communicated by:
Lehr- und Forschungsgebiet Informatik 9

Prof Dr. Ulrik Schroeder

Diese Arbeit wurde vorgelegt am Lehr- und Forschungsgebiet Informatik 9
The present work was submitted to Learning Technologies Research Group

Konzeption und Entwicklung einer Android-Library zur Darstellung von Antworttypen
Conception and Development of an Android-Library for the Presentation of Response Types

Bachelorarbeit
Bachelor-Thesis

von / presented by

Sabouni, Walid
362896

Prof. Dr.-Ing. Ulrik Schroeder

Prof. Dr. Bernhard Rumpe

Aachen, 8. April 2019

Inhaltsverzeichnis

Danksagung

An dieser Stelle möchte ich mich bei allen Beteiligten bedanken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Bachelorarbeit beigetragen haben.
Ein besonderer Dank gebührt Svenja Noichl, die mich bei der Themenfindung und der anschließenden Betreuung der Bachelorarbeit unterstützt hat.
Weiterhin bedanke ich mich bei Herrn Prof. Dr.-Ing. Ulrik Schroeder vom Lehr- und Forschungsgebiet Informatik 9, für die Genehmigung der Arbeit und die Möglichkeit, in diesem Themengebiet meine Bachelorarbeit verfassen zu dürfen. Ich danke Herrn Prof. Dr. Bernhard Rumpe vom Lehrstuhl Software Engineering für die Betreuung als Zweitgutachter.
Nicht weniger Dank gilt meiner Familie und meinen Freunden, die mich besonders zu dieser Zeit immer tatkräftig unterstützt haben und mir zahlreiche Tipps und Anregungen verschafft haben.
Mein Herzlichen Dank gilt zu einem großen Teil Lasse. Er stand mir von Anfang an bei, hat mich stets aktiv mit seinen Tipps unterstützt und hat sich viel Zeit genommen, diese Arbeit auf grammatikalische und sprachliche Fehler zu überprüfen.
Ebenfalls möchte ich mich bei allen Probanden der Studie bedanken, die zum Erfolg dieser Abschlussarbeit beigetragen haben.

Kurzfassung

Es existiert kaum ein Bereich der modernen Gesellschaft, der nicht durch die Digitalisierung berührt wird [34]. Gerade leben die jüngeren Generationen in dieser digitalisierten Lebenswelt, während älteren Menschen mit diesen neueren Techniken nicht groß geworden sind und weniger Berührungspunkte damit haben [30]. Das Interagieren mit digitalen Fragen und Antworten, etwa in Form einer E-Klausuren [34] sowie digitale Fragebögen [23] und ähnliche Anwendungen gehören ebenfalls zu dieser Digitalisierung. Außerdem hat Android heutzutage den größten Marktanteil in der Welt der Smartphones [17].
Infolgedessen werden in dieser Arbeit verschiedene Darstellungsmöglichkeiten von Frage- und Antworttypen auf Android-Geräten ermittelt und in Prototypen implementiert und dann auf Verständnis von unterschiedlichen Altersgruppen geprüft. Basierend auf den Ergebnissen werden die bevorzugten Darstellungsvarianten in einer Android-Bibliothek umgesetzt, sodass diese von Entwicklern einfach, schnell und einheitlich in verschiedenen Applikationen anwendbar ist.

Teil I

Einleitung und Einführung

Kapitel 1 Einleitung

Digitale Technologien sind für die meisten Menschen zu einem festen Bestandteil ihres Lebens geworden [14]. Egal ob Smartphone, Tablet oder soziale Netzwerke, die Digitalisierung hat unser Leben von Grund auf verändert.
Smartphones sind im Alltag die meistgenutzten Geräte für den Internetzugang [19]. Außerdem haben im Jahr 2018 etwa 88 Prozent aller weltweit an Endverbraucher verkauften Smartphones das Betriebssystem Android [31]. Die Digitalisierung hat viele Bereiche getroffen, vor allem den Bildungsbereich z.B. durch die Anwendung elektronischer Prüfungen oder E-Tests [34], aber auch im Bereich Studiendurchführung z.B. durch die Anwendung von digitalisierten Fragebögen [23]. Die Gemeinsamkeit dieser Anwendungen ist, dass sie Fragen und Antworten erzeugen, speichern und verwalten.
Unterschiedliche Altersgruppen können unterschiedliche Ansprüche an digitalen Anwendungen haben. So nutzen ältere Menschen moderne technische Geräte wie das Smartphone seltener als jüngere Menschen [29].
Deshalb bietet es sich an, die Ansprüche unterschiedlicher Altersgruppen an die Darstellung von Fragen und Antworten auf Android-Geräten zu untersuchen und eine Android-Bibliothek zu entwickeln, um die Darstellung verschiedener Frage- und Antworttypen für unterschiedliche Altersgruppen zu vereinfachen.

1.1. Zielsetzung

Ziel dieser Arbeit ist es verschiedene Antworttypen für digitale Fragebögen, Quizzes oder Lernangebote zu ermitteln und dann entsprechend unterschiedliche Darstellungsvarianten für diese Fragen- und Antworttypen auf Android-Geräten zu untersuchen. Dabei behandelt diese Arbeit unter anderem die Fragen, welche Fragen- und Antworttypen es gibt, wie diese in mobilen Anwendungen dargestellt werden können und ob sich diese Darstellungen auch für unterschiedliche Altersgruppen eignen, nämlich Menschen, die über 50 Jahre alt, zwischen 30 und 50 und jünger als 30 sind. Anschließend daran werden die Varianten in einer Android Bibliothek umgesetzt, damit diese schnell und einheitlich in der Entwicklung von Applikationen verwendet werden können.

1.2. Aufbau der Arbeit

Dieses Kapitel umfasst eine Motivation für das Projekt. Ebenso werden die Ziele dieser Arbeit genannt. Der Rest der Arbeit ist wie folgt organisiert: Zuerst werden in Kapitel 2 die wichtigsten relevanten theoretischen und technischen Grundlagen kurz eingeführt. Danach werden verwandte Arbeiten und ähnliche Android Bibliotheken sowie deren Funktionalitäten in Kapitel 3 vorgestellt. In Kapitel 4 wird die Konzeption der Kategorisierung von Frage- und Antworttypen nach deren Darstellungsmöglichkeiten erläutert. Außerdem werden zu jeder Kategorie Prototypen in der Form einer Applikation implementiert. Anschließend daran werden diese Prototypen durch eine Studie von Personen unterschiedlicher Zielgruppen evaluiert. In Kapitel 5 wird dann der Aufbau dieser Studie sowie ihre Durchführung erklärt. Ferner werden die Ergebnisse und deren Auswertung

präsentiert. Anschließend daran und basierend auf die Ergebnisse der Studie wird in Kapitel 6 unsere Android-Bibliothek ***RespoLib*** sowie ihre Implementation und Installation präsentiert. Im Schluss dieser Arbeit befindet sich in Kapitel 7 die Schlussfolgerung der Bachelorarbeit sowie ein Ausblick, in welchem die Zukunftsvisionen und möglichen Erweiterungen der ***RespoLib*** vorgestellt werden.

Teil II

Grundlagen

Kapitel 2 Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen sowie die technischen Grundlagen beschrieben, welche für das Verständnis dieser Arbeit erforderlich sind. Bezüglich der theoretischen Grundlagen werden die Fragentypen erläutert, hinsichtlich der technischen Grundlagen werden wichtige Konzepte der Android-Entwicklung eingeführt.

2.1. Theoretische Grundlagen

In dem theoretischen Grundlagen werden die Fragentypen sowie die Gestaltungsrichtlinien für Seniorinnen und Senioren. Diese werden in den folgenden Abschnitten näher erläutert.

2.1.1. Fragentypen

Fragen können in zwei Kategorien unterteilt werden, diejenigen die auf den Inhalt bzw. die Zielrichtung abzielen, beispielsweise Fragen bzgl. Meinungen, Überzeugung und Verhalten, oder dem gegenüberstehend jene, die die Form behandeln wie zum Beispiel offene, halboffene und geschlossene Fragen [26]. Allerdings ist die erste Art der Unterscheidung für diese Arbeit weniger von Bedeutung, weshalb nun die Kategorisierung nach der Form der Frage erläutert wird.

Offene Frage

Bei der offenen Frage wird keine Antwortmöglichkeiten vorgegeben sondern die befragte Person beantwortet in ihren eigenen Worten [27]. Ein konkretes Beispiel ist in Abbildung 2.1 dargestellt. Dieser Fragetyp bietet den Vorteil, dass die befragte Person die Möglichkeit hat, so zu sprechen, wie sie es gewohnt sind. Allerdings ist die Auswertung von solchen Fragen schwer ist, da die Verkodung der Antworten zusätzlichen Aufwand erfordert [26].

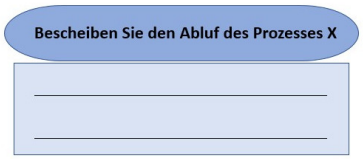


Abbildung 2.1.: Beispiel für eine offene Frage.

Geschlossene Frage

Bei einer geschlossenen Frage handelt es sich um eine Frage mit begrenzten vorgegebenen Antwortmöglichkeiten. Zum Beispiel bei Fragen, wo aus mehreren Antwortalternativen genau eine Antwort (Einfachauswahl) oder mehrere (Mehrfachauswahl) ausgewählt werden kann [27]. Dieser Fragetyp bietet den Vorteil, dass er die Auswertung sehr einfach macht [26]. Bei geschlossenen Fragen kann eine Trennung zwischen dichotomen und polytomen Fragen getroffen werden. Bei dichotomen Fragen (bekannt auch als binäre Fragen) händelt es sich um eine Frage, die genau zwei komplementäre Ausprägungen hat. Die Belegung solcher Fragen ist äquivalent zu Boolschen Variablen in der Informatik. Beispiele für Antwortvorgaben bei dichotomen Fragen sind richtig/falsch oder männlich/weiblich[27]. Ein konkretes Beispiel ist in Abbildung 2.2 dargestellt.
Bei polytomen Fragen händelt es sich um eine Frage mit mindestens drei Antwortoptionen, die bevorzugt als Auswahlliste umgesetzt werden [27]. Wie zum Beispiel in Abbildung 2.3

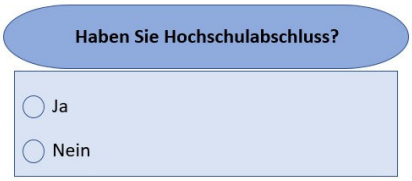


Abbildung 2.2.: Beispiel für eine dichotome Frage.

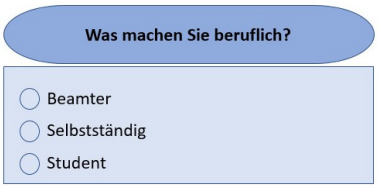


Abbildung 2.3.: Beispiel für eine polytome Frage.

Halboffene Frage

Halboffene Fragen werden als eine erweiterte Form der geschlossenen Fragen bezeichnet, bei der zwar Antwortmöglichkeiten vorgegeben werden aber trotzdem hat die Befragte Person die Möglichkeit eine eigene Antwort zu geben, die sie in ihren eigenen Worten formulieren kann [27]. Beispielsweise wird bei einer an sich geschlossenen Frage eine zusätzliche Kategorie z.B. "Sonstiges" angehängt, wie in Abbildung 2.4 zu sehen ist.

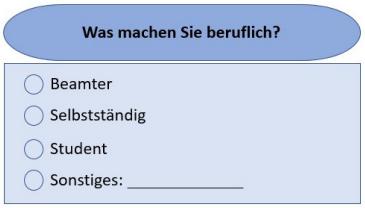


Abbildung 2.4.: Beispiel für eine halboffene Frage.

2.1.2. Gestaltungsaspekte für Seniorinnen und Senioren

Seniorinnen und Senioren haben besondere Bedürfnisse und Anforderungen an Smartphones und Tablets [12]. Diese wurden schon in mehreren wissenschaftlichen Arbeiten erforscht und durch die *Österreichische Forschungsförderungsgesellschaft* wurde ein Leitfaden zur Verbesserung der Nutzung mobiler Geräte für Seniorinnen und Senioren entwickelt [12]. Allerdings erleichtert die Beachtung dieses Leitfadens die Nutzung mobiler Geräte nicht nur für alte Menschen sondern auch für jüngere Menschen[12]. Hierzu werden einige wichtige Aspekte vorgestellt:

- **Icons**
  Es ist besonders wichtig, dass Buttons so groß wie möglich gestaltet sind, weil zu kleine Buttons zu Bedienungsproblemen führen. Deshalb müssen die Buttons mindestens 48 x 48 dp groß sein, um ein leichteres Antippen zu ermöglichen. Für eine gute Tauglichkeit wurde vorgeschlagen, dass Buttons immer umrandet oder hervorgehoben gestaltet werden [12]. Außerdem sollte immer ein Abstand zwischen verschiedenen Bedienelementen eingehalten werden. Für Android beträgt der Mindestabstand 8dp.

- **Schriftgröße**
  Mit dem Alter nimmt die Sehstärke bei den meisten Menschen ab [12]. Dies sollte bei der Gestaltung der Bedienelemente berücksichtigt werden. Die Schriftgröße sollte deshalb mindestens 11pt betragen [12]. Außerdem sollte die Schriftgröße individuell einstellbar sein, damit sie sich für unterschiedliche Altersgruppen anpassen lässt.

- **Farbe und Kontrast**
  Durch schlechte Auswahl der Farbe oder des Kontrasts kann die Sichtbarkeit von Bedienungselementen verschlechtert werden. Deswegen sollten die Kontraste und Farben lesefreundlich sein [12].

- **Nutzer-Eingabe**
  Die Eingaben der Nutzer, etwa bei der Nutzung eines Bedienelements müssen bestätigt werden. dies soll durch ein verstärktes Feedback erfolgen zum Beispiel durch Töne, Farbänderungen oder visuelle Bestätigungsnachricht [12].

2.2. Technische Grundlagen

In diesem Abschnitt werden die wichtigsten Begriffe und Techniken, die zur Implementierung unserer Bibliothek relevant sind, näher erläutert.

Softwarebibliothek

Eine Softwarebibliothek (Library) ist eine Sammlung von Daten und Programmiercode z.B. Klassen, Methoden und Funktoinen, die zur Entwicklung von Softwareprogrammen und Anwendungen verwendet wird. Bibliotheken werden entwickelt, um den Programmierer bei der Erstellung von Software zu unterstützen. [21]

Android Bibliothek

Bibliotheken in Android lassen sich in zwei Gruppen unterteilen. Diejenigen, die die Ausführung des Betriebssystems ermöglichen, und gegenüberstehend jene, die die Entwicklung von Applikationen unterstützen. Android stützt sich bei der Ausführung des Betriebssystems auf etwa hundert dynamisch geladene Bibliotheken [17]. Solche Bibliotheken sind nicht relevant für dieser Arbeit. Die zweite Art von Bibliotheken enthält normalerweise ein oder mehrere Pakete, die Sammlungen von .class-Dateien sind, und werden als .jar oder .class Dateien veröffentlicht [17]. Diese Bibliotheken werden bei der Entwickelung von Applikationen benutzt, damit diese einige vorprogrammierte Funktionalitäten haben können. Unsere Bibliothek gehört zu dieser Art von Bibliotheken.

Native Bibliotheken

Native Bibliotheken sind C/C++-Bibliotheken, die häufig aus der Open Source-Gemeinschaft stammen und die notwendigen Dienste für die Android Anwendungsschicht zur Verfügung stellen. Dazu zählt unter anderem SQLite. [13]

SQLite

SQLite ist eine Open Source-Datenbank, die stabil läuft und bei Kleingeräten, unter anderem Android-Geräte, beliebt ist. SQLite ist für den Einsatz in der Entwicklung von Android-Apps ausgezeichnet geeignet, weil sie ohne Konfigurationsaufwand einsetzbar ist [13]. Außerdem benötigt sie keinen Server um Daten zu speichern. Allerdings besteht auch die Möglichkeit sie leicht mit einem Server zu verbinden, um Daten auf einem Server zu speichern. Ein wichtiger Vorteil bei SQLite ist die Stabilität unter widrigen Umständen so gehen bei Stromausfall oder Speichermangel keine Daten verloren [3].

Android-Versionen und API-Levels

Im Zuge der Weiterentwicklung der Android-Plattform und der Veröffentlichung neuer Android-Versionen erhält jede Android-Version eine eindeutige ganzzahlige Kennung, den sogenannten API-Level (Application Programming Interface-Level). Daher entspricht jede Android-Version einem einzelnen Android API-Level. Außerdem bestimmt der API-Level, welche Android-Versionen eine Anwendung ausführen können und welche nicht [13]. Wenn etwa der API-Level einer Applikation auf 17 eingesetzt ist heißt das, dass die Anwendung dieser Applikation nur auf Android-Versionen mit dem angegebenen API-Level oder einem höheren API-Level funktioniert. In der folgenden Tabelle 2.1 sind Daten über die aktuell benutzten Android-Versionen sowie deren eindeutigen API-Level und die relative Anzahl von Geräten, die eine bestimmte Version der Android-Plattform ausführen, dargestellt.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.5%
4.3		18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1		22	14.4%
6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1		25	10.1%
8.0	Oreo	26	14.0%
8.1		27	7.5%

Tabelle 2.1.: Verteilung der aktuell benutzten Android-Geräte auf die Android-Versionen bzw. den API-Level. die Informationen wurden am 26.Oktober 2018 veröffentlicht [4].

Context

Der Application-Context ist eine Schnittstelle zu globalen Informationen über die Anwendungsumgebung und den Prozess, in dem alle Komponenten laufen. Er ermöglicht den Zugriff auf anwendungsspezifische Ressourcen und Klassen sowie den Aufruf von Vorgängen auf Anwendungsebene, wie z.B. das Starten von Aktivitäten, Senden und Empfangen von Absichten usw. [13].

Activity

Activity (Aktivität) bezeichnet ein einzelnes Fenster, in dem die App ihre Benutzeroberfläche zu einem Zeitpunkt zeichnet. Eine App besteht meistens aus mehreren Aktivitäten und im Allgemeinen implementiert eine Aktivität einen Bildschirm in einer App [13]. So kann beispielsweise eine der Aktivitäten in einer App den Einstellungsbildschirm implementieren, während eine andere Aktivität die Startseite implementiert.

2.2.1. Views und Layouts

Android organisiert seine UI-Elemente in Views und Layouts. Alles, was für den Nutzer sichtbar ist, zählt als View wie zum Beispiel Buttons, Labels, Textfelder usw.. Mehrere Views werden durch Layouts Organisiert. Wie zum Beispiel gruppieren einen Button und ein Label oder andere Elemente [13]. Layouts sind also Behälter für verschiedene Views und Layouts wie in Abbildung 2.5 zu sehen. Views werden in Android gelegentlich als Widgets bezeichnet. Android bietet verschiedene Layout, die ihre Kinder unterschiedlich anordnen. Dazu zählen beispielsweise ConstraintLayout, LinearLayout, FrameLayout, RelaiveLayout und AbsoluteLayout. Im folgenden werden die relevanten Layouts zu dieser Arbeit kurz eingeführt.

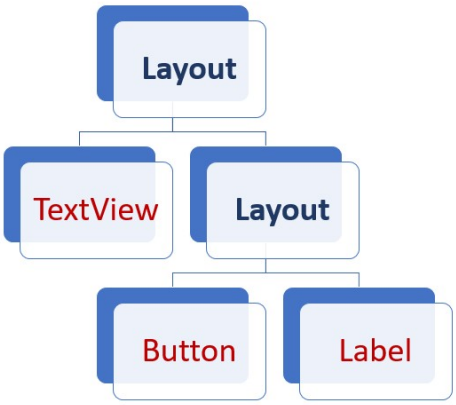


Abbildung 2.5.: Darstellung einer View Hierarchie, die ein UI-Layout definiert.

ConstraintLayout

ConstraintLayout ermöglicht es, komplexe Layouts mit einer flachen Ansichtshierarchie (keine verschachtelten Ansichtgruppen) zu erstellen. Innerhalb des ConstraintLayouts werden alle Views gemäß den Beziehungen zwischen den anderen Views und dem übergeordneten Layout gestaltet [5]. Ihr besonderer Vorteil besteht in der perfekten Unterstützung durch den Layout-Editor im Android Studio, die ein direktes Editieren der XML-Datei weitgehend überflüssig macht. Jedes Element in einem ConstraintLayout benötigt mindestens eine horizontale und eine vertikale Beschränkung, um seine Position zu definieren [3].

LinearLayout

LinearLayout ist eine Ansichtsgruppe, die alle Kinder bzw. Views in eine einzige Richtung nebeneinander ausrichtet, entweder vertikal oder horizontal. [13]

RadioGroup

RadioGroup ist ein besonderes LinearLayout, die *Radiobuttons* enthalten kann, wobei bei der Auswahl eines *Radiobuttons* der vorherige *Radiobutton* automatisch deaktiviert wird. [6]

TableLayout

Ein Layout, das seine Kinder in Zeilen und Spalten anordnet. Ein TableLayout besteht aus einer Reihe von TableRow-Objekten, die jeweils eine Zeile definieren. TableRow hat viele Gemeinsamkeiten mit dem horizontal orientierten LinearLayout und zwar, dass in dem die Kinder bzw. Views horizontal nebeneinander angeordnet werden. [13]

FrameLayout

Ein FrameLayout blockiert einen Bereich auf dem Bildschirm und wird verwendet, um eine einzelne View anzuzeigen. Generell sollte FrameLayout eine einzelne View enthalten, da es sonst zur Überlappung von Views kommen kann [7]

ScrollView

ScrollView ist ein besonderes FarmeLayout, das das Scrollen durch seine Elemente ermöglicht. Allerdings kann ein ScrollView nur einen einzigen View als direktes Kind enthalten [8]

Teil III

Verwandte Arbeiten

Kapitel 3 Verwandte Arbeiten

Im folgenden werden wir verwandte Arbeiten näher betrachten, die sich mit ähnlichen Themen befassen. Die erste Arbeit befasst sich mit der Umsetzung von elektronischen Fragebögen zur Unterstützung von Psychologie Patienten auf mobilen Endgeräten. Die zweite Arbeit behandelt die Realisierung eines mobilen Frameworks zur Unterstützung tinnitusgeschädigter Patienten. Die dritte Arbeit befasst sich mit der Erstellung und Anwendung von elektronischen Klausuren in der Hochschullehre.
Außerdem werden in diesem Kapitel zwei ähnliche Open-Source Bibliotheken vorgestellt, die die Implementation von Fragebögen und Formularen auf Android-Geräten ermöglichen.

3.1. Fragebögen

In der Forschung werden Frage- und Antworttypen und deren Anwendungen oft behandelt. Die Erstellung und Auswertung von Fragebögen ist in der Forschung eine bewährte Methode zur Datenerhebung, da sich mithilfe eines Fragebogens vereinfacht zuverlässige Aussagen vieler Personen untersuchen lassen [28].

3.1.1. Erstellung der Fragebögen in Psychologie

In der Psychologie werden Fragebögen als Instrument zur Sammlung von Informationen eingesetzt. Allerdings wird der Großteil der in der Psychologie verwendeten Fragebögen immer noch auf Papier gefertigt [28], weil es zum einen einen Mangel an der IT-Kenntnis in diesem Bereich gibt und zum anderen es bei der Entwicklung von softwaregestützten Systemen höhere Kosten verursacht. Allerdings bringt dieser papierbasierte Ansatz viel Zusatzaufwand mit, vor allem bei der Analyse der erhobenen Daten, da diese manuell in digitale Auswertungstabellen übertragen werden müssen.
Infolgedessen führt das *Institut für Datenbanken und Informationssysteme der Universität Ulm mit der Klinischen Psychologie und der Klinischen Neuropsychologie der Universität Konstanz* ein Projekt durch, um den Prozess der Befragung von Patienten mit dem Einsatz von mobilen Endgeräten wie beispielsweise Tablets oder Smartphones zu automatisieren [27].
Hierzu wurden Frage- und Antwottypen behandelt. Bezüglich der Antwottypen wurde es festgestellt, dass die Wahl des Antworttyps mit dem gewünschten Messniveau, der Form der Frage und der Antwortquantität(dichotome oder polytome Frage) stark zusammenhängt. Außerdem wurden die folgenden Antworttypen aufgelistet und kurz mit möglichen Umsetzungen in grafischen Benutzerschnittstellen beschrieben:

Listing 3.1: übernommen von Scher14 und angepasst

- **Freitext-Antwortt:** Die Eingabe erfolgt in einem Textfeld [27].

- **Zahl:** Die Eingabe erfolgt im Nummernfeld oder durch Verschiebung eines Schiebereglers (Slider) auf den gewünschten Wert [27].
- **Datum:** Die Eingabe erfolgt in eine Formatvorlage aus mehreren Nummernfeldern oder durch die Wahl aus einem Datumswähler [27].
- **Dichotome:** Die Eingabe erfolgt mittels Schalter oder Auswahlliste [27].
- **Einfachauswahl:** Die Eingabe erfolgt durch die Auswahl genau einer Antwort mit exklusiven Optionsfeldern (Radio-Buttons) oder durch eine Auswahlliste (Dropdown-List) [27].
- **Mehrfachauswahl:** Die Eingabe erfolgt durch die Auswahl von beliebig vielen Antworten mit inklusiven Optionsfeldern (Checkboxes) [27].
- **Ratingskalen:** Die Eingabe erfolgt durch die Einstellung eines horizontalen Schiebereglers (Slider) [27].
- **Ranking:** Die Eingabe erfolgt durch Rangnummern zur Sortierung der vorgegebenen Antwortmöglichkeiten oder durch Verschieben der Antwortoptionen (Drag-and-Drop) [27].

Durch die Standardisierung der Gestaltung und Eingabemöglichkeiten entstehen unerwünschte Effekte [27]. Zum Beispiel bei einer Frage mit den Antwortmöglichkeiten Ja/Nein, die mittels Checkboxes dargestellt ist, ist also immer ein Zustand erfüllt, sodass implizit eine Vorbelegung der Variable existiert. Hat die Checkbox zum Beispiel zur Initialisierung den Zustand nicht ausgewählt, lässt sich nach der Beantwortung keine Aussage darüber machen, ob der Befragte die Frage mit Nein beantwortet oder übersprungen hat. Um diesem Problem zu vermieden, wurde ein Schalter mit drei Zuständen für das mobile Betriebssystem iOS von Apple entwickelt (siehe Abbildung 3.1). Das Resultat ist ein sogenannter Tristate Switch. Die Eingabe der Antwort erfolgt durch die Einstellung des Schiebereglers aus dem neutralen Zustand in der Mitte auf die gewünschte Position [27]. Ein ähnliches Problem der Vorselektion von Antworten besteht bei Slidern. Wie zum Beispiel bei Ratingskalen mit einer impliziten Ausgangsstellung (Anker) der Schieberegler [27]. Für das Problem wurde auch ein Konzept entwickelt, in dem es eine neutrale Initialposition gibt und die Möglichkeit besteht, eine bereits eingegebene Antwort zurückzusetzen (siehe Abbildung 3.2).

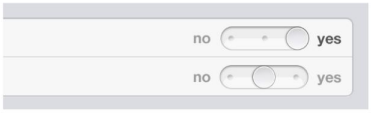


Abbildung 3.1.: Schalter mit neutralem Initialzustand, übernommen von [27].

Abbildung 3.2.: Schieberegler mit neutralem Initialzustand, übernommen von [27].

3.1.2. Design einer Anwendung zur Unterstützung Tinnitus-Patienten

Das Symptom Tinnitus bezeichnet die bewusste Wahrnehmung von Tönen oder Geräuschen, wobei es in der Umgebung keine echte physikalische Ursache für das akustische Signal gibt [16]. Tinnitus kann nur vom Betroffenen beschrieben werden. Deshalb ist eine genaue Messung dieser Schwankungen nur schwer möglich [16]. Aufgrund dessen wurde in der Arbeit [16] die App *Track Your Tinnitus* für Android und IOS entwickelt, das die Betroffenen ermöglicht, die individuellen Schwankungen ihrer Wahrnehmung des Tinnitus zu überwachen [16]. Diese App erfasst und misst die Schwankungen durch gezielte Fragen. Außerdem soll die App dabei helfen, die Ursachen des Tinnitus besser registrieren zu können.
In dieser Arbeit geht es darum, die vorhandene App "track your tinnitus" nach den Design Styleguides der Android, IOS und Windows Phone Betriebssysteme neu zu designen und konzeptionieren [2]. Dabei sollen die Tinnitusbeobachtungen einheitlich und einfachst möglich dargestellt werden. Um das zu realisieren, wurden mehrere Darstellungsmöglichkeiten ermittelt und dementsprechend Personen durch eine Online Umfrage befragt, welche der verschiedenen Darstellungsmöglichkeiten sie als am besten und effizientesten empfunden haben. In der Umfrage wurden Nutzer über die Optik, Selbsterklärbarkeit und Bedienbarkeit der Fragen zum Hauptmenü, zu den Ja/Nein Fragen und den Fragen mit Auswahlmöglichkeit befragt [2]. Es wurden zwei Typen der Fragen behandelt: Ja/Nein Fragen und Fragen mit Auswahlmöglichkeit. Dabei wurden Ja/Nein Fragen mittels Checkboxes, RadioButtons und Dropdown-List dargestellt (siehe Abbildung 3.3), gegenüberstehend wurden die Fragen mit Auswahlmöglichkeit mittels CheckBoxes, RadioButtons, Ja/Nein Buttons und SwitchButtons dargestellt (siehe Abbildung 3.4).

Abbildung 3.3.: Verschiedene Möglichkeiten zur Beantwortung von Ja/Nein-Fragen, übernommen von [2].

Abbildung 3.4.: Fragen mit Auswahlmöglichkeit in verschiedenen Varianten, übernommen von [2].

Bei der Auswertung der Umfrage kam heraus, dass die Teilnehmer sich sowohl bei Ja/Nein Fragen als auch bei Fragen mit Auswahlmöglichkeit Checkboxes gegenüber die anderen Darstellungsmöglichkeiten [2] bevorzugen. Dementsprechend wurde diese als einheitliche Darstellungsmöglichkeit übernommen.

3.2. E-Learning

„E-Learning“ bezeichnet den Einsatz von verschiedenen modernen Medien, die vor allem auf neuen Technologien basieren, in den Lehr- und Lernprozessen zum Beispiel computerunterstützte Präsentationen, E-Klausuren, Lern- und Übungssoftware und Simulationssoftware [18]. E-Learning ist an deutschen Hochschulen mittlerweile in mehreren Formen weit verbreitet mit dem Ziel einer Verbesserung der Hochschullehre [35]. In der folgenden Arbeit wird die Gestaltung von E-Klausuren genauer vorgestellt.

3.2.1. E-Klausuren

Immer öfter wird der Unterricht mit modernen Lehrmethoden ergänzt [20]. Während Videos, interaktive Online-Sitzungen und webbasierte Übungen den Lernprozess unterstützen, werden digitale Tests angewendet, weil sie sowohl den Studierenden als auch den Lehrenden vielfältige Vorteile anbieten [33]. Ein großer Vorteil für die Studierenden besteht darin, dass sie die Ergebnisse ihrer Klausur in der Regel wesentlich schneller erhalten können. Außerdem können sie die eingegebenen Antworten während der Durchführung der Klausur schnell und einfach beliebig ändern, ohne Spuren zu hinterlassen [33]. Hinsichtlich der Lehrenden bieten sich auch viele Vorteile an, die sich in drei Gruppen teilen lassen, nämlich Vorteile bei der Erstellung, Durchführung und Auswertung von E-Klausuren. Bei der **Erstellung** steht in der Regel eine große Auswahl verschiedener Fragetypen sowie Darstellungen zur Verfügung [33]. Bei der **Durchführung** können die Fragenreihenfolge und/oder die Antwort-Optionen einfach randomisiert werden, um das Abschreiben während der Klausurdurchführung zu erschweren [33]. Andererseits gibt es bei der **Auswertung** den Vorteil, dass die Klausuren automatisch korrigiert werden können und damit eine große Zeiteinsparung erfolgt [32]. Es wird bei E-Klausuren zwischen zwei Aufgabenformate unterschieden, nämlich geschlossenen und offenen Aufgaben [33], die sich in weitere Aufgabentypen unterteilen lassen. Geschlossene Aufgaben bezeichnen Aufgaben, auf die mögliche Antworten vorgegeben sind, während offene Aufgaben sich auf Aufgaben, in denen der Lernende die jeweilige Antwort selbst erstellen muss, beziehen [32]. In den Arbeiten [33] und [32] wurden verschiedene geschlossene und offene Aufgabentypen vorgestellt. Dabei wurde jeweils erläutert, wie der Einsatz in E-Klausur Systemen aussehen könnte. In den Tabellen 3.1 und 3.2 gibt es eine Zusammenfassung der geschlossenen bzw. offenen Aufgabentypen sowie eine Beschreibung der Aufgabetypen und der möglichen Eingabeformen.
In der Tabelle 3.1 bezeichnet *Drag und Drop* die Eingabeform, bei der Elemente im Bereich des Bildschirms an einen bestimmten Zielort verschoben werden müssen. Außerdem bezeichnet *Matching* die Eingabeform, bei der Elemente von zwei gegenübergestellten Listen einander durch Anklicken durch Linien zugeordnet werden. In der Tabelle 3.2 ist zu berücksichtigen, dass die Eingabeformen bezüglich Antworten auf der offenen Fragen immer durch eine Texteingabe erfolgt.

3.3. Open-Source Bibliotheken

Als Open Source wird eine Software bezeichnet, deren Quelltext öffentlich ist und von Dritten eingesehen, geändert und genutzt werden kann. Open-Source-Software kann meistens kostenlos genutzt werden. Im Folgenden werden verschiedene Open-Source Android-Bibliotheken vorgestellt, die ähnliche Funktionalitäten wie unsere ***RespoLib*** beinhalten. Die folgenden Bibliotheken sind kostenlos und öffentlich auf GitHub zur Verfügung gestellt [9].

Aufgabeart	Beschreibung	Eingabeformat
True/False	z.B: Ja/Nein bzw. Richtig/Falsch-Fragen	Radiobuttons, Checkboxes
Auswahlaufgaben	Markierung einer oder mehrerer Antwortalternativen	Radiobuttons, Checkboxes
Bildmarkierungsaufgaben	Markierung eines oder mehrerer Bilder	Radiobuttons, Checkboxes
Zuordnungsaufgaben	Begriffe zuordnen	Drag & Drop , Matching
Sortieraufgaben	Begriffe oder Sätze in eine bestimmte Reihenfolge bringen	Drag & Drop, Pfeiltasten oder Eingabe von Positionsziffern
Lückentext-Aufgaben	Lücken mit Antwortalternativen	Dropdown-List

Tabelle 3.1.: Aufgabenart, Beschreibung und mögliche Eingabeformat von geschlossenen Aufgaben, zusammengefasst von [33] und [32]

Aufgabeart	Beschreibung
Kurztext	Kurze Antwort muss eingegeben werden
Langtext	Antwort in einer oder mehreren Zeilen muss eingegeben werden
Lückentext	Wie bei Kurztext aber mit bestimmten Context und Lücken
Teilmenge	Bestimmte Anzahl von Antworten aus der richtigen Antworten mussen eingegeben werden

Tabelle 3.2.: Aufgabenart und Beschreibung von offenen Aufgaben, zusammengefasst von [33] und [32]

3.3.1. SurveyLib

Mit Hilfe dieser Bibliothek können Fragen und Antworten im Form eines Fragebogens auf Android-Geräten dargestellt werden [1]. Das ziel dahinter ist an erster Stelle, Feedback von Nutzern zu sammeln. Die Bibliothek bietet eine statische Darstellung der Frage und zwar an der Oberseite vom Bildschirm kombiniert mit Antwortalternativen, die nach Auswahl von einer Eingabeform dargestellt werden. Die möglichen Antworttypen, die zum Auswahl stehen, sind *String*, *StringMultiline*, *Number*, *Radioboxes* und *Checkboxes* [1]. Dabei könnte mit dem Typ *String* die Antwort als kurzer Text in einer Zeile eingegeben werden, während *StringMultiline* es erlaubt, die Antwort in einer oder mehreren Zeilen einzugeben. Beim *Number* Typ besteht die Möglichkeit, die Antwort nur als eine oder mehrere Ziffern einzugeben. *Radioboxes* bezeichnet, dass genau eine Antwort mittels eines Radiobutton ausgewählt werden kann, während mittels *Checkboxes* mehrere Antworten gleichzeitig ausgewählt werden können. Die Eingabe der Frage, des Antworttyps und der Antwortmöglichkeiten erfolgt durch eine JSON Datei, in der auch die Möglichkeit besteht, extra Optionen einzugeben etwa eine extra Beschreibung der Frage oder ob die Eingabe der Antwort Pflicht ist. Außerdem könnte auch eingestellt werden, ob die Reihenfolge der Antwortalternativen zufällig sein sollte. Ein Beispiel für die Eingabe ist in Listing 3.2 zu sehen.

```
1 {
2   "questions": [
```

```
    {
4     "question_type": "Checkboxes",
      "question_title": "What were you hoping the XYZ mobile app would
         do?",
6     "description": "(Select all that apply)",
      "required": false,
8     "random_choices": false,
      "choices": [
10      "thing #1",
        "thing #2",
12      "thing #3",
        "thing #4"
14    ]
    }
16  ]
}
```

Listing 3.2: Eingabe der Frage und der Antwortmöglichkeiten in einer JSON Datei

Nach der Eingabe von Listing 3.2 sieht dann die Darstellung wie in Abbildung 3.5 aus.

Abbildung 3.5.: Darstellung eine Frage vom Typ Checkboxes mittels SurveyLib [1]

Eine Anpassung der Farbe der Frage bzw. Antwortmöglichkeiten ist ebenfalls möglich und zwar durch die Anwendung von HTML code in der JSON Datei bei den *question_type* bzw. *choices* Attributen [1].

Mit *ServyLib* kann zwar Frage und deren Antwortmöglichkeiten dargestellt werden, aber allerdings biete sie nur wenige Darstellungsmöglichkeiten an. Außerdem werden mit dieser Bibliothek keine Antworte gespeichert, was dazu führt, dass sie allein keinen sinnvollen Nutzen bringt.

3.3.2. Simple Questionnaire

Bibliothek, die Formulare auf Android-Geräte mit Hilfe von *Google Forms* darstellt [25]. *Google Formes* ist ein kostenloser, webbasierter Dienst, der von Google angeboten wird und in *Google Drive* integriert ist. Google biete neun unterschiedliche Fragetypen zur Auswahl an. Diese sind in Abbildung 3.6 dargestellt.

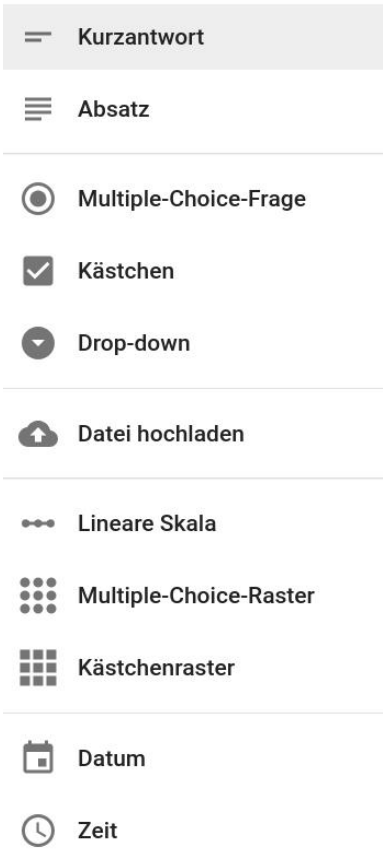


Abbildung 3.6.: Fragetypen in Google Forms

Diese Bibliothek dient dazu, mit *Google Formes* erstellte Formulare einfach in Android-Applikationen zu integrieren, in dem sie die folgenden Schritte durchführt:

- Ersetzen des CSS durch mobilfreundliche Ansicht.
- Formulare in ein WebView einbinden.
- Schließen des WebViews, wenn der Benutzer den Fragebogen beantwortet hat.
- Speichern, ob der Benutzer den Fragebogen bereits beantwortet hat.

Für die Nutzung dieser Bibliothek ist ein Google Konto erforderlich. Außerdem muss das Formular erst auf der *Google Forms*-Webseite erstellt werden und dann eine URL erstellen, die als Eingabe für die Bibliothek nötig ist. Das Speichern der Antworten erfolgt nur, wenn eine Verbindung mit dem Internet besteht, da die Antworten nur online auf *Google Drive* gespeichert werden können. In der Tabelle 3.3 befindet sich ein Vergleich zwischen beiden erwähnte Bibliotheken.

	SurveyLib	Simple Questionnaire
Antworttypen	String, StringMultiline, Number Radioboxes, Checkboxes	Siehe Abbildung 3.6
Daten Speichern	Unmöglich	Nur Online
Erstellung	Lokal im JSON Format	Auf Google-Forms Webseite
Design Optionen	Beschreibung, Farbe	Beschreibung, Bilder Video

Tabelle 3.3.: Vergleich zwischen den Bibliotheken

Teil IV

Konzeption und Entwurf

Kapitel 4 Konzeption

In diesem Kapitel wird die Konzeption der Kategorisierung der Fragen- und Antworttypen erläutert und die Kategorien sowie deren Umsetzungsmöglichkeiten auf Android-Geräten eingeführt.

4.1. Idee und Konzept

Die Idee ist, verschiedene Fragentypen mit unterschiedlichen Darstellungsmöglichkeiten in einer Android-Applikation zu Implementieren. Diese werden in unserer Arbeit Prototypen genannt. Mittels einer Studie werden die Prototypen dann von Personen unterschiedlicher Altersgruppen evaluiert. Basierend auf den Ergebnissen dieser Studie werden die von den Teilnehmern bevorzugten Varianten verbessert und in einer Android-Bibliothek umgesetzt.

4.2. Kategorisierung der Frage- und Antworttypen

Für die Auswahl der Fragen- und Antworttypen wurde berücksichtigt, dass sie für unterschiedliche Zwecke geeignet sind, beispielsweise für die Entwicklung von Lernanwendungen, Fragebögen und Formularen. Die Auswahl wurde von den Verwandten Arbeiten 3.1 und 3.3 inspiriert.
In unserer Arbeit werden Fragentypen in neuen unterschiedlichen Gruppen kategorisiert, nämlich:

- **Dichotome**
  Steht für Fragen, die genau zwei Komplementäre Ausprägungen als Antwortmöglichkeiten haben (siehe 2.1).
- **Single Choice**
  Steht für Fragen, bei denen aus mehreren Antwortalternativen genau eine Antwort ausgewählt werden kann.
- **Multiple Choice**
  Steht für Fragen, bei denen aus mehreren Antwortalternativen eine oder mehrere Antworten ausgewählt werden können.
- **Skalierung**
  Steht für Fragen, bei denen auf einer Skala von a bis b genau eine Antwort ausgewählt werden können.
- **Multi-Skalierung**
  Steht für Fragen, bei denen auf einer Skala von a bis b eine oder mehrere Antworten ausgewählt werden können.
- **Freie Antwort**
  Steht für Fragen, bei denen die Antwort als Texteingabe erfolgt.

- **Anordnen**
  Steht für Fragen, bei denen eine Reihe von Elementen in einer bestimmten Reihenfolge angeordnet können.
- **Zuordnen**
  Steht für Fragen, bei denen Elemente einer bestimmten Lücken oder einem Feld zugeordnet können.
- **Datum und Zeit**
  Steht für die Fragen, bei denen die Antwort ein Datum oder eine Zeit ist.

4.3. Darstellungsmöglichkeiten

Um verschiedene Darstellungsvarianten in einer Applikation umzusetzen, wurden verschiedene Bedienelemente und Techniken angewendet. Die Entscheidung für diese Bedienelemente wurde getroffen, da diese in der erste Stellte von Android-Studio zur Verfügung gestellt sind. Außerdem werden sie häufig verwendet, etwa bei der Erstellung von digitalen Fragebögen oder E-Klausuren, wie in der verwandten Arbeiten gezeigt wurde. die folgenden Bedienelemente und Techniken werden verwendet:

- **Radiobuttons**
  Bedienelemente in der Form eines Kreises, die bei der Auswahl markiert wird. In dieser Arbeit sind Radiobuttons auf eine Art und Weise implementiert, sodass gleichzeitig genau ein Radiobutton markiert werden kann (Siehe Abbildung 4.2).
- **Checkboxes**
  Bedienelemente in der Form eins Vierecks, die bei der Auswahl mit einem Tick markiert werden (Siehe Abbildung 4.2).
- **Switch**
  Bedienelement in der Form eines Schalters, der durch Ziehen markiert wird (Siehe Abbildung 4.2).

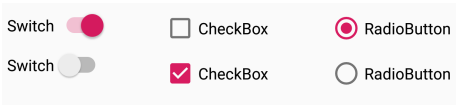


Abbildung 4.1.: Radiobuttons, Checkboxes und Switch.

- **Imagebutton**
  Bedienelement in der Form eines Bild mit einem Radiobutton. der durch Druüken ausgewählt und markiert wird. Dabei ist zu betrachten, dass sowhol das Bild als auch der Radiobutton auswählbar sind (Siehe Abbildung 4.3).

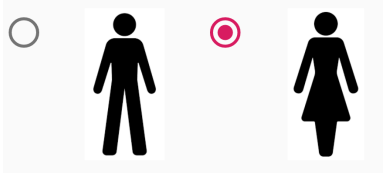


Abbildung 4.2.: Imagebutton

- **Dropdown-List**
  Bedienelement, das beim Drücken eine Liste von Auswahlmöglichkeiten anzeigt (Siehe Abbildung 4.3).

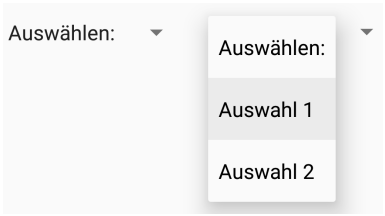


Abbildung 4.3.: Dropdown-List

- **Seekbar**
  Bedienelement in der Form einer Linie mit einem Start- und Endpunkt und einem Initialwert, der beliebig zwischen den Start- und Endpunkt durch bewegen umgestellt wird.

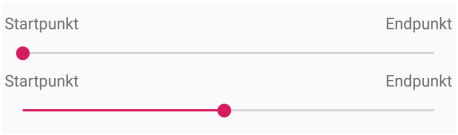


Abbildung 4.4.: Seekbar

- **Discrete Seekbar**
  Eine *Seekbar*, die mit diskreten Punkten in Bereiche zwischen dem Start- und Endpunkt geteilt ist (Siehe Abbildung 4.5).

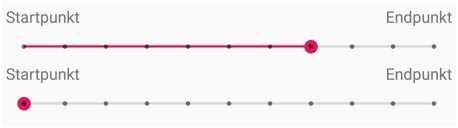


Abbildung 4.5.: discret Seekbar

- **Rating Stars**
  Bedienelemente in der Form eines Satzes von Sternen, die bei der Auswahl markiert werden (Siehe Abbildung 4.6).

Abbildung 4.6.: Rating Stars

- **Date-Time Picker**
  Bedienelement in der Form eines Kalenders bzw. einer Uhr, bei dem ein Datum bzw. eine Zeit ausgewählt werden kann (Siehe Abbildungen 4.7 4.8).

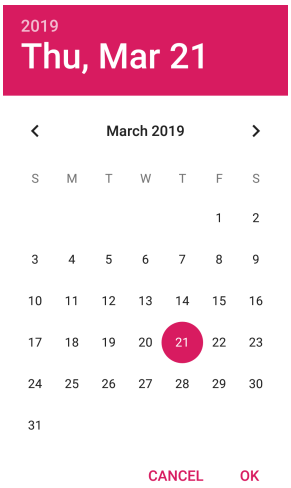


Abbildung 4.7.: Date Picker

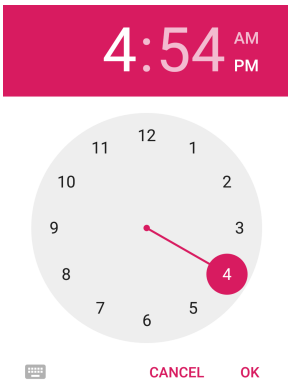


Abbildung 4.8.: Time Picker

- **Radiobuttons Matrix**
  Ein Satz von Radiobuttons in Form einer Matrix.
- **Textview**
  Bedienelement, in dem Text angezeigt werden kann.
- **Textbox**
  Bedienelement in der Form eines Kastens, bei dem die Eingabe mit der Tastatur erfolgt.
- **Paintbox**
  Bedienelement in der Form eines Kastens, bei dem die Eingabe mit dem Finger erfolgt.
- **Sort-List**
  Bedienelement in der Form einer Liste, die andere Bedienelemente enthält, die durch Ziehen und Ablegen sortiert werden können.
- **Drag&Drop**
  Drag and Drop steht für Ziehen und Ablegen. Es ist eine Technik, mit der Bedienelemente angeklickt und durch gedrückt halten verschoben werden können.

Mit Hilfe dieser Bedienelemente und Techniken wurden zu jedem Fragentyp mindestens zwei Darstellungsvarianten implementiert. Um festzustellen, welche Darstellungsvarianten in Frage kommen, wurden ähnliche Arbeiten sowie online Plattformen wie *SurveyMonkey* [11] und *QuestionPro* [10] betrachtet und anschließend die am häufigsten verwendeten Darstellungsvarianten prototypisch implementiert In der Tabelle 4.1 sind die Fragentypen mit deren Darstellungsvarianten zusammengefasst. Diese werden dann in einer Applikation umgesetzt und in der Durchführung der Studie im nächsten Kapitel verwendet, um die Darstellungsvarianten zu evaluieren.

Fragetypen	Darstellungsvarianten
Dichotome	Radiobuttons, Switch, Dropdown-List, Imagebuttons
Single Choice	Radiobuttons, Dropdown-List, Switch
Multiple Choice	Checkboxes, Dropdown-List, Switch
Skalierung	Seekbar, Discrete Seekbar, Radiobuttons Matrix, Rating Stars
Multi Skalierung	Checkboxes Matrix, Textboxes Matrix
Freie Antwort	Textbox , Paintbox
Anordnen	Drag&Drop, Sort-List
Zuordnen	Textboxes, Dropdown-List, Drag&Drop
Datum und Zeit	Date-Time Picker, Textboxes, Dropdown-List

Tabelle 4.1.: Fragetypen mit deren Darstellungsmöglichkeiten

Teil V

STUDIE

Kapitel 5 Studie

In diesem Kapitel wird die durchgeführte Studie vorgestellt. Dabei werden das Ziel, der Aufbau und die Durchführung erläutert. Außerdem wird die implementierte Applikation vorgestellt, die die Prototypen der Darstellungsvarianten hat und in dieser Studie verwendet wird. Anschließend werden die Ergebnisse und die Auswertung der Studie präsentiert. Außerdem werden das Feedback und die Wünsche der Teilnehmer in der Studie erwähnt. Im Schluss werden die wichtigsten Beobachtungen zusammengefasst.

5.1. Aufbau und Ziel der Studie

Das Ziel dieser Studie war es, die besten Darstellungsformen jedes Fragetyps herauszufinden, mit der Personen unterschiedlicher Altersgruppen am intuitivsten, einfachsten und effizientesten arbeiten können. Außerdem war das Ziel ein Feedback sowie Wünsche und Verbesserungsvorschläge zu bekommen, die dann bei der Implementierung unsere Bibliothek eingesetzt werden können.
Um das zu realisieren wurde eine Studie mit 14 Teilnehmern unterschiedlicher Altersgruppen durchgeführt. In der ersten Gruppe sind junge Menschen, die unter 30 Jahre Alt sind und in der zweiten Gruppe sind Personen, die zwischen 30 und 50 Jahre alt sind. In der dritten Gruppe sind Personen, die über 50 Jahre alt sind. Eine genauere Verteilung ist in der Tabelle 5.1 zu sehen. Das Ziel dieser Einteilung ist zu überprüfen, ob junge Menschen und ältere Menschen die selben Darstellungen bevorzugen, obwohl ältere Menschen generell Smartphones schlechter bedienen können[24]. Außerdem war das Ziel die Anforderungen unterschiedlicher Altersgruppen (falls es Unterschiede gibt) herauszufinden, um sie spätere bei der Implementierung unserer Bibliothek zu berücksichtigen. In dem Rest dieser Arbeit werden die Personen der ersten zwei Gruppen mit U50 (unter 50 sind) bezeichnet, während die Personen die über 50 sind mit Ü50 bezeichnet werden.

Alter der Altersgruppe	Anzahl der befragten Personen
<30	5
Zwischen 30 und 50	5
>50	4

Tabelle 5.1.: Verteilung der Teilnehmer nach dem Alter.

Hier wurden Personen zu jeder Darstellungsform gefragt, wie einfach sie die Eingabe der Antwort, wie klar sie die Darstellung der Antwortalternativen fanden und wie komplex die Darstellung im Allgemein empfunden wurde. Zusätzlich wurde am Ende gefragt, welche Darstellungsform sie für diese Art von Fragen bevorzugen. Die folgende Abbildung 5.1 zeigt einen Ausschnitt aus der Umfrage.

1-

	Leicht					Schwer
Die Eingabe der Antwort	1	2	3	4	5	6

2-

	Klar					Unklar
Die Darstellung der Antwortmöglichkeiten	1	2	3	4	5	6

3-

	Einfach					Komplex
Die Darstellung im Allgemein	1	2	3	4	5	6

4-

Welche Darstellung bevorzugen Sie	

Abbildung 5.1.: Ausschnitt von der Umfrage

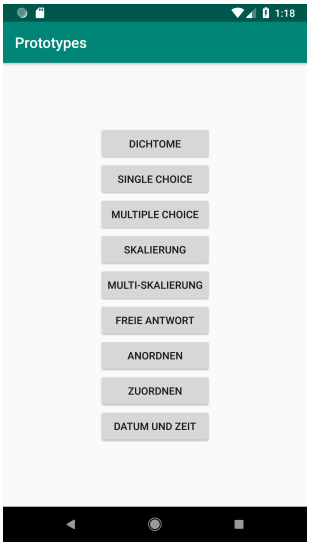


Abbildung 5.2.: Startseite der Applikation

5.1.1. Prototypen-Applikation

Um die unterschiedlichen Darstellungsvarianten evaluieren zu können, wurden die Fragentypen mit deren Darstellungsvarianten, wie in der Tabelle 4.1, in einer Android-Applikation umgesetzt, die in dieser Studie verwendet wird. Auf der Startseite der Applikation gibt es neun Tasten, die jeweils eine Gruppe der Fragentype bezeichnet (Siehe Abbildung 5.2). Jede Kategorie enthält genau eine Frage, die mit mehreren unterschiedlichen Darstellungsmöglichkeiten dargestellt ist. Jede Darstellung wird immer in einer neuen Seite (Activity) angezeigt. Außerdem gibt es in jeder Darstellung die Taste *Weiter*,

die auf die nächste Darstellung wechselt. Bei der letzten Darstellung gibt es die Möglichkeit mittels der Taste *Fertig* wieder auf die Startseite zu wechseln. In der Abbildung 5.3 und 5.4 sind die Start bzw. Endseite von der Kategorie *Dichotome* als Beispiel dargestellt. Eine Übersicht zu allen Darstellungsvarianten der Prototypen-Applikation befindet sich im Anhang B.

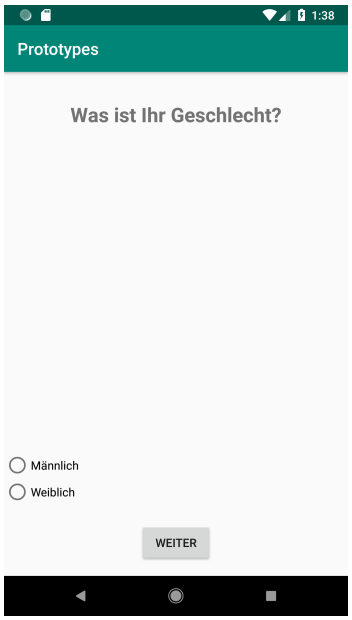


Abbildung 5.3.: Darstellung der dichotome Frage mit der Taste Weiter.

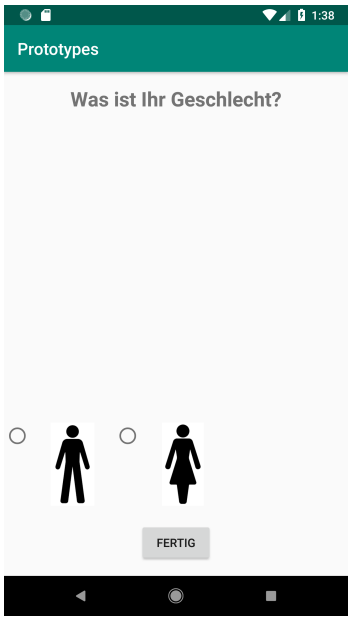


Abbildung 5.4.: Darstellung der dichotome Frage mit der Taste Fertig.

5.1.2. Durchführung

Für die Durchführung der Umfrage werden das Handy *Galaxy Note 4* von Samsung und das Tablet *Ipad Air* von Apple mit einem passenden Stift verwendet. Auf dem Handy wurde die Prototypen-Applikation installiert, während auf dem Ipad die Befragung mittels eines Fragebogens durchgeführt wurde. Bei der Durchführung der Studie ist es wichtig, dass alle Teilnehmerinnen und Teilnehmer ein möglichst ähnliche Vorkenntnisse in der Bedienung der Geräte haben, die während der Umfrage verwendet werden, da es das Auftreten von verfälschten Werten reduziert und den Vergleich der ermittelten Daten vereinfacht. So wurden ihnen anfangs notwendige Interaktionskonzepte des Handys und Ipads vorgeführt. Dazu gehört die Nutzung von -*Weiter* und *Fertig* Buttons in der Prototypen-Applikation und die Zurück Taste auf dem Handy, die die Vorangegangene Darstellungsform anzeigt und die Tastatur ausblendet falls diese eingeblendet ist. Außerdem wurde die Nutzung des Ipads und sein Stift zur Beantwortung des Fragebogens vorgeführt. Vor dem Start der Umfrage wurde eine Einleitung mit einer Kopie des Fragebogens gegeben. Zusätzlich wurde die Bedeutung der Fragen der Fragebogen sowie die Eingabe der Antwortmöglichkeiten erklärt.
Nachdem die App gestartet wurde, bekommt der Teilnehmer die Startseite zu sehen (Abbildung 5.2). Hierbei wählt der Teilnehmer eine Kategorie aus. Nach der Bearbeitung der Fragen einer Kategorie erhält er eine Übersicht zu den Darstellungsvarianten dieser Kategorie (Siehe Anhang X) und dementsprechend beantwortet er auf die Fragen 1 bis 3

(Siehe Abbildung 5.1) einmal für jede Darstellungsmöglichkeit. zu der letzten Darstellungsmöglichkeit bekommt er die Frage 4 zusätzlich zu Beantworten (Siehe Abbildung 5.1).
Die Durchführungszeit der Umfrage betrug durchschnittlich etwa 30 Minuten pro Teilnehmer, die sich folgendermaßen aufteilen: etwa 20 Minuten bei der Beantwortung der Fragebögen und etwa 10 Minuten für Feedback und Wünsche der Teilnehmer. Während der Durchführung der Studie wurde keine weitere Hilfe gegeben und sich auf die Intuition der Teilnehmer verlassen. Außerdem wurden währenddessen Beobachtungen mitgeschrieben. Dazu gehören unter anderem die Schwierigkeiten, die die Teilnehmer während der Bearbeitung der Fragen begegnet sind, wie zum Beispiel Schwierigkeiten bei der Eingabe der Antwort. Diese werden dann später im Fazit zusammengefasst.

5.2. Ergebnis

In den folgenden Abschnitten werden die Ergebnisse aller Teilnehmer an der Umfrage präsentiert. Hierbei werden nur die Ergebnisse der Frage *Welche Darstellung bevorzugen Sie?* veranschaulicht. Da nur diese für die Entscheidung, welche Darstellungen später in der Bibliothek implementiert werden, verwendet werden. Dann werden die anderen Ergebnisse lediglich als Hinweise interpretiert, die bei der Analyse der Ergebnisse verwendet werden. Die komplette Ergebnisse der Umfrage befindet sich im Anhang C. Außerdem werden in der folgenden Abschnitte einige interessante Fälle, wo es einen auffälligen Unterschied zwischen den Altersgruppen gibt, ausführlich beschrieben.

5.2.1. Ergebnisse der Fragen vom Typ *Dichotome*

Bei den dichotomen Fragen ist es zu bemerken, dass bei allen Teilnehmern die Darstellung mittels *Radiobuttons* die anderen Darstellungsvarianten dominiert (Siehe Abbildung 5.5). Einige Teilnehmer bezeichneten diese Darstellung als *übersichtlich und benötigt genau eine Schritt, um die Antwort zu wählen*.

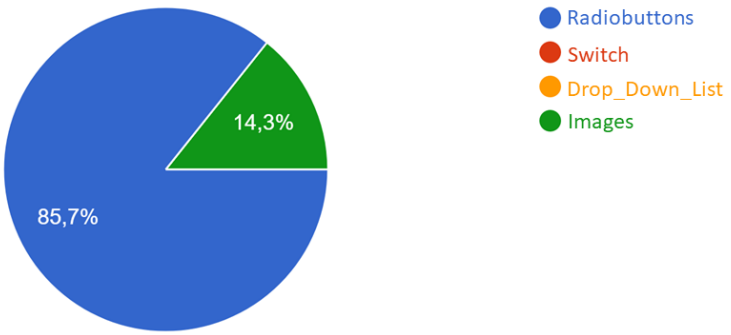


Abbildung 5.5.: bevorzugte Darstellung aller Teilnehmer bzgl. dichotome Fragen

5.2.2. Ergebnisse der Fragen von Typ *Single Choice*

Bei den Fragen von Typ *Single Choice* haben sich die meisten Teilnehmer ebenfalls für die Darstellung mit *Radiobuttons* entschieden, wie in Abbildung 5.6 zu sehen ist.

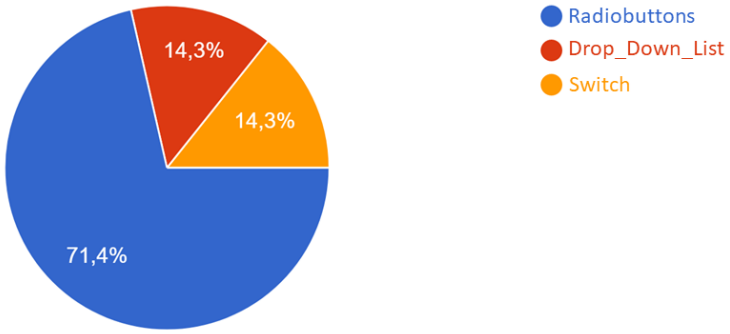


Abbildung 5.6.: bevorzugte Darstellung aller Teilnehmer bzgl. Fragen von Typ *Single Choice*

5.2.3. Ergebnisse der Fragen vom Typ *Multiple Choice*

Bei den Fragen vom Typ *Multiple Choice* haben die meisten Teilnehmer die Darstellung mittels *Checkboxes* bevorzugt. Einige Teilnehmer bezeichneten diese Darstellung als *einfach zu bedienen*. Diese sind in der Abbildung 5.7 dargestellt.

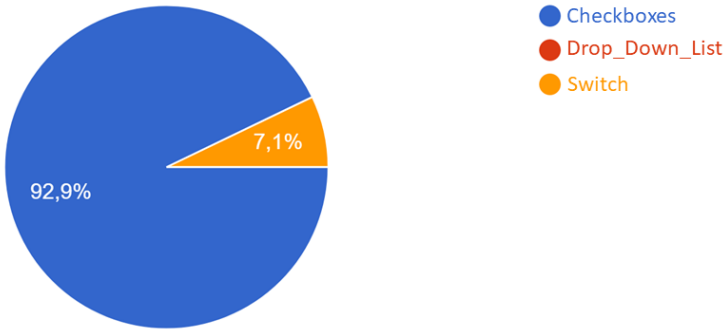


Abbildung 5.7.: bevorzugte Darstellung aller Teilnehmer bzgl. Fragen vom Typ *Multiple Choice*

5.2.4. Ergebnisse der Fragen vom Typ *Skalierung*

Bei den Fragen vom Typ *Skalierung* lässt sich aus den Ergebnissen kein eindeutiger Favorit herauslesen, wie in der Abbildung 5.8 zu sehen ist. Deswegen wurden die Ergebnisse der unterschiedlichen Altersgruppen getrennt betrachtet. Dabei wurde festgestellt, dass es einen großen Unterschied bei den Ergebnissen zwischen den Altersgruppen gab, wie in den Abbildungen 5.9 und 5.10 dargestellt ist, da die Altersgruppe Ü50 nur die Darstellungen mit *Seekbar* und *Radiobuttons Matix* bevorzugt hat, während die anderen Altersgruppen die Darstellungen mit *Discrete Seekbar* und *Rating Star* besser fanden. Gründe dafür sind, laut den Teilnehmern über 50, dass die Darstellung mit *Seekbar* weniger Beschriftungen hat, was sie übersichtlicher macht. Die anderen Altersgruppen tendieren hingegen dazu, solche Fragen mit der Darstellung *Rating Stars* zu bevorzugen.

5.2.5. Ergebnisse der Fragen vom Typ *Multi-Skalierung*

Bei den Fragen vom Typ *Multi-Skalierung* haben sich alle Teilnehmer geeinigt, dass die Darstellung mit *Checkboxes Matrix* besser als die andere Darstellung ist, wie in der Abbildung 5.12 zu sehen ist.

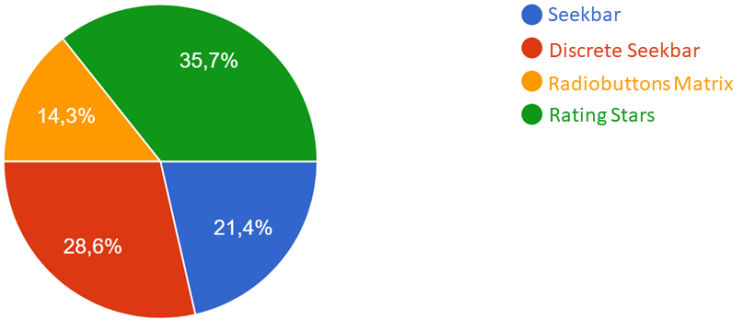


Abbildung 5.8.: bevorzugte Darstellung aller Teilnehmer bzgl. Fragen vom Typ *Skalierung*

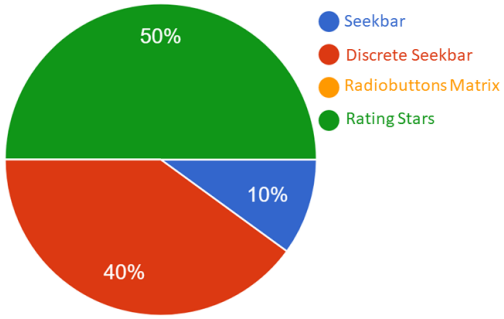


Abbildung 5.9.: Teilnehmer unter 50 Jahre alt

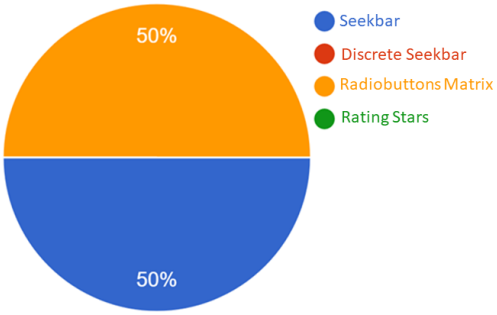


Abbildung 5.10.: Teilnehmer über 50 Jahre alt

Abbildung 5.11.: bevorzugte Darstellung bzgl. der Fragen vom Typ *Skalierung*.

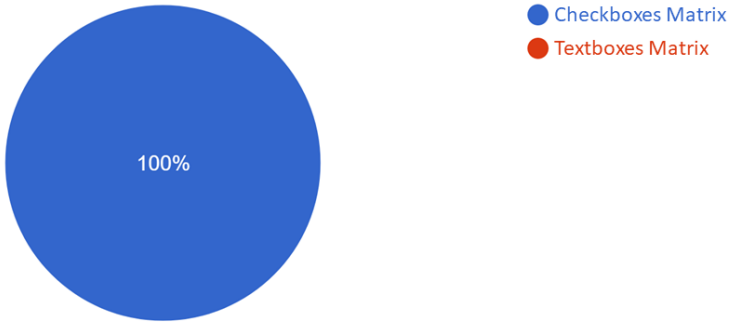


Abbildung 5.12.: bevorzugte Darstellung aller Teilnehmer bzgl. Fragen vom Typ *Multi-Skalierung*

5.2.6. Ergebnisse der Fragen vom Typ *Freie Antwort*

Bei den Fragen vom Typ *Freie Antwort* haben sich die meistens Teilnehmer für die Darstellung mit *Textbox* entschieden wie in Abbildung 5.13 dargestellt ist. Einige Teilnehmer fanden die Darstellung mit *Paintbox* effektiver, da sie die Möglichkeit bietet, Dinge genauer zu beschreiben, zum Beispiel durch die Eingabe eines Logos.

5.2.7. Ergebnisse der Fragen vom Typ *Anordnen*

Bei den Fragen vom Typ *Anordnen* haben die meisten Teilnehmer die Darstellung mit *Sort-List* bevorzugt, wie in der Abbildung 5.14 zu sehen ist. Einige Teilnehmer bezeichneten diese Darstellung als *weniger Schritte verlangt bis alle Elemente sortiert sind*.

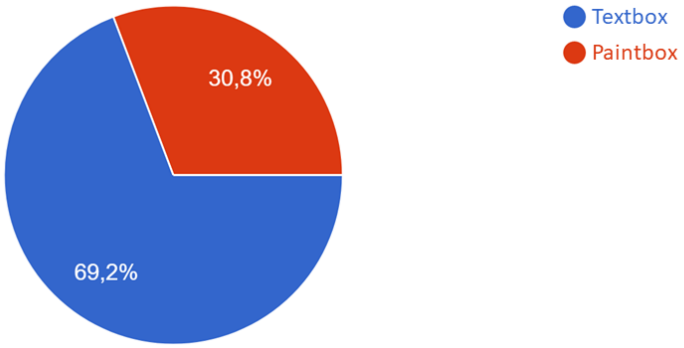


Abbildung 5.13.: bevorzugte Darstellung aller Teilnehmer bzgl. Fragen vom Typ *Freie Antwort*

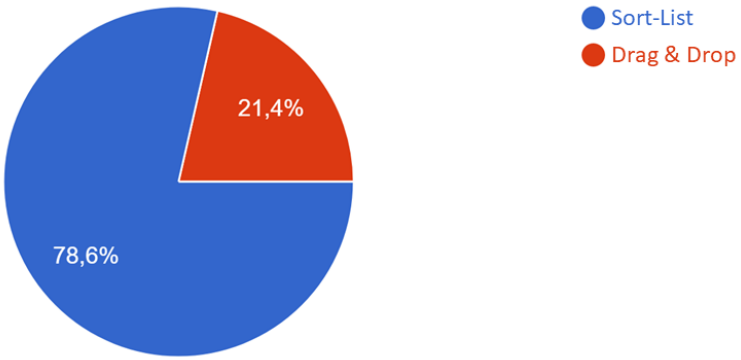


Abbildung 5.14.: bevorzugte Darstellung aller Teilnehmer bzgl. Fragen vom Typ *Anordnen*

5.2.8. Ergebnisse der Fragen vom Typ *Zuordnen*

Bei den Fragen vom Typ *Zuordnen* gab es ebenfalls keine eindeutige bevorzugte Darstellung von allen Teilnehmern, wie in Abbildung 5.15 zu sehen ist. Es wurde Außerdem festgestellt, dass es einen auffälligen Unterschied bei der Auswahl zwischen die Teilnehmer unterschiedlicher Altersgruppen gibt. Dabei haben sich die Personen aus der Altersgruppe U50 nur für die beiden Darstellungen *Dropdown-List* und *Drag & Drop* mit jeweils 60% und 40% entschieden, während 50% die Personen aus der Altersgruppe Ü50 sich für die Darstellung mit *Textboxes* entschieden haben, wie in den Abbildungen 5.16 und 5.17 dargestellt ist.

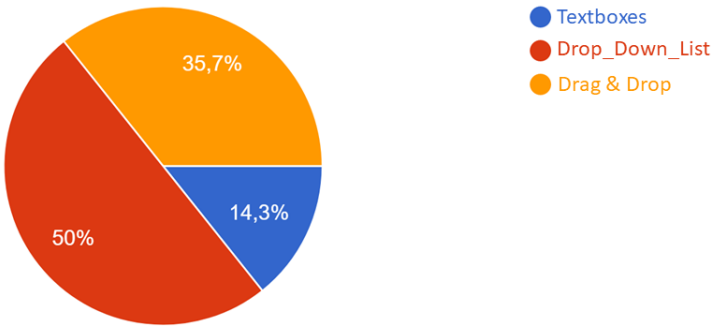


Abbildung 5.15.: bevorzugte Darstellung aller Teilnehmer bzgl. Fragen vom Typ *Zuordnen*

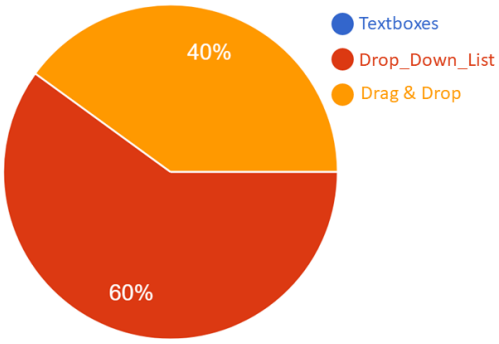


Abbildung 5.16.: Teilnehmer unter 50 Jahre alt

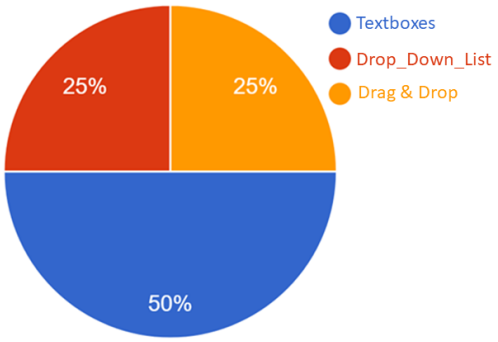


Abbildung 5.17.: Teilnehmer über 50 Jahre alt

Abbildung 5.18.: bevorzugte Darstellung bzgl. Fragen vom Typ *Zuordnen*

5.2.9. Ergebnisse der Fragen vom Typ *Datum und Zeit*

Auch bei den Fragen vom Typ *Datum und Zeit* lässt sich bei den Ergebnissen aller Teilnehmer keinen eindeutigen Favorit auslesen, wir in der Abbildung 5.19 zu sehen ist. Da die meisten Teilnehmer der Altersgruppen U50 die Darstellung mit *Date und Time Picker* bevorzugen, während die Teilnehmer der Altersgruppe Ü50 sich für die Darstellung mit *Textboxes* entschieden haben, wie in den Abbildungen 5.20 und 5.21 dargestellt ist.

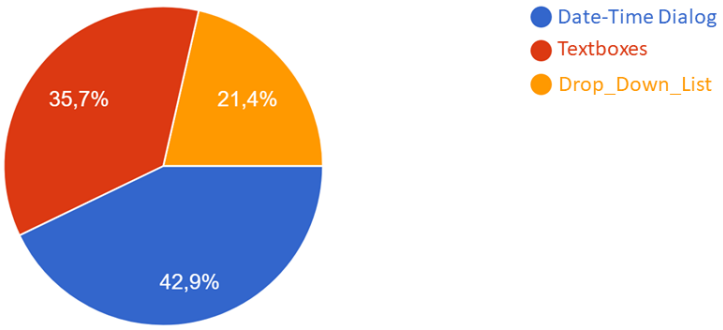


Abbildung 5.19.: bevorzugte Darstellung aller Teilnehmer bzgl. Fragen vom Typ *Datum und Zeit*

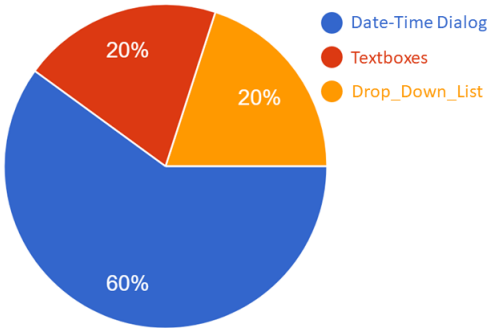


Abbildung 5.20.: Teilnehmer unter 50 Jahre alt

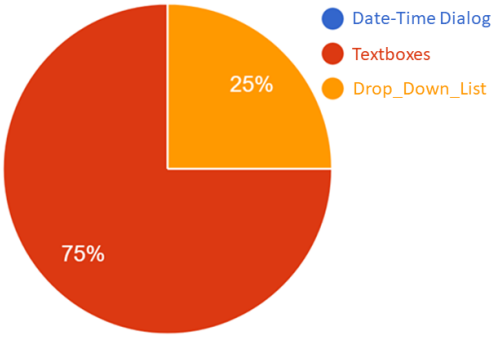


Abbildung 5.21.: Teilnehmer über 50 Jahre alt

Abbildung 5.22.: bevorzugte Darstellung bzgl. Fragen vom Typ *Datum und Zeit*.

5.3. Auswertung der Umfrage

Ausgewertet wurde per Hand und ohne eine bestimmte Auswertungssoftware. Dabei wurde nach den bevorzugten Darstellungen und auffälligen Eingaben gesucht. Außerdem wurden das Feedback und die Wünsche der Teilnehmer einbezogen. Es wurde festgestellt, dass bei einigen Fragentypen keine eindeutige bevorzugte Darstellung erkannt werden konnte, wie in der Abbildung 5.8 zu sehen ist. Es wurde außerdem festgestellt, dass es bei solchen Fragentypen einen auffälligen Unterschied zwischen der Gruppe Ü50 und den jüngeren Teilnehmern gab. Das ist zum Beispiel der Fall bei den Fragen vom Typ *Skala*, *Zuordnen* und *Date and Time*. Als Grund wurde in dem Interview festgestellt, dass die Personen aus der Gruppe U50 Darstellungen bevorzugen, die am schnellsten und mit so wenig Aufwand wie möglich bedienbar sind, wo also die Eingabe der Antwort schnell und mit wenigen Schritten erfolgt. Dies sehen wir beispielsweise bei den Fragen nach Datum und Zeit. Hier haben sich die meisten Personen aus der Gruppen U50 für die Darstellung mit *Date-Time Picker* und nicht für *Textboxes* entschieden, da die manuelle Eingabe laut den Teilnehmern aufwändiger ist und mehr Zeit kostet als *Date-Time Picker*. Die meisten Personen aus der Gruppe Ü50 bevorzugten bei diesem Fragentyp die Darstellung mit *Textboxes*. Ein Grund dafür wurde ebenfalls in dem Interview festgestellt, nämlich, dass die Personen dieser Gruppe Darstellungen bevorzugen, wo die Antwortalternativen klar Dargestellt sind und die Eingabe der Antwort leicht ist (Siehe Abbildung 5.1). Diese Tendenz sehen wir ebenfalls bei den Fragen vom Typ *Zuordnen*, da keine Person aus der Gruppe U50 die Darstellung mit *Textboxes* bevorzugt hat, während die Hälfte der Personen aus der Gruppe Ü50 sich schon für diese entschieden haben.
Mittels unserer Auswertung der Ergebnisse konnten keine auffälligen Unterschiede zwischen den Personen aus den Gruppen U50 (unter 30 Jahre alt und zwischen 30 und 50) erkannt werden. In einigen Fällen war sich die Mehrzahl der Personen aus allen Altersgruppen für eine Darstellung einig, wie zum Beispiel Bei Fragen vom Typ *Dichotome*, *MultipleChoice*, *Multi-Skala* und *Anordnen*. Dies ergibt sich nach unserer Analyse dadurch, dass diese bevorzugten Darstellungen sowohl schnell als auch einfach bedienbar sind.
Infolgedessen werden wir in unserer Bibliothek zuerst solche Darstellungen implementieren und für die Fälle, bei denen Personen unterschiedlicher Altersgruppen sich nicht eindeutig für eine Darstellung entschieden haben, werden die bevorzugten Darstellungen aus den Ü50 und U50 Gruppen jeweils einsetzt werden.

5.4. Feedback und Wünsche

Im Anschluss an die Umfrage wurde ein Interview mit den Teilnehmern durchgeführt. Durch dieses haben sich die folgenden Wünsche und Bemerkungen ergeben: Bei den Darstellungen der Fragen vom Typ *Freie Antwort* haben sich einige Teilnehmer gewünscht, Hinweise zur Bedienung dieser Darstellung zu bekommen, etwa ein Piktogramm. Es wurde ebenfalls bemerkt, dass Teilnehmer, vor allem, die, die sich nicht gut mit Android auskennen, bei den Darstellungen, bei denen es eine *Scrollbar* gab, die *Scrollbar* komplett übersehen haben. Die Teilnehmer haben bemerkt, dass die *Scrollbar* nicht gut sichtbar war. Außerdem haben sich die Teilnehmer bei der Anwendung der Technik *Drag & Drop* beschwert, dass die Reaktionszeit auf das Ziehen eins Elements sehr lang ist und sich gewünscht, diese zu verkürzen. Aufgrund dessen werden wir in unserer Bibliothek die folgenden Verbesserungen einsetzen: Das Anzeigen eines Hinweises in Form eines Texts in der *Textbox* zu ermöglichen, und außerdem eine bessere angepasste *Scrollbar* zu entwerfen. Leider konnten wir im Rahmen dieser Arbeit die Reaktionszeit der Technik *Drag & Drop* nicht verbessern, da der von Android zur Verfügung gestellte *Drag & Drop* Prozess eine *Drag* Aktion erst nach einer gewissen Zeit erkennt.

5.5. Fazit

In diesem Kapitel wurde der Aufbau der Studie erläutert. Außerdem wurde die Durchführung dieser Studie ausführlich erklärt und die Ergebnisse präsentiert. Bei der Auswertung der Ergebnisse wurde festgestellt, dass Personen, die unter 50 sind, dazu tendieren, Darstellungen zu bevorzugen, die *am schnellsten und mit wenig Aufwand bedienbar* sind, während die Personen, die über 50 sind, *verständlich dargestellte und einfach bedienbare* Darstellungen, bevorzugen. Personen unterschiedlicher Altersgruppen haben sich für Darstellung geeinigt, die *einfach und schnell Bedienbar* sind. Aus diesen Ergebnissen ergaben sich die Darstellungsvarianten, die später in unserer Android-Bibliothek umgesetzt werden. Diese sind in der Tabelle 5.2 zusammengefasst. Außerdem wurden Feedback und Wünsche der Teilnehmer erwähnt, die ebenfalls später in der Implementierung unserer Bibliothek eingesetzt werden. Unter anderem der Entwurf einer angepassten Scrollbar.

Darstellungsmöglichkeiten
Radioboxes
Imagebuttons
Checkboxes
Seekbar
Rating Stars
Checkboxes Matrix
Textboxes
Sort-List
Dropdown-List
Date-Time Picker

Tabelle 5.2.: Darstellungsvarianten, die in unserer Bibliothek umgesetzt werden

Teil VI

Implementation

Kapitel 6 Implementation

Basierend auf den Ergebnissen des vorherigen Kapitels wird in diesem Kapitel die Implementation unserer Bibliothek eingeführt. Zuerst wird die Struktur der Bibliothek präsentiert. Danach werden die implementierten Klassen mit deren Funktionalitäten sowie die verwendeten Bibliotheken erläutert. Anschließend werden die in unserer Bibliothek umgesetzten Gestaltungsaspekte vorgestellt.

6.1. Struktur der Bibliothek

Die Bibliothek wurde mit der Programmiersprache Java mit Hilfe von der Entwicklungsumgebung Android-Studio implementiert. Die API-Version unserer Bibliothek wurde auf 17 gesetzt, da mit niedrigeren Versionen die Möglichkeit nicht besteht, erzeugten Bedienelementen eine ID zuzuweisen. Allerdings ist unsere Bibliothek trotz dieser Wahl mit etwa 98% der Android Geräte kompatibel, wie in Tabelle 2.1 zu sehen ist. Die Bibliothek besteht aus zwei Sorten von Klassen: Klassen für die Umsetzung der unterschiedlichen Darstellungsmöglichkeiten (Generatorklassen) und eine Klasse, die für das Speichern der Ergebnisse ist (DatabaseHelper). In den folgenden Abschnitten wird die Implementation dieser Klassen näher erläutert. Eine ausführliche Dokumentation der Generatorklassen befindet sich im Anhang D.

6.1.1. Generatorklassen

In der Bibliothek gibt es acht verschiedene Generatorklassen, die unterschiedliche Darstellungsvarianten von Fragen- und Antworttypen umsetzen. Dabei repräsentiert jede Klasse einen Fragentyp. In der ersten Version unserer Bibliothek wurden die Klassen so implementiert, dass jede Klasse Methoden hat, die zu einem Fragentyp die entsprechende Darstellung erzeugen. Um diese Methoden zu nutzen musste jedoch ein Objekt der Klasse erzeugt werden.
Die Aktivierung von zusätzlichen Funktionalitäten für eine Darstellung, etwa ob die Antworten auf diese Darstellung gespeichert werden sollen oder nicht, wurde durch Eingabe von Parametern in dem Konstruktor ermöglicht. Aber durch diese Implementierung wurde später festgestellt, dass für jede neue Frage ein neues Objekt erstellt werden muss, das dann die Eigenschaften der Darstellung dieser Frage enthält, was dazu geführt hat, dass am Ende so viele Objekte in der Applikation erzeugt sind. Außerdem war der Aufruf einer Methode ausschließlich über das erzeugte Objekt möglich. In der aktuellen Version unserer Bibliothek wurden die Klassen so implementiert, dass sie lediglich statische Methoden haben. Diese haben den Vorteil dass sie unabhängig von einem Objekt existieren und über die Klasse aufgerufen werden können, in der sie definiert sind. So ist es für den Entwickler einfacher die Bibliothek zu benutzen und dabei kann Speicherplatz gespart werden, da keine Instanzen von Objekten erzeugt und verwaltet werden müssen. Jeder Aufruf einer Methode erzeugt dann genau eine eindeutige Darstellung einer Frage, deren Eigenschaften durch die Eingabe von Parametern spezifiziert werden können. Der

Nachteil dieser Struktur ist, dass alle Methoden innerhalb einer Klasse überladen sind, was die Klasse unübersichtlicher macht. Jede Methode hat andere Parameter hat, was dem Entwickler Flexibilität bei der Darstellung einer Frage bietet.

Signatur der Methoden

Die Signatur aller Methoden hat immer mindestens zwei Parameter, die bei dem Aufruf einer Methode übergeben werden müssen, nämlich eine Activity und ein Layout. Die Activity ist hier wichtig um festzustellen, in welchem Fenster der Applikation eine Darstellung erzeugt werden soll. Die Entscheidung für die Eingabe eines Layouts wurde getroffen, um eine dynamische Positionierung einer Darstellung zu ermöglichen, wobei der Entwickler selber entscheiden kann, wo eine Darstellung genau erzeugt werden soll. Außerdem besteht dadurch für den Entwickler die Möglichkeit, dem Layout andere Views hinzuzufügen. Deshalb ist die Eingabe einer Activity sowie eines Layouts bei allen Methoden einer Genratorklasse erforderlich.
Außerdem hat die Signatur aller Methoden zusätzlich die Parameter *saveButtonID* und *questionID*. Durch die Eingabe der Parameter *saveButtonID* wird die Taste festgelegt, durch die dann die Methode zum Speichern der Daten aufgerufen wird. Die Eingabe der Parameter *questionID* ist notwendig für das Speichern der Antwort in einer Tabelle in der Datenbank, wie oben erklärt wurde. Allerdings ist die Eingabe dieser zwei Parameter optional und erfolgt bei dem Aufruf der entsprechenden Methode.
Um eine sinnvolle Nutzung unserer Bibliothek in der Entwicklung von Lernanwendungen unter anderem bei der Erstellung von E-Tests bzw. E-Klausuren zu ermöglichen, ist die Implementation einer automatischen Korrektur der eingegebenen Antworten nötig. Deshalb wurde eine zusätzliche Funktionalität (checkAnswer) implementiert, die durch die Eingabe der richtigen Antwort die eingegebene Antwort einer Darstellung auf die Korrektheit überprüft. Diese erfolgt in unserer Implementation durch die Eingabe des Parameters *CorrectAnswer* bei dem Aufruf der Methode einer Darstellung.

Scrollbar Erweiterung

Die Anzahl der erzeugbaren Antwortalternativen in einer Activity ist von der Größe des Bildschirms eines Android-Gerätes abhängig. Wenn beispielsweise ein Handy einen kleinen Bildschirm hat, dann ist es unmöglich beliebig viele Antwortalternativen in einer Activity sinnvoll zu erzeugen, vor allem bei der Darstellung von Antwortalternativen mit Hilfe einer *Radiobuttons* oder *Checkboxes*. Infolgedessen haben wir uns für die Implementierung einer zusätzlichen Funktionalität *Scrollbar* entschieden. Die Anwendung dieser Funktionalität ist in unserer Bibliothek bei der Darstellung der Antwortalternativen in Form von *Radiobuttons* oder *Checkboxes* möglich ist und erfolgt durch eine freiwillige Eingabe des Boolean Parameters *Scrollbar* bei dem Aufrufen der entsprechenden Methode.

Halboffene Frage Erweiterung

Damit die Anwendung unserer Bibliothek nicht nur auf die geschlossenen und offenen Fragen beschränkt wird, haben wir uns entschieden eine zusätzliche Funktionalität (extraAnswer) zu implementieren. Durch diese erfolgt eine Erweiterung der geschlossen Fragen auf halboffene Fragen. Diese Funktionalität steht ebenfalls bei der Darstellung der Antwortalternativen in Form von *Radiobuttons* oder *Checkboxes* zur Verfügung und erfolgt durch eine freiwillige Eingabe des Boolean Parameters *extraAnswer* bei dem Aufrufen der entsprechenden Methode.

Erzeugen eines neuen Layouts

Bei der Erzeugung einer Darstellung wird nicht immer die ausgewählte Darstellung in dem eingegebenen Layout als direktes Kind erzeugt, sondern bei einigen Darstellungen erst ein zwischen Layout erzeugt, das die ausgewählte Darstellung beinhaltet. Dementsprechend wird dieses Layout zu dem eingegebenen Layout hinzugefügt, wie in der Abbildung 6.1 dargestellt ist. Dies ist der Fall bei der Implementierung der Darstellungen, die mit Hilfe von *Checkboxes*, *RadioButtons*, *Sort-List* und *Seekbar* erzeugt werden.

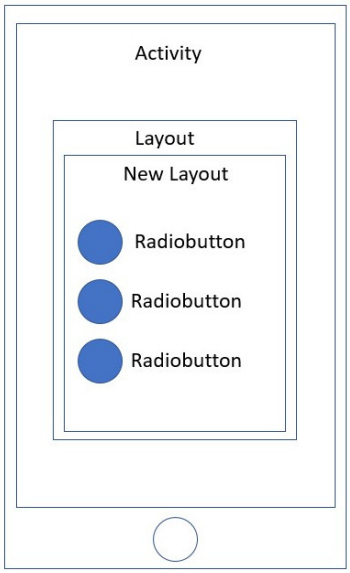


Abbildung 6.1.: Beispiel für die Erzeugung von Radiobuttons in einem Layout.

Für diese Implementierung gab es mehrere Gründe: Zum einen, dass es bei einigen Layouts beispielsweise *FrameLayout* technisch unmöglich ist, mehrere Elemente als direkte Kinder dazu hinzuzufügen. Um das Problem zu beseitigen, wird ein Layout erzeugt, das beliebig viele Elemente enthalten kann. Allerdings wurde in unserer Bibliothek *FrameLayout* verwendet, um die Eigenschaft *Scrollbar* zu erzeugen, die mit Hilfe einer *Scrollview* erfolgt.
Ein neues Layout zu erzeugen ist ebenfalls bei der Erzeugung von *Radiobuttons* nötig. Da *Radiobuttons* in unserer Bibliothek gemäß den Ergebnissen der Studie für die Darstellung des Fragentyps *Dichotome* und *SingleChoice* benutzt werden. Das heißt mit Hilfe von *Radiobuttons* darf gleichzeitig genau eine Antwort ausgewählt werden. Um das zu realisieren, muss eine *Radiogroup* erzeuget werden, in der sich diese *Radiobuttons* befinden.
Bei der Darstellung einer Frage in Form einer *Sort-List* ist es ebenfalls technisch unmöglich eine solche Liste direkt in dem eingegeben Layout zu erzeugen, da die Darstellung dieser sortierbaren Liste mit Hilfe eines angepassten *DragLayout* erfolgt. Dieses Layout wird mit Hilfe der externe Bibliothek *DragLinearLayout* erzeugt (Diese wird später genauer erläutert). Außerdem ist es hier zu beachten, dass die Elemente dieses Layouts lediglich vertikal angeordnet werden können. Diese Beschränkung liegt an der verwendeten Bibliothek.

Bei der Darstellung einer Frage vom Typ *Skalierung* ist die Erzeugung eines zwischen Layouts ebenfalls wichtig, da hier mehrere Bedienelemente erzeugt werden sollen, nämlich entweder *Seekbar* mit *Textboxes* als Beschriftung oder *Rating Stars*, die ebenfalls *Textboxes* als Beschriftung hat. Um mehrere Bedienelemente sinnvoll zu positionieren, wird ein neues Layout erzeugt, wie in Abbildung 6.2 zu sehen ist.

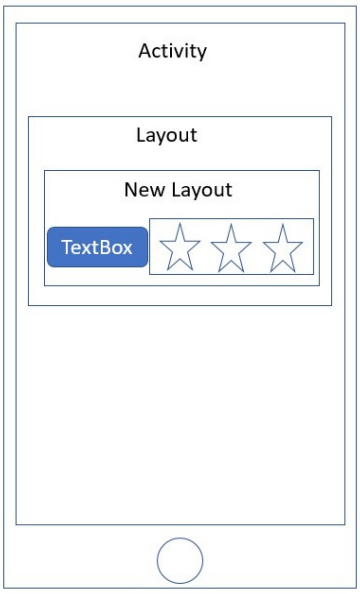


Abbildung 6.2.: Beispiel für die Erzeugung von Ratings Stars in einem Layout.

Eingabe der Antwortalternativen

Bei allen Methoden erfolgt die Eingabe der Antwortalternativen beim Aufruf der Methode durch die Eingabe des Parameters *Answers*. diese Parameter kann bei den meisten Methoden als Array vom String eingegeben werden. Allerdings gibt es ein paar Ausnahmefälle, wie zum Beispiel bei der Erstellung einer Darstellung mit Hilfe von *Imagebuttons*. Hierbei sollte ein Array von den IDs der Bilder eingegeben werden, da hier davon ausgegangen wird, dass der Entwickler die Bilder bereits in dem eingegeben Layout eingebettet hat. Dabei ist zu beachten, dass jedes Bild eine ID haben muss, die als Repräsentant eines Bildes gilt. Diese ist erforderlich sowohl für die Übergabe beim Aufruf der Methode als auch für das Speichern der Antwort in der Datenbank, da das ausgewählte Bild selber nicht in der Datenbank gespeichert werden kann.
Bei den Fragen vom Typ *Datum und Zeit* mit Hilfe von *Date-Time Picker* können keine Antwortalternativen eingegeben werden sondern diese werden mit einem Dialog zur Auswahl angezeigt. Deswegen wird hier ein Button als Eingabe verlangt, der beim Drücken dann den entsprechenden Dialog anzeigt. Um dem Entwickler mehr Flexibilität zu bieten, wurden mehrere Varianten implementiert, nämlich eine, bei der automatisch in dem eingegebenen Layout ein Button erzeugt wird und eine, die einen Button als Eingabe bekommt, der dann nach dem Aufruf unserer Methode den entsprechenden Dialog erzeugen kann.

Hilfestellung

Gemäß den Ergebnissen der Studie in dem vorherigen Kapitel haben sich einige Teilnehmer gewünscht, Hinweise zur Bedienung einiger Darstellungen zu haben. Diese wurde bei einigen Darstellungen in Form eines textuellen Hinweises umgesetzt. Zum Beispiel kann bei der Darstellung einer Frage vom Typ *Freie Antwort* ein Hinweis als zusätzlicher Parameter beim Aufruf der entsprechenden Methode eingegeben werden und dies wird dann in der *Textbox* angezeigt, wie in Abbildung 6.4 zu sehen ist. Auch bei der Darstellung einer Frage vom Typ *Dropdown-List* kann ein Hinweis als Parameter eingegeben werden, der dann als vorgewählter aber nicht auswählbarer Eintrag angezeigt wird.

Feedback

Gemäß den in der Grundlagen vorgestellten Gestaltungsrichtlinien für Seniorinnen und Senioren 2.1.2 sollte für die Eingabe der Nutzer ein Feedback zurückgeben werden. Alle verwandten Bedienelemente geben Standardmäßig beim Bedienen ein Feedback zurück etwa, in dem sie markiert werden. Allerdings ist das nicht der Fall bei dem Speichern der eingegebenen Antwort. Deswegen wird mit Hilfe des sogenannten *Toast* eine Rückmeldung gegeben, ob die Daten erfolgreich oder nicht erfolgreich gespeichert wurden.

6.1.2. Speichern der Daten

Um eine sinnvolle Nutzung unserer Bibliothek zu ermöglichen, etwa zur Datenerhebung oder Datenanalyse, ist es notwendig, dass die eingegebenen Antworten zu den Fragen, die mittels unserer Bibliothek dargestellt sind, gespeichert werden können. Unter Android stehen mehrere Möglichkeiten zur Verfügung um dies zu ermöglichen, zum Beispiel das Speichern der Daten in einer Datei oder in einer Datenbank. In unserer Bibliothek werden die Daten in einer Datenbank gespeichert, um den einfachen Zugriff auf die gespeicherten Daten zu einem späteren Zeitpunkt zu ermöglichen. Als Datenbank wurde die von Android unterstütze SQLite-Datenbank verwendet, da sie sowohl als Nativ-Bibliothek in Android zur Verfügung gestellt ist als auch unter widrigen Umständen robust ist. Zu diesem Zweck wurde die Klasse *DatabaseHelper* als Erweiterung der Klasse *SQLiteOpenHelper* implementiert.
Um das Speichern der Daten von verschiedenen Geräten in einer Datenbank einfach und automatisch zu ermöglichen, wurde die Klasse *DatabaseHelper* so implementiert, dass die Daten sowohl lokal auf einem Gerät als auch online auf einem Server gespeichert werden können.

Lokal Speichern

Dabei wird für jede *Generatorklasse* eine entsprechende Tabelle in der Datenbank erzeugt, in der die eingegebenen Antworten auf Fragen einer Darstellung gespeichert werden. Alle Tabellen in der lokalen Datenbank haben sechs Spalten, wie in Tabelle 6.1 zu sehen ist.

Table 1					
ID	QuestionId	SelectedAnswer	IsCorrect	Status	TIMESTAMP

Tabelle 6.1.: Beispiel für eine Tabelle in der Datenbank

Das Attribut *ID* ist nötig, um die verschiedenen Eingaben zu Identifizieren. Die ID wird automatisch bei jeder neuen Eingabe inkrementiert. Das Attribut *QuestionId* ist ebenfalls

nötig, um zu identifizieren zu welcher Frage die eingegebene Antwort gehört und kann bei der Erstellung einer Darstellung als Parameter eingegeben werden.
In der Spalte *Status* wird ein Attribut des Typs Boolean gespeichert, das für den Status der gespeicherten Daten steht. Es hat den Wert 0, wenn eine Antwort aus irgend einem Grund nicht erfolgreich auf dem Server gespeichert wurde (etwa wenn es keine Verbindung mit dem Internet gibt), damit diese dann später mit dem Server synchronisiert werden kann. Sonst hat der Status den Wert 1. In der Spalte *SelectedAnswer* werden die eingegebenen Antworten gespeichert, während in der Spalte *IsCorrect* die Korrektheit der eingegebenen Antworten gespeichert wird. In der Spalte *TIMESTAMP* werden Zeitstempel gespeichert. Diese repräsentieren die Uhrzeit sowie das Datum der Eingabe der Antwort. Somit hat der Nutzer die Möglichkeit, die eingegebenen Antworten nach Datum und Zeit zu sortieren.

Online Speichern

Für das Speichern der Daten auf einem Server muss eine Verbindung zwischen der Applikation und dem Server erstellt werden. Es gibt zwei Möglichkeiten dies zu realisieren: Entweder eine direkte Verbindung zwischen der Applikation und der Datenbank oder eine Verbindung über einen Webservice. Die direkte Verbindung ist zwar einfacher zu implementieren aber sie ist nicht sicher, da die Zugangsdaten im Quellcode gespeichert werden müssen. Deswegen wurde die Entscheidung getroffen, eine Schnittstelle (Webservice) zu implementieren, über die die Verbindung mit dem Server erfolgt. Die Schnittstelle wurde mit der Programmiersprache PHP implementiert. Sie erstellt eine Verbindung mit der Datenbank und bekommt die von der Applikation geschickte *POST* Anfrage und dekodiert die JSON Daten, um sie in der Datenbank zu speichern. Außerdem gibt sie eine Antwort zurück an die Applikation in Form von JSON und zwar *OK* wenn die Daten erfolgreich auf dem Server gespeichert wurden, sonst gibt sie die Antwort *FAILED* zurück. Die Antwort ist allerdings nötig, um sie in dem Attribut *Status* in der lokalen Datenbank zu speichern. In Abbildung 6.3 ist dieser Prozess genauer veranschaulicht. Um die Daten online speichern zu können, ist die Erstellung einer Datenbank sowie die Verwendung unserer Schnittstelle erforderlich. Zur ersten Erstellung der Datenbank wurde ein Skript geschrieben, das manuell auf einer *Mysql* Datenbank durchführbar ist. Dieses Skript erstellt die entsprechende Datenbank sowie alle notwendigen Tabellen. Die Tabellen haben genau die selben Spalten wie die in der lokalen Datenbank 6.1 bis auf der Spalte *Status*, da sie hier keinen Sinn mehr macht. Es wäre möglich, die Datenbank in der Schnittstelle zu initialisieren, sodass die Erstellung der Datenbank sowie die notwendigen Tabellen automatisch erfolgt. Dies wäre einfacher für den Nutzer, allerdings muss dann bei jedem Speichervorgang eine Anfrage an dem Server geschickt werden um zu überprüfen, ob die Datenbank und die Tabellen erstellt sind, was Zeit und Aufwand kostet. Im Gegensatz dazu ist das Ausführen dieses Skripts nur einmalig erforderlich.

Daten Synchronisieren

Wie oben erklärt wurde, werden die Daten, falls keine Verbindung mit dem Server hergestellt werden kann, lediglich lokal in einer Datenbank gespeichert. Um diese Daten auch online auf dem Server speichern zu können, wurde die Klasse *NetworkStateChecker* implementiert, die dann zum späteren Zeitpunkt die lokale Datenbank mit der online Datenbank synchronisiert, sodass die Daten, die nur in der lokalen Datenbank gespeichert wurden, automatisch auf dem Server gespeichert werden. Diese Klasse wurde als

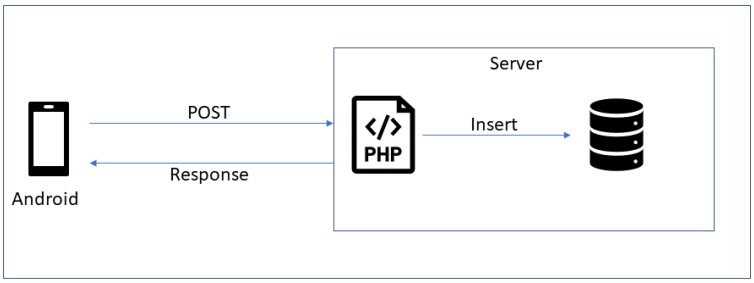


Abbildung 6.3.: Speichern der Daten auf einem Server.

Erweiterung von *Broadcastreceiver* implementiert. Jedes mal wenn die Applikation mit dem Internet verbunden ist, werden die lokale Daten mit dem Server synchronisiert. Das bietet den Vorteil, dass alle Daten zur jeder Zeit, in der eine Verbindung sowohl mit dem Internet als auch mit dem Server zur Verfügung steht, vollständig online auf dem Server sind.

6.2. Externe Bibliotheken

Für die Implementation unserer Bibliothek wurden zwei externe Bibliotheken verwendet. Diese sind in der folgenden Abschnitten näher erläutert.

6.2.1. Volley Bibliothek

Volley ist eine HTTP-Bibliothek, die das Networking für Android-Anwendungen einfacher und schneller macht. Diese Bibliothek wurde von *Google* entwickelt und ist auf GitHub erhältlich [15].
Mit Hilfe dieser Bibliothek kann ein Objekt vom Typ *JSONObjectRequest* erstellt werden, das benutzt wird, um Daten im JSON Format an einen Server schicken zu können. Dies erfolgt, in dem ein *POST Request* mit den Daten, die auf dem Server gespeichert werden sollen, an eine Schnittstelle geschickt wird. Außerdem kommt eine Antwort von der Schnittstelle zurück, die bestätigt, ob die Daten erfolgreich gespeichert wurden.

6.2.2. DragLinearLayout Bibliothek

Mit Hilfe dieser Bibliothek kann ein LinearLayout erzeugt werden, in dem das Ziehen und Ablegen von Elementen möglich ist. Diese Bibliothek wurde in unserer Arbeit bei der Implementierung der *Sort-List* Methode verwendet. Die Bibliothek ist ebenfalls auf GitHub erhältlich [22].

6.3. Design der Bedienelemente

In unserer Bibliothek wurden einige Bedienelemente neu entworfen und einige zusätzlich angepasst, sodass diese mit den Guidelines zur Verbesserung der Nutzbarkeit für Seniorinnen und Senioren übereinstimmen. In den folgenden Abschnitten werden diese präsentiert.

Größe und Gestaltung der Bedienelemente

Buttons, die durch eine Darstellung erzeugt werden, haben standardmäßig die Größe 48x48 dp. Außerdem werden alle Bedienelemente entweder hervorgehoben oder umrandet erstellt. Diese wurden so gestaltet, dass auch von älteren Menschen leicht getroffen können, wie in dem Grundlagen erklärt wurde 2.1.2.

Angepasster Rand

Die Bedienelemente sollten, wie oben erklärt wurde, entweder hervorgehoben oder umrandet erstellt werden. Deswegen haben wir einen angepassten Rand entworfen, der dann in unserer Bibliothek verwendet wird. Dieser Rand wird bei der Darstellung von *Textbox* verwendet, um diese sichtbar zu machen, wie in der Abbildung 6.4 zu sehen ist. Außerdem wurde er bei der Erstellung von *Sort-List* verwendet, um zwischen den Elementen in dieser Liste unterscheiden zu können.

Abbildung 6.4.: Angepasster Rand einer Textbox.

Scrollbar Design

Es wurde in der Studie in dem vorherigen Kapitel festgestellt, dass einige Teilnehmer die Scrollbar nicht bedienen können, da sie mit dem Standard Design von Android nicht gut sichtbar ist. Deswegen wurde in unserer Bibliothek in der XML Datei *Styles* eine angepasste Scrollbar entworfen. Diese hat den Vorteil, dass sie eine sogenannte *Thumb* hat, deren Farbe im Kontrast zu der Farbe der Scrollbar steht, was sie deutlich sichtbar macht. Ein Vergleich zwischen der Standard und angepassten Scrollbar ist in den Abbildungen 6.5 und 6.6 dargestellt.

Abbildung 6.5.: Standard Scrollbar.

Abbildung 6.6.: Angepasste Scrollbar.

Teil VII

Zusammenfassung und Ausblick

Kapitel 7 Zusammenfassung

Abschließend erfolgt in diesem Kapitel eine Zusammenfassung über die vorgestellten Inhalte dieser Arbeit. Außerdem wird ein Ausblick über die möglichen Erweiterungen unserer Bibliothek vorgestellt.

7.1. Fazit

Das Ziel dieser Arbeit war eine Android-Bibliothek zu entwickeln, die den Entwicklern dabei hilft, Darstellungsvarianten von Antworttypen zu erzeugen, die für unterschiedliche Altersgruppe geeignet sind und zu verschieden Zwecken verwendet werden können, etwa bei der Entwicklung von digitalen Fragebögen, Quizzes und Lernanwendungen. Deswegen wurden in dieser Arbeit zuerst verschiedene Antworttypen ermittelt sowie unterschiedliche Darstellungsmöglichkeiten von diesen Antworttypen auf Android-Geräten untersucht. Außerdem war das Ziel, die ermittelten Darstellungsvarianten auf Verständnis unterschiedlicher Altersgruppen zu überprüfen. Deshalb wurde eine Menge von Fragen- und Antworttypen ausgewählt und mit unterschiedlichen Darstellungsvarianten in einer prototypischen Applikation implementiert. Daraufhin wurde eine kleine Studie mit 14 Personen unterschiedlicher Altersgruppen durchgeführt, um die prototypischen Darstellungsvarianten zu evaluieren und die bevorzugten Darstellungen herauszufinden. Die Ergebnisse der Studie haben gezeigt, dass ältere Menschen, die über 50 sind, Darstellungen bevorzugen, die *verständlich dargestellt und einfach bedienbar* sind, während die jüngeren Menschen die Darstellungen, die *am schnellsten und mit wenigen Aufwand bedienbar* sind, bevorzugen. Allerdings waren sich die Personen unterschiedlicher Altersgruppen in vielen Fällen einig, bei denen die Darstellungen *einfach und schnell bedienbar* sind. Außerdem wurden im Rahmen der Studie Wünsche sowie Verbesserungsvorschläge von den Teilnehmern geäußert, die bei der Implementation unserer Bibliothek berücksichtigt wurden. Basierend auf den Ergebnissen der Studie wurden die bevorzugten Darstellungsvarianten verbessert und in unserer Bibliothek ***RespoLib*** umgesetzt, sodass diese Darstellungsvarianten schnell und einheitlich in der Entwicklung von Android Applikationen verwendbar sind.

7.2. Ausblick

In unserer Bibliothek wurden zwar bereits einige Darstellungsvarianten der Antworttypen umgesetzt, allerdings besteht die Möglichkeit, sie mit weiteren Darstellungsvarianten zu ergänzen. Zum Beispiel eine Darstellung für Fragen vom Typ *Anordnen*, die mit Hilfe der Eingabe von Rangnummern Elemente anordnet. Bei den Fragen vom Typ *Zuordnen* kann eine angepasste *Drag & Drop* Technik entwickelt werden, sodass die Reaktionszeit auf das Ziehen eines Elements kurzer als die Reaktionszeit der von Android zur Verfügung gestellten Variante ist. Eine andere Möglichkeit wäre, die Elemente durch das ziehen eines Pfeils der richtigen Position zuzuordnen. Außerdem könnte die Bibliothek so erweitert, dass eine individuelle Einstellung der Schriftgröße sowie die Farbe der angezeigten Antwortalternativen durch den Entwickler ermöglicht wird. So kann

der Entwickler zum Beispiel die Schriftgröße je nach der Zielgruppe schnell und einfach einstellen.

Anhang A Literaturverzeichnis

[1] Dan Andrei. Surveylib bibliothek. https://github.com/AndreiD/surveylib, 2015.

[2] Aliyar Aras. *Design und Konzeption einer mobilen Anwendung zur Unterstützung tinnitusgeschädigter Patienten*. PhD thesis, University of Ulm, 2014.

[3] Bernhard Baltes-Götz. Einführung in die entwicklung von apps für android 8. 2018.

[4] Android developer webseite. https://developer.android.com/about/dashboards. Eingesehen am 12.03.2019.

[5] Android developer webseite. https://developer.android.com/training/constraint-layout. Eingesehen am 12.03.2019.

[6] Android developer webseite. https://developer.android.com/reference/android/widget/RadioGroup.

[7] Android developer webseite. https://developer.android.com/reference/android/widget/FrameLayout.

[8] Android developer webseite. https://developer.android.com/reference/android/widget/ScrollView.

[9] Github. https://github.com/.

[10] Questionpro webseite. https://www.questionpro.de/.

[11] Surveymonkey webseite. https://www.surveymonkey.de/.

[12] Dorothea Erharter and Elka Xharo. Developer-guideline usability von apps für seniorinnen und senioren. *Developer-Guideline Usability von Apps für Seniorinnen und Senioren. Entstanden im Rahmen des Projektes „mobi. senior. A “. Wien*, 2016.

[13] Marko Gargenta. *Einführung in die Android-Entwicklung*. O'Reilly Germany, 2011.

[14] Vera Gehlen-Baum. *Mobile Geräte in der Präsenzlehre: Ablenkung oder Lernchance? Von der unstrukturierten Nutzung von Smartphone & Co. hin zu einem orchestrierten Modell für Vorlesungen*, volume 5. LIT Verlag Münster, 2017.

[15] Google. Volly bibliothek. https://github.com/google/volley.

[16] Jochen Herrmann. *Konzeption und technische Realisierung eines mobilen Frameworks zur Unterstützung tinnitusgeschädigter Patienten*. PhD thesis, University of Ulm, 2014.

[17] Wenhui Hu, Damien Octeau, Patrick Drew McDaniel, and Peng Liu. Duet: library integrity verification for android applications. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, pages 141–152. ACM, 2014.

[18] Matthis Kepser. E-learning an der hochschule–eine kritische einführung. In *Neue Impulse in der Hochschuldidaktik*, pages 199–228. Springer, 2010.

[19] Alexander Koerner/Getty. Die meisten deutschen surfen mit dem smartphone. `https://www.zeit.de/digital/internet/2016-10/ard-zdf-onlinestudie-internetnutzer-deutschland-zuwachs`, 2016.

[20] Abgeschlossenes Lehrprojekt, Sören Dressler, Dirk Förster-Trallo, Sandra Dressler, and Thomas Rachfall. E-tests als moderne prüfungsform in den wirtschaftswissenschaften: Wie wirken sich e-tests auf die psychische belastung und die akademische un-ehrlichkeit von studierenden aus? *CARF Luzern 2018*, page 352, 2018.

[21] Christian Lindig. *Algorithmen zur Begriffsanalyse und ihre Anwendung bei Softwarebibliotheken*. PhD thesis, Braunschweig University of Technology, Germany, 1999.

[22] Justas Medeišis. Draglinearlayout bibliothek. `https://github.com/justasm/DragLinearLayout`.

[23] Lars Miltkau. *Konzeption und Realisierung einer Web-Plattform zur Veröffentlichung und Bereitstellung digitaler Fragebögen*. PhD thesis, Ulm University, 2018.

[24] Hazwani Mohd Mohadisdudis and Nazlena Mohamad Ali. A study of smartphone usage and barriers among the elderly. In *2014 3rd International Conference on User Science and Engineering (i-USEr)*, pages 109–114. IEEE, 2014.

[25] Christoph Neijenhuis. Simplequestionnaire biblothek. `https://github.com/cneijenhuis/SimpleQuestionnaire`, 2012.

[26] Rolf Porst. Im vorfeld der befragung: Planung, fragebogenentwicklung, pretesting. 1998.

[27] Steffen Scherle. *Konzeption und Evaluierung einer domänenspezifischen Modellierungsumgebung für prozessorientierte Fragebögen*. PhD thesis, University of Ulm, 2014.

[28] Maximilian Scheurich. *Konzeption und Realisierung eines Konfigurators zur Erstellung webbasierter Fragebögen*. PhD thesis, Ulm University, 2017.

[29] Alexander Seifert. Technikakzeptanz älterer menschen im internetzeitalter. *Medien & Altern*, 1:67–74, 11 2016.

[30] Alexander Seifert and Hans Rudolf Schelling. *Digitale Senioren: Nutzung von Informations-und Kommunikationstechnologien (IKT) durch Menschen ab 65 Jahren in der Schweiz im Jahr 2015*. Pro Senectute, 2015.

[31] Statista. The statistics portal, worldwide. `https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/`, 2018.

[32] Jona Florian Stieler. *Validität summativer Prüfungen. Überlegungen zur Gestaltung von Klausuren.* Florian Stieler, 2011.

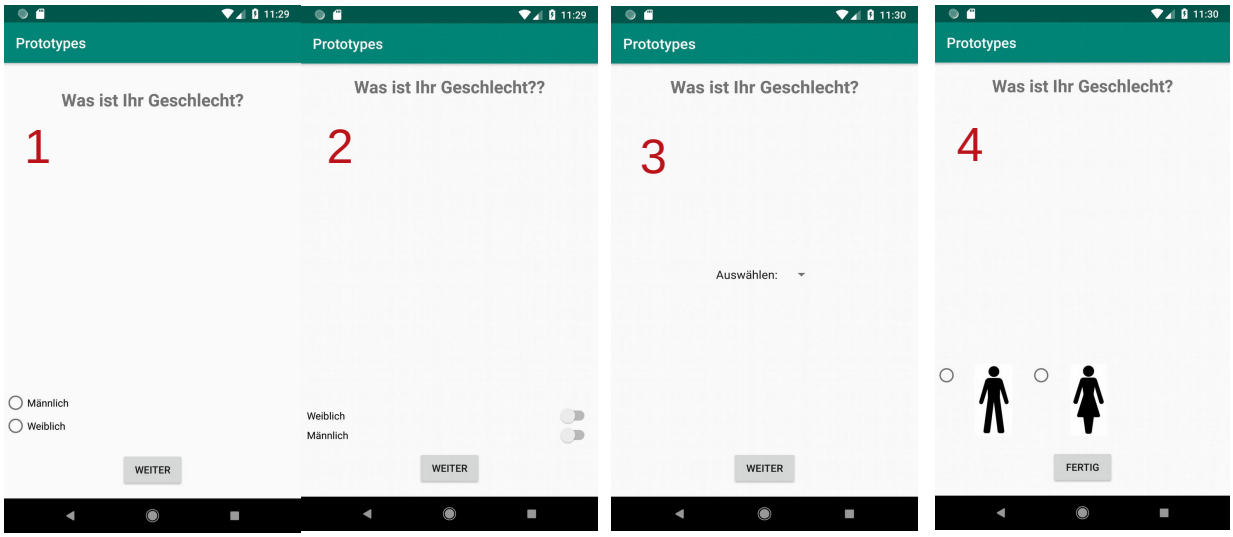
[33] Michael Vogt and Stefan Schneider. E-klausuren an hochschulen. *Koordinationsstelle Multimedia, Gießen: JLU*, 2009.

[34] Klaus Wannemacher, Unter Mitwirkung von Imke Jungermann, Sven Osterfeld, Julia Scholz, and Anna von Villiez. *Organisation digitaler Lehre in den deutschen Hochschulen*. Hochschulforum Digitalisierung beim Stifterverband für die Deutsche ..., 2016.

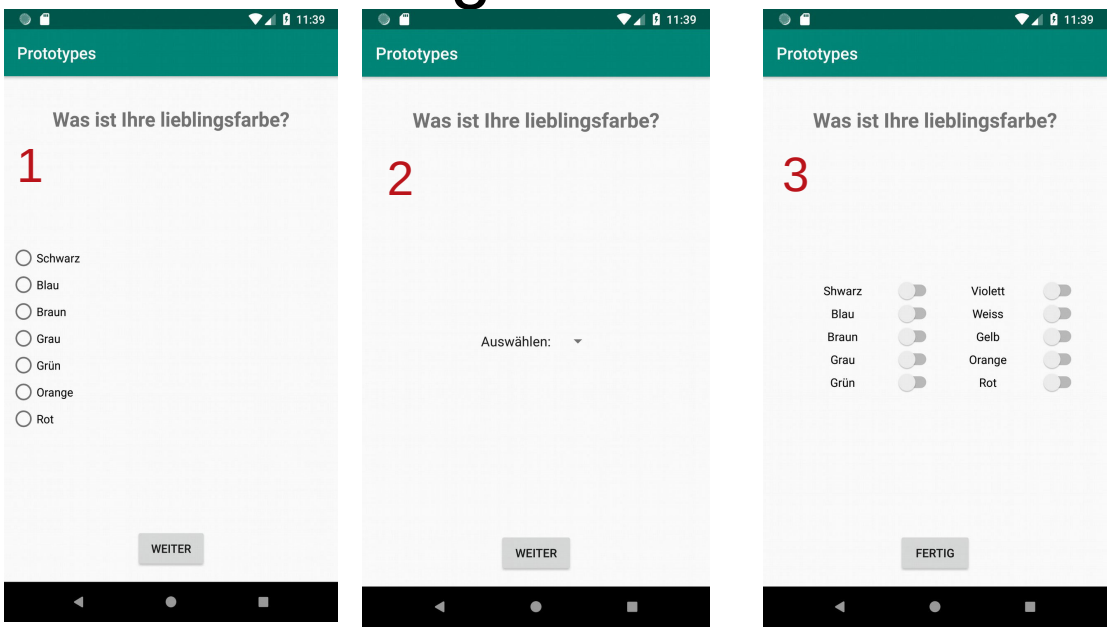
[35] René Wegener, Philipp Bitzer, Sarah Oeste, and Jan Marco Leimeister. Motivation und herausforderungen für dozenten bei der einführung von mobile learning. 2011.

Anhang B Prototypen

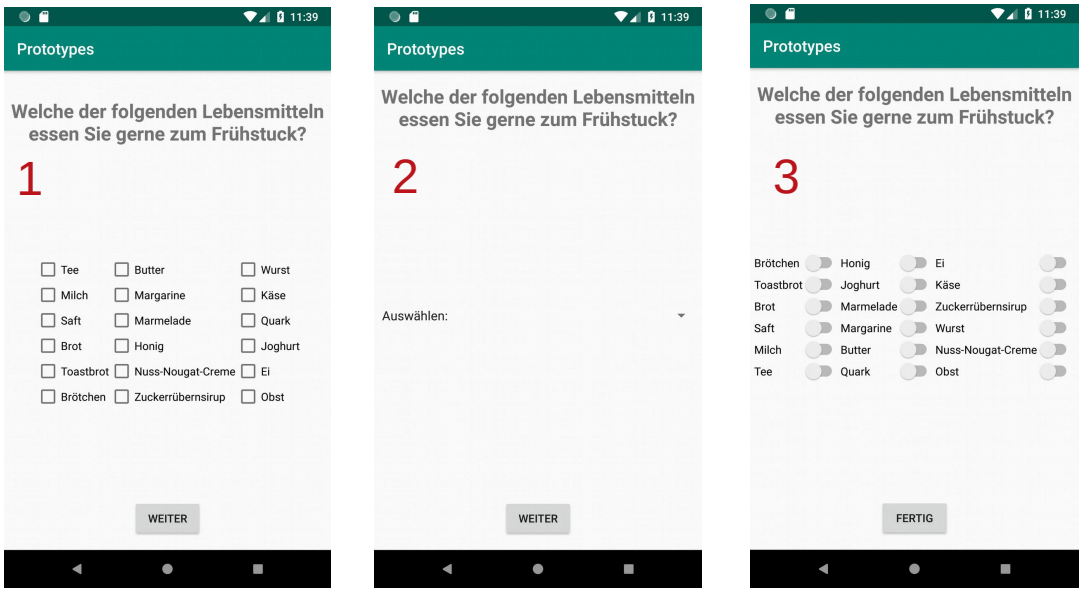
Dichtome



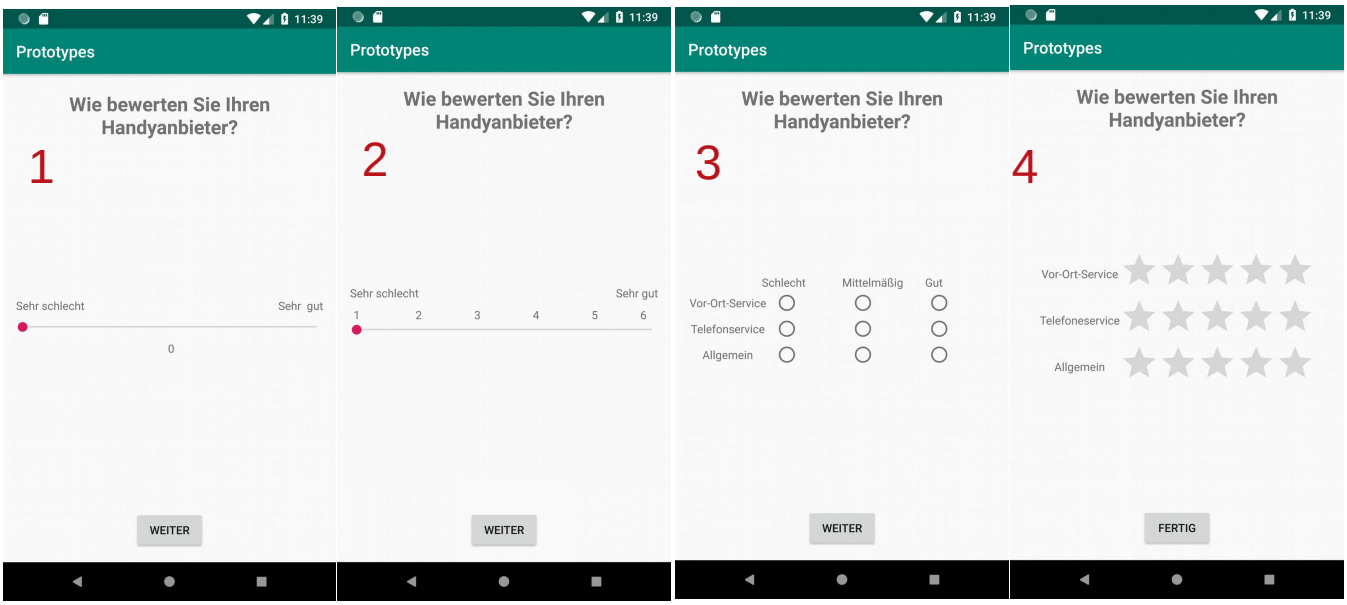
Single Choice



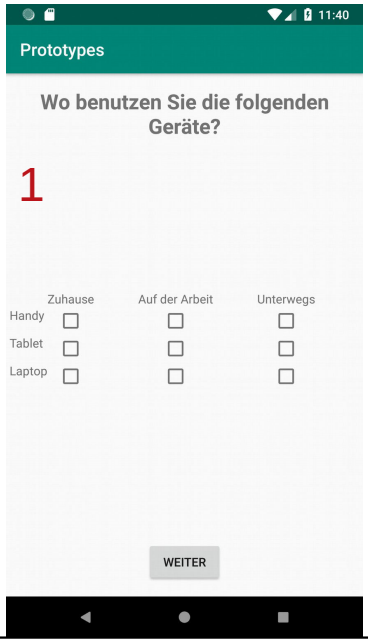
Multiple Choice



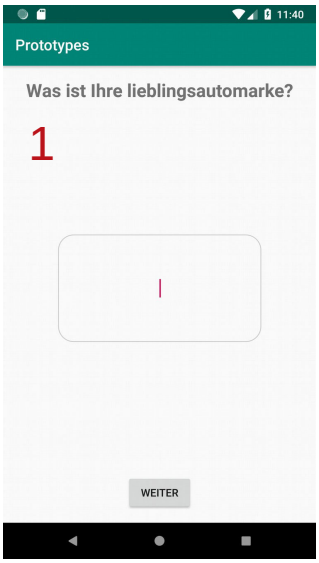
Skala

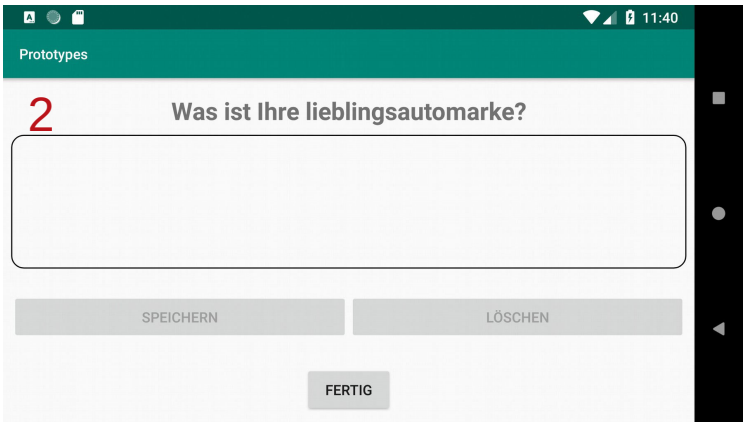


Multi Skala

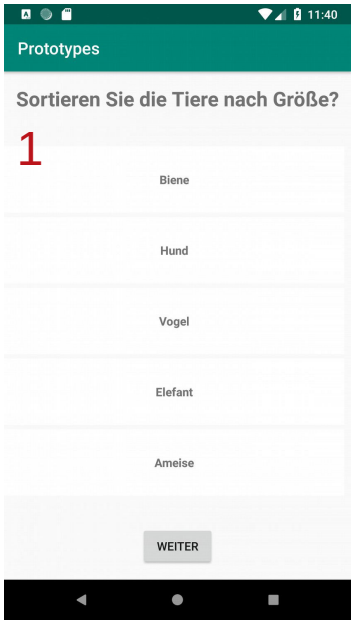


Freiantwort

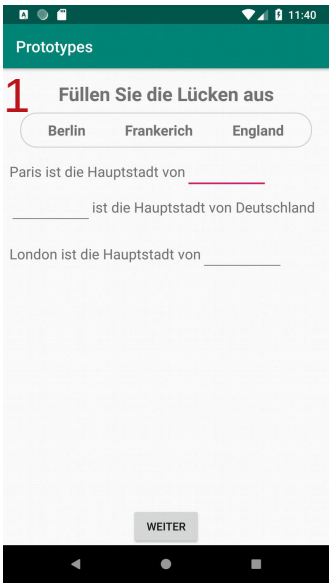


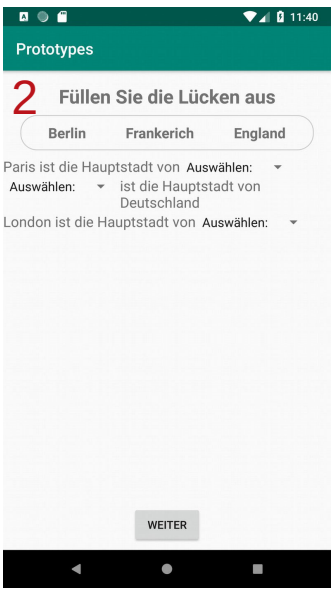


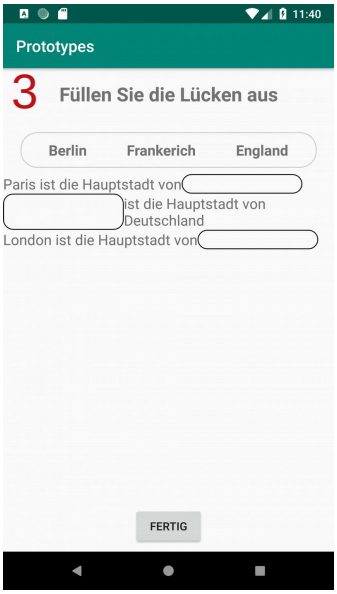
Anordnen



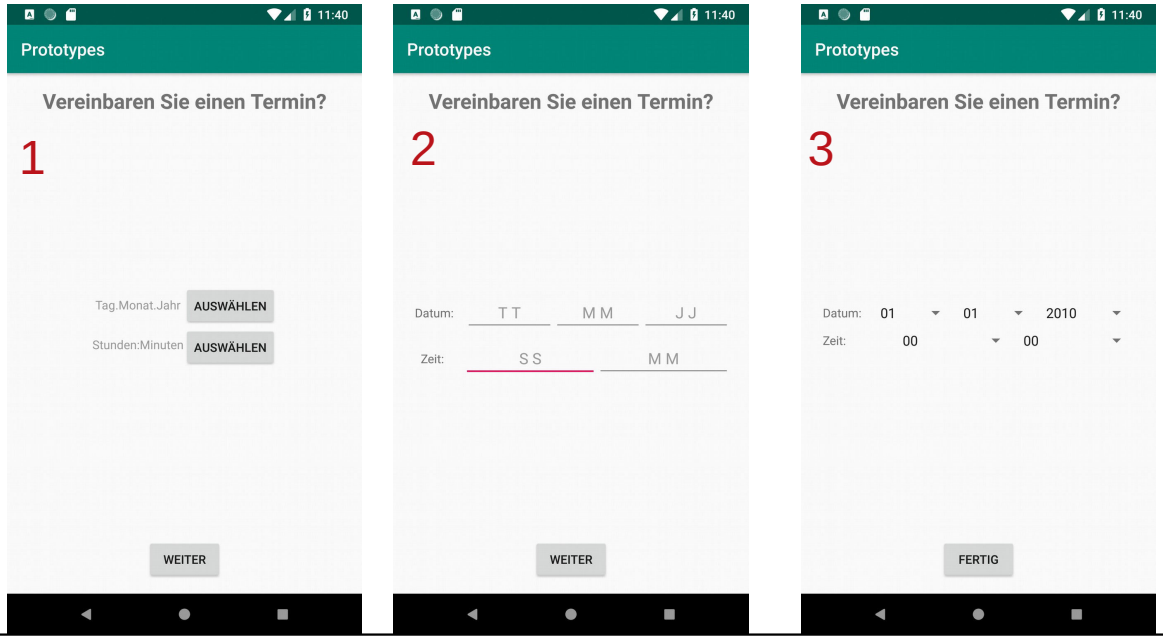
Lücken





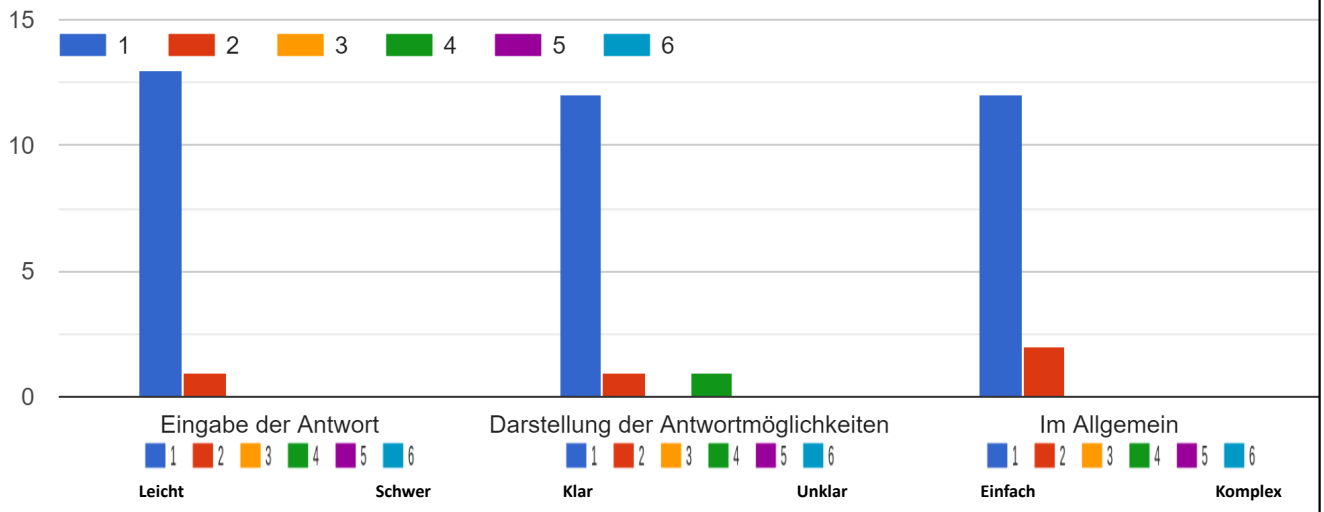


Datum

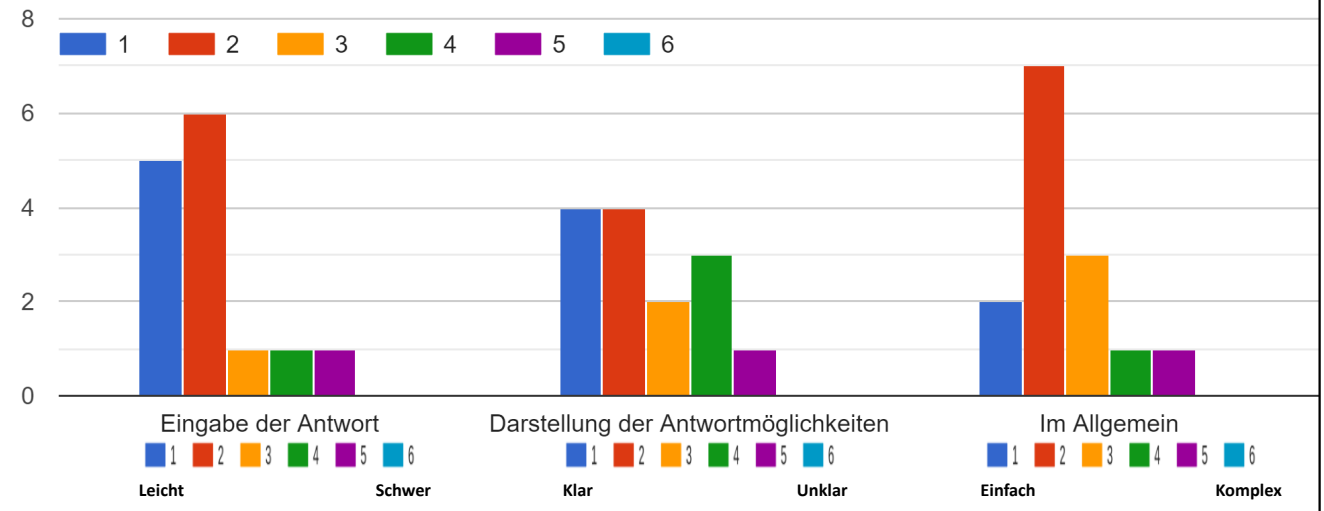

11:40
Prototypes
Vereinbaren Sie einen Termin?
1
Tag.Monat.Jahr
AUSWÄHLEN
Stunden:Minuten
AUSWÄHLEN
WEITER
11:40
Prototypes
Vereinbaren Sie einen Termin?
2
Datum:
T T
M M
J J
Zeit:
S S
M M
WEITER
11:40
Prototypes
Vereinbaren Sie einen Termin?
3
Datum:
01
01
2010
Zeit:
00
00
FERTIG

Anhang C Ergebnisse der Studie

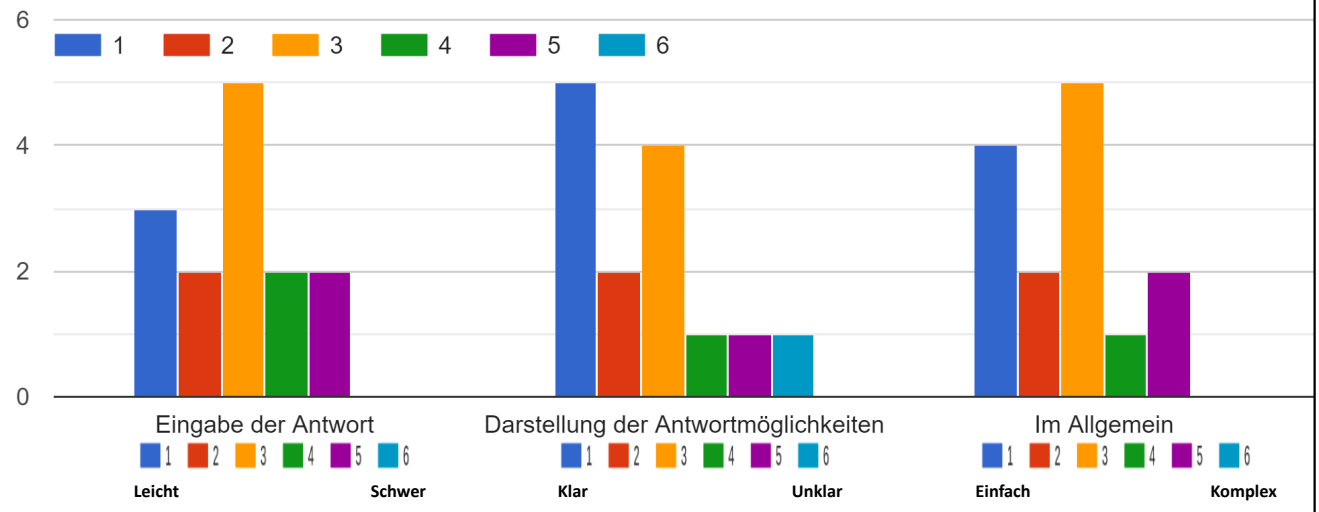
Dichtome 1



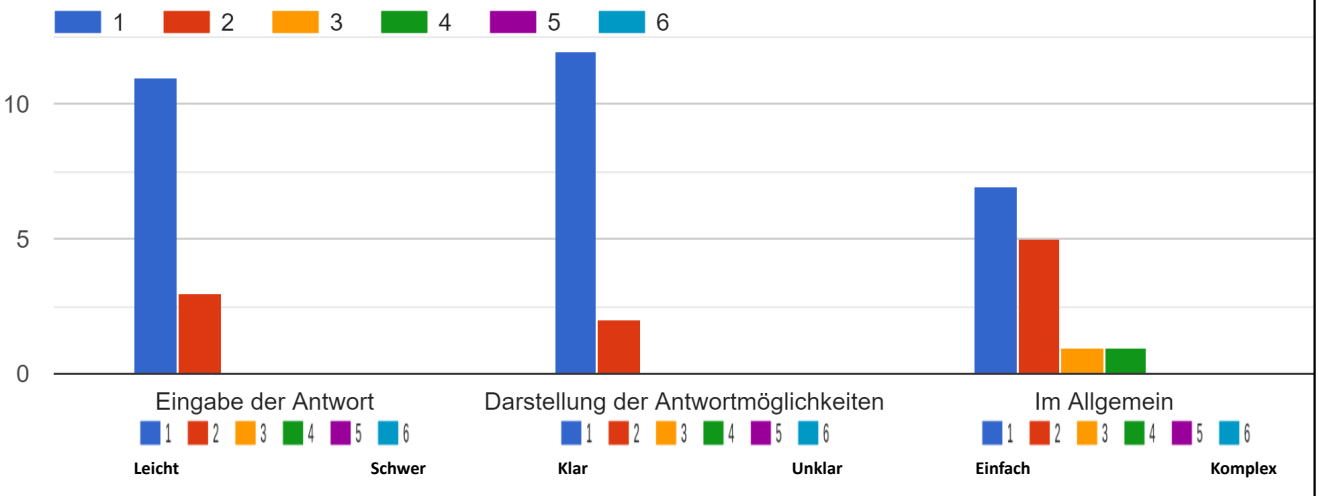
Dichtome 2



Dichtome 3

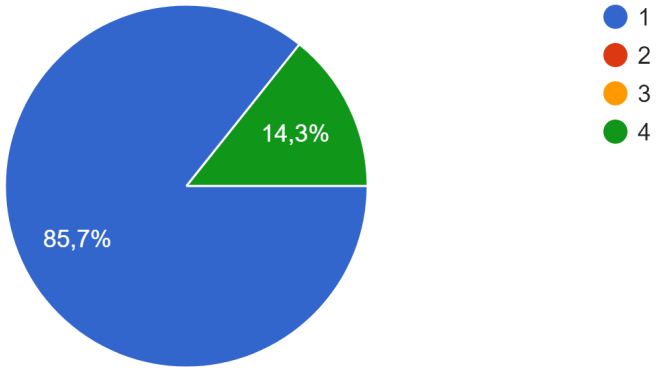


Dichtome 4

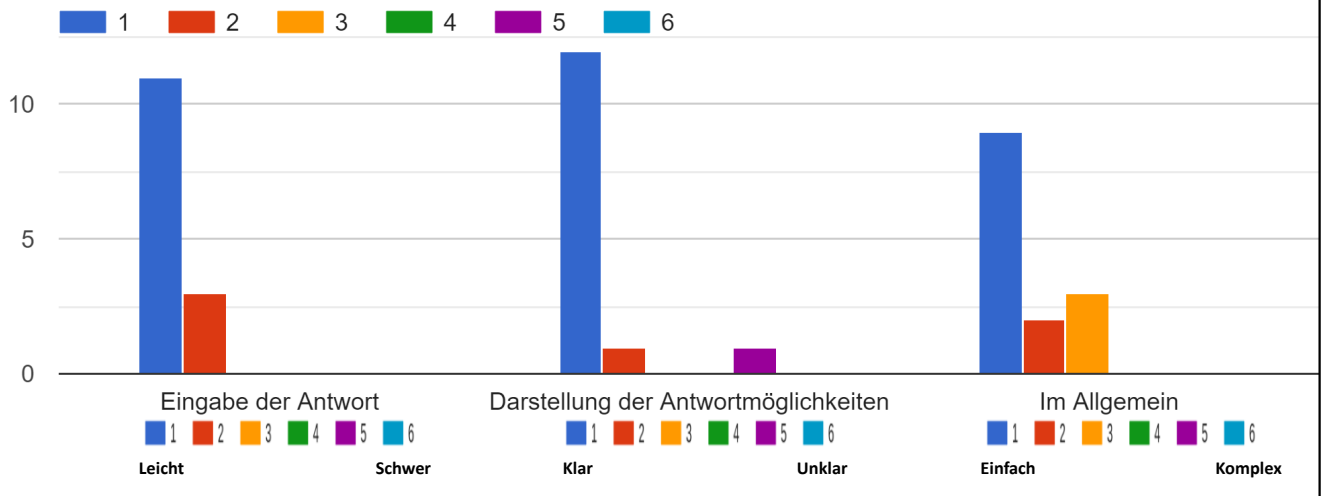


Bevorzugte Darstellung

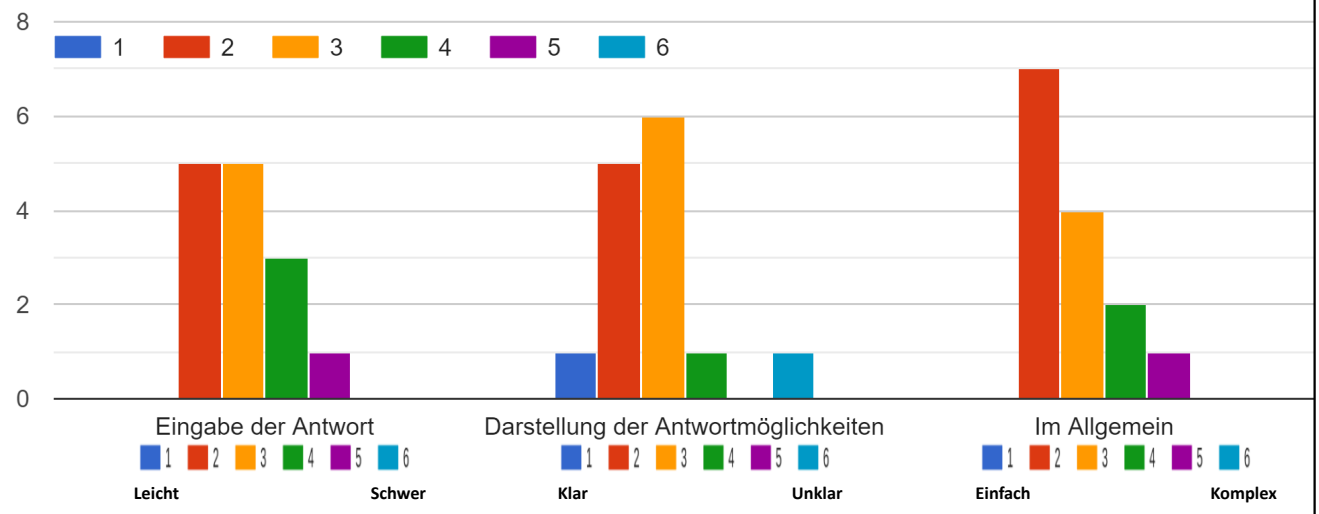
14 Antworten



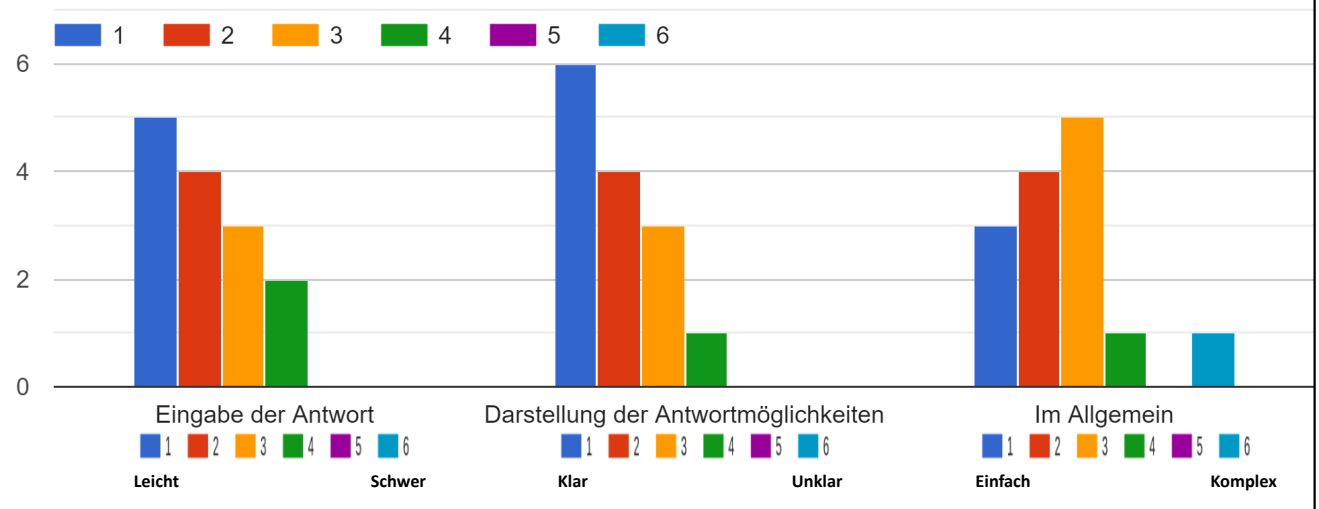
Single Choice 1



Single Choice 2

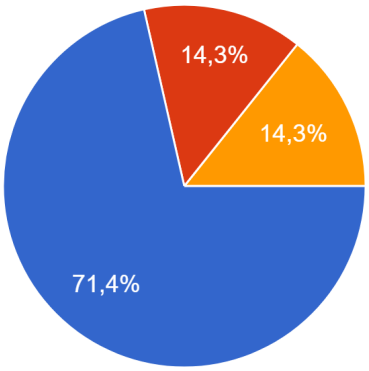


Single Choice 3

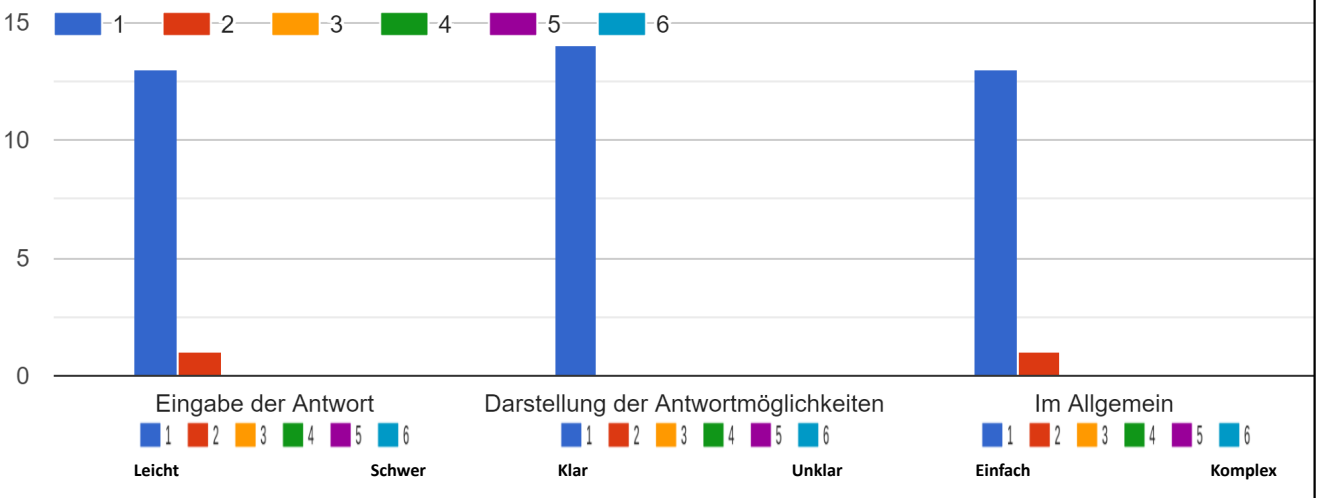


Bevorzugte Darstellung

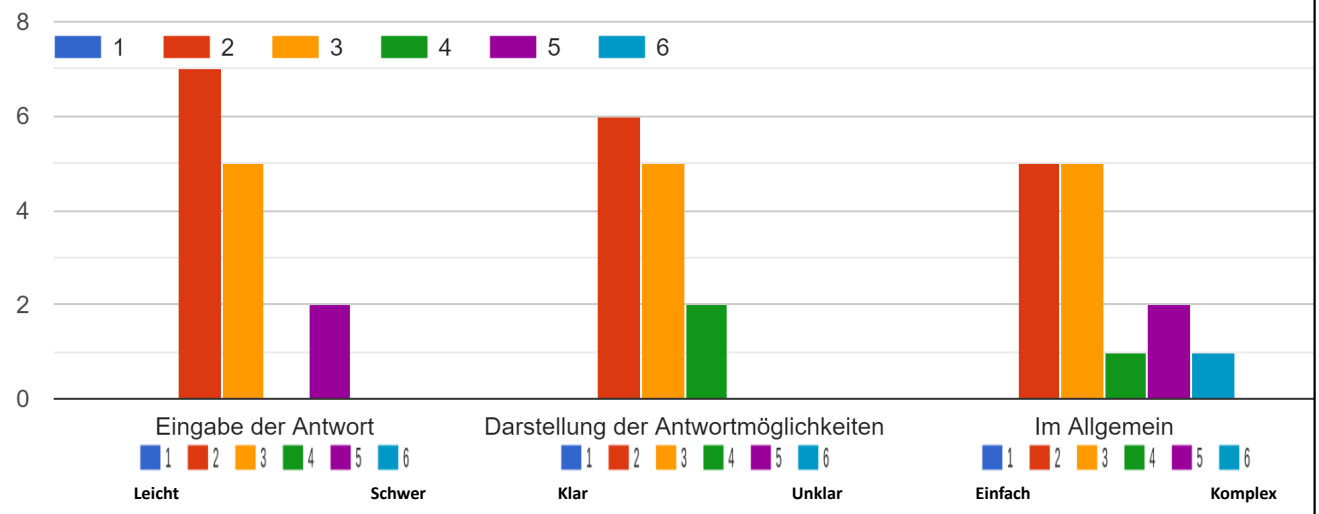
14 Antworten



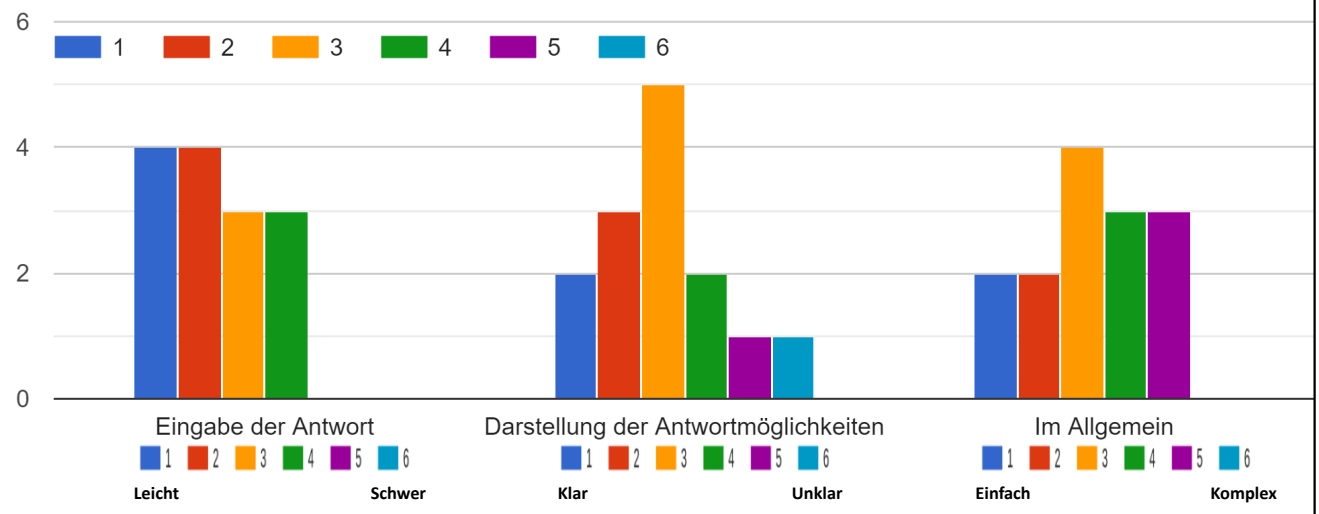
Multiple Choice 1



Multiple Choice 2

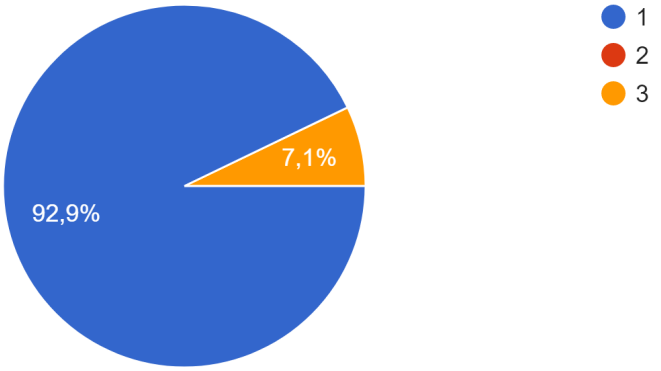


Multiple Choice 3

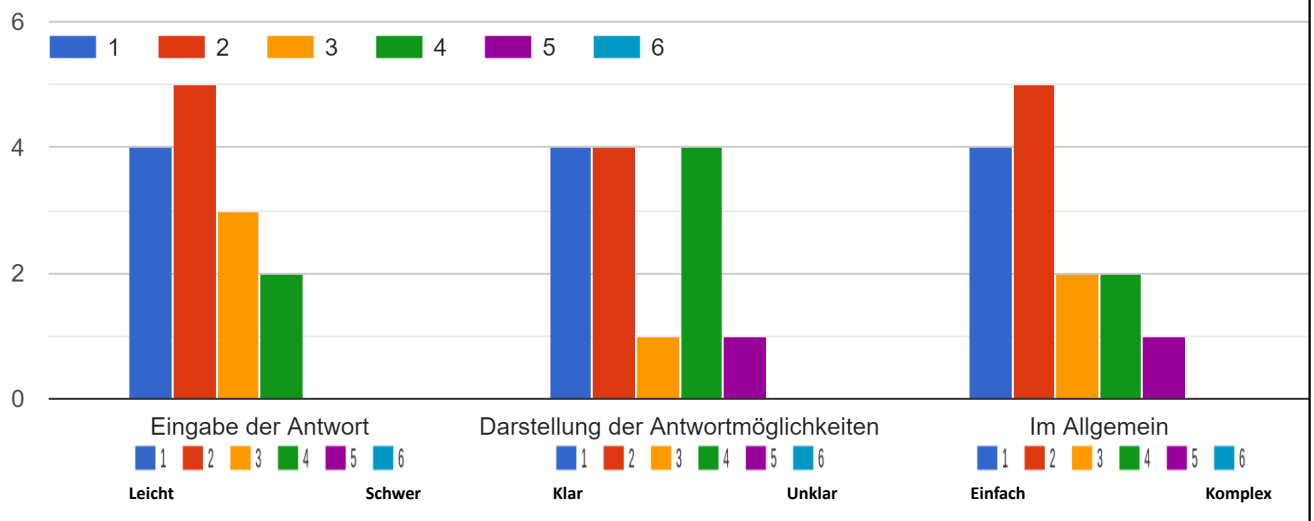


Bevorzugte Darstellung

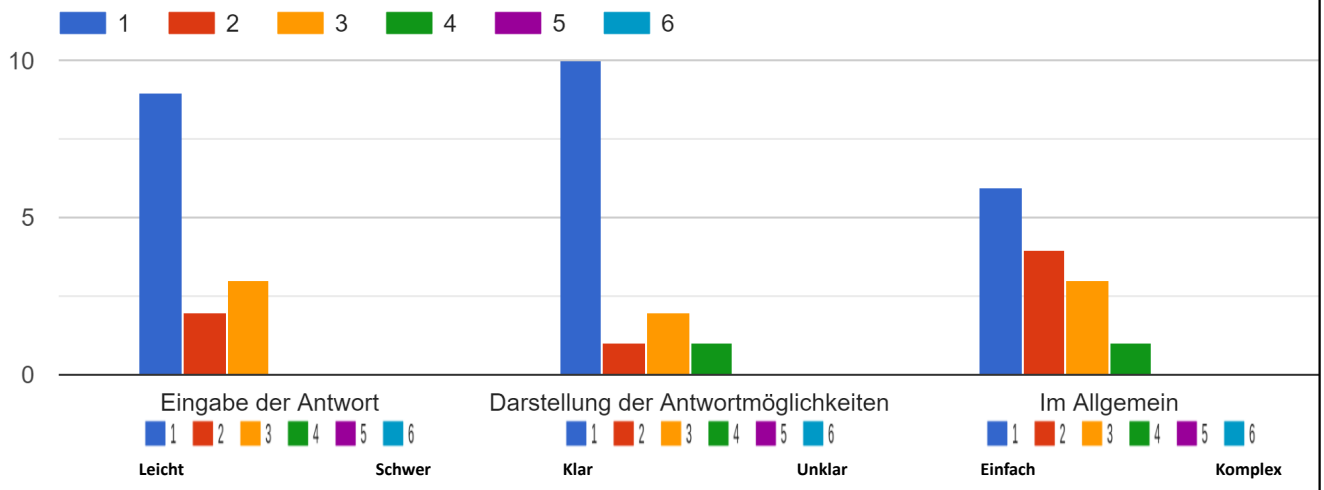
14 Antworten



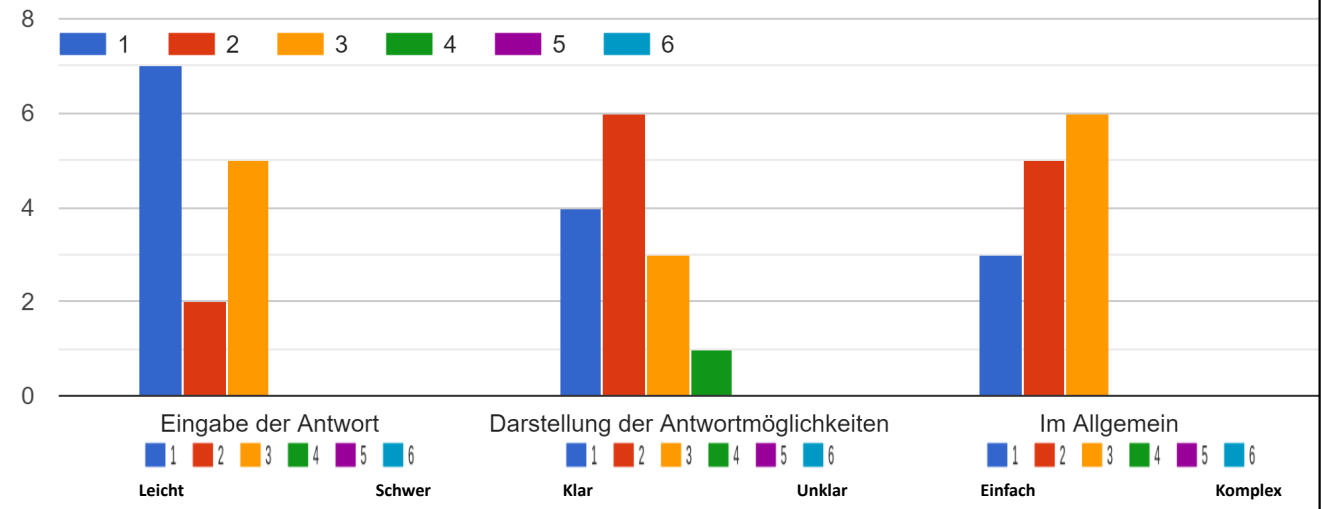
Skala 1



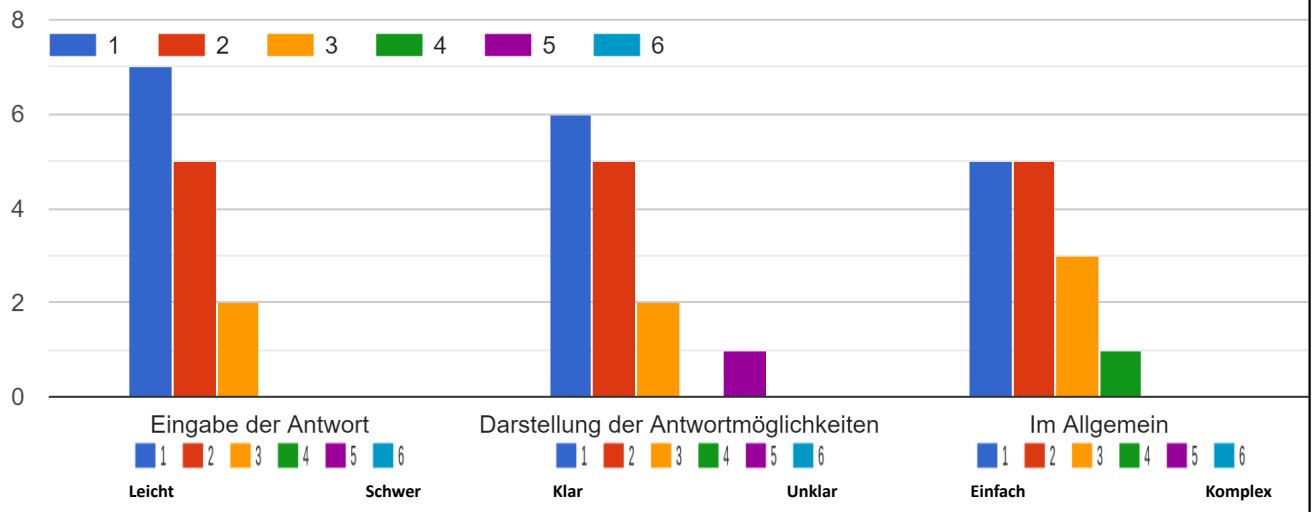
Skala 2



Skala 3

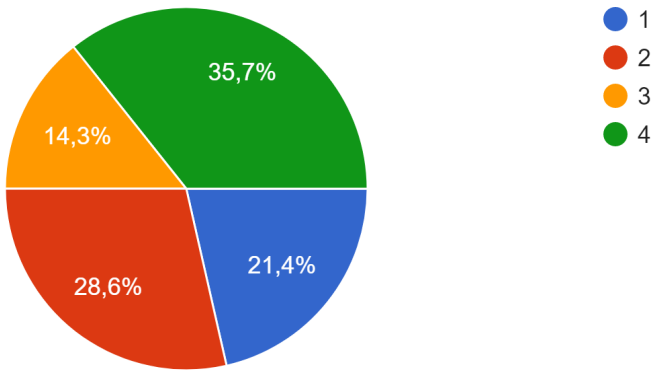


Skala 4

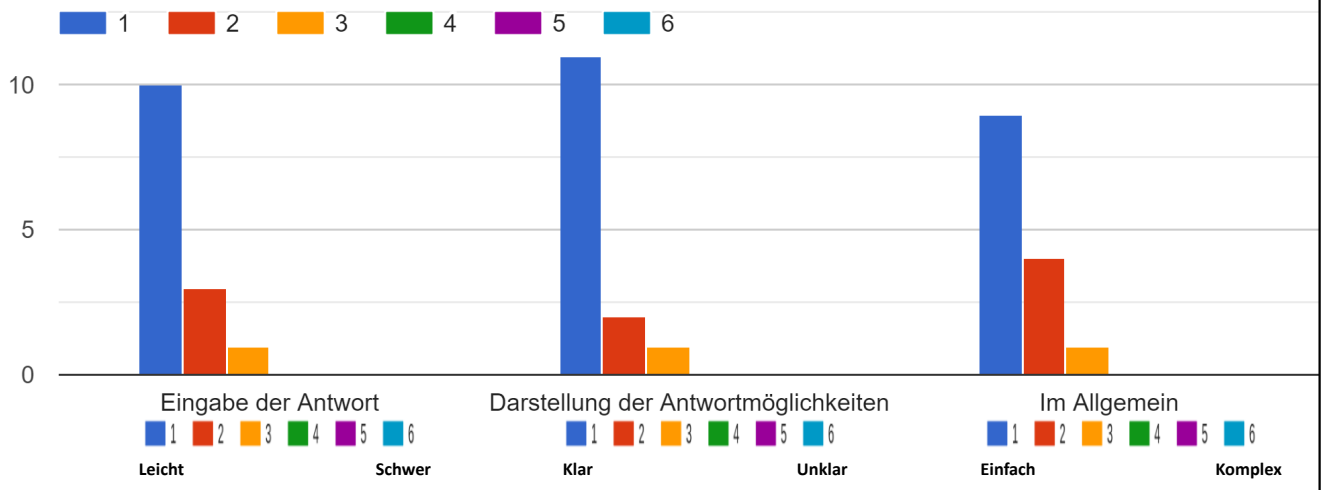


Bevorzugte Darstellung

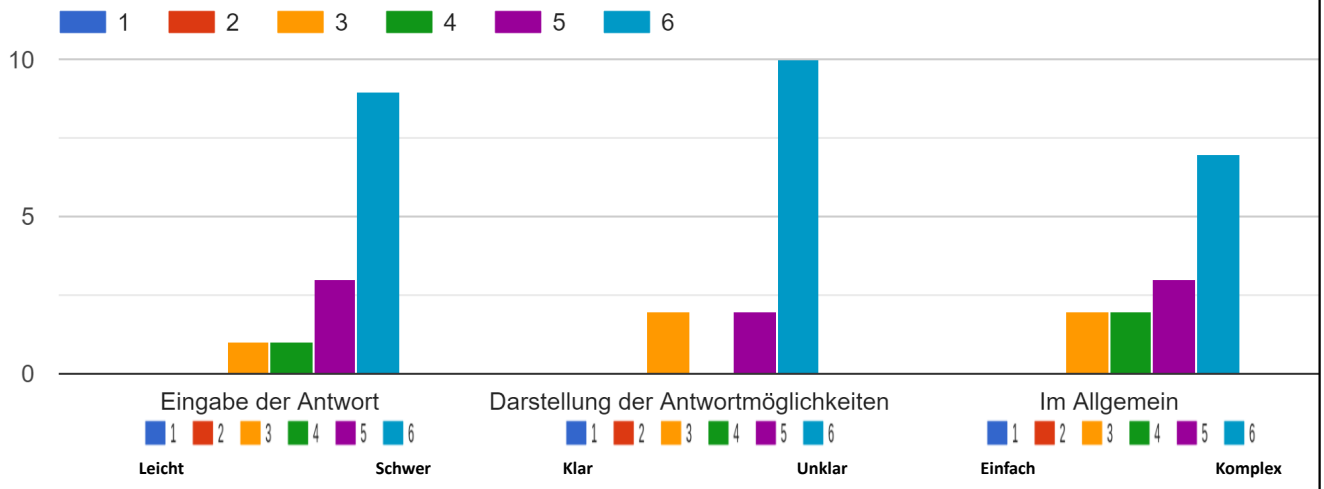
14 Antworten



Multi Skala 1

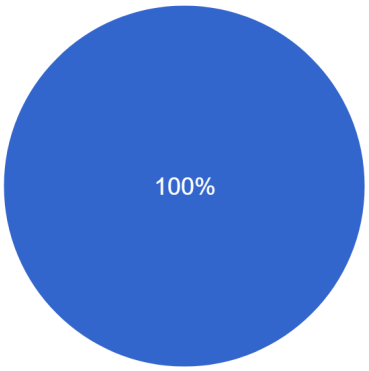


Multi Skala 2

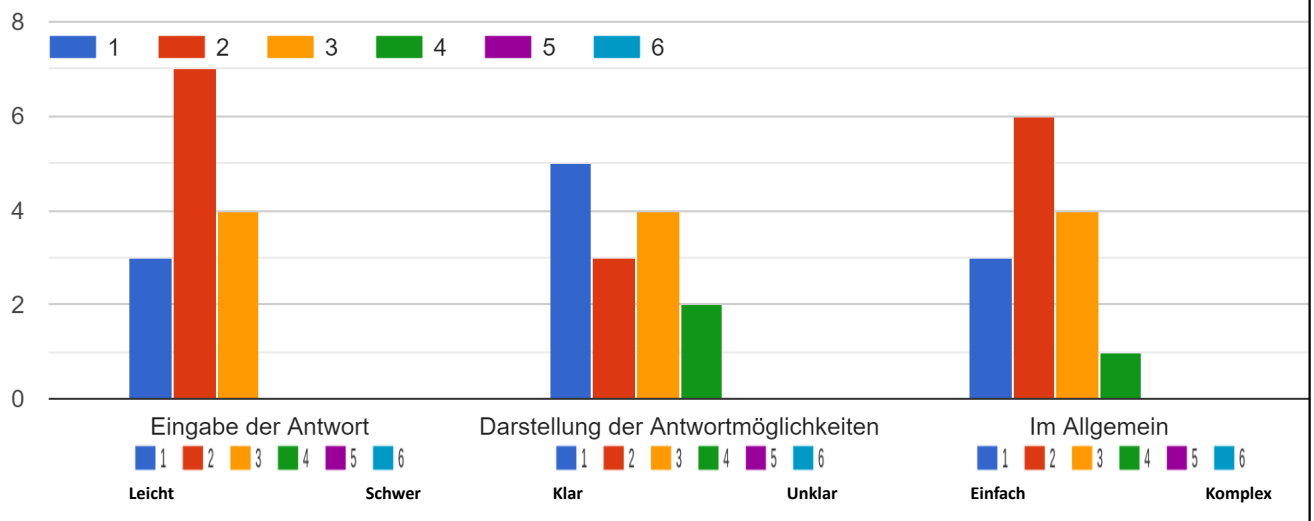


Bevorzugte Darstellung

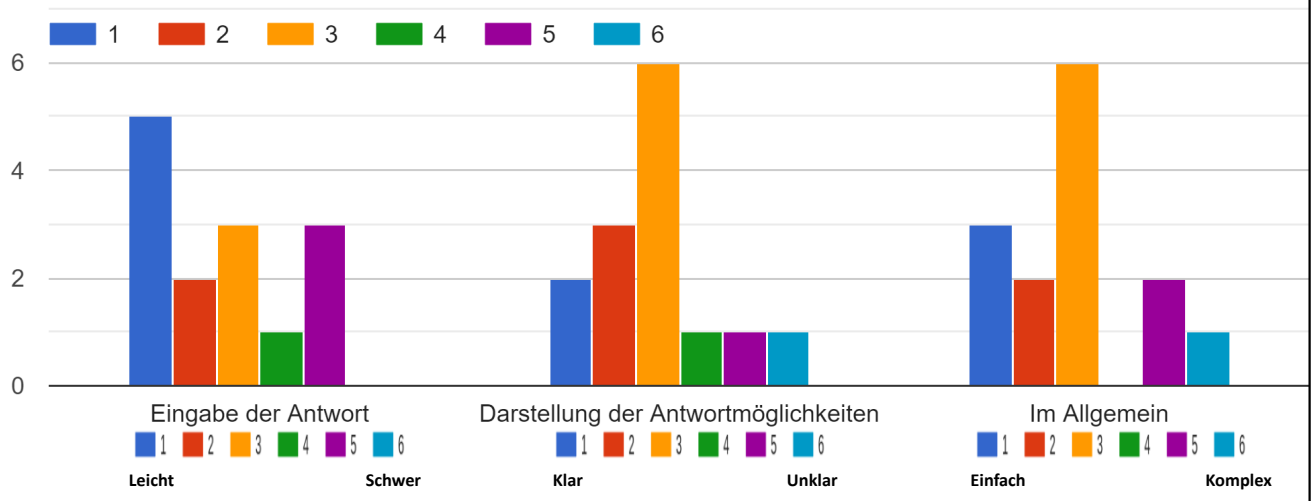
13 Antworten



Freie Antwort 1

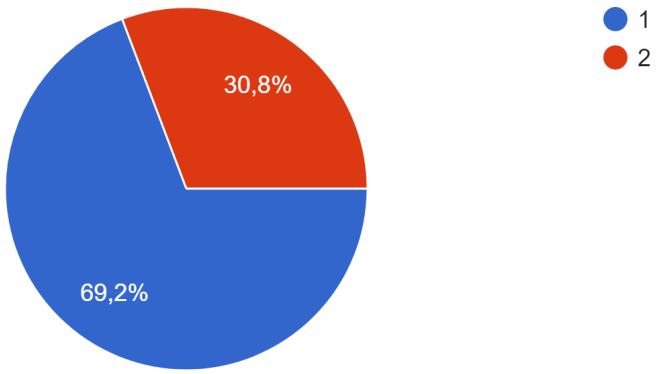


Freie Antwort 2

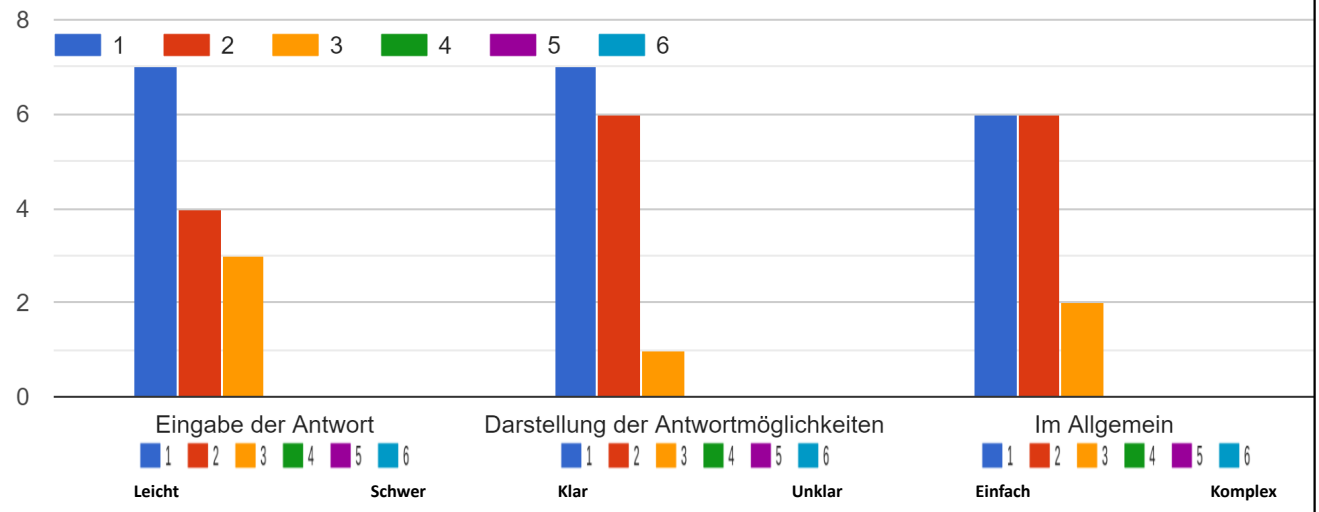


Bevorzugte Darstellung

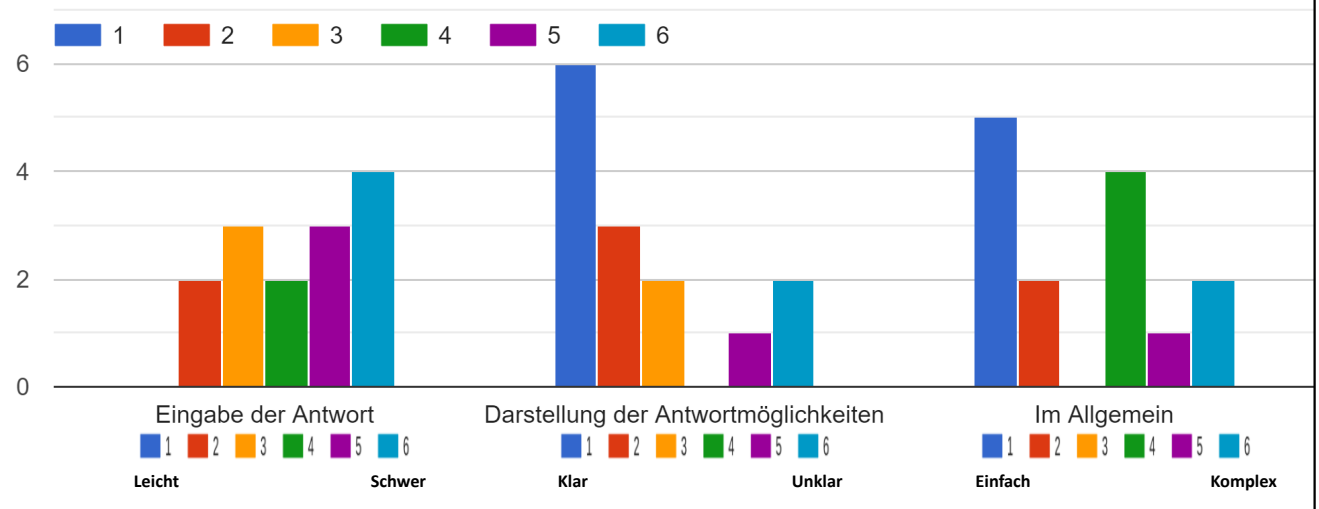
13 Antworten



Anordnen 1

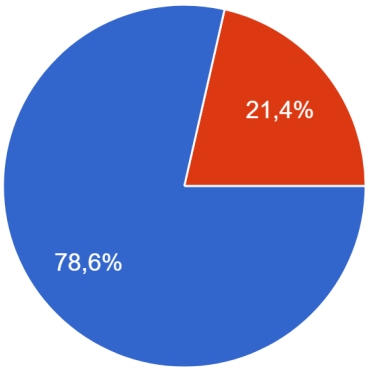


Anordnen 2

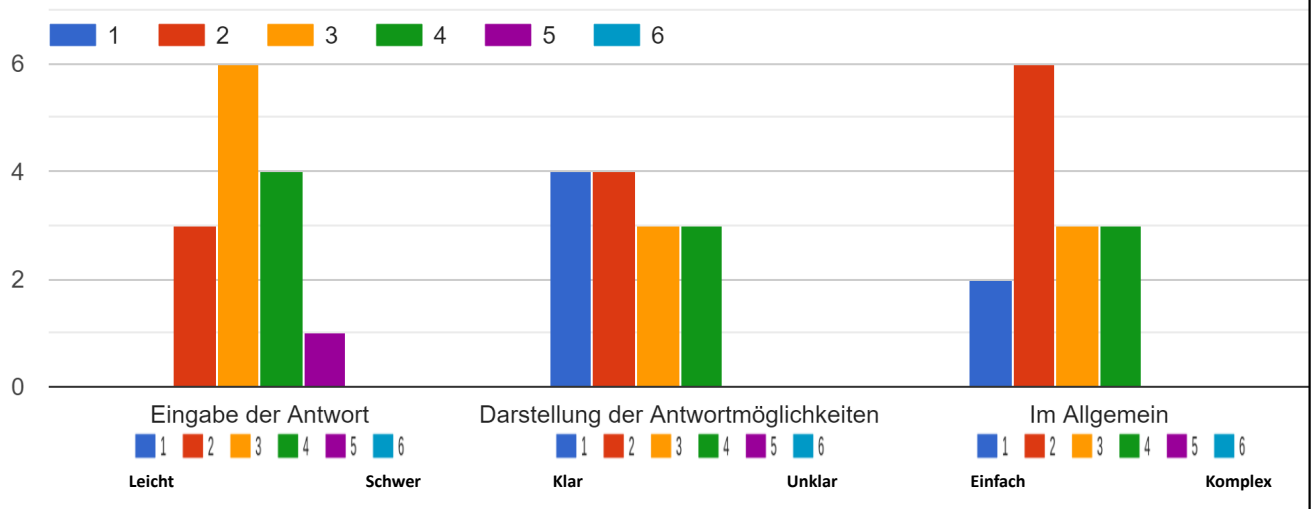


Bevorzugte Darstellung

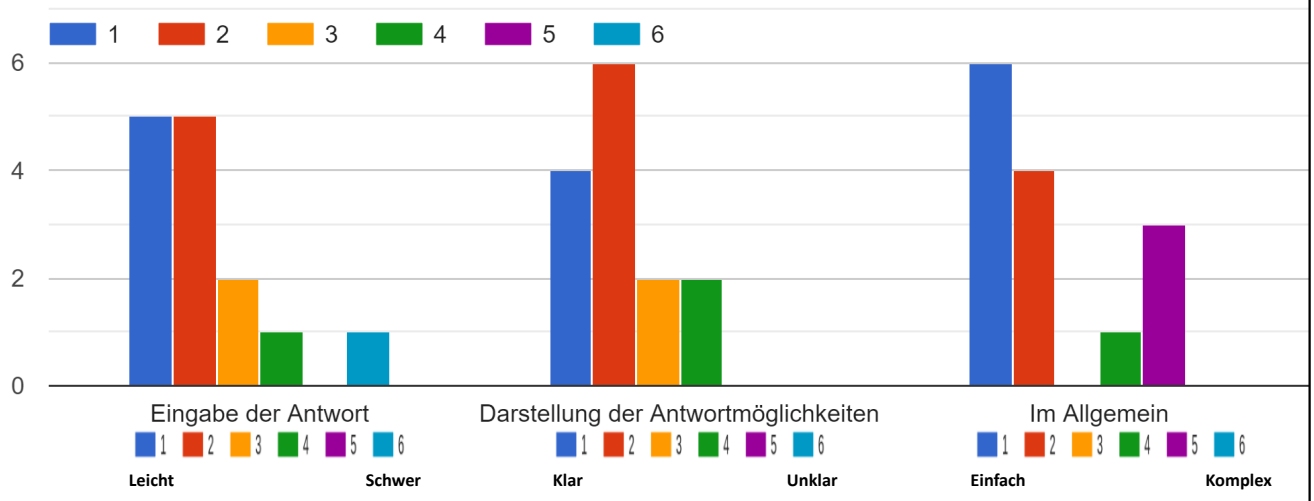
14 Antworten



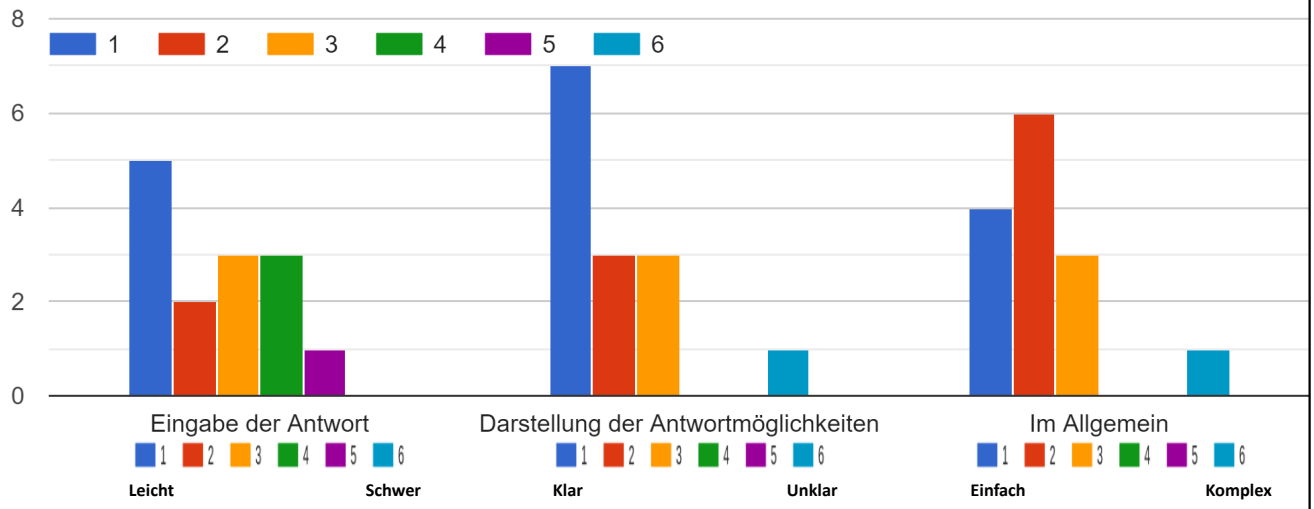
Zuordnen 1



Zuordnen 2

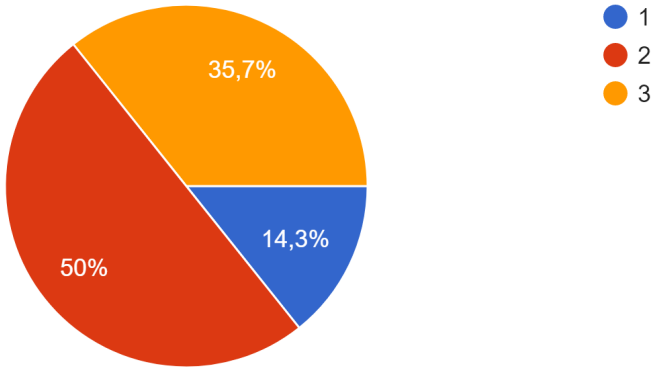


Zuordnen 3

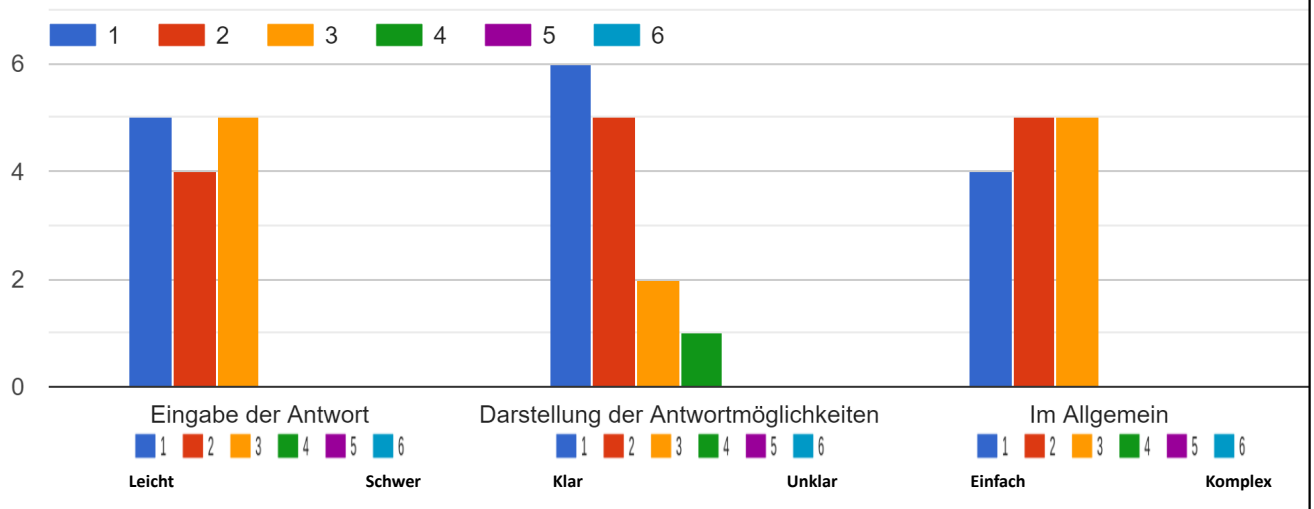


Bevorzugte Darstellung

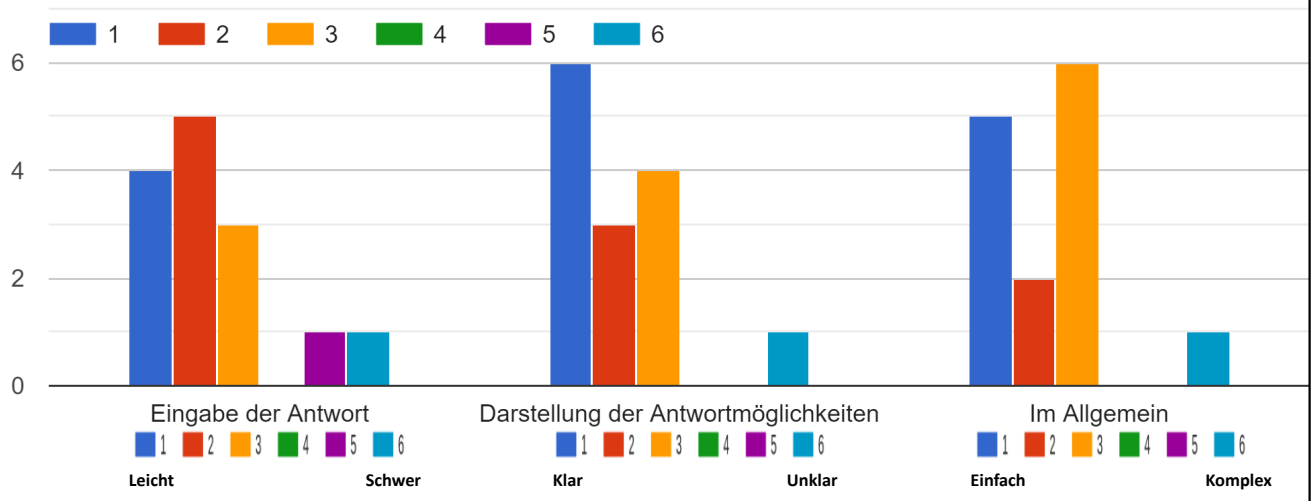
14 Antworten



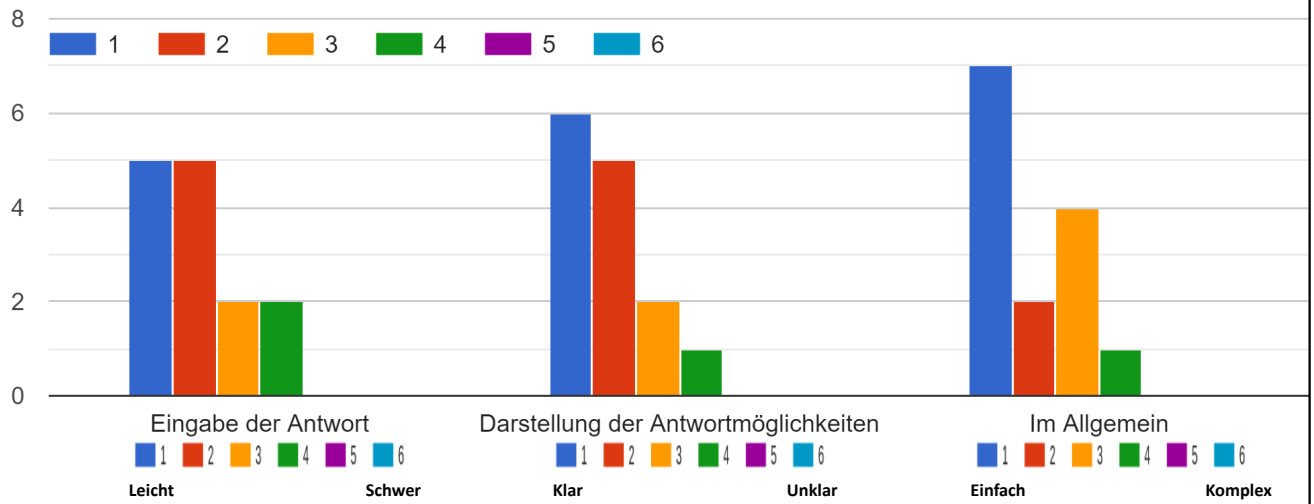
Datum 1



Datum 2

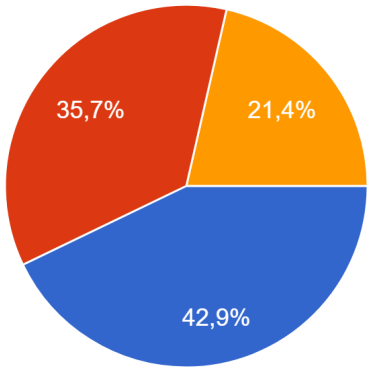


Datum 3



Bevorzugte Darstellung

14 Antworten



Anhang D Code Dokumentation

D.1. Klassen der Bibliothek

D.1.1. DatabaseHelper Klasse

Ziel der Implementation dieser Klasse ist das Speichern der Antworten in einer Datenbank sowohl lokal auf dem Gerät als auch Online auf einem Server. Diese Klasse ist eine Erweiterung der Klasse *SQLiteOpenHelper*, die aus der Native-Bibliothek SQLite stammt. Mittels dieser Klasse wird eine SQLite Datenbank erstellt. Diese Datenbank wird *Answers* genannt. Alle Tabellen in dieser Datenbank haben sechs Spalten, wie in Tabelle D.1 zu sehen ist.

Table 1					
ID	QuestionId	SelectedAnswer	IsCorrect	Status	TIMESTAMP

Tabelle D.1.: Beispiel für eine Tabelle in der Datenbank

Diese Klasse hat zwei Methoden, die Daten speichern: Die Methode *saveToLocalStorage* für das lokale Speichern der Daten und die Methode *saveToAppServer* für das Speichern der Daten auf einem Server.
Das lokale Speichern der Daten erfolgt mittels der *saveToLocalStorage* Methode, welche die Daten als Parameter bekommt und sie lokal in einer Tabelle in der *Answers* Datenbank speichert. Diese Methode ist vom Typ Boolean und gibt den Wert 1 zurück, wenn die Daten erfolgreich gespeichert wurden, und sonst den Wert 0. Allerdings ist das Speichern der Daten nur möglich, wenn schon bei Erstellung einer Darstellung die Parameter *saveButtonId* und *questionId* initialisiert wurden.
Das Online Speichern erfolgt durch die Methode *saveToAppServer*, die mit Hilfe einer Schnittstelle die Daten auf einem Server speichert. Die Schnittstelle wurde mit der Programmiersprache PHP implementiert. Diese bekommt die von der Applikation geschickte *POST* Anfrage und dekodiert die JSON Daten, um sie in der Datenbank zu speichern. Außerdem gibt sie eine Antwort zurück und zwar *OK* wenn die Daten erfolgreich auf dem Server gespeichert wurden, sonst gibt sie die Antwort *FAILED* zurück. In der Abbildung 6.3 ist dieser Prozess dargestellt.
Um die Daten online speichern zu können, ist die Erstellung einer Datenbank sowie die Verwendung unserer Schnittstelle erforderlich. Zur ersten Erstellung der Datenbank wurde ein Skript geschrieben, das auf einer *Mysql* Datenbank durchführbar ist. Dieses Skript erstellt die *Answers* Datenbank sowie alle notwendigen Tabellen. Dabei ist zu beachten, dass das Ausführen dieses Skripts nur einmalig erforderlich ist.

Typ des Parameters	Name des Parameters	Übergabe
Activity	activity	Erforderlich
int	LayoutId	
String []	Answers	
int	saveButtonId	Optional
String	questionId	
String	correctAnswer	

Tabelle D.2.: Parameter der Methoden

D.1.2. NetworkStateChecker Klasse

Diese Klasse ist eine Erweiterung der *BroadcastReceiver*, die bei der Erstellung einer Verbindung mit dem Internet in der lokalen Datenbank nach Daten sucht, die nur lokal gespeichert wurden, nämlich den Status 0 haben. diese Klasse hat die Methode *saveToServer*, die aufgerufen wird und dementsprechend die lokale Daten automatisch auf dem Server speichert.

D.1.3. Parameter der Methoden der Generatorklassen

Generatorklassen bestehen aus verschiedenen statischen Methoden und Attributen. Es gibt bei all diesen Klassen gleiche Attribute, die für die Implementation wichtig sind, nämlich die Attribute *SelectedAnswer*, *MyDb* und *TABLE_NAME* Das Attribut *selectedAnswer* ist vom Typ String und wird genutzt, um die eingegebene Antwort zwischen zu speichern. Das Attribut *myDB* ist nötig, um die Datenbank zu identifizieren, in der die eingegebene Antwort gespeichert wird. Das Attribut *TABLE_NAME* bezeichnet den Namen der Tabelle, wo die Antworten einer Generatorklasse gespeichert werden sollen. Die Attribute *myDB* und *TABLE_NAME* sind konstant und schon definiert.
Außerdem haben die Methoden aller Generatorklassen ebenfalls gleiche Parameter, die für die Implementation der Darstellungsvarianten wichtig sind. Diese sind in der Tabelle D.2 aufgelistet.
Dieser Parameter können beim Aufruf einer Methode übergeben werden, was dann eine entsprechende Funktionalität bei der Erzeugung einer Darstellung aktiviert.
Die Initialisierung des Parameters *activity* ist erforderlich. Mittels dieses Parameters wird festgestellt, in welcher Aktivität eine Darstellung erzeugt werden soll. Außerdem wird das Parameter benutzt, um den *Context* festzustellen.
Der Parameter *saveButtonId* ist wichtig, um die eingegebene Antwort einer Darstellung speichern zu können. Der Parameter *questionId* ist nötig, um zu identifizieren zu welcher Frage die eingegebene Antwort gehört. Bei der Initialisierung dieses Parameters und des Parameters *saveButtonId* können dann die eingegebenen Antworten in einer Datenbank gespeichert werden.
Durch die Initialisierung des Parameters *correctAnswer* wird in der Datenbank gespeichert, ob die eingegebene Antwort richtig ist.

D.1.4. Generatorklassen

In der Bibliothek gibt es neun verschiedene Generatorklassen, die unterschiedliche Darstellungsvarianten von Fragen- und Antworttypen umsetzten. In den folgenden Abschnitten werden diese genauer erläutert.

RadioButtonsGenerator Klasse

Diese Klasse stellt die Methode *addRadioButtons* zur Verfügung, die die Darstellung der Antwortalternativen mittels *RadioButtons* ermöglicht. Diese Methode bekommt als Eingabe ein Layout, in dem die *RadioButtons* erzeugt werden sollen, und die Antwortalternativen. Dementsprechend werden diese Antwortalternativen mit Radiobuttons in diesem Layout dargestellt.
Die Methoden dieser Klasse haben in ihren Signatur zusätzlich zu den oben in der Tabelle D.2 erwähnten Parametern zwei Parameter: Der Parameter *Scrollbar* und das Parameter *extraAnswer* vom Typ Boolean. Der Parameter *Scrollbar* bietet die Möglichkeit, eine Scrollbar zu der Darstellung hinzuzufügen, während das Parameter *extraAnswer* die Erweiterung der geschlossene Frage auf eine halboffene Frage mittels *Textbox* ermöglicht.

ImageButtonsGenerator Klasse

Diese Klasse stellt die Methode *addImageWithRadioButton* zur Verfügung, die Bilder als *Imagebuttons* darstellt. Diese Methode bekommt als Eingabe ein Layout, in dem Bilder bereits angelegt sind, und ein Array von IDs dieser Bilder. Dementsprechend werden diese Bilder als *Imagebuttons* in dem Layout dargestellt werden. Dabei ist zu beachten dass diese Bilder bereits mit einer ID initialisiert werden müssen, da diese für die Methode als Eingabe erforderlich ist. Außerdem gilt diese ID als Repräsentant eines Bildes. Diese ist ebenfalls erforderlich, um die Antwort in der Datenbank zu speichern, da das ausgewählte Bild selber nicht in der Datenbank gespeichert werden kann.

ChcekBoxesGenerator Klasse

Diese Klasse stellt die Methode *addCheckboxes* zur Verfügung, die die Darstellung der Antwortalternativen mittels *Checkboxes* ermöglicht. Diese Methode bekommt als Eingabe ein Layout, in dem die *Checkboxes* erzeugt werden sollen, und die Antwortalternativen. Dementsprechend werden diese Antwortalternativen mit *Checkboxes* in diesem Layout dargestellt. Diese Methoden dieser Klasse haben, ähnlich wie die Methoden der Klasse *RadioButtonsGenerator*, zwei zusätzliche Parameter, nämlich den Parameter *Scrollbar* und den Parameter *extraAnswer*, die genau wie in *RadioButtonsGenerator* funktionieren.

SkalaGenerator Klasse

Diese Klasse stellt drei Methoden zur Verfügung. Die erste Methode ist *addIntSeekbar*, die die Darstellung einer Skala von a bis b ermöglicht, wobei a und b Integer Werte sind und an der Grenze der Skala angezeigt werden. Diese Methode bekommt als Eingabe ein Layout, in dem die Skala erzeugt werden sollen, sowie zwei Werte *min* und *max* von Typ Integer. Dementsprechend erzeugt sie in diesem Layout eine Skala mit *min* als unterer Schranke und *max* als oberer Schranke. Außerdem wird ein *Textview* erzeugt, das die aktuelle Stellung dieser Skala anzeigt. Die zweite Methode ist *addStringSeekbar*, die die Darstellung einer Skala von a bis b ermöglicht, wobei hier a und b Werte von Typ String sind und als Beschriftung der oberen und unteren Grenze gelten. Diese Methode bekommt als Eingabe ebenfalls ein Layout und zwei Werte, allerdings vom Typ String und zusätzlich einen Wert *max* von Typ Integer, der die maximale Anzahl der Schritte bezeichnet. Hierbei wird ebenfalls mit Hilfe eines *Textview* die aktuelle Stellung dieser Skala angezeigt. Bei den ersten zwei Methoden ist zu beachten, dass ein neues ConstraintLayout erstellt werden muss, damit *min*, *max* und *Textview* richtig positioniert werden. Die dritte Methode ist *addRatingBar*, die die Darstellung *Rating Stars* ermöglicht. Diese Methode

bekommt wie die andern Methoden ein Layout als Eingabe. Außerdem bekommt sie als Eingabe die Anzahl der Sterne, die erzeugt werden sollen, sowie die Größe der Schritt als Wert vom Typ Integer und eine Beschriftung als Wert vom Typ String. Dementsprechend erzeugt sie ein entsprechendes *Rating Stars*. Dabei ist zu beachten, dass die kleinste Schrittgröße ein halber ist, welche durch die Eingabe von Null bei der Parameter *setpSize* gesetzt wird.

MultiSkalaGenerator Klasse

Diese Klasse stellt die Methode *addMultiSkala* zur Verfügung, die die Antwortmöglichkeiten in einer *Checkboxes Matrix* darstellt. Diese Methode bekommt als Eingabe ausschließlich ein TableLayout sowie die Anzahl der *Checkboxes* in einer Zeile und die Anzahl der *Checkboxes* in einer Spalte, die erzeugt werden sollen. Außerdem bekommt sie als Eingabe die Antwortmöglichkeiten in Form eines zweidimensionalen Array, wobei die erste Dimension eine Zeile und die zweite Dimension eine Spalte bezeichnet. Dementsprechend wird eine Matrix mit den eingegebenen Antwortmöglichkeiten in dem TableLayout erzeugt.

FreieAntwortGenerator Klasse

Diese Klasse stellt die Methode *addFreieAntwort* zur Verfügung, die eine *Textbox* erzeugt. Diese Methode bekommt ein Layout als Eingabe und erzeugt in diesem Layout eine *Textbox*. Außerdem bekommt sie einen Hinweis vom Typ String, der auf dem *Textbox* angezeigt wird.

SortListGenerator Klasse

Diese Klasse stellt drei Methoden zur Verfügung, die die Darstellung der Antwortmöglichkeiten in Form einer *Sort-List* ermöglicht. Diese Methoden bekommen als Eingabe ein Layout. Außerdem bekommen sie als Parameter einen Wert von Typ Boolean, der entscheidet, ob die Antwortalternativen zentriert dargestellt werden sollen.
Die erste Methode ist *addTextSortList*, die als Eingabe zusätzlich die Antwortalternativen als ein Array von String bekommt und diese in dem Layout so darstellt, dass sie sortierbar sind.
Die zweite Methode ist *makeTextSortList*, die ähnlich zu der ersten Methode ist, aber anstatt die Antwortalternativen als array vom Typ String einzugeben, können hier IDs von bereit erstellten *Textview*s als ein Array vom Typ Integer eingegeben werden. Außerdem muss genau das Layout eingegeben werden, in dem sich diese *Textview* befinden. Diese Methode bietet den Vorteil, dass der Entwickler selbst *Textview*s erstellen und designen kann, die dann in Form einer sortierbare Liste dargestellt werden können.
Die dritte Methode heißt *makeImageSortList*. Diese Methode ist vom Prinzip her genau wie die zweite Methode, aber anstatt *Textview*s in einer sortierbare Liste darzustellen, können hier IDs von bereit angelegten Bildern eingegeben werden. Diese Bilder werden in einer Liste so dargestellt, dass sie sortierbar sind. Dabei ist zu beachten, dass die angelegten Bilder das Attribut *ID*, das als Repräsentant eines Bilds gilt, initialisiert haben müssen. Der Grund dafür ist, wie oben für die Methode *addImageWithRadioButton* bereits erklärt wurde, dass der Inhalt der *ID* in der Datenbank als Antwort gespeichert wird. So wird zum Beispiel bei der Sortierung von drei Bildern X, Y und Z in der Reihenfolge [Y, X, Z], die jeweils das Attribut *ID* mit den Werten *BildX*, *BildY* und *BildZ* haben, dann die Antwort in der Datenbank als [BildY, BildX, BildZ] gespeichert.

DropDownListGenerator Klasse

Diese Klasse stellt die Methode *addDropDownList* zur Verfügung, die Antwortalternativen in einer Liste mit Hilfe von *Dropdown-List* darstellt. Diese Methode bekommt als Eingabe ein Layout sowie die Antwortalternativen und einen Hinweis. Dieser Hinweis ist vom Typ String und wird als vorgewählter aber nicht auswählbarer Eintrag angezeigt. Nach einer gültigen Eingabe dieser Parameter wird eine entsprechende *Dropdown-List* in dem eingegeben Layout mit den Antwortalternativen und dem Hinweis erzeugt.

DateAndTimeGenerator Klasse

Mit Hilfe dieser Klasse können zwei Darstellungsmöglichkeiten von Fragen nach Datum und Zeit erzeugt werden: Die erste Darstellung erfolgt mit Hilfe von *Date-Time Picker* und die zweite Darstellung erfolgt mit Hilfe von *Dropdown-List*. Für die erste Darstellungsmöglichkeit stellt diese Klasse zwei Sorten von Methoden zur Verfügung: die Methoden erster Sorte bekommen nur ein Layout als Eingabe und dementsprechend wird in diesem Layout sowohl einen Button als auch ein *Textview* erzeugt. Beim Drücken auf den Button wird ein *Time- oder Date Picker* angezeigt. In dem *Textview* wird dann die ausgewählte Antwort angezeigt. Die Methoden der zweiten Sorte bekommen als Eingabe die ID eines bereits erstellten Buttons sowie die ID eines bereits erstellten *Textview*. Beim Drücken dieses Buttons wird ein *Time bzw. Date Picker* erzeugt und in dem eingegebenen *Textview* wird die ausgewählte Antwort angezeigt. Die Methoden *addTimeDialog* und *addDateDialog* gehören zu den ersten zwei Gruppen.
Für die Darstellungsmöglichkeit *Dropdown-List* stellt diese Klasse zwei Methoden zur Verfügung. Die erste Methode ist *addDropDownTime* und die zweite ist *addDropDownDate*. Diese Methoden bekommen als Parameter nur ein Layout und erzeugen in diesem Layout das Datum bzw. die Uhrzeit in Form einer *Dropdown-List*.

D.2. Installation

Um unsere Bibliothek bei der Entwicklung einer Applikation verwenden zu können, muss sie in dem Projekt, wo diese Applikation entwickelt wird, importiert werden. Außerdem muss für das online Speichern der Daten eine Datenbank auf einem Server erstellt werden. Diese sind genauer in beigefügten der Datei *Installation Guide* erklärt.

Eidesstattliche Versicherung

Sabouni, Walid — 362896

Name, Vorname — Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/Masterarbeit* mit dem Titel

Konzeption und Entwicklung einer Android-Library zur Darstellung von Antworttypen

selbstständig und ohne unzulässige Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen , 8. April 2019

Ort, Datum — Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bisw 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen , 8. April 2019

Ort, Datum — Unterschrift