PR Bachelorarbeit für Informatik und Wirtschaftsinformatik, 188.939

SS 2013

Abschlussbericht zum Projekt:

”code-it”

Context-driven Exploration of the Internet of Things

Name	*Matrikelnummer*	*Studienkennzahl*
Matthias Karan	1027663	033 532
Hansjörg Hofer	1026632	033 532

Die Bachelorarbeit wurde betreut von:

Dipl. -Ing. Wolfgang Spreicer

und

DI(FH) Dr. Wolfgang Reitberger, M.Sc.

Institut für Gestaltungs - und Wirkungsforschung

Argentinierstraße 8

Technische Universität Wien

Projektbeschreibung:

Das Projekt code-it erforscht innovative Anwendungen der Internet of Things Technologie im Consumer-Bereich. Dabei wird ein Hauptaugenmerk auf den Shopping-Kontext gelegt. Durch den Einsatz der NFC-Technologie wird KonsumentInnen ein besserer Überblick über Produktionswege und CO2 Fußabdruck ihrer gekauften Produkte ermöglicht.[1]

Das Projekt code-it untersucht zunächst in einer explorativen Studie mit BenutzerInnen mögliche Applikationen auf deren Sinnhaftigkeit und Akzeptanz. Mögliche Szenarien umfassen eine persuasive und personalisiert Einkaufsunterstzützung (z.B. gesunde Ernährung) oder eine längerfristige Analyse der Einkaufsgewohnheiten. Basierend auf diesen Ergebnissen wird eine Smartphone App entwickelt und mit BenutzerInnen in realen Settings getestet.

Aufgabenstellung:

Unsere Aufgabe war es Prototypen für die Userstudie des Projekts "code-it" zu entwickeln. Einerseits soll eine Smartphone App entwickelt werden, mit der es mittels NFC-Schnittstelle möglich ist, Lebensmittel-Produkte sowie Rechnungen mit dem Handy zu scannen und Informationen über den aktuellen Kalorien-Verbrauch je nach Nahrungs-Kategorie des Users anzuzeigen. Die App steht in direkter Verbindung mit einer zusätzlichen Web-Applikation, mit welcher die Daten stets synchronisiert werden. Die Web-Applikation stellt den zweiten Teil des Prototypen dar und visualisiert den Kalorien Verbrauch der User über längeren Zeitraum. Der User soll in der Applikation persönliche Ziele für die verschiedenen Nahrungs-Kategorien einstellen können. Diagramme und Widgets zeigen dem User inwieweit die Ziele, anhand der gekauften Produkte, eingehalten wurden. Zusätzlich soll es in der Web-Applikation einen Administrator-Bereich geben, mit dem Rechnungen eingegeben werden können, Log-Einträge und Statistiken zum Userverhalten angezeigt werden können.

Die Ernährungspyramide:

Als Vorlage für die Einteilung der Lebensmittel-Produkte in unterschiedliche Kategorien dient die "Österreichische Ernährungspyramide"[2], die vom Österreichischen Bundesministerium für

[1] Homepage des Code-It Projekts, Wolfgang Reitberger
Institut für Gestaltungs- und Wirkungsforschung, URL: http://www.codeit.at (Stand: 05.07.2013)
[2] Die Österreichische Ernährungspyramide, Bundesministerium für Gesundheit Österreich, URL: http://bmg.gv.at/home/Schwerpunkte/Ernaehrung/Empfehlungen/DIE_OeSTERREICHISCHE_ERNAeHRUN

Gesundheit empfohlen wird. Die Ernährungspyramide teilt Lebensmittel in insgesamt sieben Kategorien, die sich in sechs Lebensmittelgruppen und eine Getränkegruppe unterteilen[3]. Die Produkte werden daher konkret den folgenden Kategorien zugeordnet: *"Fettes, Süßes und Salziges" - "Fette und Öle" - "Fisch,Fleisch, Wurst und Eier" - "Milch und Milchprodukte" - "Getreide und Erdäpfel" - "Gemüse, Hülsenfrüchte und Obst" - "Alkohofreie Getränke"*

Fddb Datenbank

Um Informationen über Lebensmittel-Produkte zu erhalten wurde die Fddb-Datenbank verwendet. Die Abkürzung Fddb steht für "Fooddatabase" und umfasst mittlerweile Daten zu 195350 Produkten.[4] Die Produkte werden von Usern selbst eingetragen und von Fddb überprüft.
Die Homepage www.fddb.info soll den Usern als Hilfe für gesunde Ernährung dienen.
Für Entwickler bietet fddb eine eigene API[5]. Diese ermöglicht es uns über eine REST-Schnittstelle Produktsuchen anhand von Produkt-EANs durchzuführen und entsprechende Informationen zu erhalten und zu speichern. Die Produktabfragen über die fddb-API finden serverseitig statt.

Arbeitsstruktur:

Unser Projektteam besteht aus 2 Entwicklern. Zu Beginn des Projekts waren die Rollen gleich aufgeteilt und beide Entwickler haben hauptsächlich am Entwurf für die Datenbank sowie an der Android App gearbeitet. Da sich die Arbeit jedoch schon nach kurzer Zeit in 2 große Teilbereiche, einerseits Entwicklung der Android App und andererseits Entwicklung der Web-Applikation, sinnvoll einteilen ließ, haben wir uns jeweils verstärkt auf einen Teilbereich konzentriert.
Rollenverteilungen:
- Verantwortlicher für Entwicklung der Android-App
- Verantwortlicher für Entwicklung der Web-Applikation

Die Arbeitsteilung war jedoch nie strikt aufgeteilt und beide Entwickler haben auch in den anderen Bereichen gearbeitet beziehungsweise mitgeholfen. Die Zusammenarbeit im Team hat sehr gut funktioniert. Meist wurde gemeinsam beziehungsweise zeitgleich implementiert. Die Kommunikation fand meist über Skype, aber auch über persönliche Treffen statt.

GSPYRAMIDE (Stand: 06.07.2013)
[3] Die Ernährungspyramide im Detail, Bundesministerium für Gesundheit Österreich, URL: http://bmg.gv.at/home/Schwerpunkte/Ernaehrung/Empfehlungen/Die_Ern%C3%A4hrungspyramide_im_Detail_-_7_Stufen_zur_Gesundheit (Stand: 06.07.2013)
[4] Info-Homepage der fddb-Datenbank, www.fddb.info (Stand: 06.07.2013)
[5] Fddb API Documentation, URL: http://fddb.info/api/v8/documentation/ (Stand: 17.09.2013)

Die Zusammenarbeit mit unseren Betreuern Herrn Wolfgang Spreicer und Herrn Wolfgang Reitberger funktionierte ebenfalls sehr gut. Seit dem Kickoff-Treffen am 6.März gab es alle 2-3 Wochen persönliche Treffen im Büro. Bei den Treffen haben wir wir als Entwickler laufend unsere Arbeitsfortschritte präsentiert. Gemeinsam wurden neue Arbeitsschritte und Meilensteine festgelegt, Projektbezogene Unklarheiten diskutiert und technische sowie design-spezifische Fragen geklärt. Ansonsten fand die Kommunikation zwischen Betreuern und Entwicklern hauptsächlich über e-Mail statt.

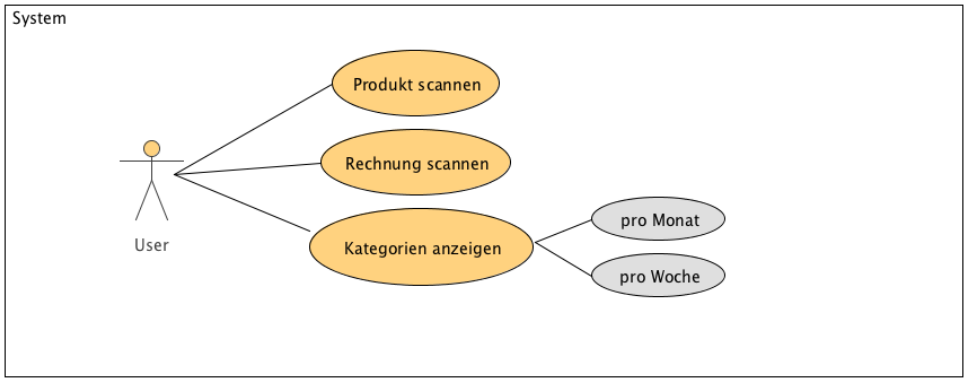
Implementierung:

Implementierung der Android-App

Für die “In-situ” Unterstützung des code-it Projekts war es die Aufgabe, eine native Android-App zu entwickeln die am Smartphone Einkaufsdaten in einer Datenbank speichert und unmittelbares, personalisiertes Feedback beim Einscannen von Produkten gibt.

Zur Implementierung wurden uns 2 Android Smartphones sowie mehrere RFID-Tags (Mifare Classic 1K, 13.56 MHz) von unseren Betreuern zur Verfügung gestellt.
Als Entwicklungsumgebung wurde Eclipse Java EE (Version: Juno Service Release 2) mit installiertem Android SDK (Android Version: 4.2.2) verwendet.
Die App wurde hauptsächlich mit dem Samsung Nexus S (Android Version: 4.1.2) getestet.

Anwendungsfälle Android-App:



Anwendungsfalldiagramm: Android-App

0. User Login:
Beim allersten Start der App gelangt der User zum Login-Screen. Nach Eingabe von e-Mail und Passwort gelangt der User zur Kategorien-Ansicht. Der Login ist nur einmal nach Installieren der App beziehungsweise falls die Datenbank gelöscht werden sollte, erforderlich.

1. Produkt scannen:
Mit dem Prototyp können Lebensmittel anhand von RFID-Tags mit dem NFC-fähigen Handy gescannt werden.
1.1 *RFID-Tag auslesen:*

Auf den RFID-Tags befindet sich pro Produkt jeweils lediglich eine maximal bis zu 13-stellige eindeutige EAN-Nummer [6]. Diese wird von der App ausgelesen und auf Gültigkeit überprüft.
Die Produkt EAN wird als Text auf die Tags mit folgender Syntax geschrieben:
"#Product# EAN". Zum Beispiel "#Product# 0123456789123". (ohne Anführungsstriche eingeben!).
Um Tags zu beschreiben empfehlen wir die Applikation "Tagstand Writer"[7] die im Google Play Store kostenlos verfügbar ist. Ist die App installiert kann man im Menü unter "Additional Tag Types" → "Text" → "Text eingeben" → "Write Tag" Tags sehr einfach beschreiben.

1.2 *EAN vearbeiten:*

Die EAN-Nummer wird daraufhin mit der lokalen Datenbank am Handy verglichen. Ist ein Produkt mit der EAN bereits in der Datenbank so wird automatisch zur entsprechend Kategorie gewechselt und das Produkt mit einem kleinen Pop-Up angezeigt. Befindet sich kein Produkt mit der EAN in der lokalen Datenbank, wird die EAN an den Server geschickt. Dieser führt, sofern nicht bereits in der Server-Datenbank gespeichert, eine Produktsuche über die fddb-API durch und schickt, sofern gefunden, Informationen über das Produkt (z.b. kcal) an die App zurück und das Produkt wird angezeigt.

1.3 *Kalorienverbrauch visualisieren:*

Nachdem ein Produkt erfolgreich gescannt wurde, erhält der User direkt Feedback darüber, wie sich dieses Produkt beim Kauf beziehungsweise Verzehr auf die "Ziele" der jeweiligen Ernährungs-Kategorien auswirken würde. Dazu wird der zusätzliche Kalorien-Anteil im entsprechenden Kategoriebild blinkend hervorgehoben. Falls der zusätzliche Kalorie-Anteil das festgelegte Ziel überschreiten würde, blinkt ein "roter Rahmen" rund um das Kategoriebild, der verdeutlichen soll, dass sich dieses Produkt schlecht auf die Ziele des Users auswirken würde.

2. Rechnung scannen:

Mit dem Prototyp können ähnlich wie zuvor bei Produkten auch Rechnungen (die mehrere Produkte enthalten) anhand von RFID-Tags gescannt werden. Die RFID-Tags sollen den Kauf zum Beispiel an einer Supermarkt-Kassa nur simulieren und haben daher keinerlei Bezug zu einem realen Bezahlvorgang.

1.1 *RFID-Tag auslesen:*

Auf den RFID-Tags befinden sich pro Rechnung jeweils mehrere Produkte (EAN-Nummer) sowie der Preis pro Produkt. Die Rechnung wird von der App ausgelesen und auf Gültigkeit überprüft.
Rechnungen werden als Text auf die Tags mit folgender Syntax geschrieben:

---

[6] European Article Number, Wikipedia, URL: http://de.wikipedia.org/wiki/European_Article_Number (Stand: 06.07.2013)

[7] Produktseite der Applikation Tagstand Writer, Android Play Store, URL: https://play.google.com/store/apps/details?id=com.tagstand.writer&hl=de (Stand: 17.09.2013)

“#Bill# Anzahl;EAN1;Preis”.
Zum Beispiel:
“#Bill#
2;0123456789123;2,99;
3;0123456789456;1,49;”
(wiederum ohne Anführungsstriche eingeben!).

1.2 *Rechnung anzeigen:*

Zunächst wird die Rechnung mittels Pop-Up Nachricht am Handy angezeigt. Anzahl, EAN sowie Preis der einzelnen Produkte werden angezeigt. Zusätzlich wird der Gesamtpreis der Rechnung berechnet und angezeigt.

1.3 *Rechnung verarbeiten:*

Jedes Produkt wird daraufhin wieder mit der lokalen Datenbank am Handy verglichen bzw. falls nicht vorhanden an den Server geschickt. Sofern beim scannen eine Internetverbindung besteht, wird die Rechnung an den Server geschickt. Die Kalorien der jeweiligen Produkte werden zu den Kategorien aufsummiert und die Füllstände der Kategoriebilder aktualisiert.
Besteht keine Internetverbindung beim scannen, so wird die Rechnung vorerst lokal gespeichert, und bei nächster Möglichkeit, wenn wieder Internetverbindung besteht (oder alternativ bei Drücken des Sync-Buttons) an den Server geschickt.

3. Kategorien anzeigen

3.1 *Kategoriebilder*

Jede der sieben Ernährungs-Kategorien wird anhand eines Bildes dargestellt. Je nachdem wie weit sich der User dem festgelegten Ziel nähert, wird das Bild farblich gefüllt. Befindet sich der User innerhalb der festgelegten Kalorien Grenze, wird ein grüner Rahmen um das Kategorie Bild angezeigt. Ist der User bereits über der Grenze, so wird die Kategorie mit einem roten Rahmen dargestellt.

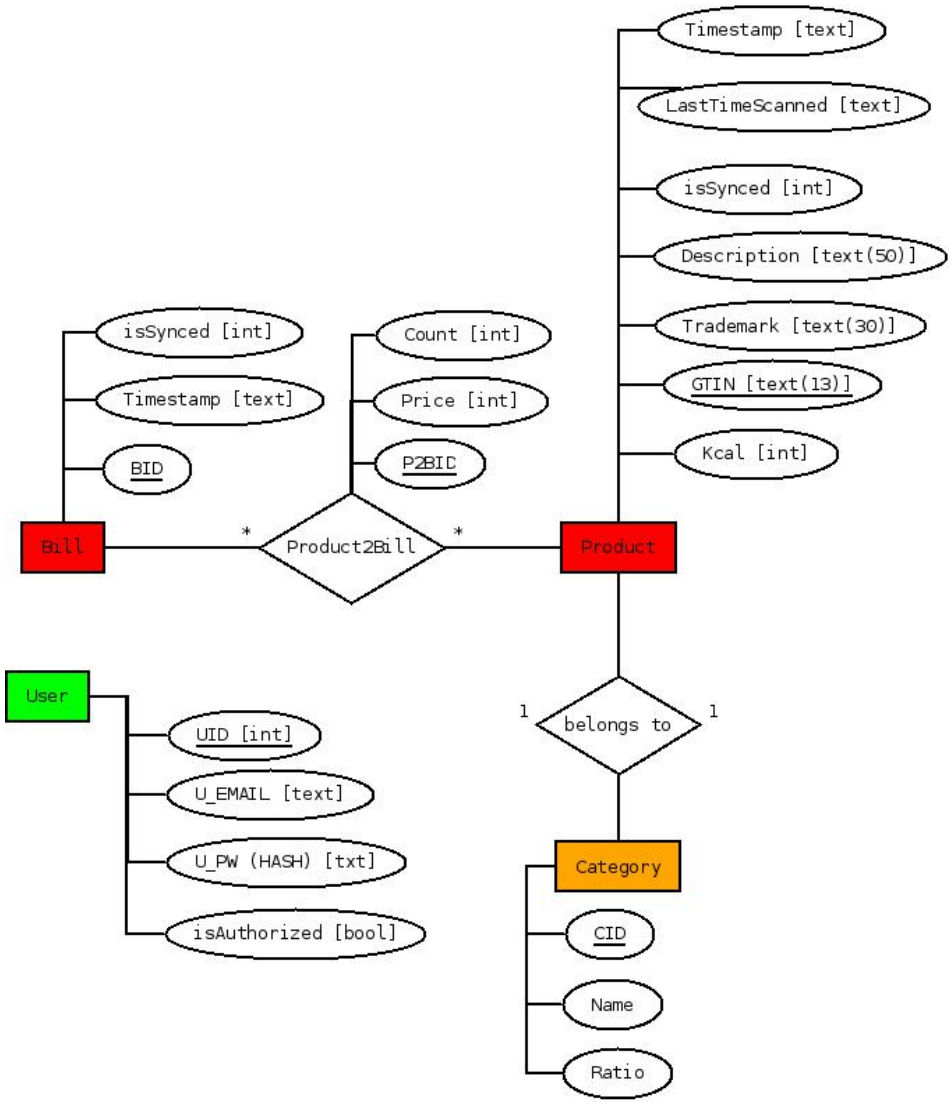
3.2 *Zwischen Kategorien wechseln*

Um zwischen den sieben Kategorien hin und her zu wechseln kann der User ganz einfach über den Bildschirm “swipen” (daher von links nach rechts oder umgekehrt) und sich nach rechts oder links durch die Kategorien bewegen.

3.3 *Woche/Monat*

Mit den Buttons “Woche” und “Monat” kann der User festlegen für welchen Zeitraum die Ziele angezeigt werden.

Datenbankmodell Android-App:



ER-Diagramm: Android-App

Die Daten des Users werden mit dem von Android empfohlenem “Content Provider” gekapselt. Content Provider sind die Standard Schnittstelle für strukturierte Daten in Android. [8] Die darunterliegende Datenbank basiert auf der SQLite3 Bibliothek[9].

[8] Developer Guidelines for ContentProvider, Android Developers, URL: http://developer.android.com/guide/topics/providers/content-providers.html (Stand: 06.07.2013)
[9] Homepage der SQLite Datenbank Library, SQLite, URL: http://www.sqlite.org/ (Stand: 06.07.2013)

Implementierung der Web-Applikation:

Um die gesammelten Einkaufsdaten auch längerfristig für die User zu veranschaulichen, war es die weitere Aufgabe, eine Web-Applikation zu entwickeln, welche die Daten anhand von Diagrammen und Widgets visualisiert, sowie die Möglichkeit bietet, gesundheitliche Ziele je nach belieben festlegen zu können.
Zusätzlich soll es einen Administrator-Bereich geben, mit dem User verwaltet werden können, Rechnungen eingetragen werden können und Log-Daten zum Userverhalten angezeigt werden können.

Für die Entwicklung der Web-Application wurde als IDE hauptsächlich Netbeans 7.3 verwendet. Als Lokale Testumgebung wurde MAMP 2.1.3 mit PHP 5.4, MySQL 4.0.1 und dem Webserver Apache genutzt. Nach dem zweiten Treffen mit den Betreuern wurde uns zusätzlich ein SSH Zugang zum Institutsserver eingerichtet. Das Release-System verwendet abgesehen vom Webserver Ngingx, die selben Spezifikationen.
Als Testgeräte für die Userstudie wurden zwei verschiedene Tablets verwendet: iPad2, Nexus 10.

Anwendungsfälle Web-Applikation:

Anwendungsfalldiagramm: Web-Applikation

0. User Login:

Um sich in die Web-Applikation einzuloggen muss der User lediglich seinen Namen aus einer Drop-Down List auswählen und bestätigen. Dies wurde eingeführt um den Test-Usern einen leichten wiedereinstieg in die Web-App zu ermöglichen falls sie versehentlich ausgeloggt wurden. Der Administrator kann mithilfe eines Links zum regulären Login wechseln. Hier wird eine Standard Authentifizierung mit Benutzername und Passwort verwendet um sich anzumelden. Da es sich bei der Web-Application um einen reinen Prototypen für Userstudies handelt wurde die Sicherheit sehr niedrig priorisiert.

1. Diagramme anzeigen:

Direkt nach dem Login werden dem User anhand eines Diagrammes die Ziele dem aktuellen Kalorienverbrauch gegenüber gestellt. Die farbigen Balken zeigen dabei an wie weit man vom jeweiligen Ziel der Kategorie entfernt ist. Um die Kategorien leichter mit den Farben des Diagramms assoziieren zu können wird im Hintergrund das jeweilige Symbol der österreichischen

Ernährungpyramide verwendet.
1.1: Zusätzlich hat der User die Möglichkeit sich den aktuellen Stand des persönliches Diagramms zu drucken beziehungsweise zu exportieren.
1.2: Mit Klick auf den Button “Ansicht wechseln” kann der User zwischen zwei verschiedenen Darstellungen des Diagramms wechseln. Dabei handelt es sich um ein Donut-Diagramm und eine Kombination von Torten- und Strahlendiagramm.

2. Ziele einstellen:
Bei Klick auf den Button “Optionen” erscheint für jede Kategorie ein Feld um das Verhältnis zu vergrößern oder zu verkleinern. Zusätzlich kann der User auch die jeweilige Kategorie “sperren”. Dadurch verändert sich das Verhältnis der gesperrten Kategorien nicht mehr und das erstellen einer individuellen Ernährungpyramide fällt leichter.

3. Rechnungen eintragen:
Im Administrator-Bereich unter dem Menüeintrag “Rechnungen” ist es möglich mehrere Rechnungen für alle User einzutragen. Dazu kann ein Datum, sowie ein zugehöriger User eingegeben werden. Daraufhin wird eine neue Rechnung erzeugt und einzelne Produkte können der Rechnung durch Eintragen des EAN-Codes, der Menge, sowie des Preis pro Stück hinzugefügt werden. Die eingegeben EAN werden zunächst mit der lokalen Datenbank verglichen und falls nicht bereits vorhanden in mit der Fddb-Produktsuche gesucht. Einzelne Produkte können vor der Speicherung auch wieder gelöscht werden. Mit Klick auf “Speichern” wird die Rechnung hinzugefügt.

4. Rechnungen anzeigen:
Im Administrator-Bereich unter dem Menüeintrag “Rechnungen” werden alle eingetragenen sowie mit der Handy-App gescannte Rechnung pro User angezeigt. Einzelne Rechnung können auch gelöscht werden.

5. Produkt suchen
Im Administrator-Bereich unter dem Menüeintrag “Produkte” können einzelne Produkte anhand von EAN-Code, Produktname oder Beschreibung gesucht werden. Die Produktsuche läuft über die Fddb-API und zeigt maximal 10 gefundene Produkte an, die nacheinander angezeigt werden und direkt gespeichert werden können.

6. Produkt hinzufügen
Im Menüpunkt “Produkt hinzufügen” kann der Adminstrator neue Produkte speichern. Dazu muss ein eindeutiger EAN-Code, Marke, Beschreibung, Menge bzw. Einheit und eine Kategorie zugewiesen

werden.
Gefundene Produkte aus dem Menüpunkt “Produkt suchen” können ebenfalls direkt gespeichert werden.

7. Produkte anzeigen

Alle gespeicherten Produkte werden in einer übersichtlichen Tabelle angezeigt und können auch gelöscht werden.

8. Benutzer verwalten

Im Menüpunkt “Benutzer” kann der Administrator alle User verwalten.

8.1 User hinzufügen

Mit Eingabe von User-Name, E-Mail Adresse und Passwort können neue User angelegt werden.

8.2 Username ändern

Der Administrator kann durch anwählen des entsprechenden Users, ganz einfach den Benutzername beliebig ändern.

8.3 Benutzer anzeigen

In einer übersichtlichen Tabelle werden alle Benutzer mit zugehörigen Daten (außer Passwort) angezeigt.

8.4 Tägliche Kcal berechnen

Für jeden Benutzer kann der Administrator den täglichen Kilokalorienverbrauch berechnen.
Die Formel für die Berechnung funktioniert wie folgt [10]:

Männer:

$$66,47 + 13,7 * Gewicht[kg] + 5 * Körpergröße\,[cm] - 6,8 * Alter[Jahre]$$

Frauen:

$$655,1 + 9,6 * Körpergewicht[kg] + 1,8 * Körpergröße\,[cm] - 4,7 * Alter[Jahre]$$

9. Widgets anzeigen

Die Pyramide:

Mit Klick auf das Widget-Symbol rechts oben in der Applikation, wird eine Pyramide sowie ein Smiley angezeigt. Die Pyramide soll längerfristig verdeutlichen, wie weit der User eine persönlich festgelegten Ziele einhält. Je weiter sich der Smiley in Richtung der Spitze der Pyramide bewegt, desto mehr nähert sich der User seinem Ziel. Zur Berechnung werden die Kalorien aller eingetragenen Rechnungen pro Kategorie verwendet. Ist der Wert *x* gleich 0 so ist das Verhältnis zwischen den Kalorien der Rechnung und der Ernährungspyramide perfekt. Der Smiley ist somit an der Spitze der Pyramide. Je größer der Wert wird desto weiter entfernt sich der Smiley von der Spitze

[10] Kalorien Grundumsatz pro Tag berechnen, Stefan Kaiser - www.gesumag.de, URL:http://www.gesumag.de/kalorien-grundumsatz-pro-tag-60/ (Stand: 17.09.2013)

der Pyramide.

$$x = \sum_{i=0}^{6} \left| 1 - \frac{\frac{Kalorien_i}{KalorienGesamt}}{\frac{Ernährungspyramide_i}{ErnährungspyramideGesamt}} \right|$$

Der Smiley:
Der Smiley soll dem User verdeutlichen, ob sich der letzte eingetragene Einkauf positiv oder negativ auf die Ziele auswirken. Lacht der Smiley, so haben sich die Einkäufe positiv ausgewirkt. Ist der Smiley traurig, so hatten die Produkte negativen Einfluss. Der neutrale Smiley soll zeigen, dass seit der letzten Rechnung zu viel Zeit vergangen ist um eine Emotion anzuzeigen.

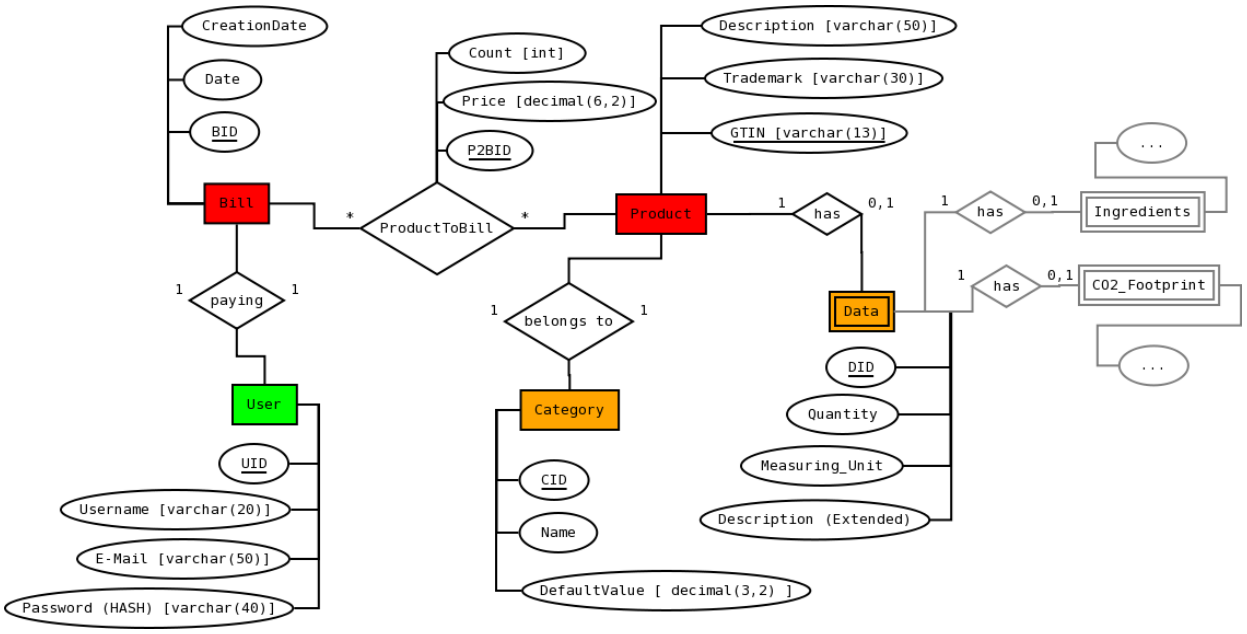
10. Statistik generieren

Für die erleichterte wissenschaftliche Analyse der Userstudies, können Statistiken zu den einzelnen Usern generiert werden. Diese beinhalten alle Kalorienwerte im Verhältnis zueinander zum Zeitpunkt der neuen Rechnung.

11. Logs anzeigen

Um das Verhalten der Test-User untersuchen zu können wird jeder Ansichtwechsel und jede Veränderung der Ernährungspyramide protokolliert. So kann später nachvollzogen werden welches Diagramm oder welche Ansicht von den Test-Usern bevorzugt wurde.

Datenbankmodell Web-Applikation:



ER-Diagramm:Web-Applikation

Bibliotheken:

Android App:

Zusätzlich zum Android 4.2.2 Paket wurden folgende erweiternde Bibliotheken verwendet:

- Jake Wharton's View Page Indicator [11]
- Jake Wharton's Action Bar Sherlock [12]

Web-Applikation:

Für die Implementierung der Grundfunktionalität (CRUD, Authentifizierung) wurde kein PHP-Framework verwendet, stattdessen wurde eine einfache Architektur mit einer Business und Service Schicht verwendet.

Die Visualisierung basiert auf JavaScript und verwenden die umfangreichen Bibliotheken JQuery[13] und HighCharts[14]. Letztere ist eine reine Chart-Library und für nicht-kommerzielle Zwecke kostenlos.

[11] Homepage der "ViewPagerIndicator" Library von Jake Wharton, Jake Wharton, URL: http://jakewharton.com/viewpagerindicator/ (Stand: 06.07.2013)
[12] Homepage der "ActionBarSherlock Library" von Jake Wharton, Jake Wharton, URL: http://jakewharton.com/actionbarsherlock/ (Stand: 06.07.2013)
[13] Homepage der JQuery JavaScript Library, URL: http://jquery.com/ (Stand: 17.09.13)
[14] Homepage der HighCharts JavaScript Chart-Library, URL: http://www.highcharts.com/ (Stand: 17.09.13)

Known Bugs:

- Die Android App läuft am Testgerät (Samsung Nexus S) einwandfrei. Andere Smartphones mit NFC-Unterstützung wurden nicht getestet, daher kann es hier möglicherweise zu Problemen kommen.
- RFID-Tags vom Typ “Mifare Classic” funktionieren einwandfrei. Andere RFID-Tags wurden nicht getestet.

Besonderheiten der Implementierung:

- Unterstützung der NFC Technologie
- Direktes Feedback zu Produkten durch Kategoriebilder mit Animationen
- Synchronisierung zwischen Smartphone-App und Web-Applikation
- Übersichtliche Darstellung des Einkaufverhalten durch interaktive Diagramme der Web-Applikation
- Einbindung der Fddb-API zur Suche von Produkten
- Web-Applikation wurde für Tablets und Touchsteuerung optimiert
- Selbständiges Synchronisieren der Diagramm-Daten.
- Anzeige von Log-Einträgen für Administratoren um Userverhalten einfach zu analysieren

Potenzial für Weiterentwicklungen:

- Der Bezahlvorgang mit NFC am Handy wurde für den Prototyp nur simuliert. Bei tatsächlichem Einsatz der App könnte der Bezahlvorgang tatsächlich über die NFC-Technologie umgesetzt werden.
- Um die App nicht nur von der NFC-Technologie abhängig zu machen, wäre es möglich zusätzlich auch das Scannen von Produkten über einen “Barcode Scanner” zu ermöglichen.
- Zusätzlich zu den gesundheitlichen Aspekten könnten auch finanzielle, oder beispielsweise CO2-Fußabdrücke abbilden um das Einkaufsverhalten der User noch weiter zu verbessern.