„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	Markus Hube
Gruppenmitglieder	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

Abbildungsverzeichnis

Abkürzungsverzeichnis

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

[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

Funktionalitäten der App

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

3.1 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 User-Account, 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.

3.2 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, welche für alle anderen Nutzer sichtbar ist.

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 Preissparprinzip generiert werden.

3.3 Abgrenzungskriterien

Die Applikation soll keine Onlineeinkaufsfunktion ermöglichen und der Nutzer soll nicht über diese Einkäufe bezahlen können.

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.

Kommentiert [A1]: Bitte noch folgende Begriffe beschreiben: Routes, Controller, Factories, Cordiva, AnhularJS,

Er meint Cordova und AngularJS

Der generelle Aufbau des Softwaresystems sieht folgendermaßen aus.

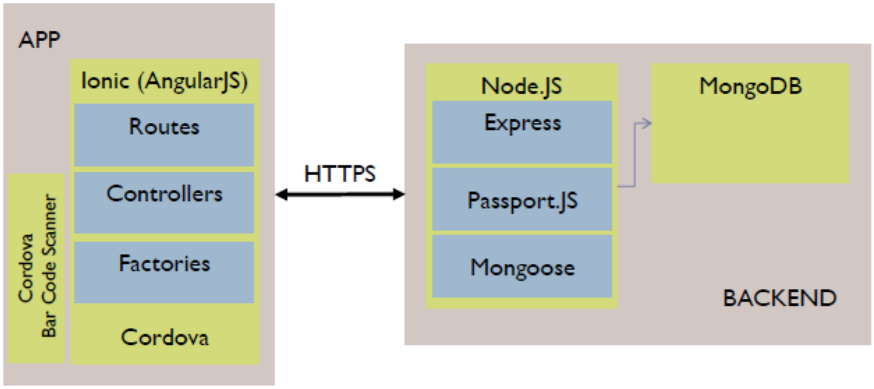

Abbildung **Fehler! Es wurde keine Folge festgelegt.**: Systemarchitektur

Wie in Abbildung 1 ersichtlich, besteht das System aus 2 Komponenten: der eigentlichen App auf Basis des Ionic Frameworks sowie ein Backend inklusive einer Datenbank.

Beide Komponenten orientieren sich an einer Model – View – Controller Architektur. Zwischen den Komponenten findet eine Kommunikation mit HTTPS statt.

In beiden Komponenten wurden existierende Frameworks zur Entwicklung von mobilen, hybriden Apps mit Javascript bzw. der Implementierung von Webservern auf Basis von Node.JS verwendet. Mit dieser Sprachenwahl wurde ein hohes Augenmerk auf die Betriebssystemneutralität der entstehenden Anwendungen gelegt.

Insbesondere Node.JS eignet sich durch seine Skalierbarkeit und den von Haus aus geringen Overhead besonders gut für Anwendungen, welche wenig auf betriebssystemspezifische Ressourcen zugreifen müssen, sowie hochperformante Webapplikationen. Da Node.JS single-threaded ausgeführt wird, eignet es sich besonders gut für Input-Output lastige Anwendungen wie z.B. WebAPIs mit geringem Rechenaufwand.

In der App Komponente wurde ein Framework (Ionic Framwork) gewählt, welches eine große Palette an vorgefertigten Design Elementen mitliefert und damit den Entwicklungsaufwand erheblich verringert. Das Ionic Framework liefert neben diesen Elementen aber auch ein logisch strukturiertes Programmiermodell, indem es die Komponenten von AngularJS (Routing, Controllers, Factories sowie Services) verwendet. Das Basis Framework, welches die Verbindung zu den Ressourcen des mobilen Betriebssystems (iOS oder Android) herstellt ermöglicht in dieser App insbesondere das Hinzufügen von Plugins.

5.1 Backend

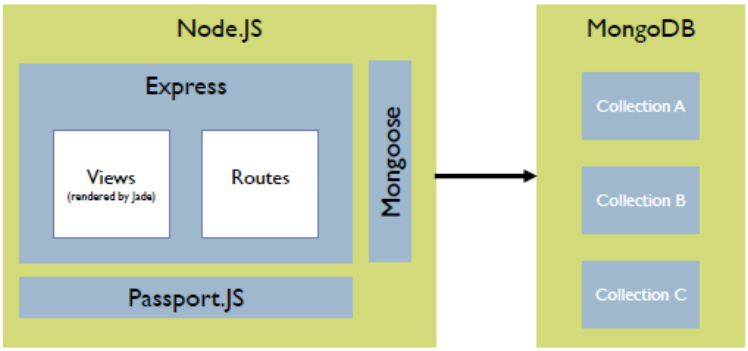

Abbildung **Fehler! Es wurde keine Folge festgelegt.**: Aufbau des Backend

Als Backend wird in diesem Projekt jeglicher Serverseitig ausgeführter Code sowie alle dort befindlichen Dritt-Applikationen bezeichnet. Konkret gliedert sich das Backend in 2 grobe Komponenten auf.

Die Node.JS basierte Webapplikation stellt mithilfe des Express.JS Webframeworks jene Endpunkte bereit, welche durch die mobile Applikation konsumiert werden können. Routen definieren Funktionen, welche bei einer http Anfrage an einen definierten Endpunkt ausgeführt werden. In der im Rahmen dieses Projektes erstellten API ist diese Funktion zumeist das Ausführen einer Datenbankoperation und das anschließende Antworten des Servers auf die Anfrage. Die Antwort erfolgt dabei immer im Form eines JSON Arrays, welches dann vom Empfänger interpretiert werden kann. Um das Produkt zu präsentieren sowie eine Verwaltung von Login und Registrierung über ein Webinterface außerhalb der mobilen App verfügbar zu machen, wurden Views implementiert. Views sind Ansichten, welche mithilfe der Rendering Engine „Jade“ dynamisch generierte Seiten im HTML Format als Antwort vom Server zum Client senden können.

Um die Endpunkte der Anwendung vor unbefugten Zugriff zu schützen wird Passport.JS verwendet. Dieses Node.JS Modul ermöglicht es eine Authentifizierung an der API durchzuführen, welche auf verschiedene Authentifizierungsprovider zurückgreifen kann. In diesem Projekt wurde die lokale Benutzerauthentifizierung auf Basis von in einer Datenbank gespeicherten Credentials gewählt. Die Authentifizierung wird mittels eines temporären Browser-Cookies auf dem Client persistiert.

Weiter oben angesprochene Datenbankoperationen werden mithilfe von Mongoose, eines Daten-Modelling-Tools für Node.JS und MongoDB, durchgeführt. Zuvor wurden in Mongoose Modelle definiert, welche bei Ausführung der Applikation automatisch dem Modell entsprechende Collections in der Datenbank anlegt. Diese Modelle sind in den Routen der Applikationen zugreifbar und können zur Manipulation der Daten in der Datenbank verwendet werden. In den meisten Fällen wurden alle CRUD (Create – Read – Update – Delete) Funktionen als Endpunkte abgebildet (siehe API Dokumentation).

Eine MongoDB als NoSQL Datenbank bildet zweite Komponente des Backends ab. Die besondere Datenmodellflexibilität ermöglicht eine stark iterativ fokussierte Entwicklungsweise, was dem engen Entwicklungszeitfenster in diesem Projekt zu Gute kam. Auch die hohe Skalierbarkeit und die einfache Implementation mittels des Tools Mongoose waren ausschlaggebende Gründe für die Wahl dieser Datenbank.

Um diese Anwendung mit Internet zu hosten wurde OpenShift ausgewählt. Die Cloudplattform von Red Hat bietet im ersten Preismodell eine kostenfreie Möglichkeit Serveranwendungen zu hosten.

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.

Kommentiert [A2]: und weiter??

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.

Kommentiert [A3]: Bitte verständlich erklären.

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.

5.2 App

In der App Komponente findet das Programmierparadigma MVC in der Umsetzung des Angular JS Frameworks Anwendung, welches die Softwarebasis für die Elemente von Cordova und somit des Ionic Frameworks ist. Im Code wird hierbei zunächst mit sogenannten „Factories“ eine klassenähnliche Struktur geschaffen, welche es ermöglicht mit den definierten Endpunkten des Backends zu kommunizieren. Routen legen die App-internen Endpunkte fest. Diesen definierten Endpunkten werden konkrete Funktionen und Ansichten über Controller zugewiesen. Der Controller ist in dieser Position das Bindeglied zwischen den Datenmodellen und dem View. Die Adaption dieser Begrifflichkeiten auf das MVC-Modell wird in Abbildung 2 zur Verdeutlichung aufgezeigt.

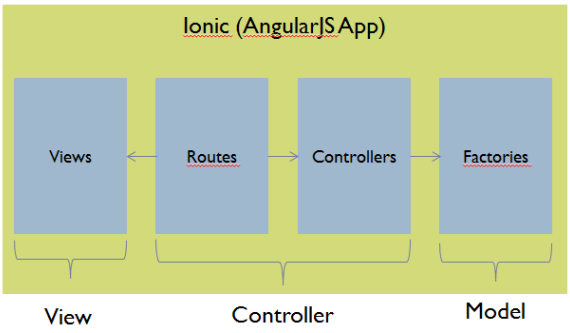

Abbildung **Fehler! Es wurde keine Folge festgelegt.**: Adaption von Model-View-Controller in der App

Use Cases

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

use case	Einkauf einlesen
actors	
	Benutzer
precondition	
	Artikel hat erkennbaren Barcode, Rahmenbedingungen das die App ausgeführt werden kann sind gegeben (Akku voll, genug Rechenleistung etc) , App ist gestartet, Kamera des Mobiltelefons ist funktionstüchtig.
main flow	Artikel bekannt
	Der Nutzer startet die App auf seinem Smartphone. Er navigiert beim Start des Einkaufes in der App durch das Menü und setzt alle Einstellungen die er haben möchte. Er startet den Einkauf, wählt den Supermarkt im Menü aus, in dem er sich befindet, er nimmt den gewünschten Artikel den er einlesen (einkaufen) möchte aus dem Regal im Supermarkt und richtet den Artikel so aus, dass er den Barcode mit dem integrierten Barcodescanner scannen kann. Wenn dieser erfolgreich gescannt ist, sieht er auf dem Screen alle erfassten Informationen. Daraufhin kann er die Menge auswählen ,die er einkaufen möchte und anklicken, ob dieser Artikel für ihn oder z. B. jemand aus seiner Gruppenverwaltung bestimmt ist und ob der Preis oder Sonderpreis gerade gültig ist. Nun kann er den nächsten Artikel Scannen
alternitive flow	Artikel nicht bekannt
	Der Nutzer startet die App auf seinem Smartphone. Anschließend navigiert er beim Start des Einkaufes in der App durch das Menü und konfiguriert alle Einstellungen nach seinen Wünschen. Wenn er den Einkauf startet, wählt den Supermarkt über das Menü aus, in dem er sich befindet. Im Markt nimmt er den gewünschten Artikel den er einlesen (einkaufen) möchte aus dem Regal und richtet den Artikel so aus, dass er den Barcode mit dem integrierten Barcodescanner scannen kann. Wenn der Artikel nicht im System hinterlegt ist, kann er diesen hinzufügen. Hierfür gibt er Preis, Inhaltsmenge, die Einheit der Inhaltsmenge und den Titel an. Nun kann er die Menge auswählen die er einkaufen möchte und anklicken, ob dieser Artikel für ihn oder z. B. jemand aus seiner Gruppenverwaltung bestimmt ist und ob der Preis oder Sonderpreis gerade gültig ist. Nun kann er den nächsten Artikel Scannen
postcondition	
	Der Artikel wurde erfolgreich gescannt. Die Daten im System wurden auf Aktuellität durch den User kontrolliert oder aktualisiert. Der Artikel ist nun im Einkaufskorb aufgenommen und für wen dieser
exceptional flow	Ausnahme 1
	Der Artikel hat keinen Barcode
exceptional flow 2	Ausnahme 2
post condition	
	Der Artikel kann nicht im System erfasst werden, da dieser keinen Barcode hat.
end	Einkauf einlesen

Kommentiert [A4]: Frage: Kommt hier noch ein Quellenverzeichnis rein?
Und wollen wir hier noch die genutzten Tools aufführen? Oder nicht?

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