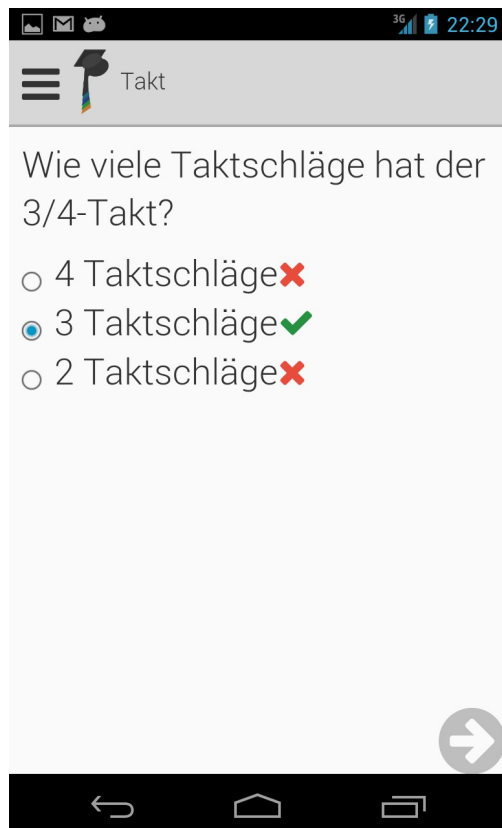


Abbildung 22: Testantwort



Abbildung 23: Auswertung