**Gruppe B**

Grundlagen Mobile Computing

Einkaufs App

THE SIMPLE ACT OF CARING IS HEROIC

(some clever person, once)

# Allgemeines

Im Zuge der Lehrveranstaltung *Grundlagen Mobile Computing* soll eine Applikation für mobile Endgeräte, vollumfassend (Frontend & Backend) erstellt werden. Eingesetzte Technologien können dabei frei gewählt werden. Die funktionale Domäne der Applikation ist dabei frei wählbar. Einzig übergeordnete Anforderungen an die Funktion der Applikation sind vorgegeben und wie folgt definiert:

* Mobile App
* Interaktion mit anderen Benutzern
* Auch verwendbar, wenn Server offline
* Daten müssen permanent persistiert werden

## Motivation

Die aktuell anhaltende Covid-19 Pandemie fordert eine neue Ordnung des täglichen Lebens der Gesellschaft. Vor allem Verkehrsbeschränkungen des täglichen Lebens erschweren dabei das Leben vor allem für besonders gefährdete Gesellschaftsgruppen.

Familien und Angehörige älterer und / oder gesundheitlich beeinträchtigter Personen müssen den Kontakt reduzieren und sind gegebenenfalls auch nicht in unmittelbarer Reichweite, um Unterstützung anbieten zu können. Zudem wird der genannten Personengruppe empfohlen Zuhause zu bleiben und vor allem größere Menschenansammlungen zu vermeiden, um so das persönliche Risiko zu minimieren.

Diese Maßnahmen sind dabei nicht nur für ältere und gebrechlichere Personengruppen empfehlenswert, sondern können generell Anwendung finden, um eine weitere Ausbreitung dieses Virus zu verhindern.

Um eine Versorgung mit Gütern des täglichen Bedarfs sicherzustellen wird eine Applikation zur gemeinschaftlichen Verwaltung der Besorgungen städtischer Wohnhäuser erstellt. Mit Hilfe dieser Applikation kann jede Partei eines Wohnhauses ihren aktuellen Bedarf preisgeben und mit den Mitbewohnern teilen. Gleichzeitig ist es auch jedem Benutzer möglich die offenen Besorgungen gesammelt zu erledigen und anschließend in der Hausgemeinschaft zu verteilen.

Zusätzlich bietet dieser Ansatz eine generelle Möglichkeit des digitalen Teilens innerhalb eines Hauses. So können neben Bedarfen für Lebensmittel und sonstige Konsumgüter auch Bedarfe für beispielsweise Werkzeug oder persönliche Unterstützungen im Haushalt erstellt, und durch die Hausgemeinschaft gedeckt werden.

Die Benutzeroberfläche wird dabei möglichst einfach und selbsterklärend gehalten um die Applikation auch für ältere Personen und Personen die noch nicht mit neueren Technologien vertraut sind, erreichbar zu halten.

# Anforderungen

In diesem Kapitel werden die, in der Gruppe identifizierten Anforderungen an die zu erstellende Applikation festgehalten. Diese dienen als Grundlage zur Auswahl eingesetzter Technologien.

Die funktionalen Anforderungen beschreiben dabei explizite Funktionen bzw. Funktionalitäten, die von der Applikation erfüllt werden müssen.

Im Abschnitt der nichtfunktionalen Anforderungen wird besonders auf die Benutzbarkeit und Zugänglichkeit der Applikation geachtet.

## Funktionale Anforderungen

Benutzeranmeldung / Datenerfassung

Die Applikation muss es Benutzern ermöglichen sich anzumelden. Als Anmeldedaten werden Name, Anschrift und Kontaktdaten erfasst. Eine Kartographische Darstellung des Wohnorts wird benötigt, um etwaige Fehlinterpretationen der eingegebenen Adresse zu verhindern, da diese für die weitere Funktionalität der Applikation von äußerster Relevanz ist.

Menüführung

Die Menüführung soll sich auf 3 wesentliche Dialoge beschränken:

* Bedarfsübersicht des Wohnhauses
* Dialog zum Anlegen einer Bedarfsliste
* Dialog zum Erfüllen von Bedarfen

Bedarf erfassen

Die Applikation muss eine Möglichkeit bieten einen bestimmten Bedarf einer Wohnpartei der angegebenen Adresse zu erfassen. Dies soll dabei möglichst offen durch die einfache Eingabe eines Freitextes erfolgen. So liegt es in der Verantwortung des Bedarfsstellers, die Anforderungen seines Bedarfs zu definieren. Eingaben wie ‚Milch‘, ‚1l Milch‘, oder ‚1l Schärdinger Milch‘ sollen dabei möglich sein, ohne den Benutzer durch das System einzuschränken.

*Bedarfsübersicht*

Eine Übersicht über aktuell offene Bedarfe des jeweiligen Wohnhauses listet sämtliche Bedarfe mit Zuordnung zur Wohnpartei. Die Entscheidung für welche Parteien eingekauft wird bzw. wer unterstützt wird, soll dabei vom jeweiligen Bedarfserfüller getroffen werden, um so Hausbewohner nicht zwangsweise miteinander in Kontakt zu bringen.

Bedarfserfüllung

Die Applikation muss die Möglichkeit bieten, aktuell offene Bedarfe zu erfüllen. Dabei wird auf das Prinzip von ‚first come, first serve‘ gesetzt. Applikationsbenutzer in der Rolle des Bedarfserfüllers können dabei die Applikation wie einen Einkaufszettel verwenden und direkt abhaken was besorgt wurde. Mit dieser Interaktion wird der Bedarf sofort als erfüllt angesehen und kann, wodurch verhindert wird, dass er von anderen Benutzern erfüllt wird. Die Möglichkeit der Verwendung als Einkaufsliste soll den Applikationsbenutzer dabei zur häufigen Verwendung der Applikation und somit zur Einbindung in tägliche Routinen incentivieren.

Der Bedarfssteller hat in jedem Fall sofortige Einsicht, dass sich die erstellten Bedarfe gerade in Erfüllung befinden und von ihm keine weiteren Schritte (externe Hilfe, Heimpflege, etc.) organisiert werden müssen.

## Nichtfunktionale Anforderungen

Platform Independency

Die Applikation soll weitestgehend plattformunabhängig erreichbar sein. Die verwendeten Geräte sind dabei Android Smartphones und Tablets, Apple Smartphones und Tablets und in wenigen Einzelfällen auch PCs.

Accessibility

Das Gestaltung der Applikation soll weitestgehend barrierefrei erfolgen. Die Zielgruppe richtet sich dabei vor allem auch an ältere Personen, wodurch besonders auf große Schriften, hohe Kontraste der Hauptinformationen sowie eine allgemein stark vereinfachte Bedienung der Applikation erforderlich ist.

Maintainability

Die Applikation soll mit Basisfunktionalität in Betrieb genommen werden und anschließend kontinuierlich erweitert und verbessert werden. Durch die gegebene Zielgruppe ist es dabei wichtig, dass die Software ohne Benutzerinteraktion am neuesten Stand gehalten werden kann. Besonders am Backend soll es nur eine operative Softwareversion geben. Im Falle von grundlegenden Strukturänderungen soll das Frontend entsprechend (einfach) mitgezogen werden können.

# UI/UX Konzept

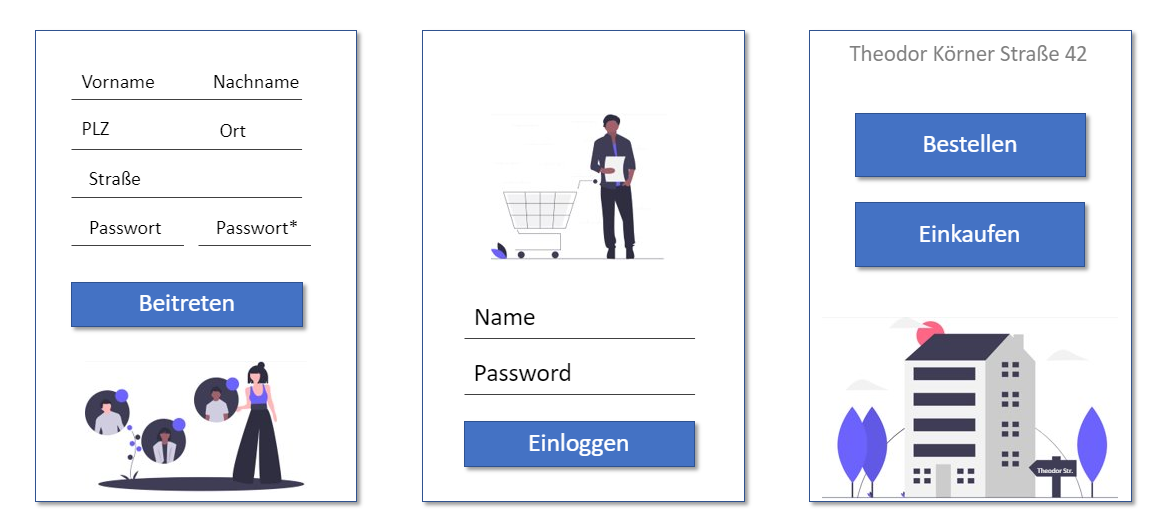
Das User Interface setzt auf sehr einfache und standardisierte Komponenten. Dabei wird die Hauptinteraktion immer mit Buttons in blauer Farbe gekennzeichnet. Der Farbcode ist dabei so gewählt, dass er sich bei Darstellung in Grautönen auch sehr klar von der restlichen Applikation abhebt.

Dies ermöglicht eine Führung des Applikationsbenutzers wie an einem roten (blauen) Faden durch die gesamten Prozesse der Applikation.

Eine zusätzliche Anforderung besteht in der maximalmöglichen Reduzierung der angezeigten Information. Der Applikationsbenutzer erhält dabei nur die Information, die er wirklich braucht, um so eine visuelle Überforderung zu vermeiden.

Die generellen Benutzereindrücke werden dabei mit semantisch sinnvollen Hintergrundgrafiken untermalt, die dabei eine zusätzliche visuelle Unterstützung für den Benutzer darstellen. Da in den Grafiken Personen illustriert werden, wird auf ethnisch korrekte Durchmischung der Abbildungen gesetzt.

Die folgenden Grafiken / Designskizzen illustrieren das Grundkonzept und dienen als Anhalt für die Implementierung der Applikation.



# Systemarchitektur

Durch die in vorherigen Kapiteln genannten Anforderungen und Designgrundlagen wurde die Auswahl der zu verwendenden Technologien bereits auf eine überschaubare Gruppe eingeschränkt. Die folgenden Abschnitte erörtern die explizite Auswahl des Technologiestacks, sowie die angedachte Systemarchitektur.

## Frontend Technology

Um eine möglichste breite Erreichbarkeit auf verschiedenen Plattformen umsetzen zu können wird auf die Technologie progressiver Webapplikationen gesetzt, da diese mit aktuell gängigen Browsern die funktionalen Anforderungen abdecken können.

Vor allem die Anforderung der hohen Wartbarkeit und der schmalen Infrastruktur in Kombination mit den sehr begrenzt verfügbaren Entwickler-Ressourcen untermauert diese Entscheidung. Bei der Wahl des Frontend-Frameworks wird auf VUE.js gesetzt, da dieses mit seinen on-board Funktionalitäten das Grunddesign der Applikation ausreichend unterstützt. Dabei ist VUE.js ein sehr einfach zu erlernendes und im Vergleich zu anderen Frameworks, zur Erstellung progressiver Webapplikation, wenig komplexes Framework ist.

Zur Vereinheitlichung der User-Interface Komponenten wird zusätzlich auf das UI-Framework Quasar gesetzt, welches gängige und vor allem bekannte Komponenten in einer sehr einfachen Form zur Verfügung stellt.

## Backend Technology

Auch am Backend wird aufgrund des Zeitdrucks und der Ressourcensituation auf ein sehr leichtgewichtiges, einfaches und gut dokumentiertes Framework – Feathers.js - gesetzt.

Feathers basiert dabei auf dem allseits bekannten Node.js mit Express und eignet sich gut zur Implementierung einfacher Backend-Services. Mit Hilfe des CLI Tools können dabei Grundstrukturen von Backends mit Datenbankanbindungen sowie vorkonfigurierter Authentifizierungsmethoden und diversen Schnittstellen automatisiert generiert werden.

Eine bereits eingebaute Datenbankanbindung u.A. zu MongoDB ist dabei ein weiterer wesentlicher Faktor für die Wahl von Feathers.js. Das Framework bietet dabei jedoch verschiedenen Datenbank Konnektoren, die ausreichend abstrahiert vorliegen, um eine entsprechende Austauschbarkeit des DBMS zu gewährleisten.

Die Datenhaltung wird in der Erstversion über eine MongoDB mit Mongoose erfüllt. Eine Grobabschätzung der zu erwartenden Datenmengen sowie die Anzahl der zu erwartenden Entitäten spricht dabei nicht zwangsweise für den Einsatz einer relationalen Datenbank. Der Einsatz einer nicht-relationalen Datenbank vereinfacht dabei erneut die gesamte Umsetzung und beschleunigt somit die ‚Time to Market‘ des ersten Grundfunktionalität.

Um gegen Datenverlust gerüstet zu sein wird wöchentlich ein Back Up mit mongodump durfgeführt. Dabei liest mongodump Daten aus einer MongoDB-Datenbank und erstellt originalgetreue BSON-Dateien, die das mongorestore-Tool zum Auffüllen einer MongoDB-Datenbank verwenden kann.

## Systemaufbau

Generell kann eine Wohnadresse als zusammengehörige Organisationseinheit innerhalb der Applikation angesehen werden. Dadurch ist eine sehr gute Abgrenzung der Daten möglich was im Weiteren mit einer relativ einfachen Architektur, gute Skalierungsmöglichkeiten eröffnet.

Die MongoDB, Back-end und Frontend werden jeweils in Docker-Containern deployed. Da die Hauptnutzungszeiten, den Geschäftszeiten gleichen, wird ein generelles Deployment zwischen 00:00 und 00:30 Uhr durchgeführt. Aufgrund der Kosten und Haupt-Nutzungszeiten, ziehen wir zu Beginn noch keine 0% Downtime in Betracht.

Beim generellen Aufbau wird ‚Containerisierung‘ mittels virtueller Maschinen gesetzt. Dabei bedient eine VM mehrere Haushalte. Erreicht die VM eine Auslastung von 66% der zugewiesenen Ressourcen, wird eine weitere VM eröffnet, die neue Haushalte bedienen kann. Die 66% Schwelle stellt dabei einen ausreichenden Leistungspuffer sicher. Für potenziell zusätzliche Benutzer innerhalb einer Hausgemeinschaft ist dabei noch ausreichend Platz vorhanden. Eine Migration einer Hausgemeinschaft zwischen Knoten wird dabei vermieden.

Die Architektur ermöglicht dabei zusätzlich ein phasenweises Hochfahren der Applikation, je nach Nachfrage am Markt. Im ersten Ausbauschritt wird ein einziger Knoten betrieben, der über einen Gatewayknoten angesprochen wird. Beim Erreichen des 66% Schwellenwerts wird ein weiterer Knoten in Betrieb genommen. Die 34% Leistungsreserve ermöglichen dabei ausreichend Zeit zur Reaktion.

Der Gatewayknoten agiert dabei zusätzlich als HOST der progressiven Webapplikation. Diese wird dabei serverseitig wie eine statische Website behandelt.

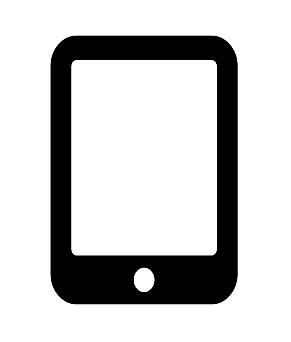
Folgende Grafik veranschaulicht den generellen Aufbau sowie die Skalierungsmöglichkeiten des Applikations-Backends.

Node …

Node n

Gateway Node

Node 1



Application

Data