3. Audit Entwicklungsprojekt interaktiven Systemen

von Merve Kabakci und Lara Löffler

Nutzungskontextanalyse

- Benutzer des Systems
- Arbeitsaufgaben
- Ausrüstung/Arbeitsmittel
- Soziale Umgebung
- Physische Umgebung

Nutzungskontextanalyse
Der Nutzungskontext bei unserem System für Halal - Produkte setzt sich zusammen aus:
1. Der Benutzer des Systems
- Hierbei handelt es sich um verschiedene Benutzergruppen.
Es gibt den primären Benutzer, welches die muslimische Bevölkerung sein kann oder auch jeder Nutzer, der auf seine Ernährung achtet. Es können Menschen sein, die sich vegetarisch Ernähren und keine Gelatinehaltigen Produkte zu sich nehmen.
- Zudem gibt es noch den Restaurantbesitzer oder Imbissbudenbesitzer, der sich in das System einträgt, wenn er ein Halal Zertifikat besitzt.
- Die Lebensmittelgeschäfte und Einzelhändler können sich ebenfalls in das System eintragen, wenn sie Halal Konforme Produkte besitzen.
2. Arbeitsaufgaben
- Die wesentlichen Aufgaben des Systems sind die Restaurants, Imbissbuden und Lebensmittelgeschäfte anzuzeigen, welche Halal Konforme Speisen und Produkte besitzen. Es soll ebenfalls das Zertifikat mit angezeigt werden.
- Das System soll über die Fähigkeit verfügen, den Standort des Benutzers zu ermitteln und ihm somit alle Geschäfte in seiner Umgebung anzuzeigen.
3. Ausrüstung/Arbeitsmittel
- Der Benutzer sollte über ein Smartphone oder ein anderes Gerät verfügen, welches einen Internetempfang hat, um seinen Standort zu ermitteln und die Geschäfte dem entsprechend angezeigt zu bekommen.
4. Soziale Umgebung
- Es handelt sich hier um den Benutzer, der das System nutzt. Das kann die Muslimische Bevölkerung sein oder auch eine Person, die sich ebenfalls nach den Halal Konformen Richtlinien ernährt.

5. Physische Umgebung
- Die physische Umgebung ist die, in der sich der Benutzer aufhält, wie zum Beispiel sein zuhause oder auf der Straße.

Mensch-Computer-Interaktion

- Methodische Rahmen der MCI
  - Grundlegende Vorgehensmodelle
  - Auswahl geeigneter Vorgehensmodelle
  - Usability Engineering Lifecycle

Grundlegende Vorgehensmodelle
Eine funktionelle Software kann sowohl nach einem usage-centered Design, als auch nach einem user-centered Design entwickelt werden. Im user-centered Design steht der Anwender und dessen Benutzererfahrung sowie -zufriedenheit im Vordergrund. Bei dem usage-centered Design liegt hingegen die Benutzung des Systems an sich, inklusive der Verbesserung der Werkzeuge und Prozesse im Fokus.
Auswahl geeignetes Vorgehensmodell
Das Ziel unserer Anwendung liegt darin dem Nutzer dabei zu unterstützen halal konforme Gastronomiebetriebe sowie Supermärkte und Kiosk in seiner Umgebung zu finden. Aus diesem Grund liegt der Dienstnutzer mit seinen Zielen, Erfordernissen und Anforderungen im Fokus. Die Motivation zur Nutzung und Akzeptanz des Systems wird hauptsächlich durch persönliche Faktoren sowie einer intuitiven Bedienung der Anwendung beeinflusst. Die Usablity des Systems muss an die Benutzergruppe und deren Anforderungen angepasst werden, um es gebrauchstauglich zu gestalten. Dazu muss der Benutzer in den Design- und Entwicklungsprozess mit einbezogen werden, indem unter anderem persönliche und soziale Faktoren des Benutzers ermittelt werden.
Unserer Meinung nach eignet sich für die oben beschriebenen Ziele ein user-centered Design besser als ein usage-centered Design. Da die Zielsetzung des Anwenders darin liegt halal konforme Produkt- und Gastronomieangebote in seiner Umgebung zu finden. Das Werkzeug bzw. der Prozess um diese Aufgabe zu lösen bzw. zu verbessern ist dabei nebensächlich. Im Fall eines usage-centered Design liegt der Fokus auf verschiedenen Arten der Nutzung.
Usability Engineering Lifecycle
Wir halten für unser System den Usability Engineering Lifecycle für vorteilhaft, da er keine unterschiedlichen Szenarien beinhaltet. Außerdem bieten uns die wiederkehrenden Iterationen ein breites Verbesserungspotential. Dieses Vorgehen bietet den wesentlichen Vorteil, dass das im Falle eines negativen Ergebnisses, schnell und einfach Verbesserungen an der Software vornehmen können. Innerhalb der Anforderungsanalyse werden sowohl Benutzerprofile erstellt als auch Aufgaben analysiert. Wir entscheiden uns für dieses Modell. Insbesondere da dieses Vorgehensmodell mit der ISO Norm 9241 konform ist.

Proof-of-concept

1. Restaurant verifizieren
2. Standort Genauigkeit

1. Restaurantverifizieren
Risiko: Restaurant können nicht angelegt werden
Ziel:
- Restaurant mit originalem Zertifikat verifizieren können
- Zugriff auf Zertifikat, dadurch verifizieren
Fail:
- Es wird kein Zugriff auf Zertifikat gewährt, da das Restaurant eventuell keins besitzt
- Verschiedene Versionen des Schächtens, dadurch nicht eindeutig, ob sie der Norm entsprechen
- Keine Verifizierung möglich
Alternative:
- Persönlich die Restaurants/Imbissbuden/Lebensmittelgeschäfte kontrollieren, ob sie ein Zertifikat besitzen
2. StandortGenauigkeit
Risiko: Kunde erlaubt keinen Zugriff auf Standort
Ziel:
- Dem Kunden alle Restaurants auflisten, welche Halal konform sind
Fail:
- Standort kann nicht ermittelt werden, aufgrund keines Zugriffs

- Kein GPS-Empfang
- Kein Betrieb wird angezeigt

Alternative:

- Google Maps oder OpenStreetMap benutzen

REST

Ressource	Methode	Semantik
/api/business/{businesstype}	GET	Listet alle vorhandenen Geschäfte auf (businesstype = restaurant; supermarket; kiosk
/api/business/{businesstype}? name=kebapland	GET	Alle jeweiligen Geschäfte mit diesem Namen werden angezeigt
/api/business/{businesstype}? adress.city=koeln	GET	Alle jeweiligen Geschäfte am Standort Köln werden ausgegeben
/api/business/{businesstype}? adress.street=..	GET	Alle jeweiligen Geschäfte auf dieser Straße werden ausgegeben
/api/business/{businesstype}? adress.zipcode=30559	GET	Alle jeweiligen Geschäfte mit der PLZ werden ausgegeben
/api/business/{businesstype}? adress.street=&adress.city= ..	GET	Alle jeweiligen Geschäfte mit der Straße und der PLZ werden ausgegeben
/api/business/{businesstype}? name=&city= ..	GET	Alle jeweiligen Geschäfte mit dem Namen in der Stadt werden ausgegeben
/api/business/{businesstype}ID	GET	Filtert die gewünschten jeweiligen Geschäfte nach ID
/api/business/{businesstype}	POST	Erstellen eines neuen Geschäftes
/api/business/{businesstype}/ID/ name	PUT	Geschäftsnamen updaten
/api/business/{businesstype}/ID/ address	PUT	Adresse eines jeweiligen Geschäfts updaten
/api/business/{businesstype}/ID	DELETE	Löscht Geschäft
/api/users	GET	Liste aller User
/api/users/ID	GET	Filtert die Users nach ID
/api/users?username=...	GET	Suche User nach Namen
/api/users/ID/locations?lat=..&lon=...	GET	liefert zwei Listen zurück, eine Liste mit Geschäften die in der Nähe sind (Koordinaten) & eine die in der Nähe sind und nach den Favoriten des Benutzers gefiltert werden
/api/users/userID/ recommendations?lat=..&lon=...	GET	Gebookmarkete Liste wird zufällig aus der Nähe des Benutzers wiedergegeben
/api/users	POST	Erstellen eines neuen Users, Benutzername darf nicht zweimal existieren
/api/users/ID	DELETE	Lösche User nach ID
/api/users/ID/username	PUT	Benutzernamen updaten
/api/users/userID/bookmarks/ busnissID	POST	Geschäfte als gebookmarked markieren
/api/users/userID/bookmarks/ businessID	DELETE	Geschäft löschen aus dem bookmarks

Technische Risiken

- Fehlerhafte Umgebebungskarte
- Verbindungsprobleme zwischen Client & Server
- Betrieb kann nicht erstellt werden

1. FehlerhafteUmgebungskarte
- Durch falsche Angaben der Geschäfte, kann es dazu kommen, dass diese
nicht angezeigt werden oder an einem anderen falschen Standort erscheinen.
2. Verbindungsprobleme zwischen Client&Server
- Wenn zwischen Client und Server keine Verbindung besteht, dass ist das
Nutzen des Systems nicht mehr gewährleistet. Dies kann entstehen durch beispielsweise fehlende Internetverbindung oder leeren Akku des Smartphones.
3. Betrieb kann nicht erstellt werden
- Es besteht das Risiko, dass ein Restaurant oder Einzelhandel sich nicht
registrieren kann.

Use Case

1.Suche

actor	goal	brief
Alle Stakeholder	Restaurants an einem bestimmten Ort suchen	Eingabe vom Standort. Das System liefert eine entsprechend gekennzeichnete Ergebnisliste, der Actor wählt gewünschtes Ergebnis aus.
Alle Stakeholder	nach bestimmten Restaurants suchen	Eingabe vom Namen des Restaurants. Ergebnis anzeigen
Alle Stakeholder	Lebensmittelgeschäft an einem bestimmten Ort suchen	Eingabe vom Standort. Das System liefert eine entsprechend gekennzeichnete Ergebnisliste, der Actor wählt gewünschtes Ergebnis aus.
Alle Stakeholder	Nach bestimmten Lebensmittelgeschäften suchen	Eingabe vom Namen des Lebensmittelgeschäftes. Ergebnis anzeigen

Use case

2.Suchergebnisse

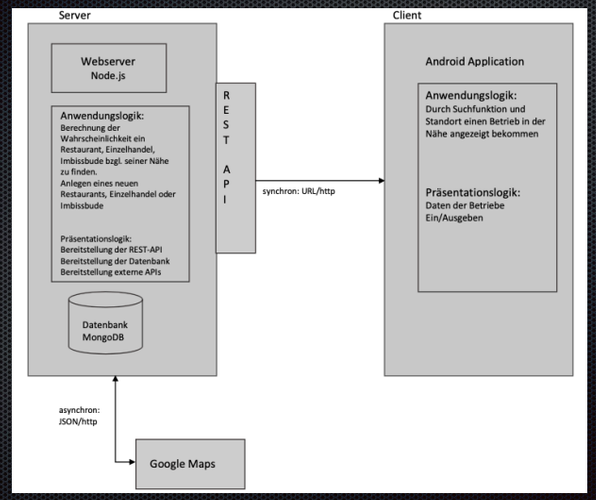
actor	goal	brief
Alle Stakeholder	Sortierung des Suchergebisses anpassen	Entsprechende Funktion des Systems nutzen und präferierten Filter wählen
Alle Stakeholder	Auswahl des am besten bewerteten Restaurant/Lebensmittelgeschäft/Imbissbude	System zeigt Bewertung der Betriebe in der Suchübersicht an

Use Case

3. Thread

actor	goal	brief
Alle Stakeholder	Ein Restaurant/Lebensmittelgeschäft/Imbissbude als Favorit markieren	Nutzen der entsprechenden Systemfunktion, der Betrieb wird zur Liste der favorisierten Betriebe hinzugefügt
Alle Stakeholder	Gewählten Laden als Favoriten löschen	In der Liste der Favoriten wird das entsprechende Geschäft bearbeitet
Alle Stakeholder	Anzeigen der Bewertungen	Das System zeigt die Bewertungen der Lebensmittelgeschäfte/Restaurants/Imbissbuden an

Architekturdiagramm



Client:
Der Client befasst eine mobile App über das System Android. Er kommuniziert sowohl synchron als auch asynchron mit dem Server. Im Client sitzt ein Teil der Anwendungslogik und der Präsentationslogik. Der Client hat die Aufgabe durch die Suchfunktion ein Restaurant oder Einzelhandel an einem Ort anzuzeigen.
Server:
Der Server kommuniziert mit allen Clients. Er kommuniziert synchron als auch asynchron. Außerdem kommuniziert er auch mit der externen API. Bei der externen API handelt es sich um Google Maps um den Standort des Nutzers zu ermitteln und ihm die gewünschten Funktionen dadurch anzuzeigen. Des Weiteren wird im Server ein neues Restaurant, Einzelhandel oder eine neue Imbissbude angelegt mit all den wichtigen Informationen.
Datenbank:
Für das Datenbanksystem haben wir uns um eine nicht-relationale Datenbank entschieden. Bei MongoDB besteht der Vorteil die Datensätze untereinander aufzuführen, dadurch kann man problemlos neue Attribute hinzufügen. MongoDB speichert die Daten im BSON-Format, dass dem JSON-Format nachgestaltet ist. Dadurch werden viele JavaScript-Datentypen unterstützt, weshalb die Nutzung von MongoDB ideal für die Node.js-Plattformen sind.

Kommunikationsmodell