Heuschrecken der Region Schaffhausen

Ein Heuschrecken-Quiz für Android

Ruben Schwarz

Betreut von: Raphael Riederer

13. Januar 2014

Maturaarbeit im Fach Informatik
Kantonsschule Schaffhausen

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Weshalb habe ich mich entschieden, diese Arbeit zu schreiben? Vorerst war ich mir noch nicht einmal sicher, die Arbeit im Fach Informatik zu schreiben. Wichtig war mir jedoch, dass ich durch die Maturaarbeit etwas lerne, was ich später auch abrufen und anwenden kann. Somit dachte ich bereits einen Schritt weiter und entschloss mich, eine App zu schreiben. Motiviert durch das Interesse an Computertechnik, insbesondere Softwareentwicklung, ferner auch durch das Bestreben, ein Informatikstudium an der ETH abzuschliessen, überlegte ich sodann, wie die App sein soll und welche Funktionen sie erfüllen soll.

1.2 Danksagung

Bevor ich das Konzept erläutere, möchte ich noch einigen Leuten danken, deren Unterstützung sehr hilfreich war und mich mit meiner Arbeit voranbrachte. An erster Stelle wäre meine Betreuungsperson, Herr Raphael Riederer, zu nennen, der mir bei der Ausarbeitung des Konzepts und grundlegenden Fragestellungen bezüglich der Datenbank und der App von grosser Hilfe war. Zudem will ich meinen Vater erwähnen, durch den ich begriff, dass ich mich nicht blind auf die App stürzen, sondern sie Stück um Stück mit weiteren Funktionalitäten ergänzen soll. Das führte dazu, dass ich meinen Code nicht unnötig oft umkrempeln musste. Nicht zuletzt gilt auch ein Dank an meinen Kollegen Markus Ferber. Seine Einführung ins Textverarbeitungsprogramm LaTeX gestaltete das Verfassen dieser Dokumentation wesentlich einfacher und zügiger.

1.3 Konzept

Zum einen soll die App einen gewissen Unterhaltungwert haben, zum anderen aber auch gewissermassen lehrreich sein. Eine Verbindung dieser Eigenschaften resultiert meistens in einem Lernspiel. Die wohl beliebtesten Lernspiele sind Quiz. Man begegnet ihnen im Alltag sehr oft. Sei es eine Quizshow im Fernsehen, ein Spiel auf einem Handy, oder ein Brettspiel, jeder kennt es. Somit ist klar, dass die Idee eines Quiz nicht neu war. Um es nicht zu allgemein zu halten, dachte ich an ein Tierquiz. Nach Absprache mit meiner Betreuungsperson entschied ich mich für ein Heuschrecken-Quiz, das alle Heuschrecken der Region Schaffhausen enthalten soll und aus einem Bild- und Tonquiz besteht.
Als nächstes stellte sich die Frage, auf welcher Plattform das Quiz verfügbar sein soll. Während sich in den 80er- und 90er-Jahren die Wahl lediglich auf wenige Betriebssysteme beschränkt hätte, namentlich DOS, Windows und Mac, so ist es mit der heutigen Flut an Betriebssystemen wesentlich schwieriger geworden, sich für eines zu entscheiden. Glücklicherweise setzen die wenigsten Hersteller auf proprietäre Systeme. Betrachtenswert sind vor allem Apple iOS, Android und Windows Phone. Dem App Store von Apple wird seit Jahren nachgesagt, dass er die besten Apps enthalte. Ausser einigen exklusiven Apps landen aber die meisten iOS-Apps auch im Android Market. Der Windows

Marketplace konnte sich noch nicht richtig durchsetzen und fiel daher aus meiner Liste raus. Die Entwicklung von iOS-Apps mithilfe des SDKs von Apple entpuppt sich für den Windows-User als unzumutbar, da er sich dafür einen Mac anschaffen muss.
Die Wahl fiel also auf Android. Es gibt unterschiedliche Sprachen, mit denen man Applikationen für Android entwickeln könnte, also stellte sich die Frage, welche sich am besten eignet. Die meisten Apps werden in Java geschrieben. Nativer Code (C bzw. C++) eignet sich eher für komplexere Programme. HTML 5 hingegen kommt mehr und mehr auf. Java sprach mir aber am besten zu, auch weil ich Gebrauch von der Eclipse IDE machen konnte. Diese erwies sich als sehr nützlich, unter anderem wegen dem nützlichen Tool "Logcat". Nicht zuletzt wegen der Beliebtheit von iPad & Co. kam der Gedanke hinzu, die App auch für Tablets zu schreiben, mit einer optimierten Oberfläche für eine bessere Nutzung.
Nun war seitens des Endgeräts alles bereits entschieden, doch wo sollen sich die Inhalte der App, alle Bilder und Töne, befinden? Einerseits ist es wesentlich einfacher, die Daten lokal abzurufen, andererseits erfordert es aber einigen Speicherplatz, den man durchaus besser nutzen könnte. Deshalb schien mir die Bereitstellung der Daten durch einen Webserver und eine MySQL-Datenbank am geeignetsten.

2 Theorie[1]

2.1 Android

Android ist ein Betriebssystem für mobile Plattformen wie Mobiltelefone, Smartphones, Tablets und Multimedia-Player, das seit Oktober 2008 verfügbar ist. Es handelt sich dabei um ein quelloffenes Betriebssystem (OpenSource), das von der Open Handset Alliance entwickelt wird. Die OHA ist ein Konsortium, welches sowohl Softwareunternehmen (bspw. Google, eBay) als auch Chiphersteller (bspw. Intel, Qualcomm) und Endgerätehersteller (bspw. Samsung, HTC) als Partner enthält. Obwohl Android einen Linux-Kernel besitzt, handelt es sich nicht um eine gewöhnliche Linux-Distribution wie Ubuntu. Viele Linux-Dienste sind nicht verfügbar und es ist auch nicht möglich, Linux-Applikationen darauf auszuführen. Android ist mit der Dalvik-VM (Virtual Machine) ausgerüstet, die ähnlich der Java-VM aufgebaut ist. Applikationen für Android werden hauptsächlich in der Programmiersprache Java geschrieben, wobei auch Anwendungen unterstützt werden, die in C, C++, Flash und HTML5 geschrieben wurden. Durch weitere Laufzeitumgebungen ist es ausserdem möglich, Code in Python oder Ruby auszuführen. Für die Entwicklung eigener Apps benötigt man das von Google freigegebene Android SDK sowie das Java SDK. Als IDE (Integrated Development Environment) kommt oft Eclipse zum Einsatz, das sich auch bei Java-Programmierern grosser Beliebtheit erfreut. Die fertigerstelllte App wird als Paket in einer apk-Datei gespeichert, durch die sich auf dem Endgerät die App installieren lässt. Die Installation nicht überprüfter Apps ist im Gegensatz zu vielen anderen mobilen Betriebssystemen (Apple iOS, Windows Phone) gestattet. Die Distribution der Apps findet somit nicht nur im hauseigenen Google Play Store (ehemals Android Market) statt, sondern auch in ähnlichen Stores von Drittanbietern und vereinzelt auf den Websites der Entwickler. Standardmässig im Android Betriebssystem vorinstalliert sind einige proprietäre Apps von Google, so z.B. Gmail, Google Maps, der Play Store und ein Internet Browser. Ausserdem verwenden viele Hersteller eigene Benutzeroberflächen (soganannte Launcher), die den Funktionsumfang teils erweitern. Android ist seit dem ersten Quartal 2010 Marktführer im Markt der mobilen Betriebssysteme. Laut Google werden täglich 1.5 Millionen Geräte mit dem Betriebssystem aktiviert.

2.2 Sprachen

2.2.1 Java

Java ist eine objektorientierte Programmiersprache des Unternehmens Sun Microsystems (unterdessen aufgekauft von Oracle). Sie ist 1995 entstanden und lehnt die Syntax an C++ an. Somit ist Java für Programmierer mit Erfahrung in C bzw. C++ einfach zu erlernen. Als Konkurrenzprodukt wird häufig die von Microsoft entwickelte Progammiersprache C# (gespr. C Sharp) betrachtet, die Microsoft etwa zeitgleich mit der

[1] Die einzelnen Unterkapitel wurden teils mit Informationen aus der Wikipedia zusammengestellt. Alle dazu verwendeten Links sind in Kapitel 6: Quellenverzeichnis aufgeführt.

.NET-Plattform herausgab. Java wird häufig mit JavaScript verwechselt, das aber keine Programmiersprache, sondern eine Skriptsprache ist und vor allem auf HTML-Webseiten verbreitet ist.
Der Vorteil von Java ist unter anderem, dass in Java geschriebene Programme problemlos auf praktisch jedem System verwendet werden können. Zwingend, um ein solches Programm ausführen zu können, ist das Java Runtime Environment (JRE), eine Laufzeitumgebung, die man auf nahezu jedem Rechner antrifft. Das Ausführen eines Programms kann man sich folgendermassen vorstellen. Der Quellcode wird vom Java-Compiler in Java-Bytecode kompiliert, ein Bytecode, den der Rechner nicht direkt versteht, sondern von einer virtuellen Maschine, der Java-VM, interpretiert wird. Jetzt wird auch klar, weshalb Java grundsätzlich plattformunabhängig ist. Es wird lediglich das JRE benötigt, um eine Java-Anwendung zu verwenden. Zur Entwicklung benötigt man zusätzlich das Java Development Kit (JDK), das einige Entwicklungswerkzeuge enthält.

```java
//helloworld.java
public class helloworld {
        public void HelloWorld() {
                System.out.println("Hello World!");
        }
}
```

Listing 1: HelloWorld-Beispiel in Java geschrieben

HelloWorld-Beispiele werden nahezu in jedem Lernbuch zur Programmierung als erstes Beispiel verwendet. Sie sind sehr simpel aufgebaut und zeigen lediglich den Text “Hello World!” an. Die Anzeige kann auf unterschiedliche Art geschehen, bspw. in einem Textfeld oder als Konsolenausgabe. In diesem Beispiel geschieht letzteres. Sie eignen sich mitunter auch, um die Syntax einer Programmiersprache abstrakt aufzuzeigen.

2.2.2 XML

XML, eigentlich Extensible Markup Language, ist, wie der Name schon verrät, eine Auszeichnungssprache. XML kann zur Darstellung von Daten verwendet werden, ähnlich wie JSON, und kann somit auch auf Servern eingesetzt werden. Im Vergleich verbraucht XML aber mehr Speicher als JSON, weshalb letzteres bevorzugt zum Einsatz kommt. Ausserdem wird XML auch zur Erstellung grafischer Benutzeroberflächen (GUI) verwendet. So nutzt bspw. Android Programmcode in Java und eine GUI, die in XML geschrieben wurde. Selbstverständlich könnte man selbst die Benutzeroberfläche in Java schreiben. Allerdings ist der XML-Code wesentlich kompakter und einfacher.

```
<TextView
        android:id="@+id/txtQuestion"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:gravity="center"
        android:text="Um welche Heuschrecke handelt es sich?" />
```

Listing 2: XML-Code für eine TextView in Android

Die TextView im Listing 2 benötigt eine ID, um vom Code aufgerufen und verändert zu werden. Die restlichen Eigenschaften, die in diesem Beispiel vorkommen, bestimmen das Aussehen des Textfeldes. Um es ein wenig zu präzisieren, versteht man darunter Breite und Höhe des Feldes, sowie den Text und dessen Ausrichtung. Was die Dimensionen des Feldes betrifft, so sind zwei Werte sehr wichtig. Der eine ist "match_parent", der dem Feld die gesamte Breite, bzw. Höhe des Layouts zur Verfügung stellt. Der andere Wert nennt sich "wrap_content" und lässt dem Feld nur so viel Platz, wie benötigt wird.

2.2.3 PHP & JSON

PHP ist eine Skriptsprache, die hauptsächlich auf Websites Anwendung findet. Die Abkürzung ist ein rekursives Akronym (PHP: Hypertext Preprocessor). Aufgrund der Datenbankunterstützung wird sie sehr oft auf Webservern verwendet. Etwa 81% aller Websites verwenden PHP auf ihrem Server, was es zur meist verwendeten Sprache zur Erstellung von Websites macht. Die Funktion von PHP ist dabei folgende: Mit dem Browser wird ein PHP-Skript auf einem Server abgefragt. Das Skript arbeitet dabei auf dem Server. Es lädt die Daten vom Server und schickt sie an den Browser zurück. Oftmals wird PHP zusammen mit HTML zur Anzeige der Daten verwendet.
JSON (JavaScript Object Notation) ist ein verbreitetes Datenformat, das hauptsächlich zur Datenübertragung zwischen Server und Client dient. Als JSON-Paket können sowohl Objekte, als auch Arrays zusammengesetzt werden. Aufgrund vielseitiger Unterstützung von Programmiersprachen, kann JSON heutzutage auf fast allen Plattformen verwendet werden.

```
<?php
        echo 'Hallo Welt!';     //Beispielcode
?>
```

Listing 3: HelloWorld-Beispiel in PHP geschrieben

Beispiel einer JSON-Ausgabe:

```
{
"Name": "Mustermann",
"Vorname": "Max",
"Alter": 42,
"Beruf": "Projektleiter",
"Kinder": [ "Peter", "Fritz" ],
}
```

2.2.4 MySQL

MySQL ist eine Datenbankverwaltungssoftware, die von Oracle herausgegeben wird. Ursprünglich von MySQL AB ab 1994 entwickelt, wurde die Software 2008 von Sun Microsystems übernommen. MySQL ist das meistverwendete Open-Source-Datenbankverwaltungssystem der Welt. Es wird bspw. von Google, Facebook und Twitter verwendet. Um auf MySQL-Datenbanken zugreifen zu können, werden oft PHP-Skripte (Kapitel 2.2.3) eingesetzt. Abfragen und Einträge werden durch Querys ausgeführt, die eine bestimmte Syntax besitzen. So kann z.B. mit "SELECT" eine Spalte abgerufen werden, während mit "INSERT" eine neue Zeile erstellt wird. MySQL ist streng genommen ein Dialekt und deshalb sehr ähnlich wie SQL.

2.2.5 phpMyAdmin

Bei phpMyAdmin handelt es sich um eine grafische Benutzeroberfläche zur Verwaltung von MySQL-Datenbanken, das in PHP geschrieben wurde. Aufgestartet wird phpMyAdmin über einen Webbrowser. Vorteile sind eine grafische Darstellung der Datentabellen, sowie eine vereinfachte Erstellung und Bearbeitung der Tabellen ohne SQL-Kenntnisse. Da ich mich mit SQL kaum auskannte, kam mir diese Oberfläche sehr gelegen.

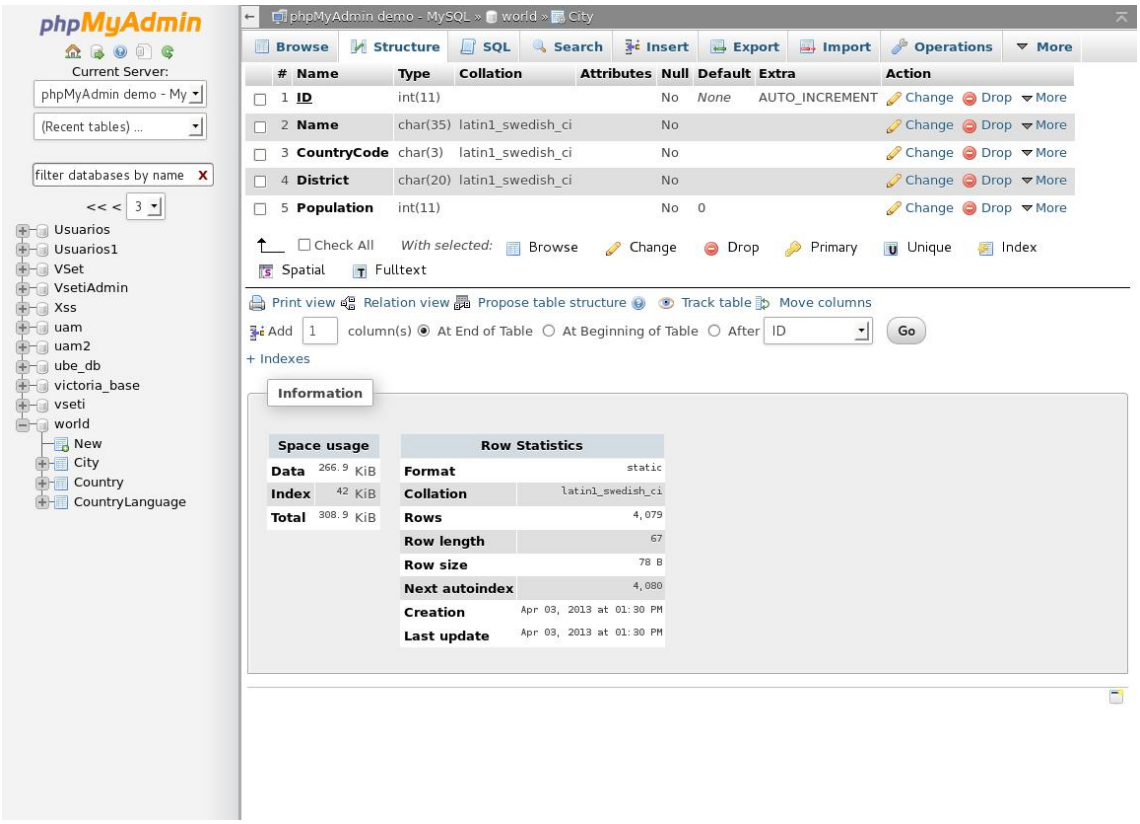

Abbildung 1: Die phpMyAdmin-Oberfläche

2.3 Heuschrecken

Folgende Heuschrecken, sind im Quiz enthalten:

- Blauflügelige Ödlandschrecke
- Buntbäuchiger Grashüpfer
- Feldgrille
- Gemeine Eichenschrecke
- Gemeine Sichelschrecke
- Gemeine Strauchschrecke
- Gemeiner Grashüpfer
- Grosse Goldschrecke
- Grünes Heupferd
- Heidegrashüpfer
- Kleine Goldschrecke
- Langflüglige Schwertschrecke
- Maulwurfsgrille
- Nachtigall-Grashüpfer
- Punktierte Zartschrecke
- Roesels Beissschrecke
- Rote Keulenschrecke
- Rotflüglige Schnarrschrecke
- Sumpfschrecke
- Waldgrille
- Wanstschrecke
- Warzenbeisser
- Weinhähnchen
- Weissrandiger Grashüpfer
- Westliche Beissschrecke
- Wiesengrashüpfer
- Zweifarbige Beissschrecke

Die aufgelisteten Heuschrecken sind in einem Häufigkeitsdiagramm des Blatts "Heuschrecken im Kanton Schaffhausen" aufgeführt. Nicht alle im Kanton vorkommenden Heuschrecken sind im Quiz enthalten. Grund dafür sind einheitliche Quellen der Bilder und Töne (siehe dazu die Bibliografie in Kapitel 6). Die Infos wurden ausnahmslos aus den jeweiligen Wikipedia-Artikeln der Heuschrecken entnommen und sind nicht in der Bibliografie aufgeführt.

3 Entwicklung

3.1 Aneignung der Sprachen

Aller Anfang ist schwer...
Da ich Java noch nicht kannte, musste ich mir diese Programmiersprache zunächst erlernen. Denn ohne Beherrschen der Sprache ist kein gezieltes Vorgehen möglich. Auch mit Datenbanken war ich noch nicht bewandert, obschon dies eine kleinere Hürde darstellte. Zwar hatte ich in der 3. Klasse den Freifachkurs Informatik besucht, von dem ich einige grundlegende Strukturen von Programmiersprachen mitnehmen konnte, bspw. Schleifen und if-else-Statements, doch erwies sich Java als ziemlich komplexer als Python. Die Verwendung eines Buches schien mir deshalb angebracht und notwendig. Entschieden hatte ich mich für "Einführung in die Android-Entwicklung" des Verlags O'Reilly. Es beschränkt sich zwar auf ein Anwendungsbeispiel, bringt aber viele grundlegende und fortgeschrittene Bausteine der Android-Architektur näher. Die übrigen "Sprachbarrieren" versuchte ich bestmöglich mit der Java-Dokumentation von Oracle zu überwinden. Zur Serverkommunikation war es nötig, einige Funktionen von SQL und PHP zu kennen. Diese liessen sich recht zügig finden. Zu Beginn hatte ich vor, ein Python-Skript zur Kommunikation zu schreiben, da mir Python recht vertraut war und ich es als angenehme Sprache empfand, liess aber schnell davon ab, weil es wesentlich umständlicher ist, als das Skript in PHP zu verfassen. Zum einen ist die Verwendung von PHP zur Serverkommunikation besser dokumentiert (es gibt nämlich Hunderte von Beispielen), zum anderen ist weniger Code nötig als bei Python. Also entschloss ich mich, diese mir neue Skriptsprache zu erlernen und ein entsprechendes Skript zu schreiben, welches mir erlaubt, die Daten in Java abrufen zu können.

3.2 Datenbank

Die Planung und Erstellung der Datenbank lief über phpMyAdmin ab, welches auf dem Server bereits vorinstalliert war. Wichtig ist es, zu wissen, welche Daten überhaupt auf der Datentabelle liegen sollen. Für meine App wurden fünf Spalten benötigt. Die erste Spalte heisst 'id' und enthält die jeweilige ID der Heuschrecke. Die Verteilung der ID's beginnt bei 1 und erhöht sich automatisch mit jedem neuen Eintrag. Die zweite Spalte enthält den Namen der Heuschrecke. Aus der dritten Spalte sind die jeweiligen Informationen, d.h. ein kurzer Steckbrief der Heuschrecke, ersichtlich. Die vierte und fünfte Spalte ist für Bilder bzw. Geräusche besetzt. Dateien könnten eigentlich als Binärdaten gespeichert werden, doch anscheinend resultiert die Verwendung von Binärdaten in einer längeren Ladezeit. Deshalb zog ich es vor, die Bild- und Tondateien auf den Server hochzuladen, von wo aus ich sie über einen Link mit der Datenbank verknüpfen konnte. Somit wurde die Datenbankgrösse auch wesentlich kleiner gehalten.

Name	Datentyp
id	INT
name	TEXT
info	LONGTEXT
image	TEXT
sound	TEXT

Tabelle 1: Die Spalten der Datentabelle

Tabelle 1 zeigt den Datentyp der einzelnen Spalten. Diese sind von grosser Relevanz, da sie zum einen beschreiben, welcher Inhalt für die Spalte zulässig ist, und andererseits helfen, den Speicherverbrauch zu optimieren. Die Spalte 'id' enthält nur kleine natürliche Zahlen, weshalb der Datentyp Integer am besten geeignet ist. In der Spalte 'name' befinden sich nur die Namen der Heuschrecken, die nicht aüsserst viele Zeichen enthalten. So auch die Spalten 'image' und 'sound', die nur Links zu den auf dem Server gespeicherten Dateien enthalten. Deshalb eignet sich für diese Spalten der Datentyp Text. Bei den Infos hingegen kann sich relativ viel Text aufhalten, weshalb Longtext wohl notwendig ist. Das Einfügen der Werte in die Spalten erwies sich als ein recht einfacher Vorgang, bei dem lediglich die Codierung des Texts zu einigen Problemen führte (siehe dazu Listing 6)

3.3 Kommunikation zwischen Server und App

Zum Austausch von Daten zwischen Server und App sind Skripte nötig. Für diese App beschränkte ich mich auf ein PHP-Skript, dass alle Daten aus der Datenbank herauslesen und sie als JSON-Paket darstellen soll. Dies bedeutet, dass dem Nutzer nur der Lesezugriff, und nicht auch noch der Schreibzugriff, gewährt ist. Somit ist es nicht möglich, die Daten auf dem Server vom Android-Gerät aus zu manipulieren.

3.4 Entwicklung der App

Nun kommt der Hauptteil ins Spiel, die Entwicklung der Android-App mit Java. Jedes Fenster wird einer sogenannten Activity zugeteilt, was eigentlich nichts weiteres ist, als eine Java-Datei mit einer Superklasse. Activities müssen nicht zwingend mit einem Layout-File verknüpft sein, umgekehrt aber schon, da das Layout-File ansonsten gar nie aufgerufen werden könnte. Im sogenannten Manifest der App sind alle Activities aufgeführt. Man muss jede neu erstellte Activity manuell dort eintragen. Ausserdem ist im Manifest auch festgesetzt, welche Activity beim Aufstarten der App geladen werden soll und welche Rechte die App besitzt. Die üblichen Rechte sind bspw. auf externen Speicher zugreifen oder die Internetverbindung verwenden. Das Heuschrecken-Quiz benötigt eine Internetverbindung und verwendet fünf Activities. Des weiteren wird auch bestimmt, welche Mindestversion von Android vorausgesetzt wird, um die App ausführen zu können. Viele Apps, die heutzutags im Play Store erscheinen, benötigen mindestens Android 2.2, da es nahezu die gesamte Funktionalität mitbringt. Deshalb habe ich mich

auch für diese OS-Version entschieden. Da die Layouts der Tablet-Version nicht viel anders aussehen als die der Smartphone-Version, werde ich mich bezüglich XML-Code auf letztere beschränken.

3.4.1 Hauptmenü

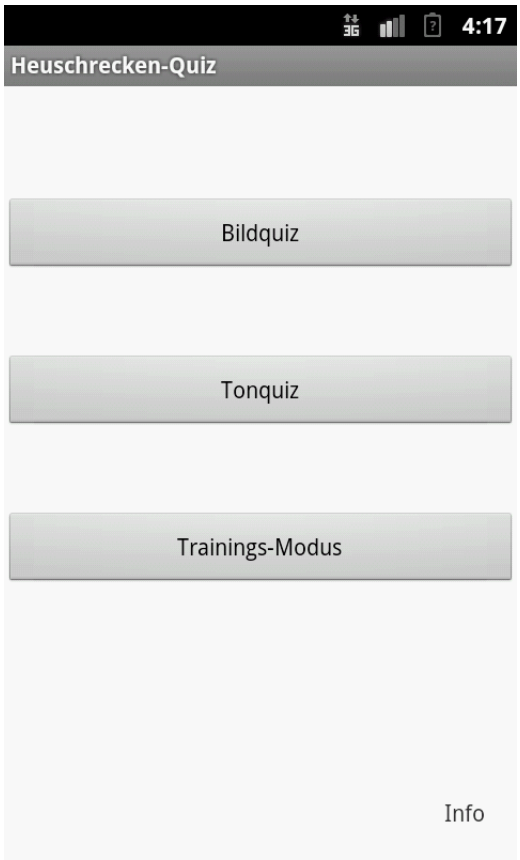

Abbildung 2: activity_main.xml : Das Hauptmenü

Wie soll nun also das erste Fenster aussehen, wenn man das Spiel startet? Gewöhnlicherweise erscheint ein Hauptmenü. Dieses enthält in diesem Quiz drei Buttons, die vertikal bzw. in einem Dreieck (in der Tablet-Version) angeordnet sind. Durch Tippen auf einen dieser Buttons gelangt man in den Spielmodus, der auf dem Button beschriftet ist. Zudem gibt es auch noch ein Textfeld namens Info, das auf Klick die InfoActivity aufruft und Infos zur App anzeigt. Man könnte das Hauptmenü selbstverständlich auch anders gestalten, bspw. mit einer Tabansicht, allerdings verliert der Spieler dabei schneller die Übersicht. Buttons sind sehr einfach zu handhaben und eindeutig.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center" >

    <Button
        android:id="@+id/btnImgQuiz"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:text="@string/app_imgquiz" />

    <Button
        android:id="@+id/btnSndQuiz"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:layout_marginBottom="50dp"
        android:text="@string/app_sndquiz" />

    <Button
        android:id="@+id/btnTraining"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_training" />

    <TextView
        android:id="@+id/txtInfo"
        android:layout_gravity="bottom"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right"
        android:layout_marginTop="130dp"
        android:layout_marginRight="20dp"
        android:text="@string/app_info" />

</LinearLayout>
```

Listing 4: XML-Code des Hauptmenüs

Wie man bemerkt, besteht das Layout aus einem einfachen LinearLayout, welches die Steuerelemente vertikal untereinender anordnet. Darin befinden sich vor allem Buttons. Jeder dieser Buttons bekommt seine eigene ID. Seine Breite, sowie seine Höhe und sein Text werden ebenfalls deklariert. Die Text-Eigenschaft kann als Wert einen String enthalten, der in der strings.xml-Datei definiert wird. Der Vorteil ist unter anderem sichtbar, wenn man die App in unterschiedlichen Sprachen verfügbar machen will. Dazu verwendet Android die Strings-Datei in unterschiedlichen Ordnern, die auf eine Sprache verweisen. Je nach eingestellter Sprache auf dem Gerät, ändert sich auch die Anzeigesprache in der App. Es hätte somit den Vorteil, dass unkompliziert innerhalb der selben App eine

andere Anzeigesprache eingestellt werden kann. Ungemein hätte es auch zur Folge, dass sich der Interessentenkreis auf nicht-germanophone Gebiete erweitern liesse. Allerdings wäre es eine zusätzliche Hürde gewesen und deshalb verwarf ich diesen Gedanken.

3.4.2 Trainingsmodus

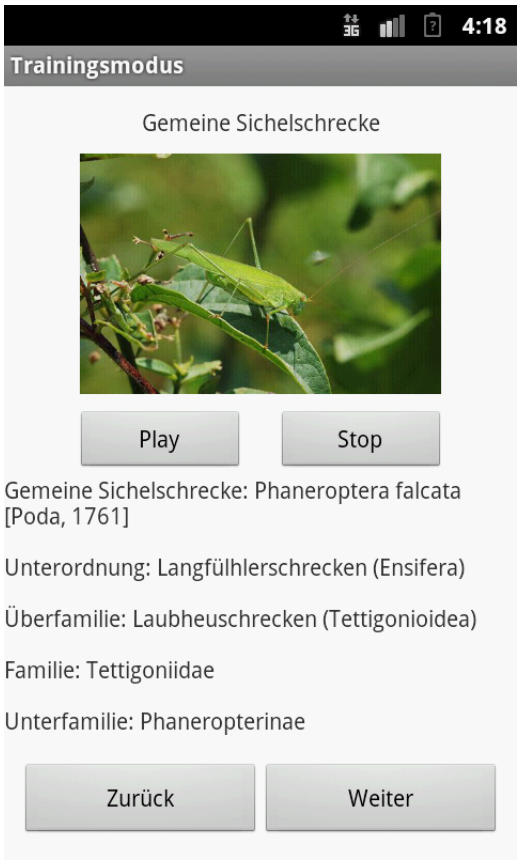

Abbildung 3: training.xml : Der Trainingsmodus

Der Trainingsmodus dient dem Einstieg ins Spiel. Der Spieler kann darin Informationen über die Heuschrecken finden, sowie auch ein Bild der Heuschrecke betrachten und deren Gesang anhören.
Betrachten wir zuerst das Layout. Man sieht zu oberst den Namen der Heuschrecke. Der Reihe nach runter steht als nächstes das Bild der Heuschrecke, gefolgt von zwei Buttons. Diese beiden Buttons dienen zur Wiedergabe des Tons, was aufgrund ihrer Beschriftungen sofort klar wird. Als nächstes sieht man die Infos zur Heuschrecke. Nicht im Bild sichtbar ist die ScrollView, die das Textfeld umgibt. Zuletzt befinden sich auf dem Bildschirm noch zwei weitere Buttons, die zur Navigation dienen. Man gelangt durch einen Klick auf sie zur nächsten bzw. vorherigen Heuschrecke. Dabei werden dann alle Felder wieder aktualisiert.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center" >

    <TextView
        android:id="@+id/txtName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center|top"
        android:layout_marginBottom="10dp" />

    <ImageView
        android:id="@+id/imgAnim"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:scaleType="fitCenter"
        android:layout_marginBottom="10dp"
        android:contentDescription="Heuschrecke" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:orientation="horizontal"
        android:gravity="center" >

        <Button
            android:id="@+id/btnPlay"
            android:layout_width="105dp"
            android:layout_height="match_parent"
            android:layout_marginRight="10dp"
            android:text="Play" />

        <Button
            android:id="@+id/btnStop"
            android:layout_width="105dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="10dp"
            android:text="Stop" />

    </LinearLayout>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="180dp" >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >
```

```
53
54              <TextView
55                  android:id="@+id/txtInfo"
56                  android:layout_width="match_parent"
57                  android:layout_height="match_parent" />
58
59          </LinearLayout>
60
61      </ScrollView>
62
63      <LinearLayout
64          android:layout_width="match_parent"
65          android:layout_height="wrap_content"
66          android:orientation="horizontal"
67          android:gravity="center" >
68
69          <Button
70              android:id="@+id/btnPrev"
71              android:layout_width="150dp"
72              android:layout_height="match_parent"
73              android:text="Zurück" />
74
75          <Button
76              android:id="@+id/btnNext"
77              android:layout_width="150dp"
78              android:layout_height="match_parent"
79              android:text="Weiter" />
80
81      </LinearLayout>
82
83  </LinearLayout>
```

Listing 5: XML-Code des Trainingsmodus

Wie auch im Hauptmenü verwendete ich im Listing 5 ein LinearLayout als oberste Layouteinheit. Darunter ist das Textfeld, welches den Namen zentriert anzeigt. Die Text-Eigenschaft ist noch nicht zugewiesen. Die nächsten Zeilen dienen der Beschreibung des ImageView. Hierbei ist offensichtlich, dass es sich um die Bildanzeige handelt. Da ich das Bild, egal wie hoch die Auflösung ist, stets in richtigem Verhältnis und getreckt anzeigen lassen will, wies ich der scaleType-Eigenschaft den Wert *"fitCenter"* zu. Auch hier wird zu Beginn kein Bild angezeigt. Erst muss der Java-Code gelesen werden, um das Bild anzuzeigen. Dies läuft aber so schnell ab, dass der Nutzer das gar nicht mitkriegt. Da ich die beiden Buttons zur Wiedergabe der Geräusche nebeneinander anzeigen wollte, und das LinearLayout die Elemente vertikal anordnet, schuf ich ein horizontales LinearLayout, welches lediglich die Buttons enthält. Es kann sein, dass man es auch anders lösen könnte, mir schien dies aber die unkomplizierteste Variante zu sein, und schliesslich wollte ich nicht zu viel Zeit mit der Gestaltung des Layouts verbringen. Die Infos werden in einem Textfeld angezeigt, welches von einem LinearLayout umschlossen ist, das wiederum von einer ScrollView eingehüllt ist. Die ScrollView ist sehr einfach zu erstellen. Man muss nur ihre Höhe und Breite definieren und fügt Steuerelemente

hinein. Zuletzt werden die übrigen beiden Buttons wie zuvor die Wiedergabe-Buttons in ein LinearLayout gepackt, um sie horizontal darzustellen.
Zugegeben, dieses Fenster ist ziemlich vollgefüllt. Es gibt aber einen Eindruck darüber, was beim XML-Code alles beachtet werden muss, um das Fenster auch so anzuzeigen, wie man es möchte.

Betrachten wir nun den Java-Code. In dieser Activity scheint viel zu geschehen, da all diese Daten geladen und verarbeitet werden müssen. Ich werde abschnittsweise die wichtigsten Auszüge des Codes erklären.

```
1  private class JsonReader extends AsyncTask<String, Void, String> {
2
3      @Override
4      protected String doInBackground(String...params) {
5          HttpClient httpClient = new DefaultHttpClient();
6          HttpPost httpPost = new HttpPost(params[0]);
7          try {
8              HttpResponse httpResponse = httpClient.execute(httpPost);
9              jsonResult = inputStreamToString(
10                 httpResponse.getEntity().getContent()).toString();
11         } catch(ClientProtocolException e) {
12             e.printStackTrace();
13         } catch(IOException e) {
14             e.printStackTrace();
15         }
16         return null;
17     }
18
19     private StringBuilder inputStreamToString(InputStream is) {
20         String line = "";
21         StringBuilder sb = new StringBuilder();
22
23         try {
24             BufferedReader reader = new BufferedReader(new
25                 InputStreamReader(is, "UTF-8"));
26             while ((line = reader.readLine()) != null) {
27                 sb.append(line);
28             }
29         } catch(IOException e) {
30             Toast.makeText(getApplicationContext(),
31                 "Fehler..." + e.toString(), Toast.LENGTH_LONG).show();
32         }
33         return sb;
34     }
```

Listing 6: JSON-Parser

Dieser Teil des Codes umfasst einen sogenannten JSON-Parser. Er besteht aus zwei Teilen. Im ersten Teil wird über das PHP-Skript ein Request an den Server geschickt. Die *HttpPost*-Methode erledigt dabei die Arbeit. Die Ausgabe des PHP-Skripts ist in

JSON codiert, was Java versteht. Der Stream wird geöffnet und in einen String umgewandelt, der *jsonResult* heisst. Im zweiten Teil wird mithilfe eines StringBuilders Zeile um Zeile gespeichert. Dabei ist auch ein BufferedReader nötig, der die Zeile liest und sie einem String zuweist, der dem StringBuilder solange hinzugefügt wird, als der String noch Zeichen enthält. Der InputStreamReader im BufferedReader enthält übrigens einen kleinen, aber dennoch wichtigen Ausdruck, "UTF-8", der dazu führt, dass Umlaute richtig dargestellt werden.
Dieser Teil der App mag durchaus der schwierigste sein, da vielleicht nicht sofort einleuchtet, was hier genau geschieht. Es ist ein wenig umständlich, aber nötig, um die Daten der Datenbank überhaupt auf dem Gerät anzeigen zu lassen.

```
1  private void setValues() {
2      try {
3          JSONObject jsonResponse = new JSONObject(jsonResult);
4          JSONArray jsonMainNode = jsonResponse.optJSONArray(
5              "testversuch");
6
7          size = jsonMainNode.length() - 1;
8
9          if(i <= (size - 1) && i >= 0) {
10             JSONObject jsonChildNode = jsonMainNode.getJSONObject(i);
11             name = jsonChildNode.getString("name");
12             info = jsonChildNode.getString("info");
13             img = jsonChildNode.getString("image");
14             snd = jsonChildNode.getString("sound");
15
16             nametxt.setText(name);
17             infotxt.setText(info);
```

Listing 7: Werte des JSON-Arrays werden eingesetzt

Der Code aus Listing 7 setzt hauptsächlich die Werte aus dem verarbeiteten JSON-Paket in die entsprechenden Platzhalter ein. Da aber nicht nur Text eingefügt wird, sondern auch Bilder und Tondateien, müssen die Steuerelemente noch programmiert werden. Zum Glück besitzt Android genügend Bibliotheken, um den Code kompakt und verständlich zu halten.
Zuerst wird der JSON-Array geladen, der die gesamte Datentabelle der Datenbank enthält. Der Integer *size* ist genauso gross wie die ID der letzten Heuschrecke im Array. Weil aber die Funktion *jsonMainNode.length()* angibt, wie lang der Array ist, und die Vergabe von IDs bei 0 beginnt, muss man 1 abziehen. In der nächsten Zeile kommt ein Zähler i zum Einsatz, der auch in den weiteren Abschnitten vorkommen wird. i repräsentiert eine Zeile der Datentabelle, bzw. ein bestimmtes JSON-Object des Arrays. Demnach ist $i < 0$ unzulässig, da die Zählung bei 0 beginnt. Der Zähler darf nur einen ganzzahligen Wert zwischen 0 und *size* annehmen. Ist diese Bedingung erfüllt, so werden die einzelnen Strings der Zeile in neue Strings gespeichert. Daraufhin werden den beiden Textfeldern die Strings *name* und *info* zugewiesen.

```
URL connURLImg = new URL(img);
HttpURLConnection connImg =
    (HttpURLConnection) connURLImg.openConnection();
connImg.setDoInput(true);
connImg.connect();

InputStream inputImage = connImg.getInputStream();
imgview.setImageBitmap(BitmapFactory.decodeStream(inputImage));
```

Listing 8: Vorbereitung des Bildes

Im Listing 8 wird die Verbindung mittels HTTP und dem Bildink hergestellt. Das Bild wird in einen InputStream geladen, der dann decodiert und in die ImageView eingefügt wird. Das Bild ist nun vollständig heruntergeladen und wird angezeigt.

```
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(snd);
```

Listing 9: Vorbereitung des MediaPlayers

Der MediaPlayer wird vorbereitet. Die erste Zeile ist nötig, um einen Stream zu laden. Die Quelle, d.h. der Link des Gesangs, wird gesetzt.

```
} else {
    onDestroy();
}
```

Listing 10: Else-Argument

Wenn die If-Bedingung nicht erfüllt ist, kehrt man zum Hauptmenü zurück. Das tritt ein wenn keine vorherigen oder nächsten Heuschrecken mehr auf der Datenbank existieren.

```
@Override
    protected void onPostExecute(String result) {
    setValues();
```

Listing 11: Prozedur nach Ausführen des JSON-Parsers

Die Methode *onPostExecute* ist Bestandteil des *AsyncTask*, der dazu dient, kurze Hintergrundprozesse auszuführen und deren Resultate in den Vordergrund zu stellen. *onPostExecute* gibt an, was nach *doInBackground* ausgeführt wird. Im Fall dieser App werden die Werte eingesetzt.

```
nextbtn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        mediaPlayer.stop();
        mediaPlayer.reset();

        i++;

        setValues();
    }
});

prevbtn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        mediaPlayer.stop();
        mediaPlayer.reset();

        i--;

        setValues();
    }
});

playbtn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        if(mediaPlayer.isPlaying()) {

        } else {
            mediaPlayer.prepareAsync();
            mediaPlayer.setOnPreparedListener(new
                MediaPlayer.OnPreparedListener() {

                @Override
                public void onPrepared(MediaPlayer mediaPlayer) {
                    mediaPlayer.start();
                }
            });
        }
    }
});

stopbtn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        if(mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
```

```
53          } else {
54
55          }
56      }
57  });
```

Listing 12: Funktionen der einzelnen Buttons

Dieser Teil des Codes dient der onClick-Methoden einiger Buttons. Als *nextbtn* wird der Weiter-Button bezeichnet, als *prevbtn* der Zurück-Button, als *playbtn* der Play-Button und als *stopbtn* der Stop-Button. Ein Klick auf *nextbtn* stoppt die Wiedergabe und setzt den MediaPlayer zurück. Der Zähler i wird um 1 erhöht, um die nächste Heuschrecke zu laden. Der Zurück-Button besitzt die nahezu gleiche Methode, i wird dabei aber um 1 vermindert, um die vorherige Heuschrecke anzuzeigen.
Der Play-Button bereitet den MediaPlayer vor und startet die Wiedergabe während der Stop-Button die Wiedergabe beendet.

3.4.3 Bildquiz

Abbildung 4: activity_img_quiz.xml : Das Bildquiz

Im Bildquiz muss der Spieler, wie der Name bereits andeuten lässt, den Namen der Heuschrecke anhand des Bildes herausfinden. Als Stütze dient ihm eine Liste aller in der Datenbank enthaltener Heuschreckennamen, aus der er den seiner Meinung nach zutreffenden Namen auswählt. Gegenüber der Eingabe des Namens gibt es einen klaren Vorteil: der Spieler muss sich nicht mit der Rechtschreibung beschäftigen.
Das Interface sieht dem des Trainingsmodus ziemlich ähnlich. Allerdings wird das Infofeld durch eine scrollbare Liste ersetzt und die beiden Buttons für die Tonwiedergabe, sowie die Buttons zum Wechseln der Heuschrecke, werden gelöscht. Daher werde ich mich jetzt auf den Code beschränken, der die Liste zeichnet.

```
<ListView
    android:id="@+id/answerList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Listing 13: XML-Code der Liste

Wie im Listing 13 ersichtlich wird, benötigt die Liste nur zwei Eigenschaften, nämlich eine Höhe und eine Breite. Die Liste ist zu Beginn leer, weil sie durch die *ImgQuizActivity* gefüllt wird.

Der Java-Code dieser Activity wird hier schrittweise erläutert.

```
@Override
protected void onPostExecute(String result) {

    setValuesImgQuiz();

    listview.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> arg0, View view,
            int position, long id) {
            answer = listview.getItemAtPosition(position).toString();

            if(answer.contentEquals(name)) {
                points = points + 20;
            } else {
                points = points - 10;
                Toast.makeText(getApplicationContext(),
                    "Falsche Antwort", Toast.LENGTH_SHORT).show();
            }
            count++;
            setValuesImgQuiz();
        }
    });
}
```

Listing 14: Klick-Aktion eines Listenelements

Der Code des Listings 14 wird ausgeführt, nachdem der JSON-Parser seine Aufgabe erledigt hat. Zuerst werden die Werte eingesetzt, ähnlich wie im Trainingsmodus. Dazu genauer noch im nächsten Codeausschnitt. Nach dem Einsetzen der Werte wird definiert, was beim Tippen auf ein Element der Liste geschieht. Dabei wird das ausgewählte Listenelement in einen String gespeichert und daraufhin mit dem richtigen Namen der Heuschrecke verglichen. Stimmen die beiden Strings überein, so wird der Integer *points* um 20 erhöht, stimmen sie nicht überein, wird *points* um 10 kleiner und es wird eine Meldung mit dem Text “Falsche Antwort” als Toast eingeblendet. In beiden Fällen wird der Zähler *count* um 1 erhöht und es werden neue Werte eingesetzt. *count* ist zu Beginn des Spiels 1 und dient dazu, dass sich das Spiel merkt, wieviele Heuschrecken bereits vorgekommen sind.

```
private void setValuesImgQuiz() {
    try {
        jsonResponse = new JSONObject(jsonResult);
        jsonMainNode = jsonResponse.getJSONArray("testversuch");

        size = jsonMainNode.length();
        Random rand = new Random();
        int i = rand.nextInt(size);

        if(count <= size) {
            if(list.contains(i)) {
                setValuesImgQuiz();
            } else {
                jsonChildNode = jsonMainNode.getJSONObject(i);
                list.add(i);

                name = jsonChildNode.getString("name");
                img = jsonChildNode.getString("image");

                URL connURL = new URL(img);
                HttpURLConnection connImg =
                    (HttpURLConnection) connURL.openConnection();
                connImg.setDoInput(true);
                connImg.connect();

                InputStream inputImage = connImg.getInputStream();
                imgview.setImageBitmap(BitmapFactory.decodeStream(
                    inputImage));

                ListDrawer();
            }
        } else {
            showResult();
        }
    }
}
```

Listing 15: Einsetzen der Werte

Wie im Trainingsmodus wird im Listing 15 zuerst der JSONArray geladen. Die Zufallszahl wird mit der Methode *Random* erstellt. Mithilfe von *nextInt* lässt sich die obere Grenze dieser Zufallszahl festsetzen, in diesem Fall *size*. Solange der Zähler kleiner oder gleich gross ist wie *size*, also solange noch nicht alle Heuschrecken vorgekommen sind, geht das Spiel weiter. Das Spiel überprüft dann, ob die ID schon einmal aufgerufen wurde. Dies geschieht mithilfe einer Liste, in welche die ID der angezeigten Heuschrecke gespeichert wird. Kommt diese ID bereits vor, wird mit einer neuen Zufallszahl gerechnet. Dieser Loop dauert aber nicht allzulange, da der Prozessor sehr schnell rechnet und der Spieler folglich nichts davon mitbekommt. Daraufhin werden die Daten wie im Trainingsmodus eingesetzt.

```
private void ListDrawer() {
    answers = new ArrayList<String>();

    for(int i = 0; i < size; i++) {
        try {
            jsonResponse = new JSONObject(jsonResult);
            jsonMainNode = jsonResponse.getJSONArray("testversuch");

            size = jsonMainNode.length();
            jsonChildNode = jsonMainNode.getJSONObject(i);
            name = jsonChildNode.getString("name");
            answers.add(name);

            initializeList();
        } catch(JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Listing 16: Liste wird mit Namen gefüllt

Nun muss die Liste der möglichen Antworten im Listing 16 noch gefüllt werden. Dafür habe ich einen *for*-Loop verwendet, der den Zähler i schrittweise erhöht, bis er die letzte Zeile gelesen hat. Aus jeder Zeile wird nur der Name der Heuschrecke extrahiert. In der darauffolgenden Prozedur (Listing 17) wird die Liste willkürlich sortiert und durch einen Adapter mit der ListView verknüpft. Die zufällige Anordnung der Listenelemente bringt eine zusätzliche Schwierigkeit ins Spiel. Ohne diese Herausforderung könnte sich der Spieler relativ schnell die Position der einzelnen Antworten merken. Hiermit gibt es eine zusätzliche Hürde.

```
private void initializeList() {
    Collections.shuffle(answers);
    adapt = new ArrayAdapter<String>(ImgQuizActivity.this,
        android.R.layout.simple_list_item_1, answers);
    listview.setAdapter(adapt);
}
```

Listing 17: Liste wird willkürlich sortiert

3.4.4 Tonquiz

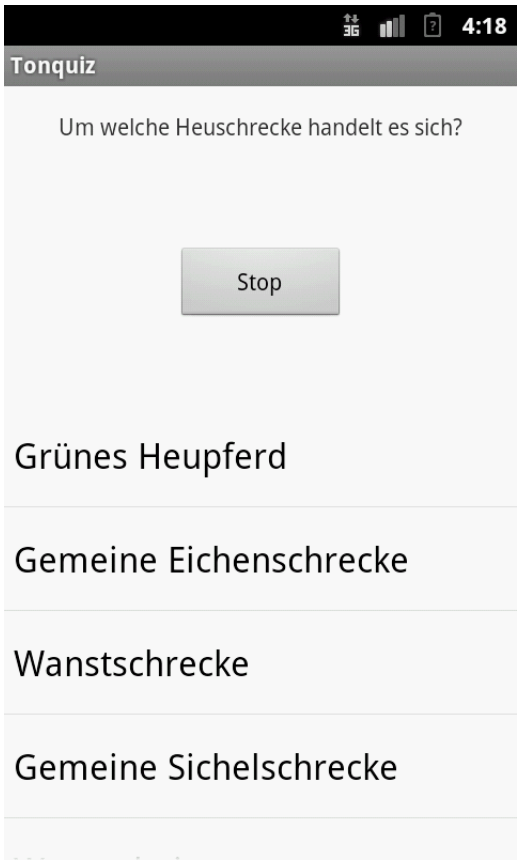

Abbildung 5: activity_snd_quiz.xml : Das Tonquiz

Das Tonquiz ähnelt im Aufbau stark dem Bildquiz. Es erscheint wieder eine Liste mit allen Heuschreckennamen, aus der man den richtigen herausfinden muss. Allerdings wird anstelle des Bildes ein Button zur Wiedergabe des Tons angezeigt. Das Layout ist daher nicht gerade spannend und ich werde den XML-Code auch nicht weiter erläutern.
Weitaus spannender sind die Prozesse, die der Benutzer nicht sieht, also der Java-Code. Zuerst werden die Daten genau so geladen wie auch im Trainingsmodus und im Bildquiz. Interessant ist, was danach geschieht. Zuerst werden die Werte eingesetzt, und zwar folgendermassen:

```
private void setValuesSndQuiz() {
    try {
        jsonResponse = new JSONObject(jsonResult);
        jsonMainNode = jsonResponse.getJSONArray("testversuch");

        size = jsonMainNode.length();
        Random rand = new Random();
        int i = rand.nextInt(size);

        if(count <= size) {
            if(list.contains(i)) {
                setValuesSndQuiz();
            } else {
                jsonChildNode = jsonMainNode.getJSONObject(i);
                list.add(i);

                name = jsonChildNode.getString("name");
                snd = jsonChildNode.getString("sound");

                mPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
                mPlayer.setDataSource(snd);
                mPlayer.prepareAsync();

                ListDrawer();
            }
        } else {
            showResult();
        }
    } catch(JSONException e) {
        Toast.makeText(getApplicationContext(), "JSON Error" +
            e.toString(), Toast.LENGTH_SHORT).show();
    } catch(MalformedURLException e) {
        Toast.makeText(getApplicationContext(), "Malformed URL Error" +
            e.toString(), Toast.LENGTH_SHORT).show();
    } catch(IOException e) {
        Toast.makeText(getApplicationContext(), "IO Error" +
            e.toString(), Toast.LENGTH_SHORT).show();
    }
}
```

Listing 18: Werte werden eingesetzt und der MediaPlayer wird vorbereitet

Die Struktur ähnelt sehr stark derjenigen des Bildquiz, auch hier wird die Zufallsvariable verwendet, um eine zufällige Heuschrecke zu laden. Allerdings muss kein Bild heruntergeladen werden.
Der MediaPlayer “mPlayer” wird so eingestellt, dass er einen Stream abspielt. Dies ist notwendig, weil die Datei vom Webserver geladen wird und sie ansonsten vollständig lokal verfügbar sein müsste. Als nächstes wird die Quelle eingestellt, die dem Link entspricht. Zuletzt wird der MediaPlayer asynchron vorbereitet, was bedeutet, dass der Vorbereitungsprozess im Hintergrund läuft. Würde er nicht asynchron vorbereitet werden, so könnte man in der Zwischenzeit die App nicht bedienen. Das könnte dann durchaus zu

Fehlern und zum Absturz der App führen.

```
mPlayer.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {

    @Override
    public void onPrepared(MediaPlayer mPlayer) {
        mPlayer.start();
        mediabtn.setText("Stop");
    }
});
```

Listing 19: Ton wird abgespielt

Nun wird im Listing 19 definiert, was passiert, wenn der MediaPlayer vorbereitet ist. Da es ein wenig mühsam wäre, wenn der Spieler zusätzlich noch auf einen Button tippen müsste, um den Ton zu hören, wird er automatisch abgespielt.

```
mediabtn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        if(mPlayer.isPlaying()) {
            mPlayer.stop();
            mediabtn.setText("Play");
        } else {
            mPlayer.prepareAsync();
            mPlayer.start();
            mediabtn.setText("Stop");
        }
    }
});
```

Listing 20: Code des MediaPlayer-Buttons

Es gibt zwei unterschiedliche Zustände, in denen sich der MediaPlayer befinden kann. Entweder spielt er gerade einen Ton ab oder nicht. Demnach sollte der Button jeweils richtig entscheiden, ob nun der Ton angehalten oder abgespielt werden soll. Das wird mit einem If-else-Statement erreicht. Jedoch muss man beachten, dass es sich um einen Stream handelt und deshalb der MediaPlayer bei erneutem Abspielen wieder vorbereitet werden muss.

```
listview.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> arg0, View view,
        int position, long id) {
        answer = listview.getItemAtPosition(position).toString();
        if(answer.contentEquals(name)) {
```

```
6               points = points + 20;
7           } else {
8               points = points - 10;
9               Toast.makeText(getApplicationContext(), "Falsche Antwort",
10                  Toast.LENGTH_SHORT).show();
11          }
12          count++;
13          mPlayer.stop();
14          mPlayer.reset();
15          setValuesSndQuiz();
16      }
17  });
```

Listing 21: Klick auf ein Listenelement

Der Klick auf ein Element der Liste ist praktisch identisch zu demjenigen im Bildquiz. Der einzige Unterschied ist, dass die Wiedergabe gestoppt und der MediaPlayer zurückgesetzt wird und somit keine Tonquelle mehr gesetzt ist.

3.4.5 Resultat

Abbildung 6: activity_result.xml : Die Anzeige der Punktzahl

Nachdem das Spiel zu Ende ist, wird die *ResultActivity* gestartet, die das Resultat, die Punktzahl, anzeigt. Dabei kann der Wert der Variable, die in den beiden Quiz für die Punktzahl verwendet wird, nicht direkt abgerufen werden. Es gibt aber dennoch eine Möglichkeit, den Wert in dieser Activity zu erhalten. Dazu verwendet man Intents, die auf beiden Klassen vorhanden sein müssen. Intents sind Schnittstellen zwischen den verschiedenen Activities. In der sendenden Klasse wir dem Intent mithilfe der *putExtra()*-Funktion die gewünschte Variable und ein Schlüssel hinzugefügt. Der Schlüssel ist notwendig, um den Wert der Variable in der empfangenden Klasse abrufen zu können. Die Punktzahl wird in eine neue Variable gespeichert, die daraufhin im Textfeld angezeigt wird.

```
final String score = getIntent().getExtras().getString("result");
scoretxt.setText("Score: " + score);
```

Listing 22: Punktzahl wird empfangen und eingesetzt

Von der *ResultActivity* aus kann man problemlos das Quiz wiederholen. Die Activity merkt sich nämlich, welches Quiz gespielt wurde. Diese Merkfunktion habe ich ebenfalls mit der Übergabe einer Variablen gelöst, die über den Intent weitergegeben wird. Auf Seite des Empfängers, der *ResultActivity*, ist also nur noch ein If-else-Argument nötig.

```
final int key = getIntent().getExtras().getInt("key");

retrybtn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        if(key == 0) {
            Intent myIntent = new Intent(ResultActivity.this,
                ImgQuizActivity.class);
            startActivity(myIntent);
            finish();
        } else {
            Intent myIntent = new Intent(ResultActivity.this,
                SndQuizActivity.class);
            startActivity(myIntent);
            finish();
        }
    }
});
```

Listing 23: Die App überprüft, welches Quiz gespielt wurde

3.4.6 Info

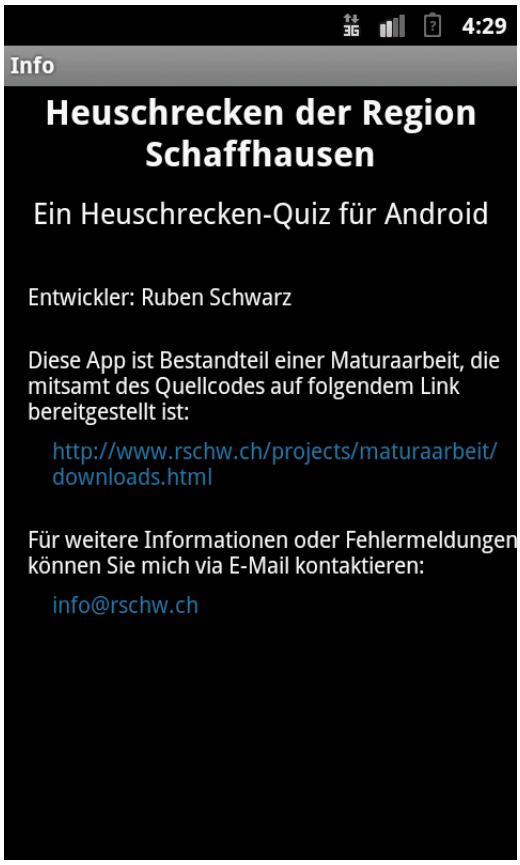

Abbildung 7: activity_info.xml : Das Infomenü

Sowohl das Layout als auch der Java-Code des Infofensters sind sehr einfach. Es dient eher dazu, allfällige Bemerkungen meinerseits festzuhalten, worunter ich vor allem Bugs (Fehler) zusammenfasse. Zudem habe ich den Link zu meiner Arbeit wie auch meine E-Mail-Adresse angegeben, um Fragen von Interessenten beantworten zu können.

4 Zusammenfassung

Vor lauter Code und Erläuterungen kann man relativ schnell den Überblick verlieren. Daher möchte ich in dieser Zusammenfassung nochmals aufzeigen, wie die App grundlegend funktioniert.
Ich möchte daran erinnern, dass die App mit einer Datenbank kommunizieren muss, um die Daten herunterzuladen.
Der Nutzer startet das Spiel. Zuerst sieht er das Hauptmenü. Im Hintergrund laufen bereits die ersten Prozesse. Es wird überprüft, ob der Spieler mit dem Internet verbunden ist, und die Buttons werden gezeichnet. Nun steht der Spieler vor der Wahl zwischen Training, Bild- oder Tonquiz. Nehmen wir an, dass der Spieler die App zum ersten Mal aufgestartet hat und kaum Erfahrung im Gebiet der Heuschrecken besitzt. Folglich entscheidet er sich für das Training. Auf dem Bildschirm erscheint nun ein neues Fenster. Es enthält den Namen der Heuschrecke, sowie auch einen kurzen Steckbrief, ein Bild und mehrere Buttons. Dabei ist folgendes geschehen: Die App hat einen Request an den Server geschickt, der von einem PHP-Skript verarbeitet und mit einer Antwort entgegnet wurde. Diese Antwort enthält alle notwendigen Daten aller Heuschrecken, die sich auf der Datenbank befinden, und ist in JSON codiert, ein Format, das von Java unterstützt wird. Die Bilder und Töne sind dabei nur Links, die auf Dateien des Webservers verweisen. Diese werden im Java-Code heruntergeladen. Drückt der Spieler den Wiedergabe-Button, so wird der Gesang der Heuschrecke abgespielt. Wählt er die nächste Heuschrecke, so wird einfach das nächste Objekt des JSON-Arrays herausgelesen und analog verarbeitet. Hat der Spieler alle Heuschrecken betrachtet, kehrt er ins Hauptmenü zurück. Nun ist er für das Bildquiz gerüstet. Er drückt auf den Button und sieht eine willkürliche Heuschrecke sowie eine Liste, aus der er seine Antwort wählen muss. Im entsprechenden Code wurde eine ähnliche Prozedur wie im Training verwendet. Allerdings wurde die zu ladende Heuschrecke durch eine Zufallszahl bestimmt und in eine unsichtbare Liste geschrieben, damit sie nicht fälschlicherweise mehrmals geladen wird. Die Liste wurde mit allen Heuschreckennamen gefüllt und, zur grösseren Herausforderung des Spielers, zufällig gemischt. Der Spieler ist aber dennoch klug genug und hat sich diese Heuschrecke aus dem Trainingsmodus merken können. Das Programm überprüft im Hintergrund, ob der Name der Heuschrecke mit seiner Antwort übereinstimmt. Sie ist richtig und er erhält 20 Punkte. Er sieht als nächstes eine neue Heuschrecke, die die vorhergehende ersetzt hat, und eine aktualisierte Liste möglicher Antworten. Diese Heuschrecke jedoch kann der Spieler nicht gut von einer anderen, ähnlichen Art unterscheiden. Ratlos entscheidet er sich für eine Heuschrecke aus der Liste. Es stellt sich heraus, dass er falsch geraten hat, denn es wird alsbald ein kleiner Balken eingeblendet mit den Worten “Falsche Antwort”. Es werden ihm 10 Punkte abgezogen. Sobald der Spieler die gesamte Liste abgearbeitet hat und das Quiz somit zu Ende ist, verschwindet das Quizfenster und wird durch ein neues Fenster ersetzt. Dieses enthält zwei Buttons und die Punktzahl des Spielers. Viel geschehen ist im Hintergrund nicht; die Punktzahl wird durch einen Intent an das neue Fenster gesendet und dort dargestellt. Der Spieler ist mit seiner Punktzahl nicht zufrieden und möchte das Spiel wiederholen, mit dem Gedanken, diesmal dank seines Kurzzeitgedächtnisses erfolgreicher zu sein. Das Spiel

lädt nochmals dieselbe Quiz-Variante. Entscheidend dafür, dass nicht das Tonquiz aufgerufen wird, ist ein weiterer Wert, der über den Intent weitergegeben wird und als ID interpretiert werden kann.
Wie man sieht, steckt viel mehr hinter der App, als man denkt. Die App ist nicht gerade die komplizierteste, und doch benötigt man recht viel Code, genauer gesagt ca. 730 Zeilen, um sie zu entwickeln. Der gesamte Quellcode ist auf der CD zu finden.

5 Verfügbarkeit der App und Ausblick

Das Heuschrecken-Quiz kann zurzeit nur über einen Link heruntergeladen werden, da die Lizenzen zur Freigabe der Bilder und Töne noch nicht erworben wurden. Der Link befindet sich am Ende dieser Seite. Die App selbst funktioniert im jetzigen Zustand und wurde bis dato auf vier verschiedenen Testgeräten (ausschliesslich des Emulators) ausgeführt, darunter auf drei Smartphones und einem 7”-Tablet. Ich werde die Entwicklung auf jeden Fall noch fortsetzen. Primär werde ich versuchen, den Code weiter zu optimieren, das Layout ansprechender zu gestalten, und vielleicht auch einige zusätzliche Funktionen einbauen.
Download-Link: http://www.rschw.ch/projects/maturaarbeit/downloads.html

6 Quellenverzeichnis

6.1 Literaturquellen

- Gargenta, Marko (2011). *Einführung in die Android-Entwicklung.* Aus dem Englischen übersetzt von Lars Schulten. O'Reilly Verlag GmbH & Co. KG, Köln.
- Roesti, Christian; Keist, Bruno (2009). *Die Stimmen der Heuschrecken.* Haupt Verlag, Bern.
- Widmer, Michael; Pfändler Ulrich (2013). *Heuschrecken im Kanton Schaffhausen.* Naturforschende Gesellschaft Schaffhausen, Schaffhausen.

6.2 Tonquellen

- Roesti, Christian; Keist, Bruno (2009). *Die Stimmen der Heuschrecken (DVD).* Haupt Verlag, Bern.

6.3 Bildquellen

- http://biofotoquiz.ch/biofotoquiz/ [Stand: 11.01.2014]

6.4 Internetquellen

- http://developer.android.com/reference/packages.html [Stand: 11.01.2014]
- http://developer.android.com/sdk/index.html [Stand: 11.01.2014]
- http://www.eclipse.org/ [Stand: 11.01.2014]
- http://notepad-plus-plus.org/ [Stand: 11.01.2014]
- http://miktex.org/ [Stand: 11.01.2014]
- http://docs.oracle.com/javase/tutorial/java/index.html [Stand: 11.01.2014]
- http://en.wikibooks.org/wiki/LaTeX [Stand: 11.01.2014]
- http://de.wikipedia.org/wiki/Android_(Betriebssystem) [Stand: 11.01.2014]
- http://de.wikipedia.org/wiki/Php [Stand: 11.01.2014]
- http://de.wikipedia.org/wiki/Mysql [Stand: 11.01.2014]
- http://de.wikipedia.org/wiki/Datentypen [Stand: 11.01.2014]
- http://img.tamindir.com/ti_e_ul/BarisYanik/p/phpmyadmin_2_1152x826.png [Stand: 11.01.2014]

- http://de.wikipedia.org/wiki/JSON [Stand: 11.01.2014]
- http://de.wikipedia.org/wiki/Xml [Stand: 11.01.2014]
- http://php.net/manual/de/book.mysql.php [Stand: 11.01.2014]
- http://www.vogella.com/tutorials/Android/article.html#gridlayout_scrollview [Stand: 11.01.2014]
- http://www.vogella.com/tutorials/AndroidIntent/article.html [Stand: 11.01.2014]
- http://www.vogella.com/tutorials/AndroidListView/article.html [Stand: 11.01.2014]
- http://codeoncloud.blogspot.ch/2013/07/android-mysql-php-json-tutorial.html [Stand: 11.01.2014]
- http://www.androidhive.info/2011/08/how-to-switch-between-activities-in-android/ [Stand: 11.01.2014]
- http://www.androidhive.info/2012/01/android-json-parsing-tutorial/ [Stand: 11.01.2014]