Gedächtnistraining mit n-back

Matthias Ramsauer nbackmemory@matthias-ramsauer.de

Andreas Fillinger s-afilin@haw-landshut.de

19. Januar 2020

Inhaltsverzeichnis

1 Beschreibung

1.1 Vision

Nachrichten, Social Media und Katzenvideos bombardieren uns mit Informationen ohne Ende - mit negativen Auswirkungen auf Konzentration und Kurzzeitgedächtnis. Wer dem entgehen möchte, muss seinen Kopf mit regelmäßigem Training auf Trab halten. Genau dafür ist diese App gedacht.

„Gedächtnistraining mit n-back“ implementiert eine Gedächtnisübung namens „n-back-Test“, welche in einem Paper aus dem Jahre 1958 erdacht wurde. Die tägliche Nutzung der App soll dabei helfen, Kurzzeitgedächtnis und Konzentrationsfähigkeit des Nutzers zu steigern. Statistiken über die tägliche Nutzung, darunter Highscores und Diagramme, legen dem Nutzer seine Fortschritte dar und bieten eine Motivation zur Weiternutzung der App.

1.2 Zielgruppe

Die App ist auf jeden ausgerichtet, der sein Gedächtnis trainieren möchte: Darunter sind Schüler, Studenten, Erwachsene und insbesondere ältere Menschen. „Gedächtnistraining mit n-back“ eignet sich auch für Personen, die wenig Zeit haben, denn wenige Minuten täglichen Trainings sollen sichtbare Ergebnisse liefern.

1.3 Die App

Allgemein läuft ein n-back-Test folgendermaßen ab: Einer Testperson wird nacheinander eine Menge von Reizen gegeben (Zahlen, Farben, Geräusche, ...). Die Aufgabe besteht darin, sich jederzeit eine bestimmte Anzahl (= n) vergangener Reize zu merken und abrufen zu können. Konkret soll in dieser App ein arithmetischer n-back-Test implementiert werden. Dabei wird dem Nutzer nacheinander ein simpler arithmetischer Ausdruck (deren Lösung einstellig ist) vorgegeben. Das Ziel ist jeweils das Ergebnis anzugeben, dass zu dem Ausdruck n Stellen zuvor gehört. (Beispielsweise wird bei n=1 das Ergebnis des vorherigen Ausdrucks erwartet, bei n=2 das Ergebnis des vorvorletzten Ausdrucks etc.)

2 Entwurf

Im folgenden Abschnitt werden die Funktionen der App beschrieben und im Anschluss mittels Illustrationen veranschaulicht.

2.1 Funktionsbeschreibung

2.1.1 Hauptmenü

Das Hauptmenü ist der Screen, der beim Starten der App immer als erstes erscheint. Es enthält Einträge, über die man auf die anderen Funktionen der App zugreifen kann. Dazu gehören:

- Das Starten der Übung
- Das Konfigurieren der Übungsparameter
- Das Einsehen der Statistiken

(Siehe Abb 1a für eine Skizze des Hauptmenüs)

2.1.2 Übung starten

Eine Übung besteht aus einer Menge an Runden. In jeder Runde wird dem Nutzer ein Ausdruck gezeigt, woraufhin er eine Zahl auswählt. Ob diese Eingabe korrekt ist (d.h. zu dem Ausdruck n Stellen zuvor passt[1]) soll dem Nutzer farblich vermittelt werden (z.B. korrekt = Eingabe grün färben; falsch = Eingabe rot, korrekte Antwort blau färben). Nach der Eingabe startet die nächste Runde mit dem Anzeigen eines neuen Ausdrucks. Wann die Übung endet hängt von den gewählten Übungsparametern (siehe 2.1.3) ab:

- Nach einer bestimmten Menge an Runden
- Nach der nächsten Runde ab Ablauf eines bestimmten Zeitlimits

Zusammenfassend enthält der Bildschirm während einer Übung folgende Elemente:

- Den aktuellen Ausdruck
- Ein Nummernfeld zur Eingabe der Antwort

[1] Für die ersten n Runden wird keine Eingabe erwartet, hier soll nach 2-3 Sekunden die nächste Runde von Selbst starten

- Die aktuelle Rundenzahl
- Die Endbedingung der Übung: Maximale Rundenzahl oder verbleibende Zeit

Ist die Übung beendet, wird der Ergebnisbildschirm angezeigt, dieser enthält folgende Informationen:

- Den Parameter n
- Die Anzahl der korrekten Eingaben
- Die Anzahl aller Eingaben
- Die Trefferquote (#korrekt / #gesamt)
- Die erzielte Punktzahl [2]
- Die höchste heute erzielte Punktzahl
- Die höchste jemals erzielte Punktzahl

Für jede Übung werden die Punktzahl, Parameter n und das Datum lokal gespeichert, um Statistiken erstellen zu können. (siehe 2.1.4)

Aus dem Ergebnisbildschirm führen den Nutzer 2 Buttons: Einer soll ihn zurück in das Hauptmenü bringen, der andere startet eine neue Übung mit den selben Parametern.
Abb. 2 stellt einen möglichen Ablauf für eine Übung mit n=1 dar.

2.1.3 Konfigurationsmenü

In diesem Menü sollen die Übungsparameter geändert werden können. Zu jedem Parameter soll es einen Eintrag bestehend aus dem Namen und dem aktuellen Wert geben. Berührung eines Eintrags öffnet ein zum Parametertyp passendes Eingabewidget, Auswahl eines Wertes aktualisiert den im Eintrag angezeigten Wert. Zusätzlich sollen dem Nutzer 3 Buttons (etwa in einer Appbar) zur Verfügung stehen:

- Änderungen übernehmen
- Änderungen verwerfen
- Defaultwerte wiederherstellen

Jeder dieser Buttons soll den Nutzer erst mit einem Popup fragen, ob er die Aktion wirklich durchführen will, bei Bestätigung dann die Aktion ausführen und den Nutzer zurück in das Hauptmenü bringen.

[2] Für die Berechnung der Punktzahl, siehe 2.1.4

Mindestens sollen folgende Parameter existieren:

- n (`type = integer, default = 1`)
  Schwierigkeitsgrad der Übung; Gibt an, zu welchem Ausdruck (relativ zum aktuellen Ausdruck) die Lösung erwartet wird.
- Übungsende (`type = Enum [ZEIT, ANZAHL], default = ANZAHL`)
  Gibt an, ob das Ende der Übung anhand eines Zeitlimits (ZEIT) oder anhand der Anzahl der Runden (ANZAHL) feststeht.
- Zeitlimit (`type = [integer, integer], default = [3, 0]`)
  Anzahl der Minuten und Sekunden, nach deren Ablauf die Übung endet (wenn als Übungsende ZEIT gewählt wurde).
- Rundenlimit (`type = integer, default = 20`)
  Anzahl der Runden, bevor die Übung endet (wenn als Übungsende ANZAHL gewählt wurde).

2.1.4 Statistiken

Zur Motivation soll der Nutzer hier Statistiken über abgeschlossene Übungen abrufen können. Um Übungen mit unterschiedlichen Parametern vergleichen zu können, soll zu jeder Übung eine Punktzahl errechnet werden. Eine mögliche Punktzahlfunktion könnte so aussehen:

$$Score = (\#korrekt - \lceil 0.5 * \#falsch \rceil) * 10^n$$

Das Statistiken-Menü soll 2 Unterfunktionen bieten, die im Folgenden beschrieben werden.

Übungsliste Hier werden alle abgeschlossenen Übung untereinander aufgelistet. Jeder Eintrag enthält Datum, n und Punktzahl der Übung. Sortiert soll die Liste anfangs aufsteigend nach Datum sein, über einen Button in der Appbar soll der Nutzer zwischen Sortierung nach Datum und nach Punktzahl wechseln können.

Diagramme Hier soll der Nutzer seine Leistung in einem von ihm gewählten Zeitraum in Form eines Diagrammes betrachten können.

Der Nutzer soll 3 Auswahlmöglichkeiten für den Zeitraum haben:

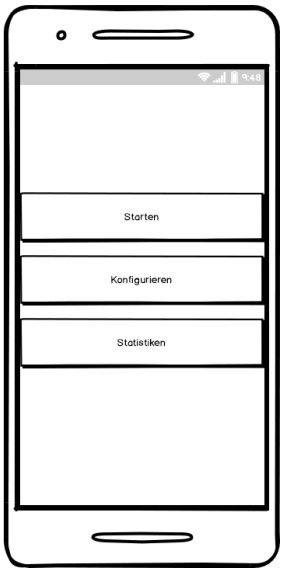
- aktuelle Woche
- aktueller Monat
- aktuelles Jahr

Nach Auswahl eines Zeitraumes soll ein Diagramm generiert werden:

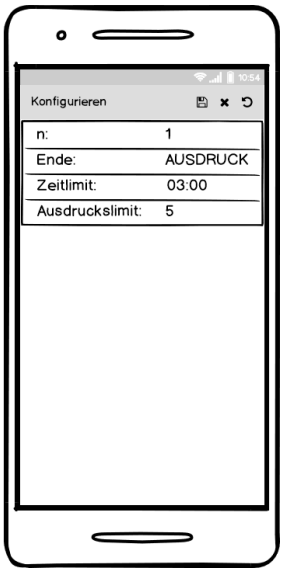
- Die x-Achse sei die Zeit
- Die y-Achse sei der Punktestand
- Jede Übung, die im gewählten Zeitraum stattfand, bildet einen Punkt mit x = Datum, y = Punktzahl

Die Eingabe für den Zeitraum soll sich auf dem selben Screen befinden wie das Diagramm. Standardmäßig wird beim Öffnen des Diagramm-Screens der Zeitraum 'Woche' gewählt und das entsprechende Diagramm für die aktuelle Woche angezeigt. Durch Auswahl eines anderen Zeitraums wird sofort ein neues Diagramm generiert.

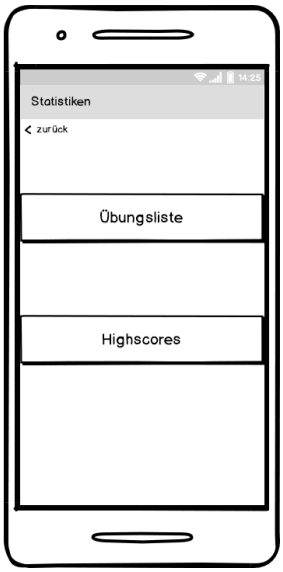
2.2 Skizzen



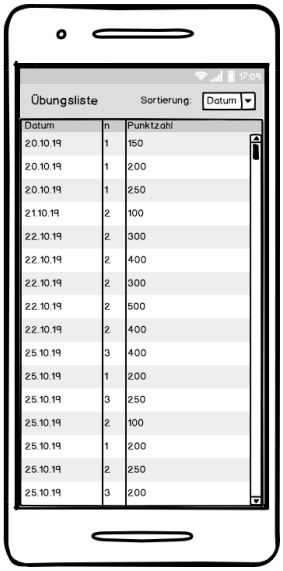
(a) Das Hauptmenü



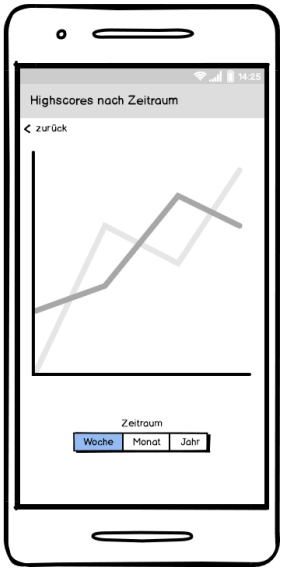
(b) Das Konfigurationsmenü



(c) Das Statistikenmenü



(d) Die Übungsliste



(e) Der Diagrammbildschirm

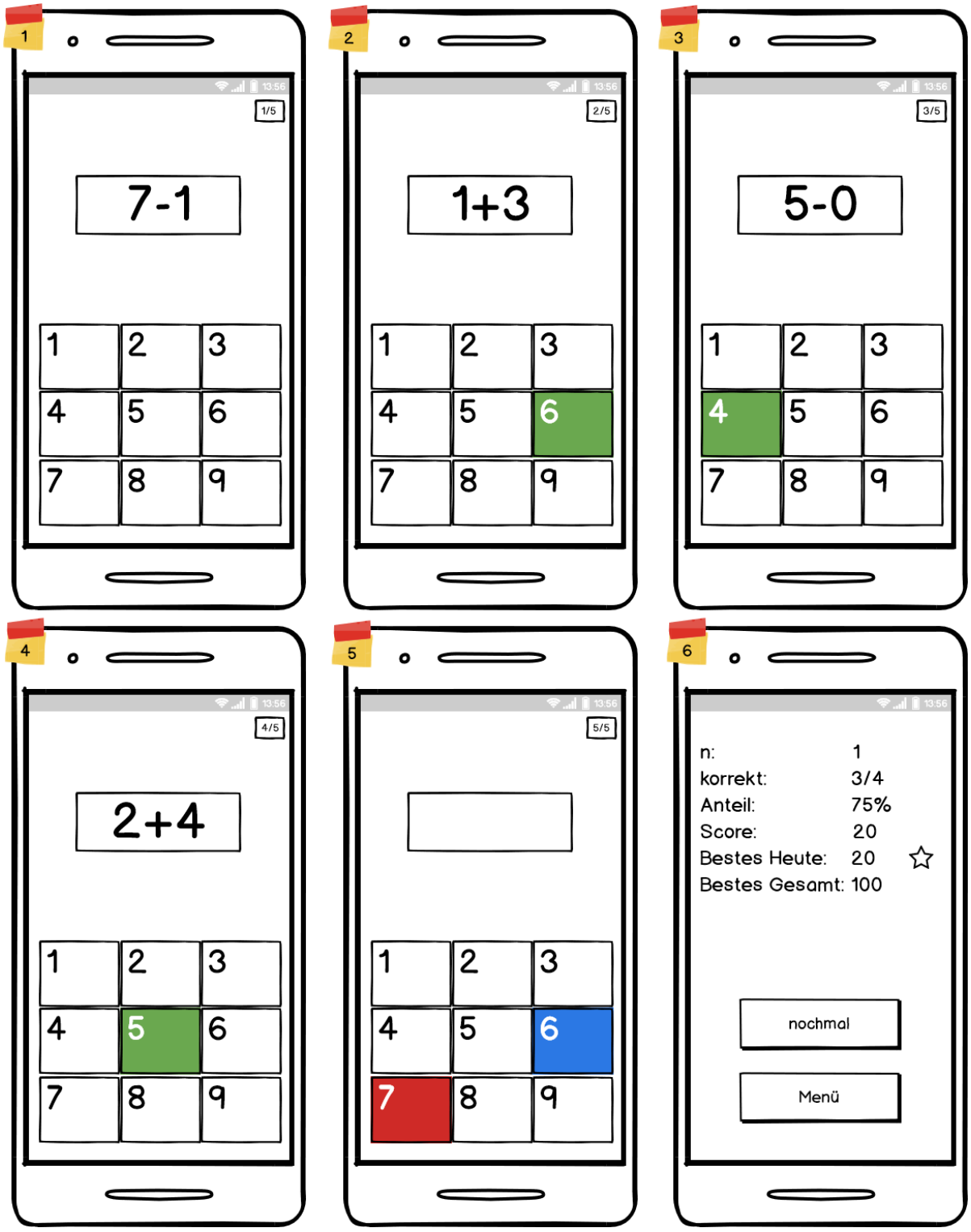


Abb. 2: Ablauf einer Übung mit n=1

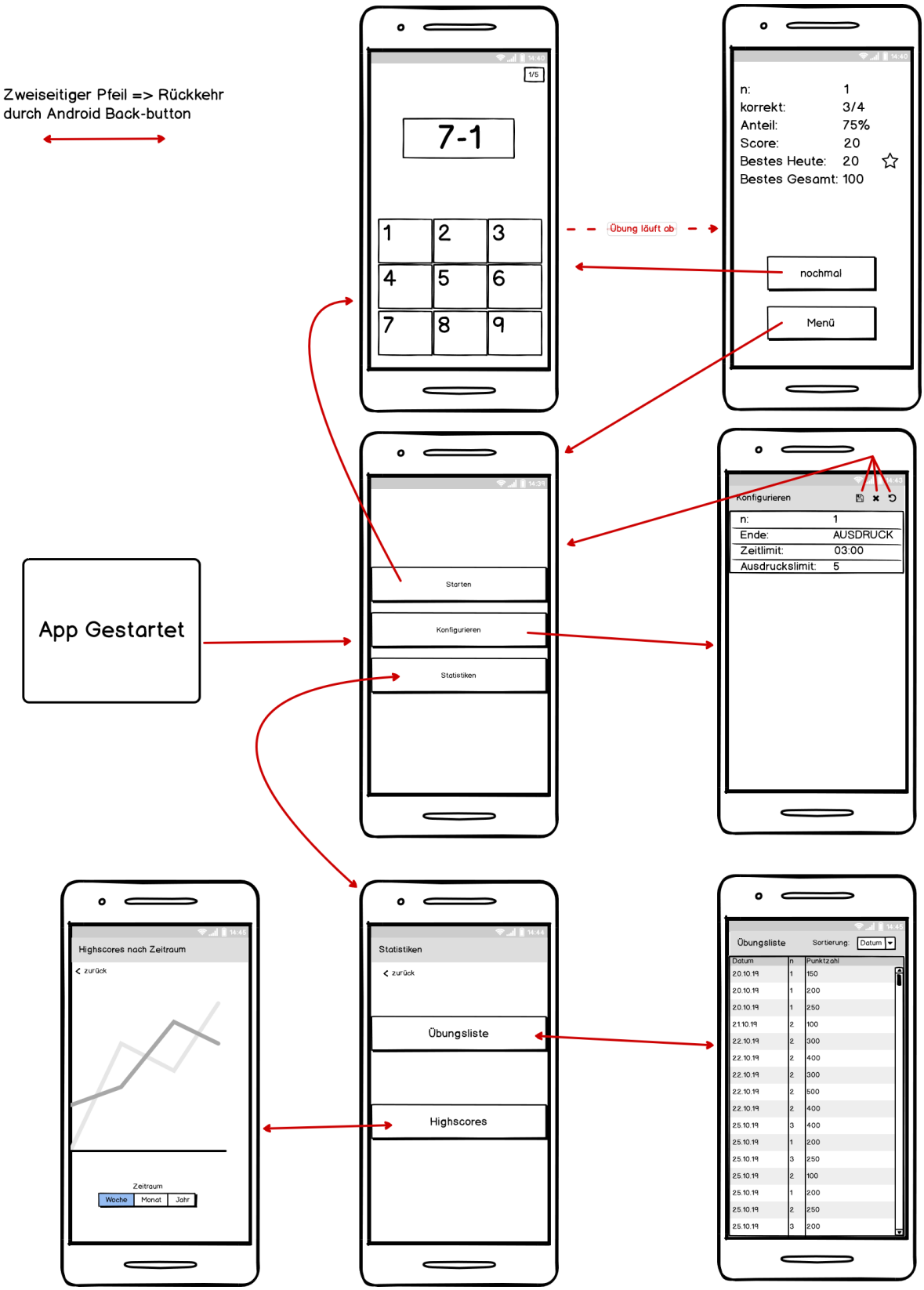


Abb. 3: Übersicht über den Ablauf der Screens

2.3 Optionale Funktion: Musikalische Untermalung

Die in diesem Abschnitt beschriebene Funktion kann implementiert werden, ist aber nicht zwingend erforderlich. Eine Implementierung muss sich auch nicht exakt an die Beschreibung in diesem Kapitel halten.

Der Nutzer kann sich eine Musikdatei aussuchen, diese wird dann während der Übung abgespielt. Folgende zusätzliche Parameter werden hierzu im Konfigurationsmenü (Abschnitt 2.1.3) angeboten:

- Musikdatei (`type = string, default = 'none'`)
  Titel einer Musikdatei auf dem Gerät
- Abspielart (`type = Enum [NOTENWEISE, STANDARD], default = STANDARD`)
  Gibt an, wie das Musikstück wiedergegeben wird

Ist eine Musikdatei gewählt (d.h. Parameter 'Musikdatei' ungleich 'none') wird während der Übung das gewählte Stück auf einer von zwei möglichen Arten wiedergegeben, abhängig vom Parameter 'Abspielart':

- `'Abspielart' = 'NOTENWEISE'`:
  Zu jeder richtigen Antwort des Nutzers wird ein Ton des Stücks abgespielt. Bei einer falschen Antwort wird entweder ein Ton übersprungen, oder ein Fehlerton abgespielt.
- `'Abspielart' = 'Standard'`:
  Zu Beginn der Übung wird die Wiedergabe des Musikstücks gestartet.

Berührt der Nutzer den Eintrag 'Musikdatei' im Konfigurationsmenü wird eine Liste verfügbarer Musikdateien angezeigt. Diese kann auf eine (oder Beide) von zwei Arten befüllt werden:

- Das Gerät wird nach Musikdateien durchsucht
- Die App stellt Musikdateien bereit [3]

[3] Hierbei muss unbedingt auf die jeweiligen Urheberbedingungen der Musikstücke geachtet werden. Eventuell muss der Urheber jedes verwendeten Musikstücks auf einem seperatem Screen genannt werden, zu dem man beispielsweise aus dem Hauptmenü gelangt.

2.4 Releaseplanung

Release	Funktion mit den wesentlichen Tätigkeiten
v0.0	• Initalisierung Android Projekt • Hauptmenü-Activity (Abb 1a) Erstellen
v0.1	• Einstellungen-Activity (Abb 1b) Erstellen • Speichern über `SharedPreferences` • Implementierung der Toolbar
v0.2	• Implementierung der `GameActivity` (Abb 2) • Implementierung der `ScoreActivity` • Implementierung der „ANZAHL“ Abschlussbedingung (Sec 2.1.3) • Erstellen der Github-Repository
v0.3	• Implementierung der „ZEIT“ Abschlussbedingung (Sec 2)
v0.4	• Implementierung der Statistiken • Speicherung der Daten über SQLite
v0.5	• Bessere Beschreibungen in den Einstellungen • Refactoring
v0.6	• Tests erstellen und Bugfixen
v1.0	• Release vorbereiten
v1.1	• Optionale Features implementieren

3 Entwurf und Implementierung

Mit Abschluss der Konzeption wird nun der Anwendungsentwurf vorgestellt und über die Implementierung berichtet.

Es wird die MVVM-Architektur verwendet. Die Applikation wird in 4 größere Abschnitte unterteilt, die möglichst voneinander getrennt sind und nur über entsprechende Abstraktionsschichten miteinander kommunizieren.

- Einstellungen
- Statistiken
- Spiel
- Datenbank

In diesen Abschnitten wird über Viewmodels bzw. Indents kommuniziert. ViewModels werden verwendet, um zwischen Fragments einer Activity zu vermitteln, während Intents im Kontext der Anwendung zum harten Wechsel zwischen Programmabläufen verwendet wird.

Die einzelnen Gui-Komponenten werden ausschließlich in XML-Format erstellt. Für die Befüllung mit entsprechenden Daten wurde jedoch verzichtet ein entsprechendes Templating Framework zu verwenden, um die Anwendung nicht unnötig aufzublähen.

Für die Anbindung wurde Room benutzt. Dabei handelt es sich um eine Abstraktionsschicht zu den Funktionen von SQLite. Mitunter wird damit auch die Migration von verschiedenen Version der Datenbank behandelt.

Die Gui-Komponenten sind aus dem Material Design Componenten von Google entnommen worden.

3.1 Activities und Fragmente

In diesem Abschnitt werden die entsprechenden Activitys der Anwendung und die dazugehörigen Fragments genauer erläutert.

3.1.1 MainActivity

Die Main-Activity ist die einfachste Activity und dient nur als Hub, um zwischen den Abschnitten zu wechseln.

3.1.2 ConfigActivity

Die `ConfigActivity` wird dafür verwendet, um das SettingsFragment zu starten und die Toolbar Aktionen abzufangen. Die Aktionen und dazugehörigen Buttons werden in den Resourcen als `"menu/configmenu.xml"` hinterlegt.

SettingsFragment Dieses Fragment erbt von `PreferenceFragmentCompat`. Das ist eine Klasse der neuen `androidx` Pakets für die Kompatibilität mit älteren Android Versionen. Sie unterscheiden sich in verschiedenen Teilen, so lässt sich unter anderem der Datentyp der einzelenen Preferences nicht mehr im XML definieren. Daher wird dies hier programmatisch eingestellt. Um den Aufwand möglichst gering zu halten, wird auch zunächst angegeben, dass die einzelnen Einstellungen persistent sind. Das führt dazu, dass die Werte aus den SharedPreferences gelesen werden. Programmatisch werden diese jedoch, nach dem Aufbau der Gui, als nicht persistent gekennzeichnet, damit die Toolbar Aktionen nicht umgangen werden.

3.1.3 AbstractGameActivity

In der AbstractGameActivity wird die Grundfunktionalität der Spielmodi `Time` und `Expression` implementiert. Dabei werden 3 verschiedene Fragmente benutzt, um die verschiedenen Spielzustände darzustellen.

GameFragment Dieses Fragment zeigt die aktuelle Expression an. Außerdem gibt sie Informationen über die den Antwortversatz (n) und den aktuellen Stand, je nach Spielmodi.

NextFragement Dieses Fragment enthält nur einen Button um entsprechend einen Zug weiter zu gehen. Es wird benötigt, um die ersten Aufgaben anzuzeigen, die noch keine Antwort verlangen.

NumpadFragment Dieses Fragment zeigt ein Nummernfeld von 1-9 an. Die einzelnen Zahlenfelder färben sich entsprechend Abb. 2 beim Drücken eines Feldes.

TimeGameActivity Benutzt einen CountDownTimer der die verbleibende Zeit im ViewModel mitspeichert. Dadurch wird bei einem recreate die Zeit am entsprechenden Zeitpunkt fortgeführt.

ExpressionGameActivity Überprüft und zeigt den aktuellen Übungsstatus an.

3.1.4 ScoreActivity

Die Score Activity wird zum Abschluss einer Übung gestartet und zeigt verschiedene Leistungsmerkmale der Übung an. Unter anderem holt sie auch die Highscores aus der Datenbank.

3.1.5 StatsActivity

Die StatsActivity besteht aus einem Tabbed View Pager und beinhaltet nur die Initalisierung des Adapters.

StatsListFragment Dieses Fragment enthält eine Liste mit beliebiger Anzahl an Elementen. Da das Laden aller Elemente nach längerer Benutzung der App schwierig werden kann, werden hierfür immer nur eine gewisse Anzahl an Items geladen und angezeigt. Für diese Funktionaliät wird ein `PagedListAdapter` verwendet.

StatsGraphFragment Dieses Fragment enthält einen Graphen, der die Statistiken über den entsprechend ausgewählten Zeitabschnitt visuell darstellt. Für die graphische Darstellung wurde die Bibiliothek `MPAndroidChart` benutzt.

3.2 Datenzugriffsschicht

Als Datenzugriffschicht wird Room verwendet. Room ermöglicht es, mit wenig Aufwand eine entsprechende Datenbank mit Schemas zu befüllen. Für diesen Zweck wurden die 4 Klassen erstellt.

Stats Diese Klasse enthält das Datenbank-Modell und besteht nur aus einer Ansammlung von Eigenschaften mit entsprechenden Annotations, die man aus der Room Dokumentation entnehmen kann. Die Annotations geben die Spaltennamen, Indexe und andere Metainformationen wieder.

StatsDao Ist ein interface mit verschiedenen Methoden, die jeweils mit einem SQL-Query annotiert sind. Dabei können die Parameter der Methode im Query verwendet werden.

StatsDatabase Diese Klasse erbt von `RoomDatabase` und enthält die Initalisierung der Datenbank und eine vorgegebene abstrakte Methode die ein `StatsDao` Objekt zurückgibt.

DateConverter Da Room mit Objekten der Klasse `java.util.Date` nicht arbeiten kann, müssen sie entsprechend konvertiert werden. In diesem Fall zu `Long` als Unix Timestamp. Um den entsprechenden Konverter bekannt zu machen, wird an `StatsDatabase` eine Annotation mit der Klassenreferenz hinzugefügt.