Philosophische Fakultät III

Sprach- , Literatur- und Kulturwissenschaften

Institut für Information und Medien, Sprache und Kultur (I:IMSK)

Lehrstuhl für Medieninformatik

Einführung in die Anwendungsprogrammierung

Modul: MEI-M03.3

Sommersemester 2017

Koch-App

Nina Hösl

1827558

Medieninformatik/Informationswissenschaft

3. Semester B.A.

Kantstraße 15

85098 Großmehring

Tel.: 0175 2954129

Email: nina.hoesl@stud.uni-regensburg.de

Jonas Jelinski

Medieninformatik/Informationswissenschaft

3. Semester B.A.

Roter-Brach-Weg 62

93049 Regensburg

Tel.: 0176 93533 778

Email: jonas.jelinski@stud.uni-regensburg.de

Schmid Roman

1570490

Informationswissenschaft/Medieninformatik

7. Semester B.A.

Altdorferstr.14

93093 Donaustauf

Tel.: 0173834444

Email: roman.schmid@stud.uni-regensburg.de

Abgegeben am 22.09.17

Inhalt

1 Projektbeschreibung

Die App soll einerseits eine klassische App zum Speichern und Abrufen von Rezepten sein, die der Nutzer nicht nur zur Organisation seiner Rezepte nutzen kann, sondern die Rezepte auch direkt beim Kochvorgang abrufen kann. Der Nutzer soll die Möglichkeit haben, erledigte Aufgaben und Zutaten markieren zu können.

Zudem hat der Nutzer die Möglichkeit, einen Timer zu starten, sobald das Gericht für eine bestimmte Zeit garen oder ruhen soll. Dieser Timer erinnert ihn rechtzeitig und lässt sich durch rasche Bewegung des Handys stoppen.

Im Gegensatz zu den marktüblichen Apps soll der Nutzer zudem die Möglichkeit haben, die Rezepte zu individualisieren. Hierzu hat er einerseits die Möglichkeit, Rezepte jederzeit zu bearbeiten, um die Zutaten, Kochzeiten und Arbeitsschritte an seinen Geschmack anzupassen. Andererseits soll ihm die Möglichkeit gegeben werden, die gespeicherte Kochzeit anzupassen, indem er den implementierten Kochzeit-Timer bei früherem Erreichen des Garzeitpunktes zu stoppen oder bei längerer Gardauer den Timer zu verlängern. Diese angepassten Garzeiten sollten entsprechen gespeichert werden und im Rezept korrigiert werden.

2 Kurzbeschreibung

Ihre perfektionierten Rezepte im Blick – immer und überall mit der Garzeit-CoockingApp der Programmier-Vereinigung JNR.

Sie möchten ortsunabhängig mal eben einen perfekten Obstkuchen backen oder die perfekte Fertigpizza zubereiten? Sie benötigen einen Timer, haben aber keine Küchenuhr mehr?
Kein Problem mit der Garzeit-CookingApp. Besonders praktisch: Sie passt die Zubereitungszeiten Ihren Wünschen und Geräten an.
Ihr Ofen braucht für Ihre Lieblingspizza immer länger als angegeben? Mit der Garzeit-CookingApp einmal die perfekte Zeit eingestellt, immer ein perfektes Ergebnis! Dabei haben Sie nicht nur die Möglichkeit vorhandene Rezepte zu laden und zu verändern. Die Garzeit-CoockingApp bietet Ihnen die Möglichkeit eigene Rezepte zu erstellen und zu speichern. So vergisst man sicher nicht, wie Omas Lieblingseintopf perfekt gelingt.
Ihr Braten hat zu viele Einzelschritte? Garzeit-CookingApp lässt Sie auch hier nicht alleine - mit unseren ad hoc Möglichkeiten zur Abarbeitung Ihres Rezeptes vergessen sie definitiv keinen Schritt.

Ihre Garzeit-CoockingApp bietet Ihnen viele nützliche Funktionen.
Hier sind alle Funktionen und Services im Überblick:

- Persönliche Rezepte
- Optimale Garzeit
- Individualisierbare Kochzeiten
- Markierung der einzelnen Schritte
- Persönliche Lieblingsrezepte einfach und unkompliziert speichern
- Keinen Timer mehr verpassen durch Vibration

- Deaktivierung des Timers durch einfaches “Schütteln” des Handys

3 Technische Voraussetzungen

Die Garzeit-CoockingApp ist eine reine Mobil-Anwendung, welche ohne eigene Server-Infrastruktur funktioniert. Vorgefertigte Rezepte können in die Datenbank geladen oder in dem dafür vorgesehenen Menü einfach erstellt werden. Für den Betrieb der Anwendung reicht eine Installation dieser auf einem Android fähigen Endgerät aus. Unterstützt werden Smartphones und Tablets, die mindestens die Version 5.0 des Android-Systems nutzen. Eine aktive Verbindung zu einem Netzwerk ist lediglich für die Installation erforderlich. Für den Betrieb der App wird ferner ein freier Speicher von wenigen MB benötigt.

4 Problemstellung und Lösungsvorschlag

Heutzutage gibt es verschiedenste Apps, die sich mit dem Thema Kochen beschäftigen. Dabei handelt es sich meist um reine Koch-Apps, welche dem Nutzer nur Rezepte zur Verfügung stellen. Für einen großen Teil der Rezepte gibt es -bis auf einzelne Ausnahmen- keine Möglichkeit diese immer wieder neu anzupassen, sei es in Bezug auf die Zutaten oder die Zubereitungszeit. In der heutigen Zeit gewinnt dieser Faktor jedoch immer mehr an Bedeutung, da sich die äußeren Umstände, unter denen ein Gericht zubereitet wird, immer wieder verändern. Sei dies nun aufgrund eines neuen Küchengerätes oder wegen einer Lebensmittelunverträglichkeit.

Solche Apps bieten dem Nutzer oftmals auch keine Option an, eigene Rezepte anzulegen und (lokal) zu speichern.

Diese und weitere Punkte haben uns überzeugt, dass der Markt reif ist für eine App, die sich immer wieder auf die Bedürfnisse des Nutzers anpasst und zudem viele bewährte Eigenschaften bestehender Apps implementiert.

- Unabhängigkeit: Die Frage welches Gericht auf den Tisch kommt, stellt sich nicht zwangsläufig immer zuhause, sondern auch unterwegs. Der User hat somit seine Rezepte immer bei sich. Möchte er spontan ein neues Rezept anlegen, hat er auch hierzu immer die Möglichkeit.
- Kompaktheit: Viele Kochbücher oder ausgedruckte Rezepte haben den Nachteil, dass diese in der (Studenten-) Küche viel Platz wegnehmen. Das Smartphone hingegen bietet dem Anwender eine kompakte, platzsparende und ressourcenschonende Alternative.
- Unterstützung: Oft werden für eine Mahlzeit mehrere Kochschritte mit unterschiedlichen Zubereitungszeiten benötigt. Hierbei ist es oft hilfreich, wenn einzelne Arbeitsschritte nacheinander dargestellt werden. Ebenso unterstützen ein oder mehrere Timer den User dabei, die Zubereitung rechtzeitig zu beenden.

- Individuell: Je nach persönlichen Präferenzen können die Zutaten oder die Garzeit variieren. So mag Nutzer A seine Pizza eher knusprig und benötigt gute 5 min länger als auf der Packungsangabe vermerkt. Nutzer B hingegen besitzt keinen Backofen, sondern nur eine Mikrowelle mit Backofenfunktion. Er benötigt also beispielsweise 13 Minuten mehr Zubereitungszeit, um ein gutes Resultat erzielen zu können.

Die Garzeit-CookingApp soll dem Nutzer eine Koch-App an die Hand geben, die ihn von der Erstellung des Rezeptes bis hin zum fertigen Gericht durchgehend begleitet. Am Anfang hat der Nutzer die Möglichkeit, gezielt nach einem vorhandenen Rezept in der Datenbank zu suchen oder sich durch die Auswahl verschiedener Kriterien ein Zufallsgericht ausgeben zu lassen. Hat der Nutzer sich für ein Rezept entschieden, erscheint dieses untergliedert in einzelne Teilabschnitte, welche hintereinander abgearbeitet und abgehakt werden können. Somit ist das Auslassen eines einzelnen Arbeitsschrittes faktisch ausgeschlossen. Als großes Extra bietet die App einen eingebauten Timer, welcher dem Nutzer optisch, haptisch und akustisch Feedback gibt, wenn die Zeit abgelaufen ist. Hierfür muss der Koch nicht mehr in der Küche oder in der Nähe eines stationären Timers sein. Ab sofort reicht es aus, das Handy in der Nähe oder in der Hosentasche zu haben und man wird trotzdem pünktlich erinnert. Stellt man noch während des Kochvorgangs fest, dass das Produkt beispielsweise schon vor Ablauf der Zeit die gewünschte Konsistenz besitzt, so lässt sich der Timer ganz einfach anpassen. Diese Änderung kann noch während oder nach des Kochvorgangs stattfinden, indem Zeit hinzugefügt oder abgezogen werden kann. Es besteht die Möglichkeit der Anpassung des Alarmtons durch verschiedene Sounds und Musikstücke. Ferner fängt das Smartphone an zu vibrieren, wenn der Countdown bei 0 angekommen ist.
Um diese Vibration zu stoppen, reicht es aus, das Handy zu schütteln oder den Bildschirm kurz zu berühren.

Wurde der für den Nutzer bestmögliche Garzeitpunkt gefunden, so kann er diesen direkt im Rezept ändern und sich zukünftig blind auf die Timerfunktion verlassen.

Als weitere Eigenschaft besitzt die App die Funktion, die Zutatenmenge anzupassen. Dies kann sowohl bei der Erstellung eines komplett neuen Rezeptes erfolgen oder aber auch während oder nach dem eigentlichen Kochvorgang. Der Nutzer hat so immer und jederzeit die volle Kontrolle über das Rezept.

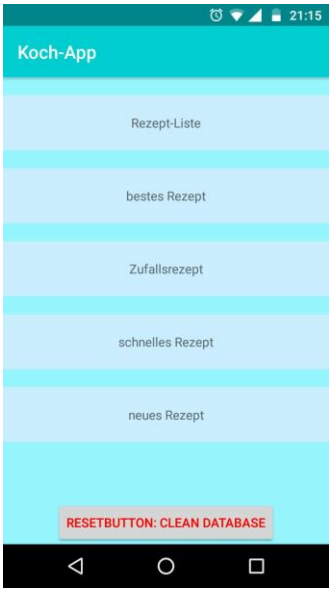
Als letztes Feature beinhaltet die Garzeit-CookingApp die Möglichkeit, Rezepte zu ranken. Dies geschieht durch die Vergabe von maximal 5 Sternen. Eine volle Sternanzahl bedeutet, dass das Gericht super schmeckt, wohingegen ein Stern darauf hinweist, dass das Rezept nicht der große Wurf ist und gegebenenfalls überarbeitet werden sollte.

Alles in allem bietet die App eine Vielzahl an verschiedenen Funktionen, die so bisher in noch keiner App vereint wurden. Sie gibt dem Nutzer die alleinige Kontrolle über sämtliche Arbeitsschritte und unterstützt ihn hierbei durch raffinierte, aber auf das Wesentliche beschränkte Methoden.

5 Design & Implementierung

5.1 User Interface

Die Anwendung besteht aus zehn verschiedenen Activities und zwölf Klassen. Eine Haupt-Activity (MainActivity) dient als Einstiegspunkt der App und stellt ein Menü mit verschiedenen Auswahlmöglichkeiten dar. Es gibt fünf verschiedene Listen mit unterschiedlichen Themenbereichen. Durch einen Klick auf einen der Listeneinträge gelangt der User zu einem der vier verschieden Rezeptmöglichkeiten. Je nach ausgewählter Liste wird eine neue Activity aufgerufen.



Grafik 1 Startbildschirm

RecipeListActivity:

Hinter der "Rezepte-Liste" verbirgt sie die RecipeListActivity. Hier werden verschiedene vorhandene Rezepte angezeigt. Durch Drücken auf ein Rezept (Beispielhaft "Beispiel-Rezept") gelangt man anschließend zu dem eigentlichen Rezept. Hierbei wird die Klasse "Recipe" aufgerufen, welche hierfür die entscheidenden Attribute bereitstellt.

Auf der nun aufgerufenen Seite bekommt der Nutzer eine Vielzahl von Informationen an die Hand gegeben. Zum einen die Rating Bar (siehe Punkt Rating_Activity**),** eine Liste mit allen Zutaten, die für das Rezept wichtig sind und zum anderen eine Option die Anzahl der Portionen zu erhöhen oder zu erniedrigen. Am Ende der Zutatenliste findet sich der Button " Timer Starten" (CookingTimerActivity), mit der der Kochvorgang gestartet wird.

Rating_Activity:

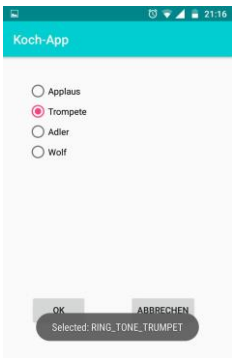
Über der Zutatenliste des Rezeptes befindet sich die Rating Bar. Diese zeigt dem Nutzer bei dem ersten Aufruf des Rezeptes 5 unausgefüllte Sterne. Der User hat direkt die Möglichkeit, über wischen oder antippen die Sterne mit Lila zu füllen.

Neben den Sternen befindet sich ein weiterer Button mit dem die Bewertung abgespeichert werden kann (Rating_Activity). Hierdurch öffnet sich eine neue Liste, wo nach Bedarf das Rating nochmals angepasst werden kann. Der Nutzer hat abschließend 2 Auswahlmöglichkeiten. Über den Button “weitere Bestandteile des Rezeptes ändern?” gelangt er direkt in die RecipeNewActivity, wo er das gesamte Rezept ändern könnte. Durch die zweite Option “Rezept nicht weiter verändern und Bewertung speichern?”, welche mit dem “Ja” Button bestätigt werden kann, kehrt der Nutzer zur Startseite zurück und die Bewertung wird in die Datenbank geschrieben.

RecipeNewActivity

Neben den Möglichkeiten sich direkt aus der Datenbank ein “bestes Rezept” oder ein “schnelles Rezept” ausgeben zu lassen, existiert noch der Reiter “neues Rezept”. Hier wird dem Nutzer eine Blankoliste ausgegeben, welche er nach persönlichen Präferenzen befüllen kann. Zur Benutzereingabe stehen die verpflichtenden Felder “Rezeptname”, „Schwierigkeitsgrad (Leicht, Mittel, Schwer)”, die „Anzahl der Portionen“ und abschließend die „Kochdauer“ zum Ausfüllen bereit. Werden nicht alle Felder ausgefüllt oder versehentlich vergessen, wird ein Feedback mit Hilfe eines Toasts erzeugt “Eine Rezepteingabe ist leer”. Am Ende des Screens stehen dem Nutzer mehrere Buttons zur Verfügung. So kann er “weitere Arbeitsschritte” oder eine “neue Zutat” auswählen. Werden keine weiteren Optionen benötigt, so gelangt der User über den Button “Speichern und zurück zum Rezept” direkt zum angelegten Rezept

Neben den nun beschrieben sichtbaren Komponenten verfügt die Anwendung über eine Vielzahl an Klassen, die zur Speicherung und zum Abrufen der Daten aus der Datenbank dienen . Eine solche zentrale Einheit bietet die CookingTimerActivity.



Grafik 2 Sound auswählen

CookingTimerActivity

Diese Activity regelt sämtliche zentrale Aufgaben rund um den Timer. Ist der Benutzer bis zum Timer gelangt, so kann er hier erneut verschiedene Einstellungen vornehmen. Die Activity verfügt oben über einen Button mit dem der Timer direkt mit den voreingestellten Werten startet. Darunter erscheint der Timer mit der noch verblei-

benden Zeit in dem Format hh.mm.ss. Der Nutzer kann die Kochzeit direkt anpassen, indem er die Minuten in den Feldern unter der Uhr eingibt. Durch den Button “Klingelton ändern” gelangt man zur ChooseRingToneActivity.

ChooseRingToneActivity

In dieser Activity wird es dem Anwender ermöglicht, über RadioButtons einen von vier Alarmtönen auszuwählen. Hierzu zählen “Applaus”, “Trompete”, “Adler” und “Wolf”. Die Änderung kann entweder über “Ok” oder “Abbrechen” bestätigt werden

Die Usability-Aspekte “Enchant me”, “Simplify my life” und “Make me amazing” wurden in der Vorlesung nur oberflächliche behandelt. Daher lag der Fokus der Arbeit vor allem auf den anderen Bereichen des Projekts. Es wurde auf eine intuitive Bedienung geachtet. Das Layout soll möglichst ansprechend sein. Insbesondere die sanften Farbkombinationen sollen auf den Nutzer entspannend wirken. Die Texte sind selbsterklärend gestaltet. Das Design ist an den Ideen von Steve Krug’s “Don’t make me think”[1] ausgerichtet. Ziel ist es den Nutzer kognitiv zu entlasten.

5.2 Implementierung

Die wesentliche Aufgabe der Anwendung besteht in der Bereitstellung der Datenbank mit ihren enthaltenen Rezepten. Initial wurde daher die Verarbeitung bzw. die Beschaffung dieser Daten getestet und implementiert.

Als Datenbank wurde GreenDao[2] gewählt, da sie die Möglichkeit der Verwendung von Java Objekten mit Verbindung der SQLite Datenbank mit großer Geschwindigkeit bietet.

Grafik 3 Datenbankschema

Vorteil dieser Methode ist, dass die serialisierbaren Objekte in Byte-Streams umgewandelt werden müssen, was sehr speicher- und zeitaufwändig ist.

Konzept: Jedes Rezept das der Nutzer abrufen oder speichern kann besteht aus drei Objekten. 1. Das Rezept der Klass Recipe.class, welches die wichtigsten Daten enthält.
2. Eine Liste von Zutaten und 3. eine Liste von Arbeitsschritten, welche in der Recipe.class, gespeichert werden. Jede einzelne Zutat ist eine Instanz der Klasse Ingrident.class und jeder Arbeitsschritt eine Instanz der Recipe-WorkStep.class. Jede Zutat und jeder Arbeitsschritt enthält die ID des Rezept zu dem es gehört. Diese Rezept-ID spielt eine zentrale Rolle in der Implementierung.

Recipe.class: Die Recipe class enthält: ID, Name, Anzahl der Portionen, Schwierigkeitsgrad, die Kochzeit und die Rezeptbewertung.

Ingrident.class: Die Ingridentklass enthält neben anderen Werten eine eigene ID und eine recipeID, durch welche ihre Instanzen zum Rezept zuordenbar sind. Daneben enthält es die Werte Einheit, Name und Menge der Zutat.

RecipeWorkStep.class: Die RecipeWorkStep.class besteht aus einem String der den Arbeitsschritt beschreiben soll, einer ID für die Instanz und einer RecipeID zur Zuordnung zum Rezept.

Die drei Klassen enthalten Anweisungen an Greendao, wie etwa @Entity, welche dem Greendao Generator sagen, wie er die einzelnen Objekte Umwandeln soll, wenn das Projekt erstellt wird. Im build.gradle(Module:app) muss nach einer Änderungen in diesen Objekten die Schema-Version erhöht werden. greendao {schemaVersion 4 }

Ansonsten ist eine Zuweisung der Objekte zur Datenbank nicht mehr möglich.

ID: Die ID der Rezepte ist der wichtigste Wert bei der Kommunikation der Activities mit der Datenbank und untereinander. Sie ermöglicht eine eindeutige Zuordnung der Rezepte zu den Datenbankeinträgen. Die Übergabe von Objekten in Intents zwischen den einzelnen Activities wäre sehr fehleranfällig. Daher schickt jede Activity der anderen Activity einfach die ID des Rezeptes und die empfangenden Activity holt sich die passenden Werte mit der RezeptID aus der Datenbank. Die gesamte Implementierung der internen Kommunikation der App ist auf diese RezeptID ausgerichtet.

DBAdapter: Zur Kommunikation mit der Datenbank wurde ein eigener Adapter implementiert. Der Adapter schränkt die Fehleranfälligkeit durch die Eingabe von Greendaobefehlen ein. Er besteht im wesentlichen aus Methoden, welche

SQLite-basierte Greendaobefehle enthalten. Optional mit Parametern. Jede Activity die mit der Datenbank kommunizieren möchte, instanziert den Adapter. Anschließend kann sie über die im Adapter befindlichen *öffentlichen* Methoden mit der Datenbank kommunizieren. Die DBAdapter.class
teilt sich in zwei Teile. Ein Teil besteht ausschließlich aus privaten Methoden. Dieser sollte so gut wie Möglich vom Nutzer des Adapters gekapselt sein. Der zweite Teil besteht aus Methoden die über die Instanzen des DB-Adapters zugänglich sein sollen. Fast alle Methoden darin sind daher public, außer diejenigen welche erst in einer zukünftigen Version der App genutzt werden sollen. **RecipeNewActivity:** Die RecipeNewActivity dient zur Erstellung neuer Rezepte und zur Änderung alter Rezepte. Ihr wird beim Aufruf ein boolean Wert mit dem NEW_RECIPE_KEY übergeben. Dieser sagt ihr, ob sie ein neues Rezept erstellen oder ein altes ändern soll.

Grafik 4 Neues Rezept

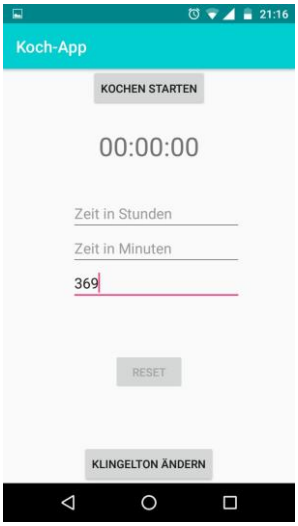
Wenn sie ein neues erstellen soll, erstellt sie ein neues über einen DBAdapterbefehl und arbeitet anschließend mit dessen ID. Wenn sie ein altes ändern soll nimmt sie die ID des alten Rezeptes welcher sich in einem Intent befindet und lädt alle Rezeptdaten in die Anzeigefenster der RecipeNewActivity. Anschließend wird diese ID verwendet.
Die Interaktion des Nutzers mit der Activity ist stark eingeschränkt, um Fehleinträge in der Datenbank zu vermeiden. So wird geprüft ob der Name des Rezeptes schon vorhanden ist und alle Felder gefüllt sind, bevor das Rezept gespeichert werden kann. Über zwei Buttons kann der Nutzer entweder die IngridentActivity oder die WorkStepActivity öffnen. Dabei wird die ID des Rezeptes mitgegeben.
IngridentActivity: Die IngridentActivity erhält die RecipeID über den gesendeten Intent. Dieser erhält jede neues Zutat, um sich selbst zuordnen zu können. Ziel dieser Activity ist, dass der Nutzer neue Zutaten erstellen kann. Dies geschieht über EditTexts und einer ListView welche die zur RecipeID zugehörigen Zutaten anzeigt. Auch hier ist der Nutzer eingeschränkt. Als Werte für die Menge werden nur Werte vom Typ NumberPasswort, also Zahlen bereitgestellt. Drückt der Nutzer den Speicherknopf werden alle neu zur ListView zugefügten Objekte über

den DBAdapter in die Datenbank geschrieben.
WorkStepActivity: Die WorkstepActitvity arbeitet ähnlich, wie die IngridentActivity. Auch ihr wird die RecipeID zur Zuordnung der Arbeitsschritte zum Objekt übergeben. Beide Activities zeigen alte Einträge des Rezeptes automatisch an. Beide Arbeiten mit Arrayadaptern zur Füllung der ListViews und OnItemClickListener.
RecipeListActivity: Die RecipeListActivity zeigt den Namen aller gespeicherten Rezepte in einer ListView an. Über OnItemClickListener wird sie gesteuert. Klickt der Nutzer kurz, wählt er das Rezept aus und die RecipeStartActivity wird gestartet. Ihr wird die ID des ausgewählten Rezeptes übergeben. Klickt der Nutzer lang auf einen Namen wird über den `AlertDialog.Builder` die Option angezeigt, ob er das Rezept löschen möchte oder nicht. Beim löschen wird das Rezept aus der Datenbank gelöscht und die ListView aktualisiert.
RecipeStartActivity: Hast der Nutzer sich zum Start der RecipeStartActivity Activity entschieden, lädt diese über die übergebene RecipeID alle zugehörigen Daten aus der Datenbank in ihre Anzeigen. Nun erhält er eine Übersicht über die Daten des Rezeptes, inklusive Arbeitsschritte und Zutaten. Die letzten beiden werden in ListViews angezeigt.
Die ArbeitsschrittListView kommt mit einem einfachen Arrayadapter aus, der sie mit einer Liste voller Arbeitschrittbeschreibung-Strings verbindet. Für die ZutatenListView wurde der eigene Adapter `IngridentListViewAdapter extends ArrayAdapter<Ingrident>`
implemntiert. Dieser überschreibt die Methoden eines Arrayadapters und ermöglicht es die Angaben für Name, Menge, Einheit jeder Zutat in jeder Zeil der ListView nebeneinander darzustellen.Die public Methode `notifyIngridentsChanged` ersetzt die `notifyDataSetChanged()` Methode des Arraydapters. Die Aktualisierung der ListViews muss in einem eigenen UI-Thread laufen, um fehlerfrei zu funktionieren.
Diese Art der Implementierung ermöglicht es den Nutzer über einen Plus und einen MinusButton die Menge and Portionen und damit die Menge pro Zutat aktiv in der ListView zu verändern. Dazu wird die Anzahl der Portionen im Verhältnis zur Menge der Zutat gesetzt und die Menge für jede einzelne Zutat in einem Programmteil umgerechnet.

Der Nutzer kann über einen Button die CookingTimerActivitystarten oder über einen anderen Button die Rating_Activity starten.

CookingTimerActivity: Die CookingTimerActivity zählt die Zeit des Rezeptes runter und startet einen Vibrations- und Musik Alarm. Dieser kann durch schütteln des Handys oder drücken eines Buttons geändert werden. Die Implementierung orientiert sich an den Vorgaben in den Übungen des Kurses. Über die Rezept-ID wird die Zeit für das Rezept ermittelt. Der Nutzer kann es noch ändern und den CountDown starten, sobald er bereit ist. Beim Start wird der CookingTimer-Service gestartet. Dieser implementiert ein Objekt der Klasse CookingTimer. Darin wird die übergebene Zeit mittels einem CountDownTimer umgerechnet. CookingTimerService und CookingTimer kommunizieren über den CookingTimerListener. Jedesmal wenn der CookingTimer eine neue Zeit errechnet hat, schickt der CookingTimerService einen BroadCast mit der neu berechneten Zeit an den CookingTimeReceiver. Dieser erbt von BroadCastReceiver. Die CookingTimerActivity instanziert diesen BroadCastReceiver und erhält dadurch die aktuelle Zeit. Die Zeit wird wieder in einen eigenen UI-Thread angezeigt werden. Ist die Zeit abgelaufen startet die CookingTimerActivity den PlayRingToneService.

PlayRingToneService: Der CookingTimerService ist ein Service der den MediaPlayer instanziert und den Android-Vibrator. Solange er läuft vibriert das Handy und der Ton wird über den MediaPlayer abgespielt. Da der MediaPlayer sehr empfindlich ist gegenüber den Zuständen in welchem er sich in seinem Lifecycle befindet, wurden eine Reihe von Listener implementiert, deren einziges Ziel ist, die Korrekte Steuerung des MediaPlayers zu gewährleisten. Ohne die Listener neigte der Player dazu weiter zu spielen, auch wenn der Service geschlossen wurde. Das lag daran, dass es zu Fehlern im MediaPlayer kam. Treten diese Fehler wieder auf, lösen die Listener die passende Methoden aus um den MediaPlayer in den passenden Zustand zu versetzen. Die CookingTimerActitvity übergibt dem RingToneService einen String, welcher den Titel des Liedes in der Datenbank enthält. Dieser wird im RingToneService in eine URI umgewandelt und das Lied ausgelesen. Gerade bei größeren Musikdateien und langen Ladezeiten ist der MediaPlayer sehr fehleranfällig. Daher sollten nur kurze Lieder verwendet



Grafik 5 Timer

werden. Zur Sicherheit wurden die oben beschriebenen ErrorListsner implementiert. “Die” Lieder im Projekt ist eigentlich *ein* Beispiellied, das *nicht* repräsentativ sind, sondern nur aus Gründen des Urheberrechts eingefügt wurde. In der raw Datei befindet sich dreimal das *gleiche* Lied mit unterschiedlichen Namen. Obwohl unterschiedliche Lieder tatsächlich ausgewählt werden können, erscheint immer der gleiche Soundtrack. Die ursprünglichen Sounds Wolf, Adler, Trompete, Ton wurden aus rechtlichen Gründen entfernt. Die CookingTimerActitvity beendeten den PlayRingToneService mittels stopService().
MovingSensor: Der RingToneService kann über den MovingSensor beendet werden. Dieser registriert und meldet Bewegungen über den Sensor und SensorEventlistener.
Die Implementierung orientiert sich an den Vorgaben im Kurs. Die Werte der Bewegungserkennung wurden geändert. Der PlayRingToneService erfährt über die überschriebene Methode des SensorEventListener, wann das Handy bewegt wurde.
Der Alarm wird dann beendet.
BeispielRezept: Beim Starten der MainActivity wird ein Beispielrezept erstellt, welches nicht gelöscht werden kann. Dieses wird den Activites verwendet, um Nullpointer Exceptions zu verhindern. Bei fehlenden Werten, werden die enthaltenen Beispielwerte verwendet.

6 Testing

Um die App zu testen, wurden 4 Probanden ausgewählt, welche die App zum realen Kochen verwenden sollten. Dabei wurden zwei Fehler entdeckt.
Der erste Fehler wurde bei der Abänderung eines bestehenden Rezeptes gefunden. Bei dem Klick rechts oben auf “Editieren” kam vom Android System die Fehlermeldung “Koch-App” wurde beendet. Nach der Bestätigung auf „ok“ wurde die App durch das System jedoch nicht beendet, sondern sie kehrte zu dem Startbildschirm der Koch-App zurück.
Fehlerursache: Durch das Fehlen einer Beispielrezeptur in der Datenbank wurde ein falscher boolean Wert (NullPointerException) ausgegeben.
Fehlerbehebung: Erstellen eines Rezeptes in der Datenbank, welches nicht mehr gelöscht werden kann, so dass der Intenet immer einen korrekten Wert übergeben bekommt.
Der zweite Fehler betrifft das Beenden der abzuspielenden Musik. Auf manchen Android Geräten hört die Soundwiedergabe erst am Ende der Audiodatei auf.
Fehlerursache: Vermutet wird ein Fehler im Lifecycle des Mediaplayers.
Fehlerbehebung: Keine Fehlerbehebung möglich, da offensichtlich auf einzelne Geräte beschränkt.

Weitere Fehler wurden während des Usability-Testes nicht aufgedeckt. Kleinere Bugs und das Einbinden vergessener Eingabefelder wurden bereits während der Entwicklung behoben. Jedoch gab es noch Rückmeldungen der Probanden bezüglich der Anpassung des Layouts und der Strukturierung der Rezepte. Größte Änderungswünsche wurden bei dem Timer geäußert. Die Funktion die Uhr, während sie läuft, mit “+” und “-” anzupassen, wurde als nicht sehr nützlich erachtet und deswegen benutzerfreundlich gelöst. Diese Änderungen wurden in der finalen Version umgesetzt.

7 Finaler Zustand und Ausblick

Die App setzt beinahe alle von uns selbst definierten Kern-Features problemlos um. Die Datenbank erlaubt dank Objekten die dauerhafte Speicherung von Eingaben und der Kochzeit auf dem Smartphone. Der Nutzer hat die Möglichkeit, jederzeit Rezepte hinzuzufügen, zu löschen und via Startbildschirm Gerichte auszuwählen. Der Aufbau der Rezepte ist untergliedert in einen “Rezeptnamen” gefolgt von dem Schwierigkeitslevel (Leicht, Mittel, Schwer). Anschließend folgt die Anzahl der Portionen und die Kochdauer.
Der Timer hat im finalen Zustand einige Änderungen bekommen. So lässt sich nun direkt nach dem Ablauf die Zubereitungszeit des Rezepts ändern. Alle anderen Eigenschaften bleiben unverändert.

Leider konnte die Anforderung “Checkboxen bei einzelnen Zutaten und Arbeitsschritten” nicht wie geplant in Gänze umgesetzt werden. Aufgrund eines vermeintlichen Fehlers in der Listview lassen sich die vorhandenen Checkboxen nicht wie gewünscht anklicken. Die Fehlerbehebung konnte trotz intensiver Versuche vor Abgabe nicht mehr erfolgen.

Folgende Zusatzfeatures konnten erfolgreich realisiert werden.
Dem Nutzer ist es in der aktuellsten Version möglich, zusätzlich zur Vibration bei Beendigung des Timers einen Signalton zu setzen. Bei der Umsetzung dieser Erweiterung wurde die Wahl zwischen mehreren verschiedenen Tönen verwirklicht. In der abgegebenen App wurde indes nur eine Audiodatei mitgeliefert, um nicht gegen mögliches Copyright zu verstoßen.

Weiteres Feature welches umgesetzt werden konnte, betrifft das Adaptive Layout. Somit kann die App auf verschieden Gerätegrößen problemlos verwendet werden. Auch wurden die einzelnen Layout Element so konfiguriert, dass Sie sich dem Display anpassen.

Ein Feature, welches aus Zeitgründen nicht mehr umgesetzt werden konnte, wäre eine Funktion, mit welcher sich die einzelnen Rezepte teilen lassen. Eine

weitere Möglichkeit die App zu erweitern, wäre die Anbindung der Datenbank an eine bestehende, wie die der Chefkoch App. Leider hat Chefkoch die erforderlichen Ressourcen hierfür nicht rechtzeitig zur Verfügung gestellt.
Des Weiteren wurde ein Usability-Problem nicht optimal umgesetzt. So existiert kein expliziter zurück Button. Der Nutzer muss in der Aktuellen Version den Back Button benutzen um immer wieder auf die Startseite oder das vorherige Menü zu kommen.

8 Projektmanagement

Das Projektteam hat sich bei regelmäßigen persönlichen oder telefonischen Treffen abgestimmt und sowohl das weitere Vorgehen als auch die Aufgabenverteilung besprochen und abgestimmt. Der Programmcode war allen Teammitgliedern auf Github zugänglich und wurde regelmäßig geupdatet. Für die Zusammenarbeit an der Projektdokumentation wurde Google Drive genutzt, um flexibel und zeitgleiche Arbeit zu gewährleisten.

Aufgabenteilung

Teilaufgabe	
Planung	Nina Hösl, Jonas Jelinski, Roman Schmid
Layout	Nina Hösl
Timerfunktion	Jonas Jelinski
Datenbankstruktur	Jonas Jelinski, Roman Schmid
Programmstruktur	Nina Hösl, Jonas Jelinski, Roman Schmid
Dokumentation	Nina Hösl, Jonas Jelinski, Roman Schmid

Quellen

[1] Krug, S (2005):Dont' make me think. Web Usability: Das intuitive Web, Pearsons Education

[2]http://greenrobot.org/greendao/ , abgerufen: 24.09.17