

**Fachbereich 4  
Informatik, Kommunikation und Wirtschaft**

# **Independent Coursework I**

**Umsetzung einer iOS Applikation basierend auf dem  
Masterprojekt Needo.com mit Swift**

**Autor:** Christian Neubauer  
Matr.Nr. 547617

**Prüfer:** Prof. Dr. Jung

**Abgabedatum:** 24.03.2016

## II Inhaltsverzeichnis

II Inhaltsverzeichnis	II
<b>1 Einleitung</b>	<b>1</b>
1.1 Ausgangslage . . . . .	1
1.1.1 Die Firma Commercetools . . . . .	1
1.1.2 Das Ergebnis . . . . .	1
1.2 Idee . . . . .	1
<b>2 Usability Analyse</b>	<b>3</b>
2.1 IST - Stand . . . . .	3
2.1.1 Login . . . . .	4
2.1.2 Start . . . . .	5
2.1.3 Menü . . . . .	6
2.1.4 Angebote und Gesuche erstellen . . . . .	7
2.1.5 Matching . . . . .	8
2.1.6 „Eigene“ anzeigen . . . . .	9
2.2 Fazit . . . . .	9
2.2.1 Positiv . . . . .	10
2.2.2 Negativ . . . . .	10
<b>3 Usability-Konzept</b>	<b>11</b>
3.1 Vor betrachtung . . . . .	11
3.1.1 UseCases . . . . .	11
3.1.2 Wie sollen diese umgesetzt werden. . . . .	12
<b>4 Umsetzung</b>	<b>14</b>
4.1 Voraussetzungen . . . . .	14
4.2 Realisierung . . . . .	15
4.3 Wie wurden die Use Cases umgesetzt? . . . . .	16
4.3.1 LandingScreen . . . . .	17
4.4 Probleme und Lösungen . . . . .	20
4.4.1 Asynchronous Programming . . . . .	20
4.4.2 Kamera . . . . .	21
<b>5 Ergebnis und Ausblick</b>	<b>22</b>
5.1 Ergebnis . . . . .	22
5.2 Ausblick . . . . .	23
5.2.1 Resümee . . . . .	23

# 1 Einleitung

## 1.1 Ausgangslage

Im Wintersemester 2014/15 und Sommersemester '15 entstand in Zusammenarbeit mit der Firma Commercetools das Masterprojekt „neeedo.com“. Die Idee dieses Projektes wurde als next-generation e-Commerce tituliert und sollte, alternative Ideen und moderne Technologien in das e-Commerce Umfeld bringen.

### 1.1.1 Die Firma Commercetools

Die Firma Commercetools entwickelt und vertreibt die e-Commerce-Plattform SSphe-re.io“, heute ”commerceTools platform“. Dies ist eine Commerce-as-a-Service Platform, die es dem Anwender erlaubt auf einfache Weise einen (Online-) Shop zu erstellen und zu verwalten. Für diese Plattform steht ebenfalls eine Entwickler-API zur Verfügung, auf der dieses Projekt aufbaut.

### 1.1.2 Das Ergebnis

Das Ergebnis des Projektes „neeedo.com“ war eine Verkaufsplattform, in der die Idee der Suche/Biete Karten, bekannt vom klassischen „Schwarzen Brett“, aufgegriffen und in einem modernen Gewand präsentiert wird. Es wurden dabei neue Ansätze für das Suchen und Anbieten von Produkten umgesetzt. Ein intelligentes Matching-System hilft den Nutzern dabei das zu finden was sie suchen.

Umgesetzt wurde das Projekt mithilfe einer selbst entwickelten Scala/Play-API, die die Verbindung zum Commercetools-Service herstellt, der die Datenhaltung und weitere interne Funktionen bereitstellt. In dieser API finden zudem alle Operationen zum Suchen und Erstellen von Artikeln statt, sowie das Matching und andere Funktionen. Welche Funktionen von der API genau erfüllt werden wird in der Umsetzung ersichtlich.

Auf dieser API aufbauend wurden eine WebApplikation und eine native Android Applikation umgesetzt, die als Schnittstelle zum Benutzer dienen.

## 1.2 Idee

Während des Projekts stand immer wieder die Entwicklung einer zusätzlichen iOS Applikation im Raum. Diese wurde aber schlussendlich nicht umgesetzt.

Dies soll nun im Rahmen dieses „Independent Courseworks“ nachgeholt werden.

Da, wie die Analyse dieser zeigen wird, markante Schwächen in der Usability der Android Applikation existieren, soll ein Schwerpunkt dieser Arbeit sein diese zu verbessern bzw.

ein neues Konzept zu entwickeln. Dies ist auch notwendig, da die aktuellen iOS-Geräte abweichende Verhalten im Vergleich zu Android Geräten aufweisen.

## 2 Usability Analyse

Da dieses Projekt auf Grundlage der Ergebnisse aus dem Projekt neeed.com entsteht, soll zuerst der IST-Stand analysiert werden.

Zuerst soll eine Betrachtung der bestehende Android - Applikation Aufschluss darüber geben was Positiv ist und wo Verbesserungen an der Usability angebracht sind.

### 2.1 IST - Stand

Bisher besteht das Projekt neeedo.com aus einer Android Applikation, einer WebApp und der verbindenden AP. Die Android Applikation ist von Verhalten und Optik her, an die Webapplikation angelehnt, und um Elemente anderer bekannter mobiler Applikationen erweitert um die geplanten Funktionen umzusetzen.

Anhand verschiedener Screenshots soll im folgenden dargestellt werden was positiv/ negativ ist und welche Lösungen als Verbesserungen umgesetzt werden könnten.

### 2.1.1 Login

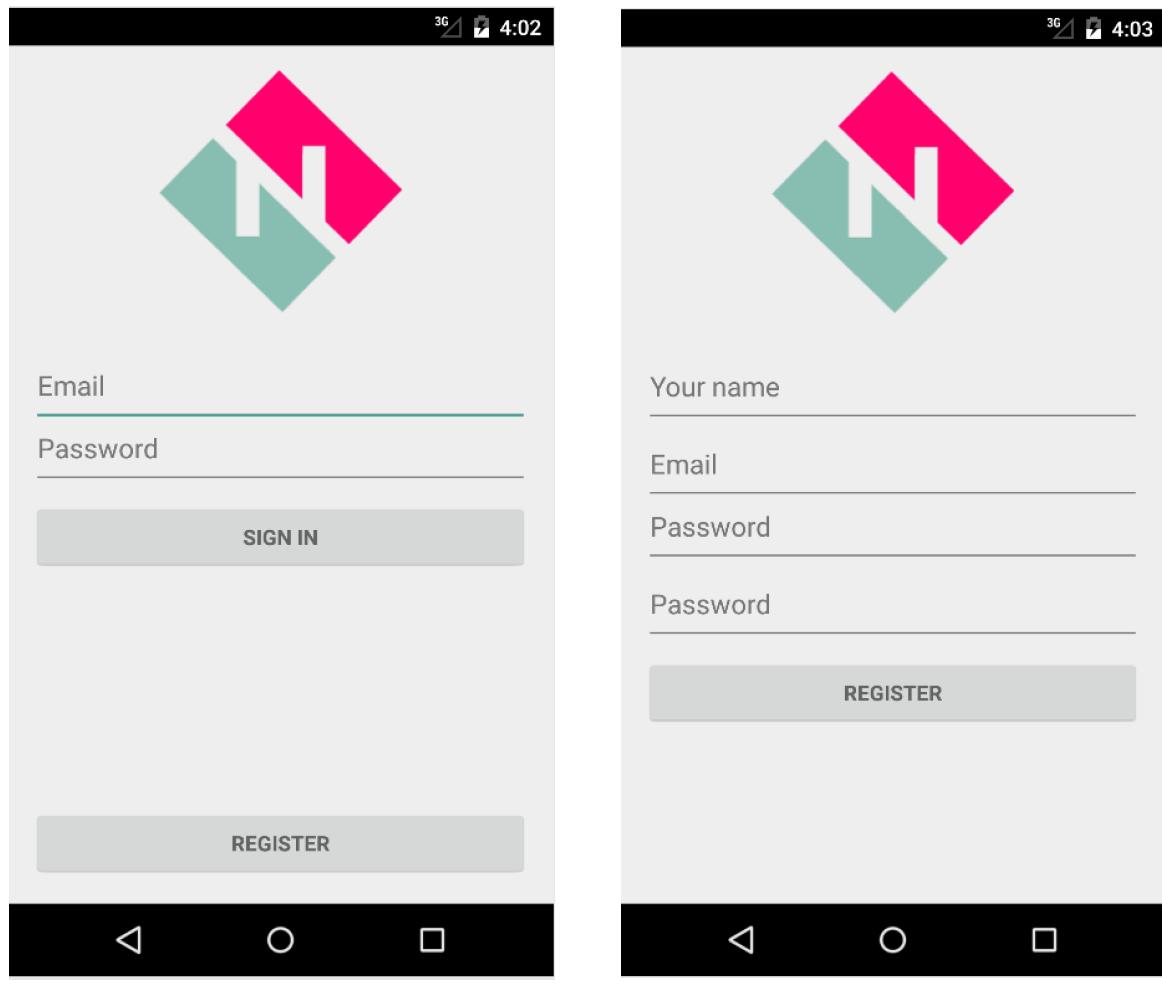


Abbildung 1

Startet man die App findet man sich zuerst auf einem LogIn/SignUp - Screen wieder. Hier kann durch die Eingabe einer gültigen Email-adresse und eines Passworts ein Account erstellt, oder sich in einen bestehenden Account eingeloggt werden. Die Umsetzung ist schlicht und funktional gehalten. Dies kann so übernommen werden.

Was erst im Vergleich mit den folgenden Seiten auffällt, ist dass das Design dieser Seite nicht in das Schema der restlichen App passt. Aus Gründen der Konsistenz sollte das angepasst werden.

### 2.1.2 Start

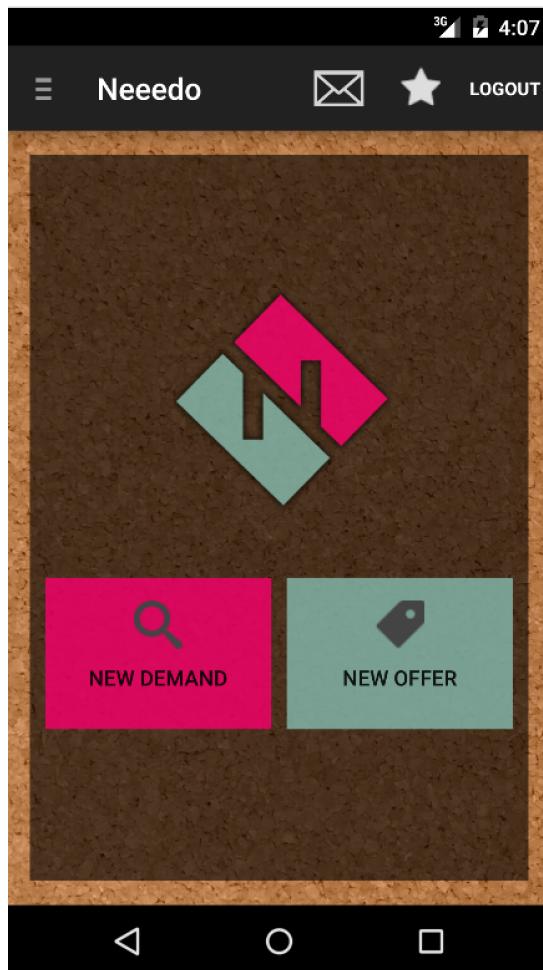


Abbildung 2: LandingScreen

Loggt man sich in die App ein landet man zuerst auf der hier abgebildeten LandingPage. Diese ist schlicht gehalten. Auf dem halbtransparenten Hintergrund finden sich zwei Button, „New Demand“ und „New Offer“, die einen wohl zu den Formularen führen mit denen man diese erstellen kann. Man sieht am oberen Rand des Screens eine Menüleiste, die einen „BurgerButton als Menü, den Schriftzug „Neeedo“, ein Briefsymbol als Hinweis auf Nachrichten und einen Logout-Button beinhaltet. Dies ist eine (Android-) typische Positionierung für diese Elemente.

Der Menü Button ist dabei der am wenigsten markante Punkt dieses Screens, obwohl man wohl davon ausgehen kann, dass er der Wichtigste ist. Der Zugang zum Menü müsste markanter sein.

### 2.1.3 Menü

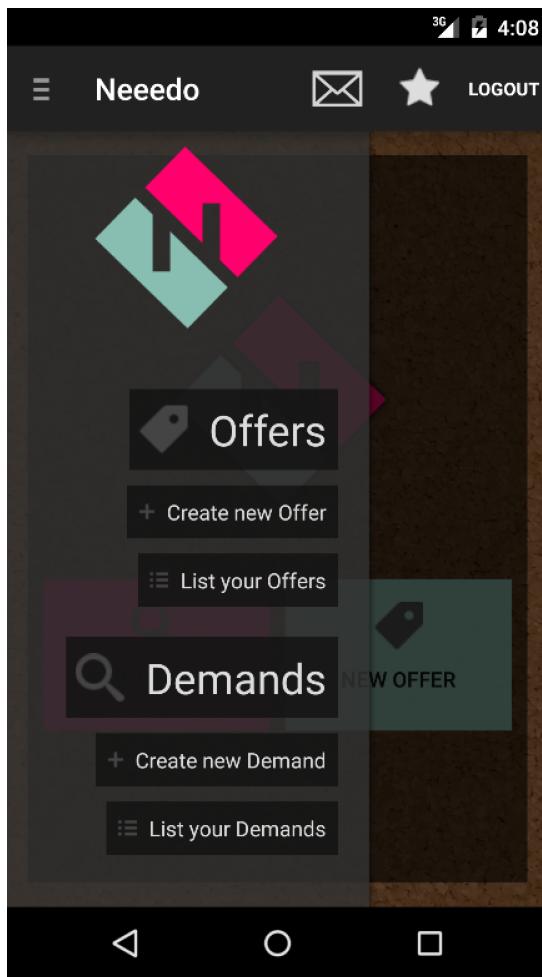


Abbildung 3: Menü offen

Benutzt man den „Menü-Button“ öffnet sich das hier abgebildete Menü. Dieses bietet, vier Funktionen an: „Neues Gesuch erstellen“, „Neues Angebot erstellen“, „Eigene Angebote auflisten“, „Eigene Gesuche auflisten“.

Die Umrandung um die Worte, hier, „Offers“ und „Demands“ lässt vermuten, dass es sich hierbei ebenfalls um Buttons handelt. Dies ist jedoch nicht der Fall. Es handelt sich hierbei lediglich um Trenner, die das Menü in drei Sektionen unterteilen.

Betrachtet man sich das Menü als Ganzes fällt auf, dass es viel zu viel Platz einnimmt und unnötig zu sein scheint. Um ein solches Menü zu rechtfertigen müßten mehr Funktionen angeboten werden.

### 2.1.4 Angebote und Gesuche erstellen

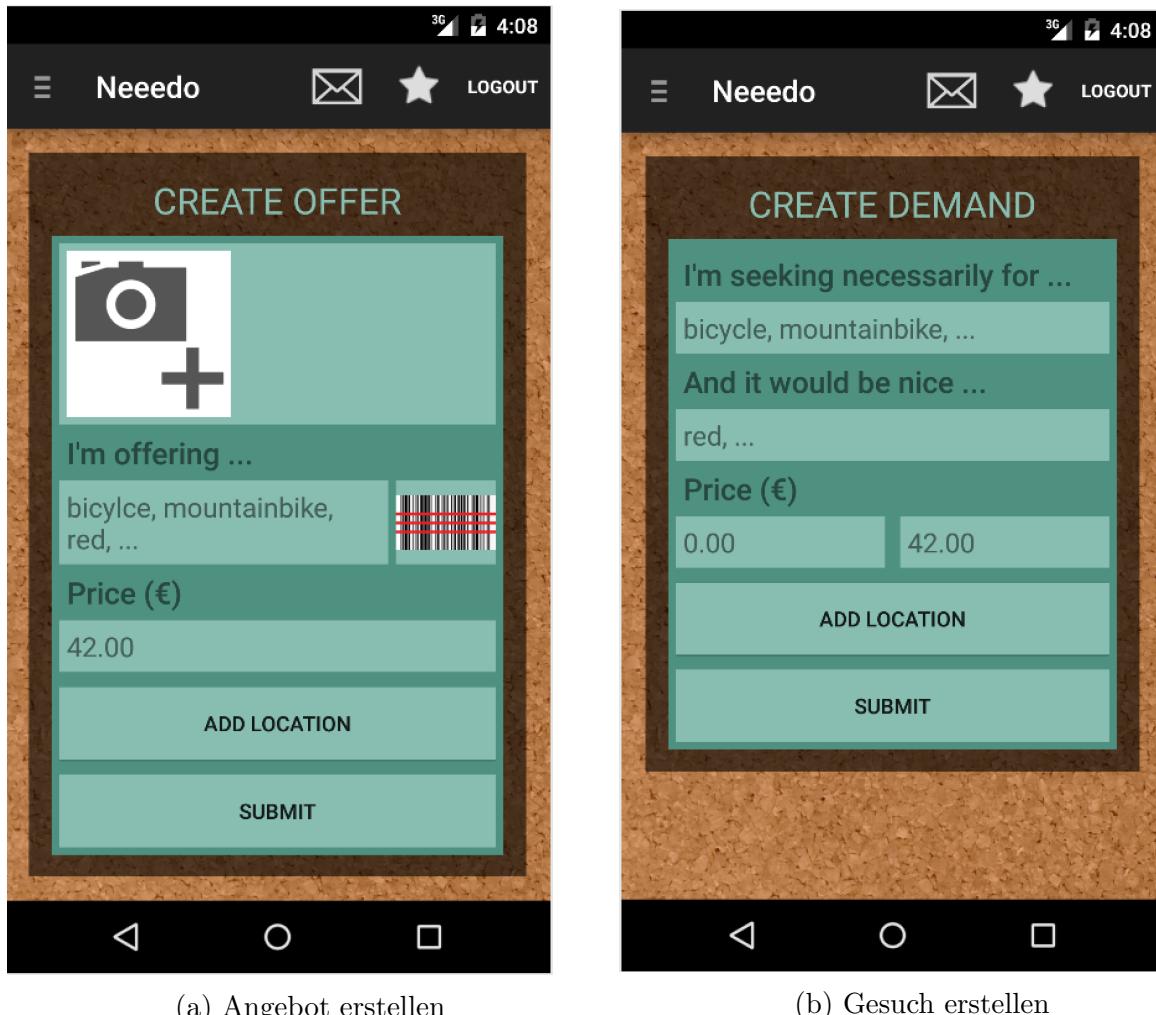


Abbildung 4

Wählt man, entweder im Menü oder auf dem StartScreen, eine der „erstellen“-Funktionen aus, landet man auf einer dieser Seiten entsprechend der jeweiligen Auswahl. Hier wird jeweils ein Formular angezeigt, in dem man eintragen kann was man sucht oder was man anbieten möchte.

Erstellt man ein Angebot/Offer ist es möglich Fotos zu hinzuzufügen. Dies geschieht entweder die Auswahl des Camera-Bildes. Dieses ist auf den ersten Blick nicht als Button erkennbar. Dies sollte klarer erkennbar sein. Ebenfalls erhält man die Möglichkeit Informationen über sein Produkt durch das Scannen eines Barcodes zu erhalten. Diese Funktionen sind sinnvoll und könnten in ähnlicher Weise übernommen werden.

Erstellt man ein Gesuch/Demand, bietet die App keine Zusatzfunktionen an. Hier muss das Formular von Hand ausgefüllt werden.

Beide Formulare besitzen die Möglichkeit eine Adresse hinzuzufügen, dies geschieht auf einer zusätzlichen Seite auf der dies durch Auswahl einer Position auf einer Karte oder ein Suchfeld geschieht.

Befindet man sich auf der Formular-Seite kann man nicht klar erkennen ob man eine Position eingeben muss oder ob es einen Standardwert gibt. Dies sollte erkennbar gemacht werden.

### 2.1.5 Matching

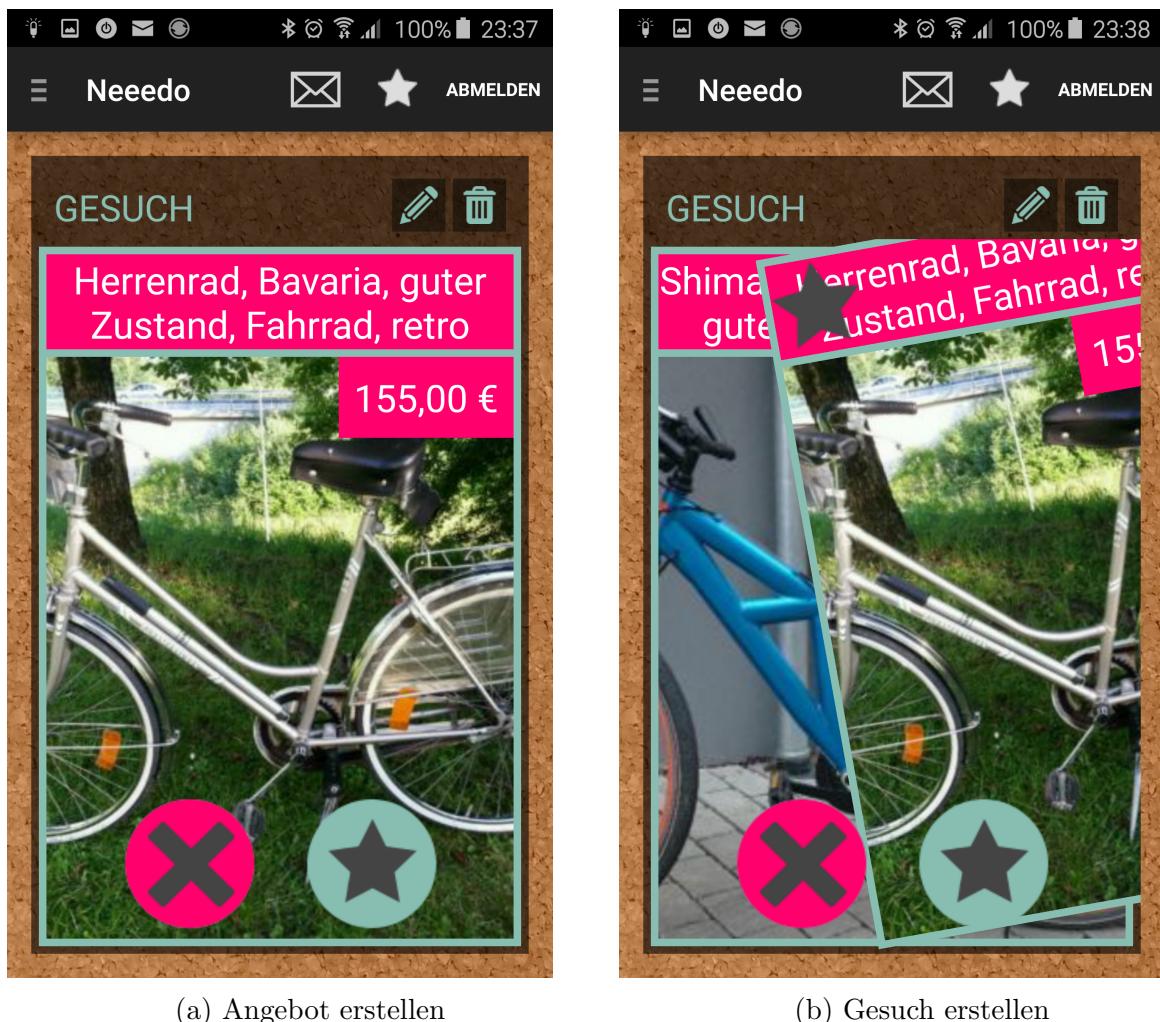


Abbildung 5

Sendet man das auf der „Gesuch erstellen“-Seite ausgefüllte Formular ab wird man auf die hier gezeigte „Matching-View“ weitergeleitet. Hier werden einem passende Angebote in einer, u.a. aus der App „Tinder“ bekannten Stapel-Darstellung präsentiert. Diese können entweder durch die Buttons oder durch eine Swipe-Geste nach rechts oder Links verworfen oder als Favorit gespeichert werden.

Um die gespeicherten Favoriten aufzurufen muss man das Stern-Symbol in der Menüleiste oben nutzen, im Screen selbst ist keine Weiterleitung möglich.

### 2.1.6 „Eigene“ anzeigen

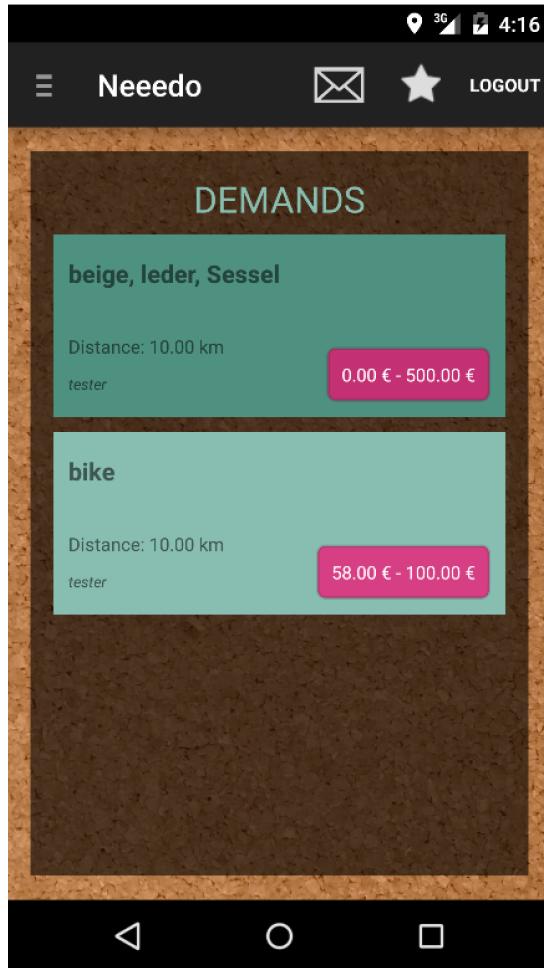


Abbildung 6: Eigene Demands anzeigen

Im Menü finden sich auch die Funktionen „eigene Angebote/Gesuche anzeigen“. Benutzt man einen dieser Buttons, wird eine Tabelle wie die hier abgebildete präsentiert. Klickt man hier auf eines der aufgeführten Elemente, wird man auf ein Formular ähnlich dem des „Erstellens“ geleitet, auf dem Änderungen durchgeführt werden können.

## 2.2 Fazit

Dies sind die wichtigsten Funktionen dieser App, weitere wie zum Beispiel der Versand von Nachrichten werden nicht näher betrachtet, da hier nur ein Standard Chat zum Einsatz kommt.

### 2.2.1 Positiv

- + Das Design ist einheitlich
- + Viele Funktionen können übernommen werden

### 2.2.2 Negativ

- Nicht alle Buttons können als solche erkannt werden
- Elemente können irrtümlich für Buttons gehalten werden.
- Seitenmenü zu groß und unpraktisch
- Login nicht im selben Design

## 3 Usability-Konzept

Basieren auf den Ergebnissen der IST-Analyse kann ein Konzept erarbeitet werden, dass die erkannten Stärken aufgreift und versucht die erkannten Schwächen zu verbessern.

### 3.1 Vor betrachtung

Aus der IST-Analyse und der gegebenen API lassen sich die folgenden Funktionen, bzw. Use Cases ableiten, die in der App umgesetzt werden sollen.

#### 3.1.1 UseCases

##### Benutzer

- Benutzer anlegen
- Benutzer löschen
- Benutzer ausloggen

##### Angebote

- Angebote erstellen
- Angebote aktualisieren
- Angebote löschen
- Eigene Angebote anzeigen

##### Gesuche

- Gesuche erstellen
- Gesuche aktualisieren
- Gesuche löschen
- Eigene Gesuche anzeigen

##### Favoriten

- Favoriten hinzufügen
- Favoriten entfernen
- Favoriten anzeigen

##### Matching

- Zeige passende Angebote

### **3.1.2 Wie sollen diese umgesetzt werden.**

#### **Benutzer**

Wie in der IST- Analyse bereits erwähnt, ist der UseCase des Nutzer-Anlegens, bzw. des User Logins bereits in einer praktikablen Weise umgesetzt. Diese soll auch in der hier entstehenden App übernommen werden.

Das Ausloggen ist in der Android Applikation durch einen Button in der Menüleiste oben umgesetzt, jedoch besteht keine Möglichkeit seinen Account zu löschen.

In der geplanten App sollen diese beiden Funktionen zusammen in einem Menü angeboten werden.

#### **Angebote, Gesuche erstellen/bearbeiten/löschen**

Grundsätzlich lässt sich an der Umsetzung des Erstellens und Aktualisierens der Angebote und Gesuche nicht viel aussetzen, außer einzelner Designfehler. Die Umsetzung mit Formularen lässt sich nur schwer umgehen und würde kaum Vorteile bieten.

Allerdings sind diese Formulare im aktuellen Zustand nicht besonders gut strukturiert und müssen angepasst werden, um Uneindeutigkeiten zu umgehen.

Das Löschen der Elemente geschieht durch einen Button auf jeweiligen Ansicht. Dies lässt sich ebenfalls aufgreifen.

In iOS haben sich inzwischen aber auch, gerade in Tabellendarstellungen, Alternative Möglichkeiten entwickelt. Diese sollen hier ebenfalls zum Einsatz kommen.

#### **Favoriten**

Das Hinzufügen von Favoriten kann nur aus der jeweiligen Offer Darstellung geschehen. Daran lässt sich nichts ändern, da die API aktuell nur Angebote als Favorit zulässt. In der aktuellen Umsetzung geschieht das durch einen „Stern“-Button auf der jeweiligen Karte.

Der Stern ist eine bekannte Art der Darstellung für Markierungen und Favoriten. Dies kann so übernommen werden.

#### **Favoriten, (Eigene) Angebote, (Eigene) Gesuche anzeigen**

Um eine Übersicht seiner eigenen Favoriten, Angebote und Gesuche darzustellen wurde in der Android Applikation eine Listen-Darstellung gewählt. Diese zeigt sich als funktional und sinnvoll.

Die Detail-Darstellung der einzelnen Elemente geschieht durch eine Karten-Darstellung auf der alle relevanten Informationen angezeigt werden. Die grundsätzliche Art der Darstellung dieser Elemente ist passend. In dieser Arbeit sollen lediglich Änderungen an inneren Struktur der Elemente durchgeführt werden.

Im Rahmen der Änderungen an der Mén”führung wird sich der Zugang zu diesen Ansichten ändern.

### **Matching**

In der Android Applikation werden die gematchten Angebote in der Tinder-ähnlichen Stapeldarstellung dargestellt. Diese Darstellung entwickelt aktuell eine gewisse Beliebtheit und wird auch in vielen anderen Apps angeboten. Es lässt sich auch kein grundlegender Fehler hieran erkennen daher kann sie in einer angepassten Weise übernommen werden.

### **Menü**

Eine große Schwäche der Android Applikation war das viel zu große Seitenmenü, dass keinen besonderen Zweck erfüllt. Da die angebotenen Funktionen hier sehr begrenzt sind bietet es sich an dieses durch eine TabBar zu ersetzen. Diese hat den Vorteil, dass die Funktionen immer präsent sind und keinen zusätzlichen Platz benötigen

## 4 Umsetzung

Die App wurde in XCode 7.2 für iOS9 entwickelt. Zur Umsetzung wurde Swift2 verwendet.

### 4.1 Voraussetzungen

Um diese App umsetzen zu können, mussten zuerst einige Voraussetzungen erfüllt werden.

#### API

Damit diese App realisiert werden konnte musste zuerst eine Verbindung zur API hergestellt werden. Diese kommuniziert über RESTful Services und ist zu großen Teilen nur über eine HTTPS-Verbindung erreichbar.

Die API kann auf GitHub unter <https://github.com/neeedo/neeedo-api> bezogen werden. Im Normalfall bietet diese für die lokale Entwicklung eine voll funktionsfähige Scala-Applikation die einfach über `./sbt run -Dhttps.port=9443` in einer Konsole gestartet werden kann. Diese ist dann unter <https://localhost:9443/>

Da Apple in der iOS-Entwicklung nur noch HTTPS-Verbindungen zu zertifizierten Diensten erlaubt, trat hier bereits das erste Problem bei der Umsetzung auf. Wenn die API innerhalb der lokalen Entwicklungsumgebung auf *localhost* betrieben wird, ist aufgrund von „Server Trust problems“ keine Kommunikation möglich. Dieses Problem lies sich weder durch die Installation eines Zertifikats, noch den Eintrag „Allow Arbitrary Loads“ in der Projektkonfiguration beheben.

Die Lösung, die am Ende zum Erfolg führte, war die Installation der API in einem DigitalOcean-Container. Hier war es möglich ein Zertifikat zu hinterlegen und eine stabile HTTPS-Verbindung aufzubauen, die eine Kommunikation zwischen API und App erlaubt.

Aktuell läuft dieser Container noch im Entwicklermodus, und muss manuell gestartet werden.

#### Konfiguration

Damit die API korrekt arbeitet müssen in der custom-application.conf - File Credentials zu einem Sphere-Shop hinterlegt werden. Um bei der Entwicklung vollen Zugriff auf die erstellten Daten zu erhalten, wurde auf <https://admin.sphere.io/> ein neuer Shop angelegt. Die nötigen Daten können dort unter Entwickler → API-Zugriff gefunden werden.

## 4.2 Realisierung

Bei der Realisierung der dieser App wurde größtenteils auf den Einsatz von zusätzlichen Libraries verzichtet, da diese nicht nötig waren.

### Networking

Die einzige externe Library die in dieser App eingesetzt wurde, ist Alamofire.

Dies ist eine in Swift geschriebene Networking Library, die Funktionen zur Benutzung von REST-Verbindungen anbietet. Eine genaue Dokumentation dieser Library kann unter <http://cocoadocs.org/docsets/Alamofire/3.2.1/index.html> eingesehen werden.

Der Hauptgrund für die Benutzung dieser Library liegt darin, dass sie den Aufwand bei der Erstellung von HTTP-Requests vereinfacht. Sie stellt dabei keine besonderen Zusatzfunktionen bereit, jedoch vereinfacht sie den die Nutzung sehr. (Der benötigte Code wird ungefähr gedrittelt). Dies ist gerade in einem Projekt dass auf sehr vielen solcher Abfragen basiert ein großer Vorteil.

### Aufbau und Design

Der Grundlegende Aufbau der App entstand durch den Einsatz in Xcode implementierten Storyboards. Hier wurden die benötigten Elemente so zusammengesetzt, dass sie die gewünschten Funktionen erlauben.

In der aktuellen Fassung wurden das Design beinahe vollständig aus Standard-UI-Elementen aufgebaut. Lediglich einige besondere Custom-Elemente wie Pins wurden zusätzlich verwendet.

Wenn die App über einen prototypische Entwicklung hinausgeht, können hier noch ausgestaltete Custom Elemente eingesetzt werden.

Ein durchgängiges Design wie es in der Android Applikation vorkommt, wurde nicht vollständig realisiert, da dies den zeitlichen Rahmen dieser Arbeit gesprengt hätte.

## 4.3 Wie wurden die Use Cases umgesetzt?

### Benutzer

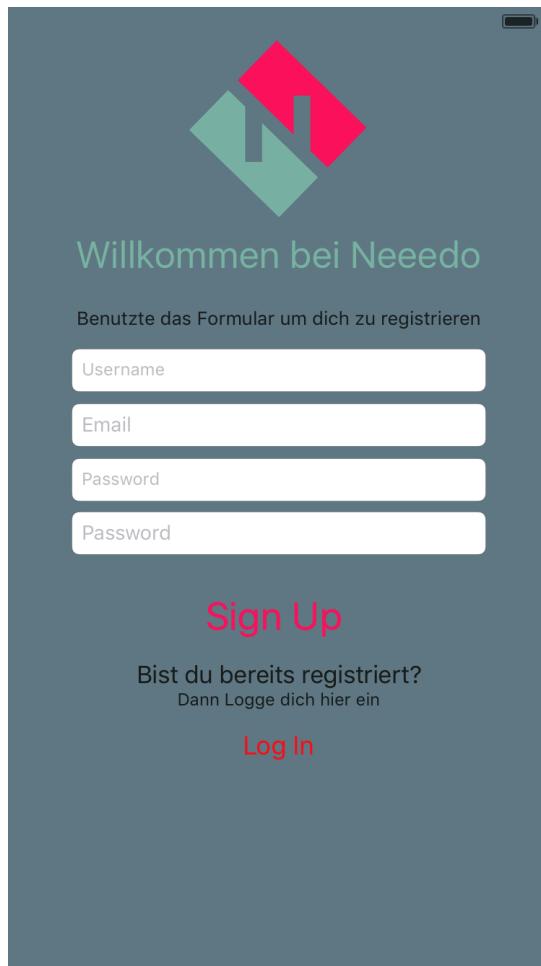


Abbildung 7: SignUpScreen

Wie im Konzept bereits erwähnt wurde an der grundlegenden Struktur des Login/SignUps nichts verändert. Lediglich farblich wurde es an den rest der Applikation angepasst

### 4.3.1 LandingScreen

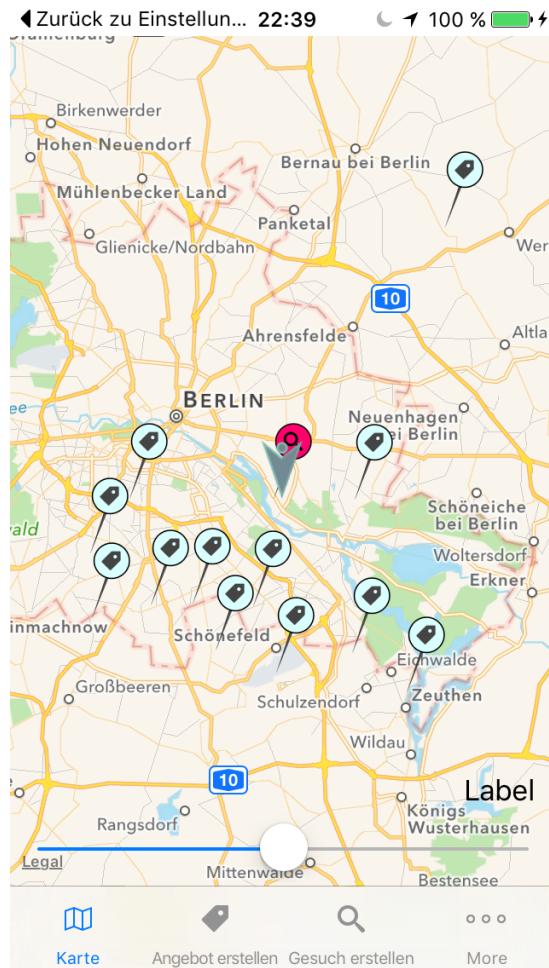


Abbildung 8: SignUpScreen

Die größte Veränderung zur Android App wurde direkt auf dem LandingScreen umgesetzt. Zuvor wurden hier nur zwei Buttons angezeigt. In der neugestalteten Fassung startet man auf einer Karte auf der direkt Angebote im Umkreis angezeigt werden. Diese Idee wurde so bereits in der WebApp umgesetzt.

### Menü

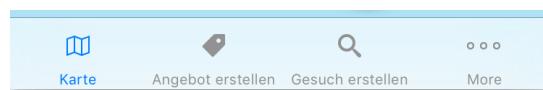


Abbildung 9: SignUpScreen

Diese Veränderung des LandingScreens war möglich, da man durch die Verlagerung des Menüs in eine Tabbar keinen Bedarf mehr für die beiden zentralen Buttons hatte. Das neue

Tabbar-Menü bietet direkten und globalen Zugang zu den wichtigen Erstell-Funktionen, Über den Button „Mehr“ können Account spezifische Funktionen wie LogOut, Nachrichten etc. erreicht werden.

### Angebote, Gesuche erstellen/bearbeiten

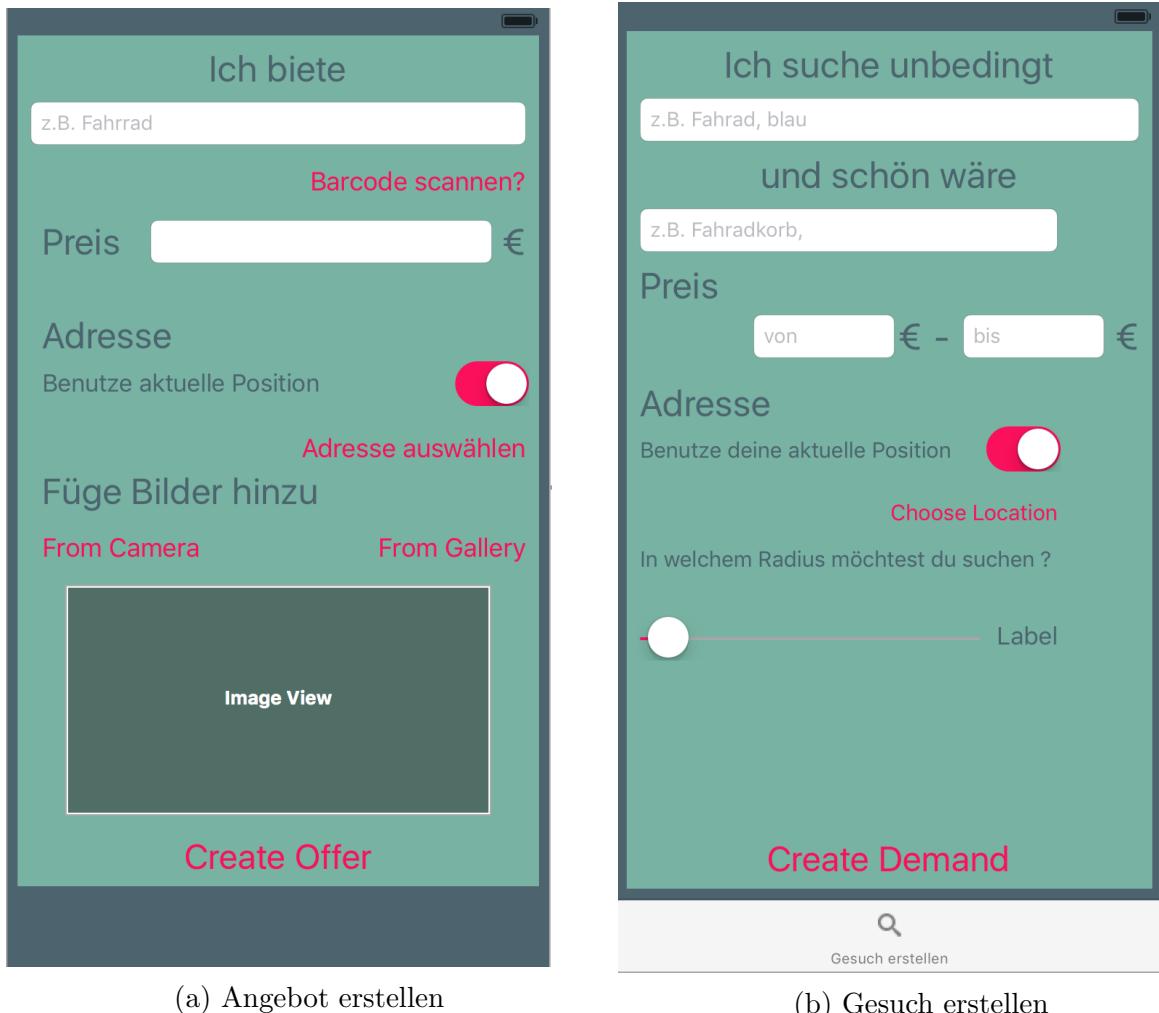


Abbildung 10

### Favoriten

Für das Erstellen und bearbeiten der Angebote und Gesuche wurde das Konzept der Formulare aus der Android Applikation übernommen. Der größte Unterschied zur vorherigen Fassung ist, dass die Auswahl der Adresse nicht mehr standardmäßig aktiv ist. Hier wurde die Möglichkeit geschaffen, direkt zu wählen, ob die aktuelle Position genutzt werden soll oder ob man eine andere wählen möchte.

Im Formular zum Erstellen eines Angebotes wurden die Funktionen des Barcodescanners, sowie das Hinzufügen von Bildern übernommen. Der Barcodescanner wurde dabei durch

in der AVFoundation-Library zugängliche Funktionen umgestetzt. Die Informationen zu den Codes werden über die outpan-API bzogen, die auch in der Android Applikation zum Einsatz kommt.

Das Hinzufügen von Bildern ist getrennt durch 2 Button dargestellt, um zu verdeutlichen, dass sowohl aus der Gallerie als auch von der Kamera Bilder bezogen werden können.

### Matching

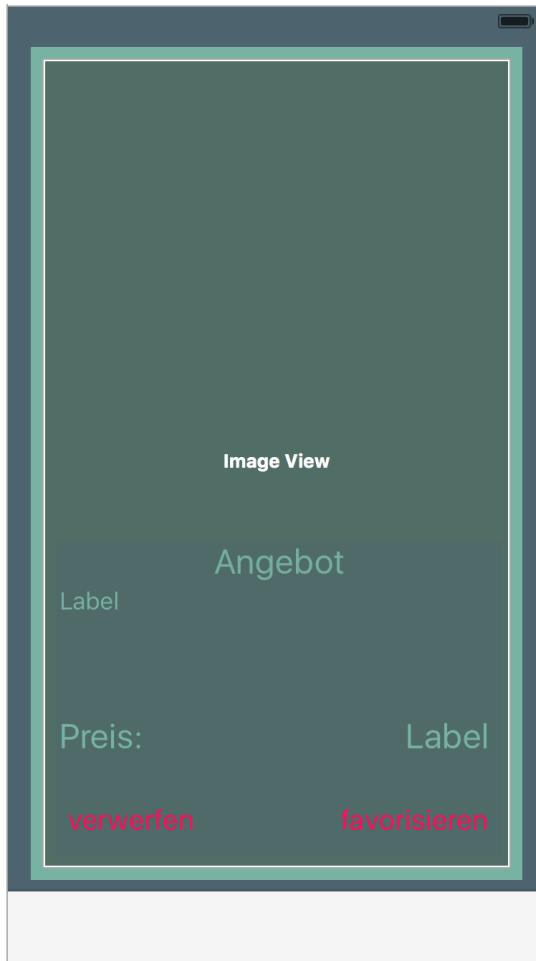


Abbildung 11: SignUpScreen

Das Matching wurde auf ebenfalls durch die Tinder-ähnliche Swipe ansicht realisiert. Was das Bild hier nicht anzeigt ist, dass das Angebotsbild im Hintergrund angezeigt wird, und man die erwarteten Swipe-Gesten nutzen kann.

## Favoriten, (Eigene) Angebote, (Eigene) Gesuche anzeigen

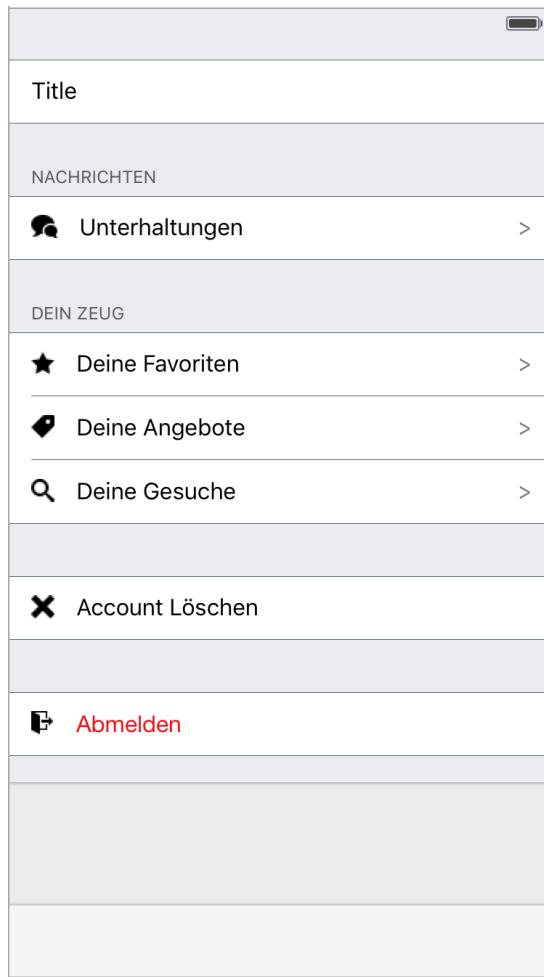


Abbildung 12: SignUpScreen

Die restlichen Funktionalitäten, die zuvor über das Kopf- oder Seitenmenü erreichbar waren, wurden zusammengefasst und unter dem Menüpunkt „Mehr“ zugänglich gemacht. Hier kann man sich jetzt seine Favoriten, Gesuche und Angebote und Nachrichten anzeigen lassen. Zusätzlich kann man hier auch seinen Account löschen und sich ausloggen.

## 4.4 Probleme und Lösungen

Während der Entwicklung dieser Applikation traten einige Probleme auf, diese sollen hier einmal mit ihren Lösungen dargestellt werden.

### 4.4.1 Asynchronous Programming

Ein Problem das bei der Verwendung von RESTful Services häufig auftritt, ist das die Antworten(Responses) auf die Anfragen(Requests) oft eine gewisse Zeit in Anspruch nehmen. Damit die App nicht blockiert während auf diese Antworten gewartet wird sind

die Funktionen die diese Anfragen bearbeiten asynchron aufgebaut. Dies ist so sowohl in Alamofire als auch im klassischen Swift realisiert.

Anfangs wurde dieser Fakt nicht beachtet, was zu einigen Problemen und Appabstürzen führte.

Die Lösung die hierbei zu verwenden ist, sind sogenannte *Completionhandler* die es ermöglichen Funktionen abhängig von der Antwort aufzurufen, d.h. diese werden dann ausgelöst wenn wirklich eine Antwort vorhanden ist.

#### 4.4.2 Kamera

Das Aufnehmen von Bildern mit der Kamera warf einige Probleme auf, die auch noch nicht behoben sind. Es scheint hier einen häufig auftretenden Bug zu geben, der beim Zugriff auf das ungespeicherte Bild auftritt. In diesem Fall erhält man den Fehler „snapshotting a view that has not been rendered results in an empty snapshot. ensure your view has been rendered at least once before snapshotting or snapshot after screen updates.“ Um das zu beheben muss das Bild zwischengespeichert oder angezeigt werden, jedoch scheinen danach in irgendeiner Art Probleme bei der Weiterverarbeitung aufzutreten.

## 5 Ergebnis und Ausblick

Im Rahmen dieses Independent Courseworks wurde versucht eine Umsetzung der Ideen des Masterprojektes neeedo.com für iOS 9 unter Verwendung von Swift2 zu erreichen. Dies sollte auf Grundlage der existierenden Android Applikation geschehen, die zuvor nach Kriterien der Usability analysiert und Umgestaltet wurde.

Bis zum Ende des Projektzeitraumes war es möglich eine prototypische Umsetzung der iOS-Applikation zu erreichen, in der ein Großteil der angestrebten Funktionalitäten implementiert werden konnte.

Das Projekt kann in seinem aktuellen Stand unter <https://github.com/Cneubauern/neeedoIOS> heruntergeladen werden.

### 5.1 Ergebnis

Die folgenden Funktionen konnten zum jetzigen Zeitpunkt umgesetzt werden.

#### Benutzer

- + Benutzer anlegen
- + Benutzer löschen
- + Benutzer ein/ausloggen

#### Angebote

- + Angebote erstellen
- + Angebote aktualisieren
- + Angebote löschen
- + Eigene Angebote anzeigen

#### Gesuche

- + Gesuche erstellen
- + Gesuche aktualisieren
- + Gesuche löschen
- + Eigene Gesuche anzeigen

#### Favoriten

- + Favoriten hinzufügen
- + Favoriten entfernen
- + Favoriten anzeigen

### Favoriten

- + Favoriten hinzufügen
- + Favoriten entfernen
- + Favoriten anzeigen

Aktuell gibt es noch Probleme beim Matching, dieses führt unter bestimmten Umständen zu Abstürzen, sowie beim ImageUpload von der Camera. Das Messaging System wurde als Ansicht implementiert ist aber noch nicht funktionsfähig. Das von der neeedo-API angebotene Tag-Suggestion-System wurde aus zeitgründen nicht mehr bearbeitet.

### 5.2 Ausblick

Für die Zukunft gilt es vor allem an der Konsistenz und am Design zu arbeiten. Es sind noch viele kleine und Mittelgroße Bugs zu beheben und die beschriebenen Funktionalitäten vollständig umzusetzen.

### 5.3 Resümee

Für mich war dieses Projekt eine interessante Herausforderungen. Zwar habe ich in der Vergangenheit bereits Erfahrungen mit der iOS Entwicklung gemacht, doch war Swift2 noch sehr neu für mich. Ebenso war dies die erste iOS-App mit Netzwerkanbindung die ich selbst entwickelt habe. Ich konnte im Verlauf des Projektes viel lernen und neue Erfahrungen machen.