Android-Projekt

Projektname:	Converter
Ausführender:	Thomas Lahn, BS INF 2007 (7. FS)
Auftraggeber:	Fernfachhochschule Schweiz (FFHS)
Zeitraum (geplant):	28.08.2010 – 22.01.2011
Stand:	28.01.2011
Version:	0.9.0

Inhalt

- Allgemeines
- Systemanforderungen
  - Allgemeine Anforderungen
  - Detailanforderungen
  - Sicherheitsanforderungen
- Use-Cases
- GUI-Prototyp
- Architektur
- Datenbankanbindung
- Internetanbindung
- Zusammenfassung
  - Implementierte Features
  - Zusätzliche Features
  - Nicht implementierte Features
  - Release-Information / Systemvoraussetzungen
  - Bekannte Fehler
  - Persönliches Fazit

Allgemeines

Converter (zuvor „Android-Umrechner“ [A-UM]) ist ein Projekt, das im Rahmen des Bachelor-Studiengangs Informatik 2007 an der FFHS durchgeführt wird.

Die Arbeit wurde im 3-Mann-Team begonnen. Aufgrund von Differenzen nach Erstellung des GUI-Prototyps wurde jedoch entschieden, dass Holger Schindler und Simon Wyssen als ein Team den Ansatz „A-UM“ weiter verfolgen werden, und Thomas Lahn ab diesem Zeitpunkt einen eigenen Ansatz verfolgen kann. Aus diesem Grund basiert diese Arbeit auf dem Stand des Projekts „A-UM“ zu diesem Zeitpunkt. Seitdem hat es keinen Austausch von Quelltext oder Dokumenten zwischen den beiden Teams mehr gegeben.

Systemanforderungen

Die Systemanforderungen beschreiben und strukturieren die Anforderungen des Systems „Converter“. Sie werden vom Auftraggeber und Auftragnehmer beidseitig als Grundlage für die Realisierung und Abnahme des zukünftigen Systems akzeptiert.

Änderungskontrolle

Version	Datum	Beschreibung, Bemerkung	Name oder Rolle
0.1	30.08.2010	Erstellung Dokument	Holger Schindler, PL
0.2	13.09.2010	Logo hinzu gefügt	Holger Schindler
0.3	13.09.2010	Review	Thomas Lahn

Definitionen

Priorität	Bedeutung
Müssen	Es ist zwingend erforderlich, entweder auf der Basis von Weisungen oder aus Gründen von „Best Practice“. Ausnahmen dürfen keine gemacht werden.
Sollen	Es ist wichtig, dass dies so gemacht wird, es können aber begründete Ausnahmen von der Regel gemacht werden.
Können	Es ist angeraten/vorgeschlagen, etwas wie dargestellt zu tun.
Dürfen	Es ist erlaubt etwas zu tun und verstösst nicht gegen „Müssen“.

Allgemeine Anforderungen

Nr.	Anforderungstext	Quelle	Priorität
2.1	Mehrere Screens	FFHS	Müssen
2.2	Implementierung gemäss Standardandroidprogrammierungsnormen	FFHS	Müssen
2.3	Verwendung mindestens eines Services	FFHS	Müssen
2.4	Nachvollziehbare Dokumentation	FFHS	Müssen
2.5	Dokumentation kleiner 20 Seiten	FFHS	Sollen

Detailanforderungen

Funktionale Anforderungen

Nr.	Anforderungstext	Quelle	Priorität
3.1.1	Hauptmenüseite	Projekt	Müssen
3.1.2	3 Untermenüseiten (Längenmasse, Temperaturen, Wechselkurse)	Projekt	Müssen
3.1.3	Mind. 2 umrechenbare Längenmasse	Projekt	Müssen
3.1.4	Mind. 2 umrechenbare Temperatureinheiten	Projekt	Müssen
3.1.5	Mind. 2 umrechenbare Währungen	Projekt	Müssen

Systemtechnische Anforderungen

Nr.	Anforderungstext	Quelle	Priorität
3.2.1	Abfrage aktueller Wechselkurse aus dem Internet	Projekt	Müssen
3.2.2	Speicherung der Währungseinheiten in SQLite-DB	Projekt	Sollen
3.2.3	Speicherung der umrechnenbaren Längeneinheiten in der SQLite-DB	Projekt	Sollen
3.2.4	Speicherung der umrechenbaren Temperatureinheiten in der SQLite DB	Projekt	Sollen
3.2.5	Speicherung der Umrechnungsformeln in der SQLite-DB	Projekt	Können

Sicherheitsanforderungen

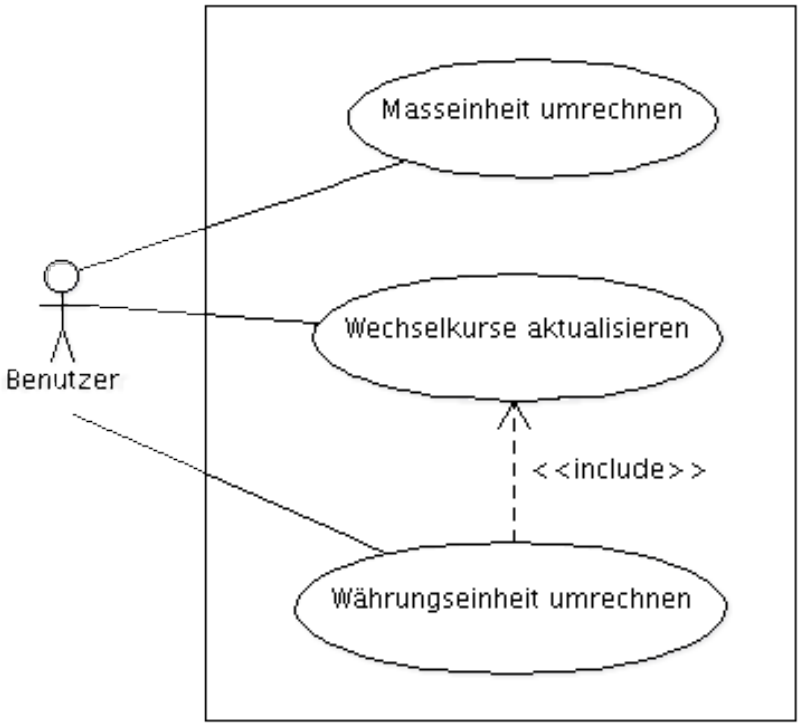
Nr.	Anforderungstext	Quelle	Priorität
4.1	Verwendung eines Zertifikates zur Authentifizierung der Applikation	Projekt	Können

Use-Cases

Änderungskontrolle

Version	Datum	Beschreibung, Bemerkung	Name oder Rolle
0.1	10.09.2010	Erstellung Dokument	Holger Schindler, PL
0.2	13.09.2010	Ergänzung Use-Case-Beschreibungen	Thomas Lahn

Use-Case-Diagramm



Use-Case-Beschreibung

Name	Masseinheit umrechnen
Kurzbeschreibung	Der Benutzer rechnet eine Masseinheit in eine andere um
Akteure	Benutzer
Auslösendes Ereignis	Der Benutzer wählt „Längenmasse“ oder „Temperaturen“ aus dem Hauptmenü
Vorbedingung	Der Benutzer hat die Applikation gestartet
Eingehende Informationen	Einheitentyp (Menüauswahl), Wert (Texteingabe); Einheit (Steuerelementauswahl)
Ablauf (essentielle Schritte)	1. Das Umrechnungsformular wird angezeigt. 2. Der Benutzer gibt in einem Feld des Formulars eine Zahl ein. 3. In den anderen Feldern werden die Zahlen für die umgerechneten Werte in den jeweiligen anderen Einheiten eingetragen.
Ausnahmefälle	• 2a. Der Benutzer gibt einen unzulässigen Wert ein (z.B. negative Länge). • 3a. In den anderen Formularfeldern wird ein Fehlerwert angezeigt. • 3b. Ein Wert ist zu gross oder zu klein, um berechnet oder dargestellt werden zu können: Im entsprechenden Formularfeld wird ein weiterer, von 3a verschiedener Fehlerwert angezeigt.
Nachbedingung	Die umgerechneten Werte oder Fehlerwerte werden angezeigt
Zeitverhalten	Die Berechnung wird nach jedem eingegebenen Zeichen durchgeführt

Name	Währungseinheit umrechnen
Kurzbeschreibung	Der Benutzer rechnet eine Währungseinheit in eine andere um
Akteure	Benutzer
Auslösendes Ereignis	Der Benutzer wählt „Wechselkurse“ aus dem Hauptmenü
Vorbedingung	Der Benutzer hat die Applikation gestartet und zuvor die Wechselkurse aktualisiert
Eingehende Informationen	Wert (Texteingabe); Währungseinheit (Steuerelementauswahl)
Ablauf (essentielle Schritte)	1. Das Umrechnungsformular wird angezeigt. 2. Der Benutzer gibt in einem Feld des Formulars eine Zahl ein. 3. In den anderen Feldern werden die Zahlen für die umgerechneten Werte in den jeweiligen anderen Einheiten eingetragen.
Ausnahmefälle	• 2a. Der Benutzer gibt einen unzulässigen Wert ein (z.B. negative Zahl). • 3a. In den anderen Formularfeldern wird ein Fehlerwert angezeigt. • 3b. Ein Wert ist zu gross oder zu klein, um berechnet oder dargestellt werden zu können: Im entsprechenden Formularfeld wird ein weiterer, von 3a verschiedener Fehlerwert angezeigt.
Nachbedingung	Die umgerechneten Werte oder Fehlerwerte werden angezeigt
Zeitverhalten	Die Berechnung wird nach jedem eingegebenen Zeichen durchgeführt
Verfügbarkeit	Die Wechselkurse wurden zuvor abgerufen

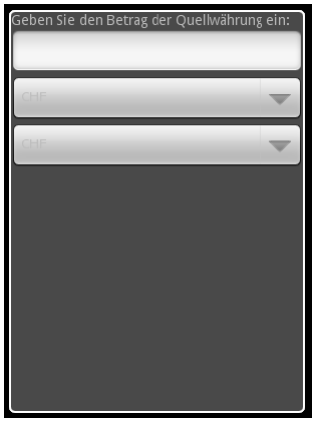
Name	Wechselkurse aktualisieren
Kurzbeschreibung	Der Benutzer aktualisiert die Wechselkurse für die Währungsumrechnung
Akteure	Benutzer
Auslösendes Ereignis	Der Benutzer wählt die Funktion „Wechselkurse aktualisieren“
Vorbedingung	Der Benutzer hat die Applikation gestartet
Eingehende Informationen	Zeitpunkt der Aktualisierung, Wechselkursdaten
Ablauf (essentielle Schritte)	1. Die Wechselkurse werden online abgerufen. 2. Aktualisierung erfolgreich: Zeitpunkt wird gespeichert. 3. Die Aktualisierung wird dem Benutzer bestätigt.
Ausnahmefälle	• 2a. Online-Service nicht verfügbar. • 3a. Fehlermeldung wird angezeigt.
Nachbedingung	Die Wechselkurse sind aktualisiert, das Aktualisierungsdatum gespeichert.
Zeitverhalten	Die laufende Aktualisierung wird abgebrochen, wenn der Benutzer die Applikation beendet.
Verfügbarkeit	Online-Verbindung muss bestehen und Online-Service muss verfügbar sein

GUI-Prototyp

Hauptmenü

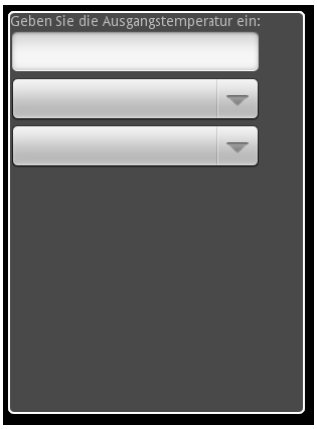
Mit den Buttons kann die entsprechende Funktion gewählt werden.

Währungsumrechner



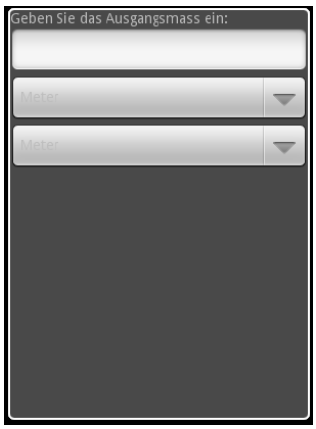
In dem ersten Eingabefeld kann der Betrag eingegeben werden, der umgerechnet werden soll.
In den unteren beiden Auswahlfeldern kann zuerst die Quell-, dann die Zielwährung ausgewählt werden.
Die „Menü“-Taste löst die Umrechnung aus.
Mit der „Zurück“-Taste kehrt man zum Hauptmenü zurück.

Temperaturen umrechnen



In dem ersten Eingabefeld kann die Temperatur eingegeben werden, die umgerechnet werden soll.
In den unteren beiden Auswahlfeldern kann zuerst die Einheit der Quelltemperatur, dann die Einheit der Zieltemperatur eingestellt werden.
Die „Menü“-Taste löst die Umrechnung aus.
Mit der „Zurück“-Taste kehrt man zum Hauptmenü zurück.

Längen umrechnen

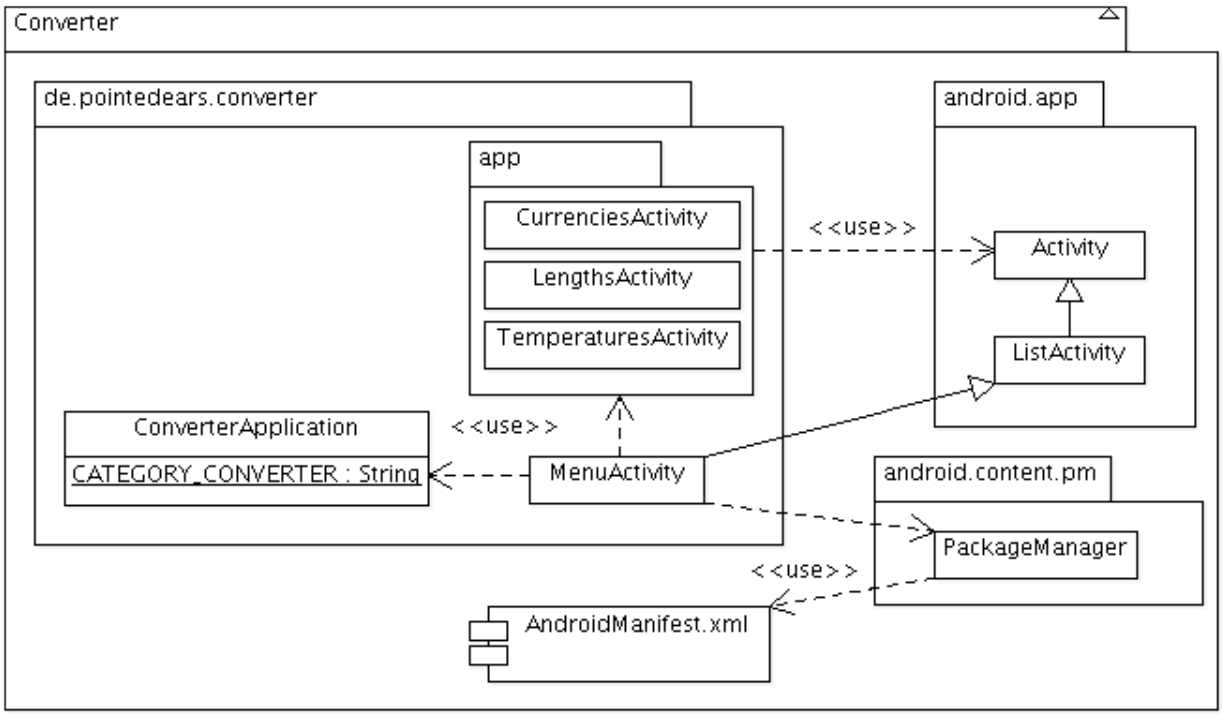


In dem ersten Eingabefeld kann das Ausgangsmass eingegeben werden, das umgerechnet werden soll.
In den unteren beiden Auswahlfeldern kann zuerst die Einheit der Quellmasses, dann die Einheit des Zielmasses eingestellt werden.
Die „Menü“-Taste löst die Umrechnung aus.
Mit der „Zurück“-Taste kehrt man zum Hauptmenü zurück.

Architektur

Die Architektur der Applikation folgt notwendigerweise dem Android-Modell.
Sie erweitert dieses Modell, indem die Standard-Activity (`MenuActivity`) als eine `ListActivity` definiert ist, die zur Laufzeit über einen `PackageManager` (im Prinzip) aus dem Android-Manifest (`AndroidManifest.xml`) die Activities der Kategorie `android.intent.category.CONVERTER` für die einzelnen Funktionen entnimmt, automatisch entsprechende, nach `android:label` aufsteigend sortierte, Listeneinträge hinzufügt und diese mit den passenden Intents belegt.

Dies ist eine bewusste Abweichung vom GUI-Prototyp. Sie soll einer Darstellung dienen, die sich anderen Android-Applikationen stärker angleicht (Usability) und dazu, die Applikation später leichter erweiterbar zu machen: Es müssten dann für neue Menüpunkte (z.B. einen Flächenumrechner), nur neue Activities im Android-Manifest hinzugefügt werden; `MenuActivity.java` könnte unverändert bleiben. (Das träfe auch auf Untermenüpunkte zu; dieser Ansatz wurde aus den SDK-2.2-`AppDemos` übernommen, siehe Quelltextkommentar.)
Dieser Anpassung musste dann auch das im GUI-Prototyp dargestellte grosse Applikationslogo weichen.



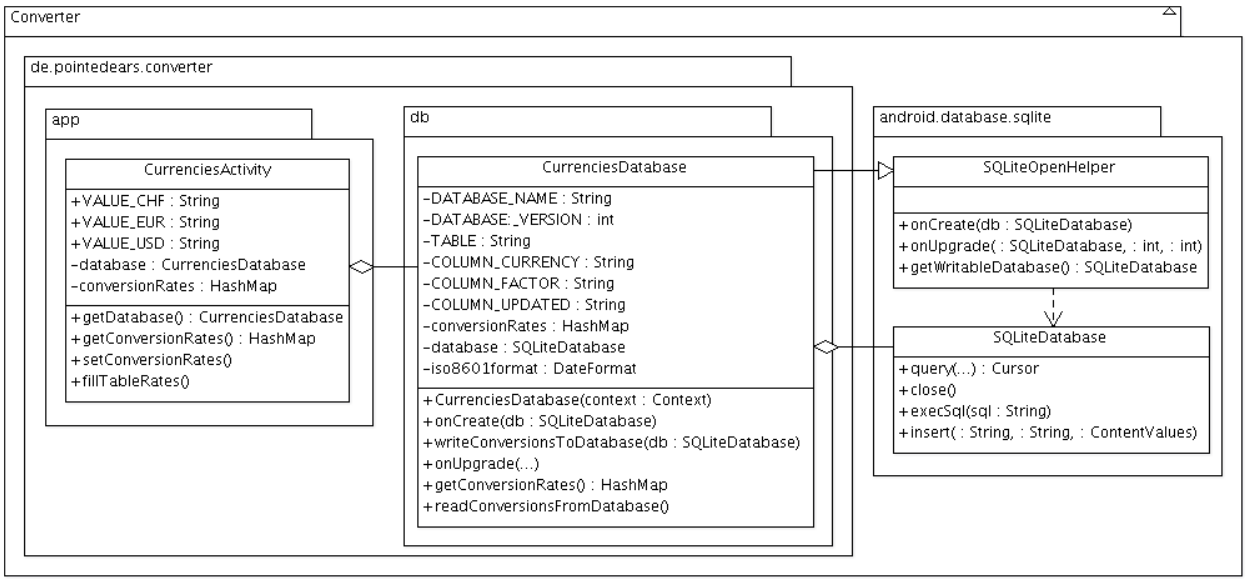
Ausserdem wird beim Start der Applikation der Service `de.pointedears.helpers.UpdateService` gestartet, der bei Erhalten eines Intents mit der Action `"de.pointedears.converter.ACTION_UPDATE"` seinerseits einen Thread startet, der die Daten vom Datenprovider abholt, parst, und Datenbank und GUI aktualisiert. Siehe hierzu die Abschnitte „Datenbankanbindung“ und „Internetanbindung“.

Datenbankanbindung

Für die Speicherung der Umrechnungskurse wird eine SQLite-Datenbank verwendet. Die Datenbankverbindung wird beim Starten der `CurrenciesActivity` erstmalig hergestellt.

Der Datenbankzugriff läuft **absichtlich** nicht in einem eigenen Thread ab, damit die aktuellen Umrechnungskurse zur Verfügung stehen, wenn das GUI angezeigt wird. Intensive Tests im SDK-2.2-Emulator und dem bereitgestellten HTC Wildfire (mit Android 2.2-Update) haben diesbezüglich keine Probleme („Anwendung reagiert nicht“) ergeben. Mit aktuelleren, leistungsfähigeren Geräten werden daher erst recht keine Probleme erwartet.

Existiert die Datenbank noch nicht oder hat sie die falsche Version, wird sie automatisch neu erstellt (bisherige Daten gehen verloren; die nächste Version könnte stattdessen eine automatische Datenmigration ermöglichen) und mit Datensätzen der zur Entwicklungszeit gültigen Wechselkurse gefüllt. Dies geschieht über die Klasse `CurrenciesDatabase`, welche von `SQLiteOpenHelper` abgeleitet ist:



Datenbankstruktur

Die Datenbank enthält nur eine Tabelle namens `currency`, welche in SQLite folgendermassen definiert ist:

```
CREATE TABLE IF NOT EXISTS currency
(currency1 TEXT, factor NUMERIC, updated TEXT,
CONSTRAINT unique_currency_pair UNIQUE (currency1) ON CONFLICT REPLACE)
```

Der Name der Spalte `currency1` ist dem Refactoring geschuldet: zuvor wurde in der Datenbank der Wechselkurs von einer Währung zur anderen gespeichert (`currency1`, `currency2`). Mit Wahl der Europäischen Zentralbank (ECB, siehe Abschnitt „Internetanbindung“) als kostenloser und relativ zuverlässiger Datenprovider, und daraus folgend Umrechnung basierend nur auf dem Euro-Kurs der jeweiligen Währung, ist die Spalte `currency2` jedoch obsolet geworden. Für den Fall, dass diese Umrechnungsart sich nicht als praktikabel/korrekt erweist, wurde der Name der Spalte für die erste Währung jedoch beibehalten.

Die Spalte `factor` speichert den Umrechnungskurs und die Spalte `updated` das Datum der letzten Aktualisierung beim Datenprovider im ISO-8601-Format „JJJJ-MM-TT HH:mm:ss“.

Für eine konsistente Speicherung darf ein Wert in der Spalte `currency1` nur einmal vorkommen (zuvor durfte eine Währungskombination nur einmal vorkommen, daher der Name des Constraints). Mit der Klausel `ON CONFLICT REPLACE` ist es in SQLite möglich, immer mit `SQLiteDatabase.insert(…)` in die Datenbank zu schreiben, wobei der die Kollision verursachende Datensatz automatisch gelöscht und mit dem einzufügenden Wert neu erstellt wird, wenn der Constraint (hier: `UNIQUE`) andernfalls verletzt würde.

Da es hier keine Verknüpfungen zwischen Tabellen gibt und zudem bei der Abfrage der Datenbank derzeit keine Filterbedingung erforderlich ist, wurde kein expliziter Primärschlüssel definiert; der SQLite-Dokumentation ist zu entnehmen, dass SQLite in diesem Fall automatisch einen internen Primärschlüssel mit der Typ-Klasse `INTEGER` erstellt.

Auf die geplante Speicherung von Umrechnungsformeln für Längen- und Temperatureinheiten wurde schliesslich verzichtet, da sich herausgestellt hat, dass diese nicht ohne erhöhten Aufwand (String-Parsing) so gespeichert werden können, dass sie in der Applikation verwendet werden können. Bei Längeneinheiten müssen je nach Operanden unterschiedliche Multiplikations- und Divisionsoperationen durchgeführt werden (v.a. bei der Konvertierung von metrischen nach englischen Einheiten), und bei Temperatureinheiten ist zwischen Grad Celsius und Kelvin (und umgekehrt) Addition, zwischen Grad Celsius und Grad Fahrenheit (und umgekehrt) jedoch zusätzlich Multiplikation erforderlich. Eine spätere Version wird dieses Feature möglicherweise beinhalten können, optimal kombiniert mit der benutzerdefinierten Speicherung der Umrechnungsformel.

Funktionsweise

Lesezugriff

Nachdem die Datenbank wie oben beschrieben ggf. neu erstellt und gefüllt wurde, werden per `SQLiteDatabase.query(…)` alle Datensätze der Tabelle `currency` ausgelesen (`CurrencyDatabase.readConversionsFromDatabase()`) und in der `HashMap` `CurrencyDatabase.conversionRates` gespeichert. Von dort können sie mit `CurrencyDatabase.getConversionRates()` abgerufen werden. (Aus Effizienzgründen werden sie in `CurrenciesActivity.conversionRates` zwischengespeichert, und können mit `CurrenciesActivity.getConversionRates()` abgerufen werden.)

Ab jetzt brauchen keine Datenbankzugriffe mehr stattfinden (Ausnahme: `UpdateService`, siehe unten) und die Datenbankverbindung wird geschlossen. Danach wird o.g. `HashMap` benutzt, um die Anzeige der (GUI-)Tabelle der Umrechnungskurse aufzubauen/zu aktualisieren.

Schreibzugriff

In die Datenbank wird nur geschrieben, wenn sie neu erstellt wird (siehe oben) oder die Währungskurse vom `UpdateService` aktualisiert werden (siehe Abschnitt „Internetanbindung“). In letzterem Fall ruft eine `RatesUpdater`-Instanz die Methode `CurrencyDatabase.writeConversionsToDatabase(…)` auf und übergibt die `HashMap` mit den Umrechnungsdaten, so dass diese per `SQLiteDatabase.insert(…)` geschrieben werden. Danach bleibt die Datenbankverbindung offen, da `writeConversionsToDatabase(…)` über `onUpgrade(…)` und/oder `onCreate(…)` auch indirekt von `readConversionsFromDatabase()` getriggert werden kann.

Internetanbindung

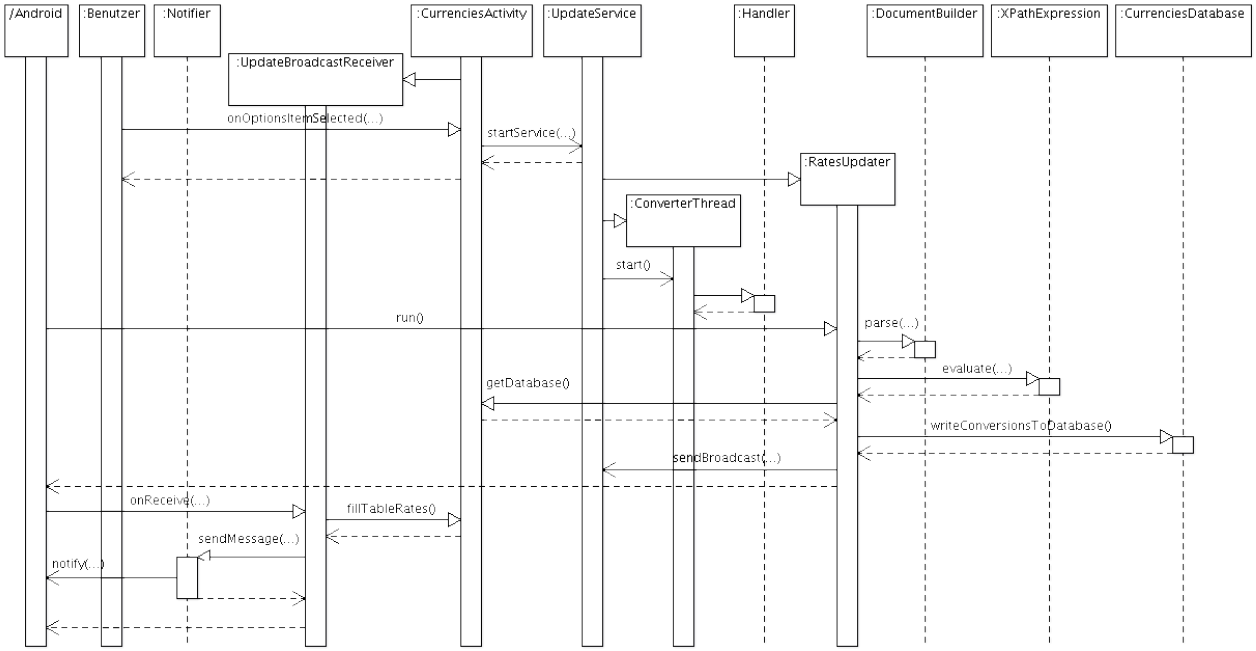
Ein Ziel der Applikation war die Nutzung der Internet-Verbindung des Android-Geräts über WLAN oder GPRS/UMTS, um die verwendeten Umrechnungskurse auf dem neuesten Stand halten zu können. Nach einiger Suche wurde die Europäische Zentralbank als zuverlässigster kostenloser Datenprovider ausgewählt (`http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml`).

Wie im Abschnitt „Datenbankanbindung“ schon angedeutet, brachte das jedoch ein erhebliches Refactoring des bisher erstellten Codes mit sich, da nun nicht mehr der eine Umrechnungsfaktor für zwei Währungen „gleichzeitig“ abgefragt werden konnte, sondern nacheinander für beide Währungen, um die Faktoren von der ersten Fremdwährung zu Euro und von Euro zur zweiten Fremdwährung zu ermitteln.

Nachdem klar war, wie die Umrechnungsdaten aussehen würden, musste eine Möglichkeit gefunden werden, diese möglichst effizient beim Datenprovider abzuholen, und möglichst effizient und mit geringer Fehlerquote zu parsen. Die Entscheidung fiel zugunsten von `DocumentBuilder.parse(URI)` und `XPathExpression.evaluate(…)` aus (in der näheren Auswahl stand auch noch ein `HTTPClient` und das Schreiben eines SAX-basierten Parsers; dies erwies sich jedoch, trotz gutem Tutorial, schnell als zu aufwändig und fehlerträchtig).

Der Ablauf bei der Aktualisierung der Wechselkurse ist folgender (vgl. untenstehendes Diagramm):

Der Benutzer wählt aus dem Optionsmenü den (einzigen) Menüpunkt „Update exchange rates“. Dies startet den `UpdateService` (falls er nicht mehr läuft) mit einem `Intent`, der die `Action "de.pointedears.converter.ACTION_UPDATE"` hat. Das startet einen *`Thread`* (`ConverterThread`), der die *`Runnable`* `RatesUpdater` ausführt. In dieser *`Runnable`* werden dann mit o.g. Methode die XML-Daten per HTTP abgeholt, in eine `Document`-Instanz geparst und mithilfe der `XPathExpression` die Umrechnungsdaten (enthalten in Attributwerten von Elementknoten) in einer `HashMap` gespeichert. Diese `HashMap` wird `CurrenciesDatabase.writeConversionsToDatabase(…)` übergeben, um die Datenbank zu aktualisieren (siehe Abschnitt „Datenbankanbindung“). Danach wird ein Intent broadcasted, der den `UpdateBroadcastReceiver` der `CurrenciesActivity` (der sich zuvor auf diesen `Intent` registriert hat) veranlasst, die (GUI-)Tabelle der Umrechnungskurse zu aktualisieren und über den `Notifier`-Helper eine `Notification` zu senden, die den Benuzter über die erfolgreiche Aktualisierung informiert. (Bisher gibt es noch keine `Notification`, die über einen Fehlschlag informiert.)


/Android
:Benutzer
:Notifier
:UpdateBroadcastReceiver
:CurrenciesActivity
:UpdateService
:Handler
:DocumentBuilder
:XPathExpression
:CurrenciesDatabase
onOptionsItemSelected(...)
startService(...)
:RatesUpdater
:ConverterThread
start()
run()
parse(...)
evaluate(...)
getDatabase()
writeConversionsToDatabase()
sendBroadcast(...)
onReceive(...)
fillTableRates()
sendMessage(...)
notify(...)

Zusammenfassung

Implementierte Features

- Hauptmenü (MUSS)
- 3 Unterseiten (MUSS)
- Verwendung mindestens eines Services: 1 Service (`UpdateService`)
- Mind. 2 umrechenbare Längenmasse (MUSS): 4 Längenmasse
- Mind. 2 umrechenbare Temperatureinheiten (MUSS): 3 Temperatureinheiten
- Mind. 2 umrechenbare Währungen (MUSS): 3 Währungen
- Abfrage aktueller Wechselkurse aus dem Internet (MUSS)
- Speicherung der Währungseinheiten in SQLite-DB (SOLL)

Zusätzliche Features

- Menü-Liste
- Notification

Nicht implementierte Features

- Speicherung der umrechnenbaren Längeneinheiten in der SQLite-DB (SOLL)
- Speicherung der umrechenbaren Temperatureinheiten in der SQLite DB (SOLL)
- Speicherung der Umrechnungsformeln in der SQLite-DB (SOLL)
- Verwendung eines Zertifikates zur Authentifizierung der Applikation (KANN)

Release-Information / Systemvoraussetzungen

Die Applikation benötigt Android 2.2, API Level 8. Sie wurde auf QVGA-Auflösung im Emulator und auf HTC Wildfire getestet.

Bekannte Fehler

Aufgrund des verwendeten flexiblen `TableLayout` kann es passieren, dass ein Eingabefeld aus dem Viewport herausragt. Da die Flexibilität dieses Layouts jedoch auch seine Vorteile bei der Zahleneingabe hat (Eingabefeld immer breit genug), wurde es beibehalten. Bei Tests ist dieser Fall auch nur eingetreten, wenn man die gleiche Einheit mit langer Beschriftung (z.B. 2x „Fahrenheit“) gewählt hat. Es wird in diesem Fall empfohlen, die horizontale Ansicht zu verwenden (Display um 90° drehen).

Persönliches Fazit

Obwohl das Projekt sehr zeitaufwändig war, und die Android-Dokumentation auf der Website von Google bei der Problemlösung nicht immer hilfreich, hat die Arbeit an diesem Projekt Spass gemacht und ich habe erneut ein tieferes Verständnis über Java-Programmierung mit Eclipse im allgemeinen, sowie Programmierung von Embedded Devices im Besonderen erlangt. Sehr wertvoll war für mich die Möglichkeit, die Applikation nicht nur im Emulator testen zu können, sondern auch das HTC Wildfire als Android-basiertes Mobilgerät genauer unter die Lupe nehmen zu können. Ich bin sicher, dass mein nächstes Smartphone eines mit Android werden wird! :)