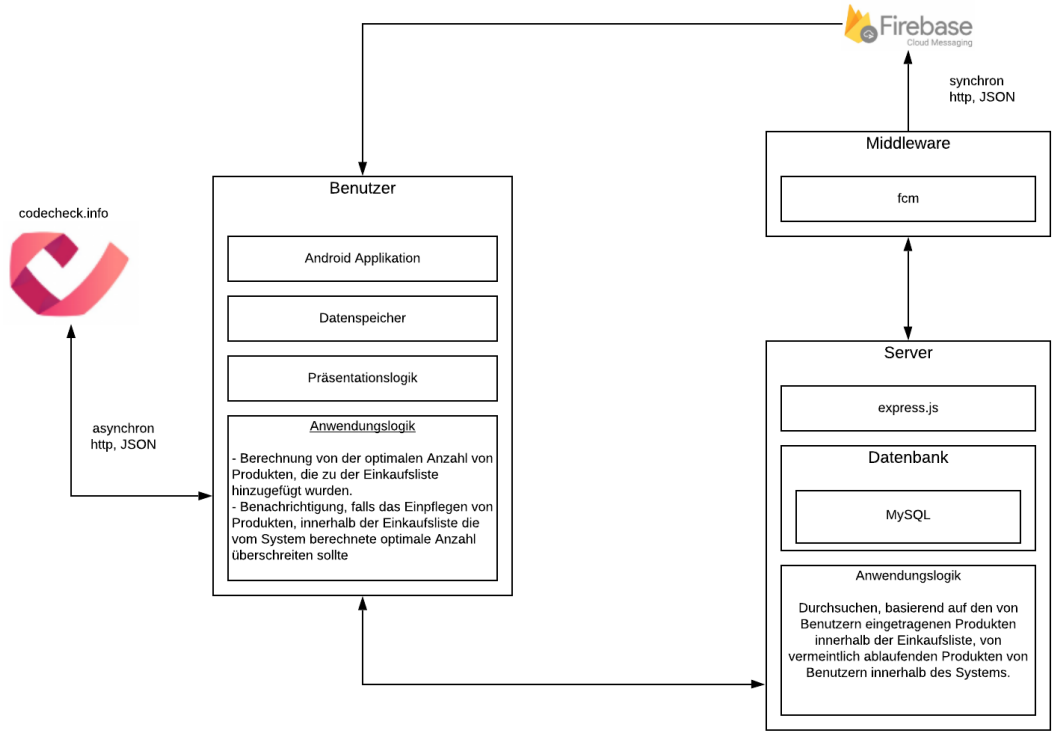
Iteration der Architektur

Die Kommunikation zwischen Client und Schnittstelle ist nötig, um bei der Erfassung von Produkten, Informationen über das Produkt zu beziehen. Ohne diese Kommunikation, könnte der Client zwar über den zu entwickelnden Barcode Scanner die European Article Number der Produkte erfassen, jedoch ohne Informationen über dessen zugehöriges Produkt.



Schnittstelle codecheck.info

Für den geplanten Barcode Scanner ist die European Article Number von hoher Bedeutung um die gesamte Geschäftslogik innerhalb des Systems zu realisieren.

Firebase Cloud Messaging

Für Die Umsetzung der Benachrichtigungen des Servers an den Benutzer Client mittels “Push-Notifications” sehen wir die Benutzung von FCM vor. Über das Software Development Kit des Herstellers ist es möglich, Benachrichtigungen basierend auf dem Publish und Subscribe Pattern oder Single Device Messaging, per Token, via HTTP APIs durchzuführen.

Datenformat: JSON

Die Wahl auf das Datenformat JSON ergab sich aus der Kommunikation zwischen dem Benutzer Client und der Schnittstelle. Innerhalb dieser Kommunikation werden die zu beziehenden Daten als JSON-Objekte übermittelt. Das System würde des Weiteren von Einfachheit und Schlankheit profitieren und sich eine ressourcenschonende Verarbeitung der Daten für mobile Applikationen eignet.

Client

Der Client ist in den drei Schichten “Datenspeicher”, ”Präsentationslogik” und “Anwendungslogik” unterteilt. Während die Präsentationslogik lediglich als Benutzungsschnittstelle dient, dient die Anwendungslogik sowohl für die Steuerung der Verarbeitung von eingehenden Informationen, als auch zur Berechnung und Benachrichtigung des Benutzers. Der Datenspeicher hingegen, dient den Benutzern die Möglichkeit, ihre bereits im System eingepflegten Daten jederzeit einzusehen, ohne dabei über einen Internetzugang zu verfügen.

Client -> Server Kommunikation

Der Client kontaktiert den implementierten REST-Server via einer http-Schnittstelle. Diese Kommunikation findet sowohl asynchron, als auch synchron statt.

Asynchron

Innerhalb der asynchronen Kommunikation mit dem REST-Server soll basierend auf den eingehenden Informationen, in Form von Produkten innerhalb von Einkaufslisten oder Inventaren, des Benutzers Informationen innerhalb des Systems gefiltert werden. Durch die einzelnen zu durchlaufenden Prozessstufen innerhalb des REST-Servers wird dem Client jeweils eine Benachrichtigung geschickt. Diese Benachrichtigung wird dem Benutzer automatisiert über so genannte “Push-Notifications” gesendet.

Synchron

Ähnlich wie bei der asynchronen Kommunikation zwischen Client und dem REST-Server, filtert dieser Informationen basierend auf den eingehenden Informationen des Clients. Jedoch wird das Durchlaufen der Prozessstufen so beeinflusst, dass der REST-Server dem Client die gewünschten Informationen sofort zur Verfügung stellt.

Client -> codecheck.info

Der Client bezieht die Informationen, für die Übermittlung von Daten zum REST-Server, über die Schnittstelle von codecheck.info. Über die bereitgestellten Informationen verschiedener Produkte werden mittels der European Article Number (EAN), sowohl der Use Case “Inventarverwaltung”, als auch die Erstellung von Einkaufszetteln realisiert. Des Weiteren ist die Kommunikation zwischen der Schnittstelle ein wichtiger Bestandteil um eine Erfassung von Produkten via Barcode Scan zu ermöglichen.

Server

Die Aufgabe des Servers ist es im Wesentlichen, Anfragen zu bearbeiten. Die vom Server verarbeitete Anwendungslogik dient dazu automatisierte Suchvorgänge, basierend auf den eingehenden Informationen des Benutzer Clients, durchzuführen.
Für die Umsetzung des REST-Servers wird die Plattform NodeJS benutzt, ein großes Hauptmerkmal dieser Entscheidung ist, dass NodeJs eine ressourcensparende Architektur bietet, die eine besonders große Anzahl gleichzeitig bestehender Netzwerkverbindung ermöglicht.

Kommunikation Server -> FCM (Firebase Cloud Messaging)

Wie bereits zuvor erwähnt führt der REST-Server, basierend auf den eingehenden Informationen des Clients, automatisierte Suchvorgänge durch. Um über eine asynchrone Kommunikation den Client die Ergebnisse dieser Suchvorgänge mitzuteilen. Jedoch benötigt der REST-Server eine Middleware um den Datenaustausch mit dem FCM durchzuführen. Für diese Middleware wird das Node Modul “fcm-node” vorgesehen, da zu diesem eine recht übersichtliche Einleitung existiert.

Anwendungslogik Client

Die Anwendungslogik des Clients dient sowohl für die Steuerung der Verarbeitung von eingehenden Informationen, als auch zur Berechnung und Benachrichtigung des Benutzers. Die Anwendungslogik beschränkt sich auf die Einkaufslisten, die vom Benutzer zu erstellen sind.

Die Berechnung eines optimalen Einkaufs- und Wegwerfverhaltens wird anhand der zu dokumentierenden Daten durchgeführt. Das System dokumentiert sowohl ausgetragene Lebensmittel aus den Inventaren einzelner Benutzer, als auch die Lebensmittel die über das System an andere Benutzer vermittelt wurden. Für die Berechnung ist es nötig, eine SQL Abfrage zu tätigen um die bereits dokumentierten Information beziehen zu können und anschließend den Durchschnitt dieser zu berechnen. Durch diese Berechnung wird eine optimale Anzahl pro Lebensmittel festgestellt und den Benutzer beim Erstellen von Einkaufslisten darauf hingewiesen. Das Sorgt dafür, dass dem Benutzer selbst überlassen ist, ob er eine Optimierung durchführen lassen will, oder eben doch das Risiko eingehen will Lebensmittel zu verschwenden.

Verteiltheit

Es wurde sich für diese Verteilung der Anwendungslogik entschieden, da das schonen von Ressourcen für den Client die Performance positiv beeinflusst. Die Anwendungslogik wäre aufgrund des hohen Traffics und die Menge der zu vergleichenden Daten für den Client sehr aufwändig. Des Weiteren müsste der Client für jede Überprüfung mit dem Server kommunizieren und da der Server zu einer direkten Anbindung zur Datenbank verfügt, ist es sinnvoll diesen Teil der Anwendungslogik auf dem Server durchzuführen. Die Berechnung der optimalen Anzahl einzelner Produkte erfolgt zwar auch über die Kommunikation mit dem Server, diese jedoch dient nur dazu um die zu dokumentierenden Daten des Benutzers zu beziehen. Die anschließenden Berechnungen die innerhalb des Benutzer-Clients durchgeführt werden sind vom Ressourcenverbrauch eher gering einzustufen.