**„EinkaufsApp“**

**Pflichtenheft**

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 5](#_Toc438636609)

[2 Problembeschreibung 6](#_Toc438636610)

[3 Funktionalitäten der App 7](#_Toc438636611)

[3.1 Einkauf 7](#_Toc438636612)

[3.1.1 Muss-Kriterien 7](#_Toc438636613)

[3.1.2 Wunsch-Kriterien 7](#_Toc438636614)

[3.1.3 Abgrenzungskriterien 7](#_Toc438636615)

[3.2 Markt 8](#_Toc438636616)

[3.2.1 Muss-Kriterien 8](#_Toc438636617)

[3.2.2 Wunsch-Kriterien 8](#_Toc438636618)

[3.2.3 Abgrenzungskriterien 8](#_Toc438636619)

[3.3 Nutzerverwaltung 8](#_Toc438636620)

[3.3.1 Muss-Kriterien 8](#_Toc438636621)

[3.3.2 Wunsch-Kriterien 8](#_Toc438636622)

[3.3.3 Abgrenzungskriterien 9](#_Toc438636623)

[3.4 Auswertung 9](#_Toc438636624)

[3.4.1 Muss-Kriterien 9](#_Toc438636625)

[3.4.2 Wunsch-Kriterien 9](#_Toc438636626)

[3.4.3 Abgrenzungskriterien 9](#_Toc438636627)

[4 Anwendungsbereiche 10](#_Toc438636628)

[5 Systemarchitektur 11](#_Toc438636629)

[5.1 Backend 12](#_Toc438636630)

[6 Use Cases 14](#_Toc438636631)

[7 Quellen 15](#_Toc438636632)

Abbildungsverzeichnis

Abbildung 1: Systemarchitektur 11

Abbildung 2: Aufbau des Backend 12

Abkürzungsverzeichnis

|  |  |
| --- | --- |
| Abb.  PHP  HTML  HfTL  DWI13  JS  CSS  App  EAN  OS  iOS  HTTP  HTTPS  Jade | Abbildung  Hypertext Preprocessor  Hypertext Markup Language  Hochschule für Telekommunikation Leipzig  Duales Studium Wirtschaftsinformatik Jahrgang 2013  JavaScript  Cascading Style Sheets  Anwendungsprogramm  European Article Number  Operating System  iPhone Operating System  Hypertext Transfer Protocol  Hypertext Transfer Protocol Secure  A Rendering Engine |

# Einleitung

Die EinkaufsApp dient dem Nutzer dazu, seine alltäglichen Einkaufserlebnisse, hinsichtlich der besuchten Geschäfte und gekauften Produkte sowie deren Preis zu dokumentieren und eine Übersicht über seine Finanzen darzustellen. Des Weiteren soll diese App als Nachschlagewerk genutzt werden, um einen Überblick über Preise und Angebote bestimmter Produkte bei bestimmten Märkten darzustellen. Der alltägliche Einkauf wird mittels eines Produkt- und Finanzmonitorings durch die App erleichtert.

# Problembeschreibung

Aufgrund der steigenden Angebotsvielfalt von Produkten und Dienstleistungen und dem daraus resultierenden wachsenden Kaufinteresses der Konsumenten, gestaltet sich das Nachverfolgen vergangener Einkäufe immer schwieriger und komplexer.

Diese Analysen sind vor allem wichtig, um zukünftige Einkäufe besser planen und sein Ausgabeverhalten besser beschränken zu können.

Der damit einhergehende zeitliche Aufwand durch eine manuelle Analyse ist

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

Die im Pflichtenheft vorgestellte EinkaufsApp soll dem Nutzer die Möglichkeit bieten seine Einkäufe aufzuzeichnen, sie nachzuverfolgen und schlussendlich durch unterschiedliche Auswertungsoptionen zu analysieren und zu optimieren.

Dabei liegt der Fokus darauf, dass nicht nur der eigene Einkauf verwaltet werden kann, sondern auch Gruppeneinkäufe. Was wiederum bedeutet, dass im Laufe eines Einkaufs, für z. B. eine Wohngemeinschaft, die einzelnen Gruppenmitglieder ihren jeweiligen Artikel direkt zugewiesen bekommen können.

# Funktionalitäten der App

In diesem Kapitel wird der Lösungsansatz zur Umsetzung der Applikation beschrieben. Hierbei gibt es die Unterscheidung in Muss-, Wunsch- und Abgrenzungskriterien.

## Muss-Kriterien

Die EinkaufsApp soll es ermöglichen eine Einkaufsliste zu generieren. Der Nutzer kann die gekauften Artikel einpflegen und anschließend unter einem selbst gewählten Namen speichern. Dafür muss es zusätzlich die Funktion „Marktauswahl“ geben, damit dieser den getätigten Einkauf einem Markt zuordnen kann. Die eingepflegten Artikelpreise können von Markt zu Markt variieren, sodass der Artikel von dem jeweiligen Markt gespeichert wird. Handelt es sich um ein Sonderangebot, kann dies vermerkt werden.

Außerdem kann der Nutzer Gruppen erstellen und verwalten um somit für z. B. eine WG einkaufen zu können. Die gekauften Artikel können den jeweiligen Gruppenmitgliedern zugeordnet werden.

Die dadurch gesammelten Daten werden je nach Bedarf ausgewertet.

Für die Auswertung soll es folgende Optionen geben:

1. Auswertung nach Kaufhäufigkeit eines Artikels in einem gewählten Zeitraum
2. Auswertung der getätigten Einkäufe in einem gewählten Zeitraum
3. Auswertung von Gruppeneinkäufen

Um die EinkaufsApp nutzen können braucht der Nutzer einen Useraccount, welcher aus Username, E-Mailadresse und Passwort besteht. Mit diesen Userdaten kann sich dieser bei der Einkaufsapp anmelden. Die getätigten Einkäufe können ihm somit genau zugeordnet werden. Wird das Passwort vergessen, kann er dies zurücksetzen.

Möchte dieser die Applikation nicht mehr nutzen, kann er sich zum Schluss über einen Log-Out Button abmelden.

## Wunsch-Kriterien

Sind die oben genannten Kriterien implementiert und laufen fehlerfrei, können weitere Funktionen in Angriff genommen werden. Beispielsweise kann der Nutzer über eine Rankingfunktion einen Markt bewerten. Diese ist für alle anderen Nutzer sichtbar.

Des Weiteren kann ein Nutzer bei einer Gruppenmitgliedschaft per Push-Notification über für ihn getätigte Einkäufe informiert werden. Dafür gibt es auch eine „Disable“-Funktionalität, falls dieser das nicht wünscht.

Generell soll in Zukunft die EinkaufsApp mit Social Media, wie Twitter oder Facebook kombiniert werden können. Dadurch soll es auch Märkten möglich sein den Nutzer über Angebote zu benachrichtigen.

Hauptsächlich soll nachdem die vergangenen Einkäufe getätigt wurden eine automatische Auswertungsfunktion Einkaufslisten nach dem Kaufhäufigkeits- und Preissparenprinzip generiert werden.

## Abgrenzungskriterien

Die Applikation soll keine Onlineeinkaufsfunktion ermöglichen. Zudem kann der Nutzer nicht über diese Einkäufe bezahlen.

# 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 durchgeführt. 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, wie der Erweiterung der Marktauswahl sowie einer erweiterten Auswertungsfunktion, die ggf. die Ausgaben auswertet als auch eine Umsatzrechnung implementiert hat. Interessant wäre zudem eine Kaufverhaltensanalyse der Kunden.

# 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.

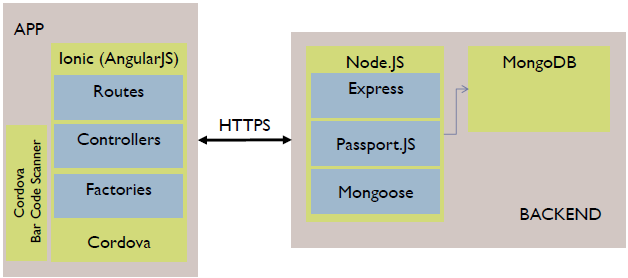


Abbildung 1: Systemarchitektur

Da die gesamte Systemlogik im Backend liegt, müssen Server und App miteinander kommunizieren. Dies geschieht über das Hypertext Transfer Protocol Secure.

Seitens des Frontends gibt es das AngularJS Framework, welches mit HTML, CSS und Javascript Komponenten zur Entwicklung von Hybridapps genutzt wird. In diesem werden die Views, also die visuelle Darstellung der App, die Controller, die die Funktionalitäten im View festlegen und den Factories, die eine Verbindung zu den in der API definierten Endpunkte stellt, verwaltet. Die Routes bilden dabei die Verknüpfung zwischen Controller und View.

Um die Funktion „Artikel hinzufügen“ im Einkaufsprozess zu realisieren wird zusätzlich über CordovanG ein Barcodescanner implementiert, welcher direkt für AngularJS ein Plugin bietet.

## Backend

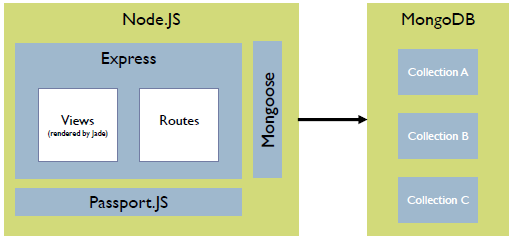


Abbildung 2: Aufbau des Backend

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 darzustellen. Die Views werden mittels Jade, einer Rendering Engine, via HTML generiert. Zur Authentifizierung des Nutzers beim Login und bei der Registrierung wird das Modul Passport.JS genutzt. Über Mongoose, einem Objektmodellierungsmodul für Node.JS, wird die Datenbankanbindung sowie die damit verbundene Businesslogik verwaltet.

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 eigenen Server aufgestellt werden müssen. Es wird das HTTPS Zertifikat genutzt wodurch keine weiteren Konfigurationen diesbezüglich anfallen.

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 den Modell-View-Controller.

Im Prozess der Entwicklung ist eine Verknüpfung zu Git möglich, einem Versionsmanagement-Tool. Wird über das Kommando „git push“ eine Änderung signalisiert, werden die Anwendungen auf dem Server automatisch gestoppt, die Änderung wird verteilt und die Anwendungen wieder gestartet.

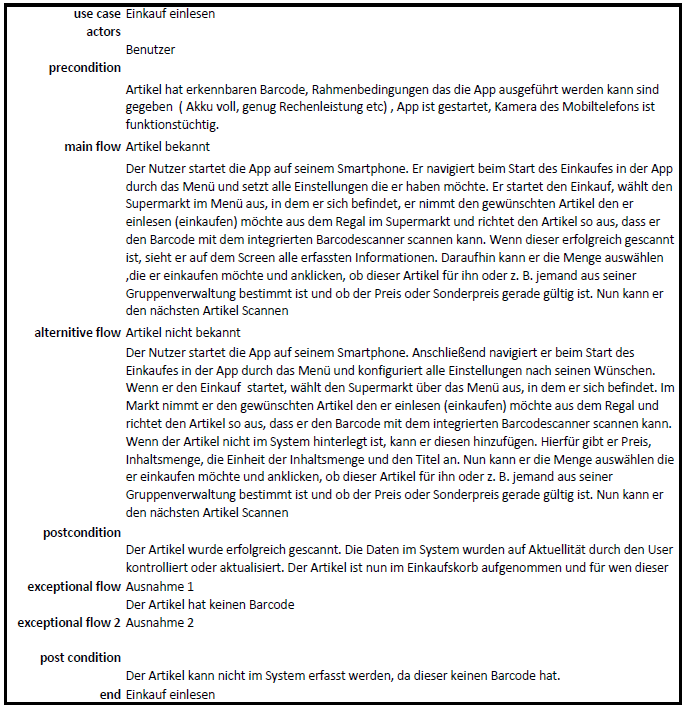
Wie schon erwähnt handelt es sich bei der EinkaufsApp um eine Hybrid-App. Aspekte einer nativen und Web-App sind hierbei vereint. Die Applikation kann auf diese Art und Weise ohne Probleme, sowohl für iOS Betriebssysteme, als auch für Android basierte Operation Systems genutzt werden. Das bedeutet im Wesentlichen, dass dadurch der Nutzerbereich erweitert 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. Dies bedeutet im konkreten, dass die Kapazitäten eines Betriebssystems nicht vollständig ausgelastet werden können, sodass einige Features, zum Beispiel die 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 kostensparender ist.

# Use Cases

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



# Quellen

Zuletzt geprüft am 19.12.2015

* Openshift Features- https://www.openshift.com/features/
* Ionic - http://ionicframework.com/
* NodeJS - https://nodejs.org/en/
* Why MongoDB - https://www.mongodb.com/blog/post/why-mongodb-popular
* AngularJS - https://de.wikipedia.org/wiki/AngularJS

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)