|  |  |  |
| --- | --- | --- |
| Pflichtenheft:  EinkaufsApp | | |
| Hochschule für Telekommunikation Leipzig Frau Dr. Sabine Wieland Gustav-Freytag-Str. 43-45, 04277 Leipzig | |  |
| Bezug:  Semestergruppe: Abgabedatum: | Softwareengineering DWI 13-2 04.01.2016 | |

Informationen zum Pflichtenheft

|  |  |
| --- | --- |
| An | Frau Prof. Dr. Wieland  Gustav-Freytag-Str. 43-45, 04277 Leipzig |
| Über | Projekt EinkaufsApp |
| Von | EinkaufsApp |

|  |  |
| --- | --- |
| Vorgelegt von |  |
| Projektleiter  Gruppenmitglieder | Markus Hube  Huong Dang  Thomas Elias  Viktor Fuchs  Florian Graupeter  Jannis Grohs  Michael Hein  Moritz Karsten  Sebastian Kiepsch  Annika Köstler  Daniel Sawadenko  Moritz Schaub  Florian Schmitt  Eric Sorgalla |
| E-Mail | markus.hube@hft-leipzig.de |

Inhaltsverzeichnis

[1 Einleitung 6](#_Toc437800046)

[2 Problembeschreibung 7](#_Toc437800047)

[3 Funktionalitäten der App 8](#_Toc437800048)

[3.1 Einkauf 8](#_Toc437800049)

[3.1.1 Muss-Kriterien 8](#_Toc437800050)

[3.1.2 Wunsch-Kriterien 8](#_Toc437800051)

[3.1.3 Abgrenzungskriterien 8](#_Toc437800052)

[3.2 Markt 8](#_Toc437800053)

[3.2.1 Muss-Kriterien 8](#_Toc437800054)

[3.2.2 Wunsch-Kriterien 9](#_Toc437800055)

[3.2.3 Abgrenzungskriterien 9](#_Toc437800056)

[3.3 Nutzerverwaltung 9](#_Toc437800057)

[3.3.1 Muss-Kriterien 9](#_Toc437800058)

[3.3.2 Wunsch-Kriterien 9](#_Toc437800059)

[3.3.3 Abgrenzungskriterien 9](#_Toc437800060)

[3.4 Auswertung 10](#_Toc437800061)

[3.4.1 Muss-Kriterien 10](#_Toc437800062)

[3.4.2 Wunsch-Kriterien 10](#_Toc437800063)

[3.4.3 Abgrenzungskriterien 10](#_Toc437800064)

[4 Anwendungsbereiche 11](#_Toc437800065)

[5 Systemarchitektur 12](#_Toc437800066)

[5.1 Backend 12](#_Toc437800067)

[6 Use Cases 14](#_Toc437800068)

Abbildungsverzeichnis

Abbildung 1 Aufbau der Systemarchitektur 11

Abbildung 2 Aufbau des Backend 11

# Einleitung

Die EinkaufsApp dient dem Nutzer seine alltäglichen Einkaufserlebnisse, hinsichtlich der besuchten Geschäfte und gekauften Produkte zu dokumentieren und

eine Übersicht über seine Finanzen darzustellen. Gleichzeitig soll diese App als

Nachschlagewerk fungieren, welches einen Überblick über Preis und Angebot bestimmter Produkte bietet. Der alltägliche Einkauf wird hinsichtlich des Monitoring

der Finanzen und Produktauswahl aufgrund der Funktionalitäten der EinkaufsApp erleichtert.

# Problembeschreibung

Mit der immer weiter steigenden Angebotsvielfalt von Produkten steigt das Kaufinteresse des Konsumenten bzw. steigt auch mit dem immer weiter wachsenden Kaufinteresse des Konsumenten das Angebot unterschiedlicher Anbieter. Dadurch gestaltet sich das Nachverfolgen vergangener Einkäufe immer schwieriger.

Dieses ist vor allem notwendig um zukünftige Einkäufe planen zu können um somit im Endeffekt nachfolgende Ausgaben einzuschränken.

Der damit einhergehende zeitliche Aufwand bei einem manuellen Monitoring ist

im heutigen digitalen Zeitalter nicht notwendig. Mehr als 50 Prozent der Einwohner Deutschlands[[1]](#footnote-1) besitzen ein Smartphone und haben somit die Möglichkeiten via Apps z. B. Einkäufe zu tracken.

Die im Pflichtenheft vorgestellte EinkaufsApp soll, wie auch schon in der Einleitung beschrieben, dem Nutzer die Möglichkeit bieten seine Einkäufe nachzuverfolgen und schlussendlich unterschiedliche Optionen hinsichtlich der Auswertung zu bieten.

Nicht nur der eigene Einkauf kann durch die EinkaufsApp verwaltet werden sondern auch Gruppeneinkäufe, was bedeutet, dass nach einem Einkauf, für z. B. eine Wohngemeinschaft, die einzelnen Gruppenmitglieder die jeweiligen Artikel zugewiesen bekommen.

# Funktionalitäten der App

Dieses Kapitel beinhaltet die geplanten Funktionen der Applikation. Die Unterteilung erfolgt in die Hauptteile: Einkauf, Auswertung und Nutzerverwaltung. Es werden jeweils pro Kategorie die Muss-Kriterien, also die Funktionen die implementiert werden müssen, Wunsch-Kriterien, also die Funktionen die nachdem die Muss-Kriterien umgesetzt wurden implementiert werden und den Abgrenzungs-Kriterien, die aus Kapazitätsgründen nicht umgesetzt werden, aber in Zukunft umgesetzt werden können.

## Einkauf

### Muss-Kriterien

Grundlegend hat die App die Funktion der Erstellung von Einkaufslisten. Hierzu gehören auch die Artikelaufnahme in diese Liste sowie die nachfolgende Bearbeitung dieser Liste bei Änderungsbedarf des Nutzers. Die Artikel werden über einen integrierten Barcodescanner mittels der European Article Number (EAN) in die Liste aufgenommen.

Zudem können Informationen über die gekauften Artikel angezeigt werden.

Zum Schluss kann der komplette Einkauf gespeichert werden. Der Einkauf wird dann abgeschlossen sobald der Nutzer dies auch bestätigt.

### Wunsch-Kriterien

Zusätzlich soll die App die Funktion des Preisvergleichs von Artikeln

in unterschiedlichen Märkten besitzen.

### Abgrenzungskriterien

Die Applikation kann Einkaufslisten basierend auf alten Einkäufen generieren können.

## Markt

### Muss-Kriterien

Bevor der Einkaufsprozess gestartet wird, soll die App den Marktstandort finden. Sobald der Nutzer vor einem Markt steht, ermittelt die App via GPS seinen Standort. Falls der Markt nicht gefunden wird, gibt es die Möglichkeit einen neuen Markt hinzuzufügen. Der Nutzer gibt dazu die Daten des neuen Marktes an.

### Wunsch-Kriterien

Es werden keine weiteren Wunsch-Kriterien implementiert.

### Abgrenzungskriterien

Ein Markt kann über die App von einem Nutzer z. B. durch ein 5-Sterne

Bewertungssystem bewertet werden. Diese Bewertungen können über soziale Medien, wie z:B. Facebook geteilt werden.

## Nutzerverwaltung

### Muss-Kriterien

Der Nutzer kann die App erst nutzen, wenn dieser ein Benutzerprofil erstellt hat. Das Profil besteht grundlegend aus Namen, E-Mailadresse und

einem Passwort. Die von ihm getätigten Einkäufe sind dann eindeutig

zuordenbar. Dies gilt auch für die Gruppenverwaltung, die Gruppeneinkäufe mit anschließender Artikelzuweisung zum jeweiligen Gruppenmitglied.

Ein Artikel kann einem Gruppenmitglied erst zugewiesen werden, wenn ein Benutzerprofil desjenigen bereits in der Datenbank im System besteht.

### Wunsch-Kriterien

Es werden keine weiteren Wunsch-Kriterien implementiert.

### Abgrenzungskriterien

Der Nutzer kann sich auch mit seinem Facebook-Profil oder via Twitter bei der App anmelden. Zudem können noch nicht registrierte Leute durch „Dummys“ ersetzt und im Nachhinein angemeldet werden.

Angemeldete Mitglieder können über Push-Notifications über den Stand der Gruppeneinkäufe benachrichtigt werden.

## Auswertung

### Muss-Kriterien

Der Nutzer kann vergangene Einkäufe auswerten lassen.

Folgende Unterscheidungen werden gemacht:

a) Kosten pro Zeitraum

b) Kaufhäufigkeit eines Artikels und die dazugehörigen Gesamtkosten

c) Kosten pro Artikelkategorie

d) monetäre Ausgaben je Käufergruppe

### Wunsch-Kriterien

Die Auswertung erfolgt nicht nur über eine reguläre Tabelle, sondern kann als Diagramm graphisch dargestellt werden.

### Abgrenzungskriterien

Personenunabhängig können Aussagen über die beliebtesten Artikel, die beliebtesten Märkte und die Durchschnittspreise eines Artikels wiedergegeben gemacht werden.

# Anwendungsbereiche

Die EinkaufsApp wird vorerst lediglich auf allen Android-fähigen Endgeräten laufen können. Eine Implementierung auf iOS-basierte Endgeräte wird nach erfolgreichem Testen der Applikation in Angriff genommen. Da es sich hierbei um eine Hybrid-App handelt, kann die Implementierung ohne großen Aufwand umgesetzt werden. Ein passendes Framework ist dementsprechend dafür vorgesehen.

Eine Internetverbindung bei aktiver Nutzung muss bestehen. Eine Offline-Nutzung ist vorerst nicht vorgesehen.

Mit der angebotenen Lösung werden hauptsächlich Privatnutzer angesprochen. Für Geschäftskunden müssten weitere Funktionalitäten implementiert werden.

# Systemarchitektur

Um die oben genannten Funktionalitäten der Applikation umzusetzen, müssen die genutzten Tools und Methoden klar definiert werden. Im weiteren Verlauf wird zunächst die generelle Struktur der Systemarchitektur beschrieben und dann auf die einzelnen Begrifflichkeiten eingegangen.

Der generelle Aufbau des Softwaresystems sieht folgendermaßen aus.

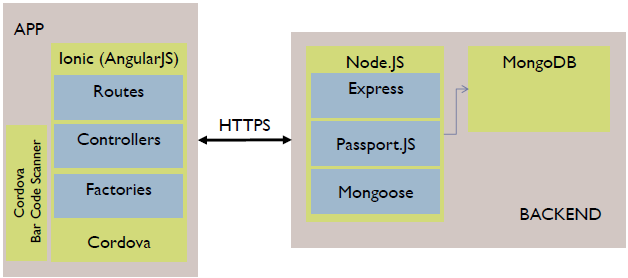


Abbildung 1

Da die gesamte Systemlogik im Backend liegt, müssen Server und App kommunizieren, was via Hypertext Transfer Protocol geschieht.

## Backend

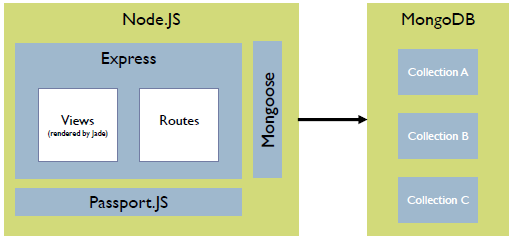


Abbildung 2

Express ist das Webframework für Node.JS, einem Tool, welches JavaScript als Serversprache ausführt, um die Serverlogik abzubilden. Dies wird benötigt um wiederum die Webseite zu bauen. Die Views werden mittels Jade, dem Rendering Engine via HTML generiert. Zur Authentifizierung des Nutzers beim Login und bei der Registrierung gibt es Passport.JS, welches ein Modul für die Authentifizierung ist. Über Mongoose, einem Objektmodellierungstool für Node.JS

Der OpenShift Server ist, im ersten Preismodell, eine kostenfreie Möglichkeit hinsichtlich der Entwicklung des Backends einer App. Es handelt sich hierbei um eine Cloudlösung, die eine schnelle Umsetzung ermöglicht, da keine eigentlichen Server aufgestellt werden müssen. Es wird das HTTPS Zertifikat genutzt und es fallen keine weiteren Konfigurationen an.

Über das Portal können die Cartridges direkt über das Webinterface verwaltet werden, anders als bei einem Rootserver wo dies über die Commandozeile geschieht. Cartidges sind z. B. die MongoDB, die für die EinkaufsApp genutzte Datenbank, NodeJS, und Express, einem Framework für die Webentwicklung auf Basis von NodeJS um Routes zu erstellen, die mit der API zusammenhängt und eine Schnittstelle zum Rendering Engin Jade bereitstellt. Jade generiert in dem Zusammenhang automatisch HTML-Code für Modell-View-Controller.

Im Prozess der Entwicklung ist eine Verknüpfung zu Git möglich, einem Versionsmanagement-Tool. Werden über das Kommando „git push“ Änderungen aktualisiert, werden die Programme auf dem Server automatisch gestartet.

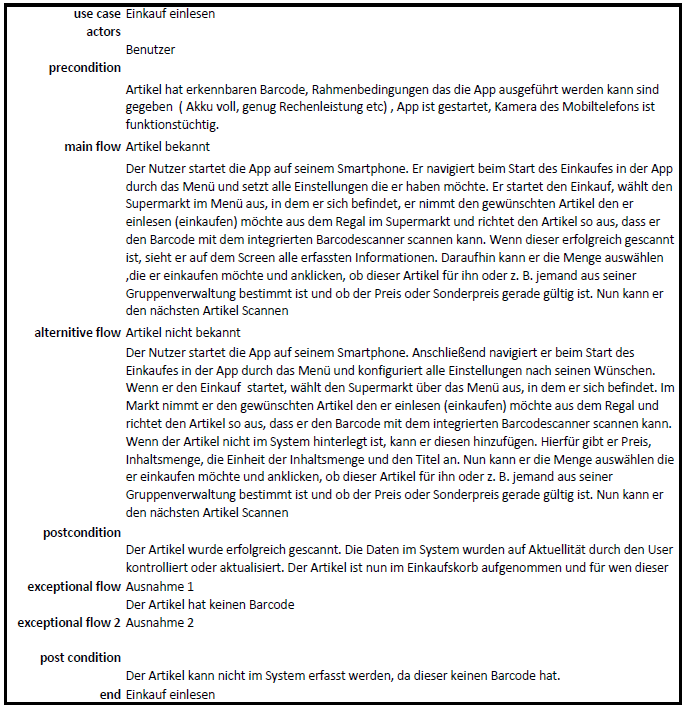
Wie schon erwähnt handelt es sich bei der EinkaufsApp um eine Hybrid-App. Aspekte einer nativen und Web-App sind hierbei vereint. Die Applikation kann auf diese Art und Weise unproblematisch, sowohl für iOS Betriebssysteme, wie auch Android basierte Operation Systems deployed werden. Das bedeutet im Wesentlichen, dass dadurch der Nutzerbereich ausgeprägter ist, als bei einer nativen App. Zukünftige Änderungen sind zudem leichter durchzuführen, da die App-Logik auf dem Server liegt.

Nachteil einer Hybrid-App sind die eingeschränkten Anwendungsbereiche, im Gegenzug zu einer nativen App. Die Kapazitäten eines Betriebssystems werden nicht vollständig ausgelastet, sodass einige Features, zum Beispiel der verbesserten Bedienbarkeit der App, nicht genutzt werden können.

Nichtdestotrotz bietet die Hybrid-App vor allem die Plattformunabhängigkeit, was bedeutet, dass keine weiteren Entwickler benötigt werden um die Software auf unterschiedliche OS zu implementieren, was im Endeffekt kostensparend ist.

# Use Cases

Der hier aufgeführte Use Case beschreibt den Anwendungsfall „Einkauf einlesen“.



1. <http://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010/>, zuletzt abgerufen 09.12.2015, 11:37 Uhr [↑](#footnote-ref-1)