Technology
Arts Sciences
TH Köln

Entwicklungsprojekt interaktive Systeme

Wintersemester 2018/2019

Modelle und Modellierungsbegründungen Meet & Remind

Dozenten

Prof. Dr. Gerhard Hartmann

Prof. Dr. Kristian Fischer

Betreuer

Daniela Reschke

Markus Alterauge

Team

Johanna Mayer

Julian Schoemaker

Inhaltsverzeichnis

1. Glossar

Begriff	Bedeutung
Benachrichtigungen	Meldungen, die an den Benutzer gesendet werden, um auf eine Erinnerung hinzuweisen. Wird auf dem Client erstellt, auch wenn das System nicht in direkter Benutzung ist.
Code-Autovervollständigung	Automatisches vervollständigen von Namen für Variablen, Funktionen oder Klassen. Vereinfacht den Prozess der Programmierung und verringert Rechtschreibfehler innerhalb des Codes.
Erinnerungen	Erinnerungen an ein Gespräch mit einem Gesprächspartner.
Gespräch	Ein Gespräch ist eine Art der Kommunikation zwischen mindestens zwei Gesprächspartnern.
Gesprächspartner	Person mit der gesprochen wird. Z.B. Kollegen, Familienmitglieder oder Freunde.
Medien	Möglichkeit zum Hinzuziehen von Themen für ein Gespräch. Z.B. Internet, Zeit oder TV.
Ort	Ort an dem das Gespräch stattfindet.
Quelle	Quelle des Themas des Gesprächs. Umfasst Medien und das Umfeld.
Syntax-Highlighting	Hervorheben der Struktur und Reihenfolge von Code innerhalb einer Entwicklungsumgebung, zur besseren Übersicht.
System	System der personenbezogenen Erinnerungen. Auch “Meet & Remind”.
Umfeld	Möglichkeit zum Hinzuziehen von Themen.

2. Einleitung

In diesem Dokument geht es um die Modelle und die Modellierungsbegründungen zum im Konzept beschriebenen System Meet & Remind. Es befasst sich mit dem gewählten Vorgehensmodell der Software-Entwicklung und beantwortet die Fragen zur Erstellung des User Interfaces. Desweiteren wird auf die Umsetzung der Systemkomponenten eingegangen. Dabei ist es wichtig die im Vorgehensmodell gezogenen Erkenntnisse zu beachten und für die Begründung mit einzubeziehen.

3. Vorgehensmodell

Zuerst wird auf die Wahl des Vorgehensmodells eingegangen, die bereits im Konzept begonnen wurde. Durch weitere Begründungen in den nächsten Abschnitten wird ersichtlich, warum die genannten Vorgehensmodelle eine gute Grundlage für die Modellierung und Implementierung des Systems Meet & Remind bieten.

3.1. DIN EN ISO 9241-210

Diese Norm befasst sich mit dem menschzentrierten Gestaltungsprozess. Dies ist ein Vorgehen, das bei der Entwicklung von interaktiven Systemen eingesetzt werden kann. Dabei soll sich unter Anderem auf die Verwendung des Systems und auf die Gebrauchstauglichkeit konzentriert werden. [Quelle einfügen]
Dieses Vorgehen passt zu dem zu entwickelnden System, da der Benutzer, und auch die Stakeholder im Allgemeinen, eine (im Alltag) unterstützende Anwendung erhalten sollen und ihre Aufgaben (und Ziele) im Bezug zum System effektiv, effizient und zufriedenstellend erledigen können sollen.

In Abbildung 1 wird der Prozess der menschzentrierten Gestaltungsaktivitäten gezeigt. Zuerst muss der Nutzungskontext verstanden und beschrieben werden. Dies geschah für “Meet & Remind” bereits im Konzept. Hier wurden die Domäne recherchiert, Probleme beschrieben, Ursachen gesucht und Stakeholder aufgelistet. Somit wurde der Nutzungskontext genauer untersucht, jedoch ist dieser Teil ein iterativer Prozess und wird ggf. nochmals aufgegriffen und überarbeitet.

Wenn der Nutzungskontext verstanden wurde folgt die Spezifizierung der Anforderungen. Hierbei wird in funktionale, organisationale und technische Anforderungen unterschieden. Diese sind Voraussetzung für den weiteren Prozess und den ersten Gestaltungslösungen.

Bei der Entwicklung von Gestaltungslösungen wird ein weiteres Vorgehensmodell, das “Usage Centered Design” angewendet, um den Verwendungszweck des Systems besser zu verstehen und damit die Benutzung des Systems zu vereinfachen.

Am Ende des Gestaltungsprozess steht die Evaluierung. Hier muss die Perspektive der Benutzer eingenommen werden oder ggf. Tests mit ihnen durchgeführt werden, um zu entscheiden, welche Teilaspekte des Prozess iteriert werden sollten, um am Ende ein gebrauchstaugliches Sytsem zu besitzen.

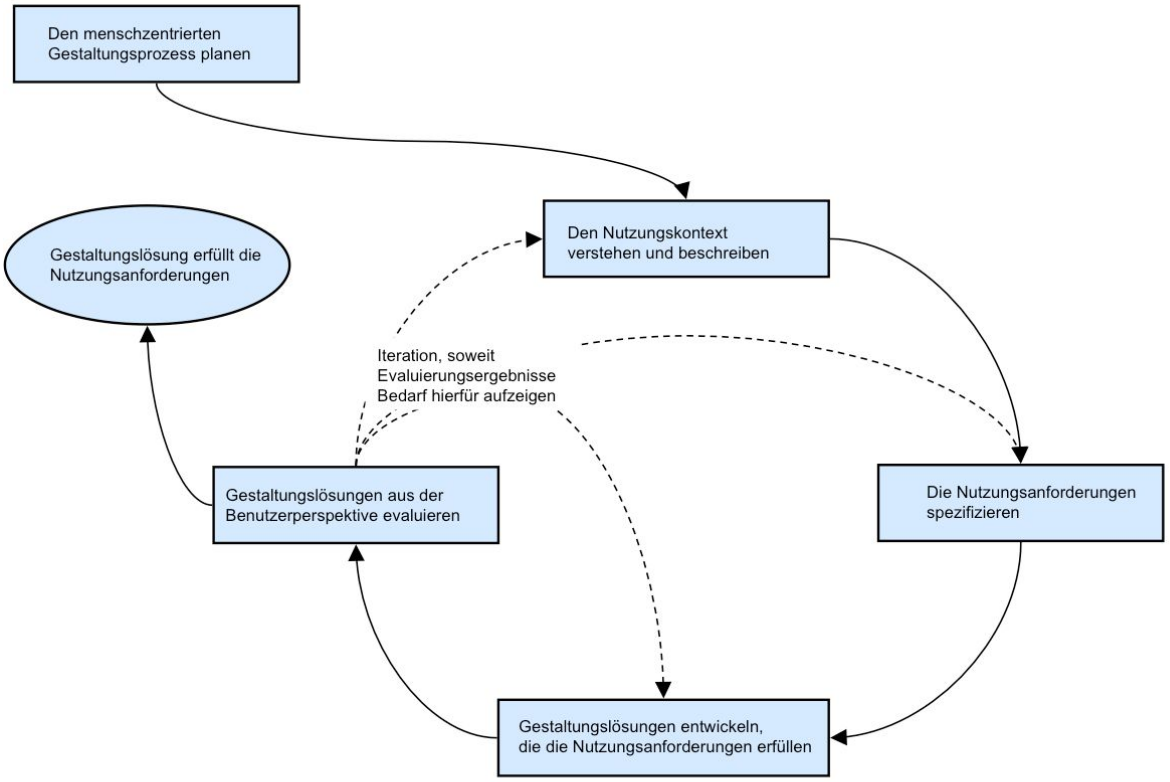


Abb. 1: aus DIN EN ISO 9241-210: Wechselseitige Abhängigkeit menschzentrierter Gestaltungsaktivitäten

3.2. Usage Centered Design

3.2.1. Beschreibung

Das Vorgehensmodell des Usage Centered Design basiert auf der modellgetriebenen Entwicklung in der Software-Entwicklung. Aus dem Namen lässt sich erkennen, dass hierbei vor allem die Einsatzmöglichkeit und Benutzung des Systems im Vordergrund steht. Die drei Modelle Role Model, Task Model, Operational Model und Content Model bilden die Grundlage für die Erstellung des Visual Design im Implementational Model. Außerdem liefern sie Erkenntnisse für die fachlichen Datenmodelle und beeinflussen damit nachhaltig die Systemarchitektur.

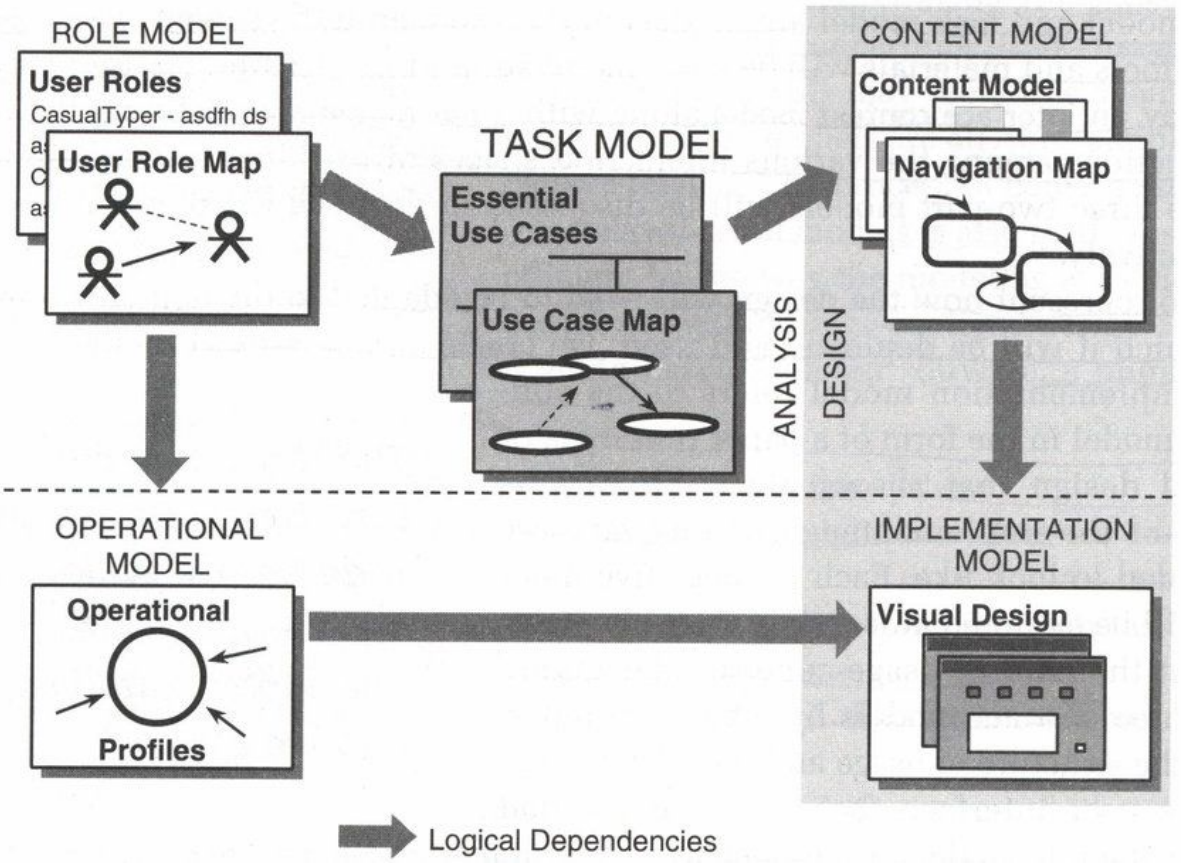


Abb. 2: aus Software for use: Essential models

3.2.2. Begründung

Im Konzept wurde bereits kurz beschrieben, warum dieses Vorgehensmodell positiv zur Entwicklung von Meet & Remind beitragen würde. Zudem lässt sich aus dem vorhergehenden Abschnitt erkennen, dass das Usage Centered Design für den Entwicklungsprozess förderlich ist.
Hier wurde auch das Scenario Based Usability Engineering als passend genannt, welches jedoch bei der weiteren Betrachtung der Modellierung als weniger geeignet eingeschätzt wurde. Die erarbeiteten Problemszenarien für dieses Vorgehensmodell wichen nicht weit genug voneinander ab, als dass man aus ihnen Schlüsse für das visuelle Design ziehen konnte.
Bei der ersten Stakeholderanalyse und dem Aufstellen der Kommunikationsmodelle konnte festgestellt werden, dass sich die Benutzer im Bezug auf das System nicht weit voneinander unterscheiden. Die Benutzer spielen also weniger eine Rolle für das System als die Art der Aufgaben, die sie mit dem System bearbeiten. Aus diesem Grund wurde auch das Vorgehensmodell des User Centered Design ausgeschlossen.

3.2.3. Einsatz

Beim Einsatz des Usage Centered Design wurde auf iterative Anwendung des Vorgehensmodells geachtet. Das bedeutet, dass alle Modellierungen mehrmals durchlaufen wurden und dabei Verbesserungen vorgenommen worden sind. Dadurch konnte sichergestellt werden, dass die Schlüsse aus späteren Modellierungen auch mit in der

Überlegungen für die vorhergehenden Modellierungen genommen wurden. Zur besseren Übersicht wurden die Iterationen in diesem Dokument zusammengefasst.
Zudem wurde die DIN EN ISO 9241-210 mit in die Überlegungen einbezogen, da sich die beiden Modelle gut verbinden lassen. Die Gebrauchstauglichkeit wurde dadurch noch mehr sichergestellt. Im kompletten Prozess wurde jedoch der Fokus auf das Usage Centered Design gerichtet.

4. Erfordernisse

“Eine notwendige Voraussetzung, die es ermöglicht, den in einem Sachverhalt des Nutzungskontexts enthaltenen Zweck effizient zu erfüllen” [QUELLE https://www.dakks.de/sites/default/files/71_sd_2_007_leitfaden_usability_1.3_0.pdf]

Der Cash Flow Manager muss wissen, welche Rechnungen zu welchen Zeitpunkten bezahlt werden sollten, um das Saldo des Firmenkontos über Null zu halten. (Voraussetzung + Zweck)

1. Als Benutzer muss man alle verbundene Kontakte als Liste verfügbar haben, um Erinnerungen für den jeweiligen Kontakt erstellen zu können.
2. Als Benutzer muss man ein Textfeld verfügbar haben, um Erinnerungen zu erstellen.
3. Als Benutzer muss man an erstellte Erinnerungen beim Aufeinandertreffen des jeweiligen Kontakts erinnert werden können, um diese beim Gesprächspartner ansprechen zu können.
4. Als Benutzer muss man Labels zu Erinnerungen hinzufügen können, um passende Gesprächsthemen vorgeschlagen zu bekommen.
5. Als Benutzer muss man mobil und ohne dauerhafte Internetverbindung für erstellte Erinnerungen benachrichtigt werden, um die Sicherheit zu haben, keine Erinnerung zu verpassen.

5. Domain Model

Bei der Vorgehensweise des Usage Centered Design wird die Prozess des Domain Model für die weitere Modellierungsbegründung genutzt. Hier kommen unsere Domänenrecherchen aus dem Konzept zu tragen.

5.1. Domänenmodell

Um einen besseren Überblick für den Modellierungsschritt Role Model zu erhalten, haben wir uns das Domänenmodell zur Domäne “Gespräch” wieder einbezogen. Innerhalb der Erarbeitung des Konzeptes wurde das Domänenmodell bereits iteriert.

Eine weitere ausführliche Iteration ist unserer Meinung nach nicht notwendig, da die Domäne wie in der Abbildung gezeigt bereits genügend Umfang bietet, um ein verteiltes System mit Anwendungslogik zu entwickeln.

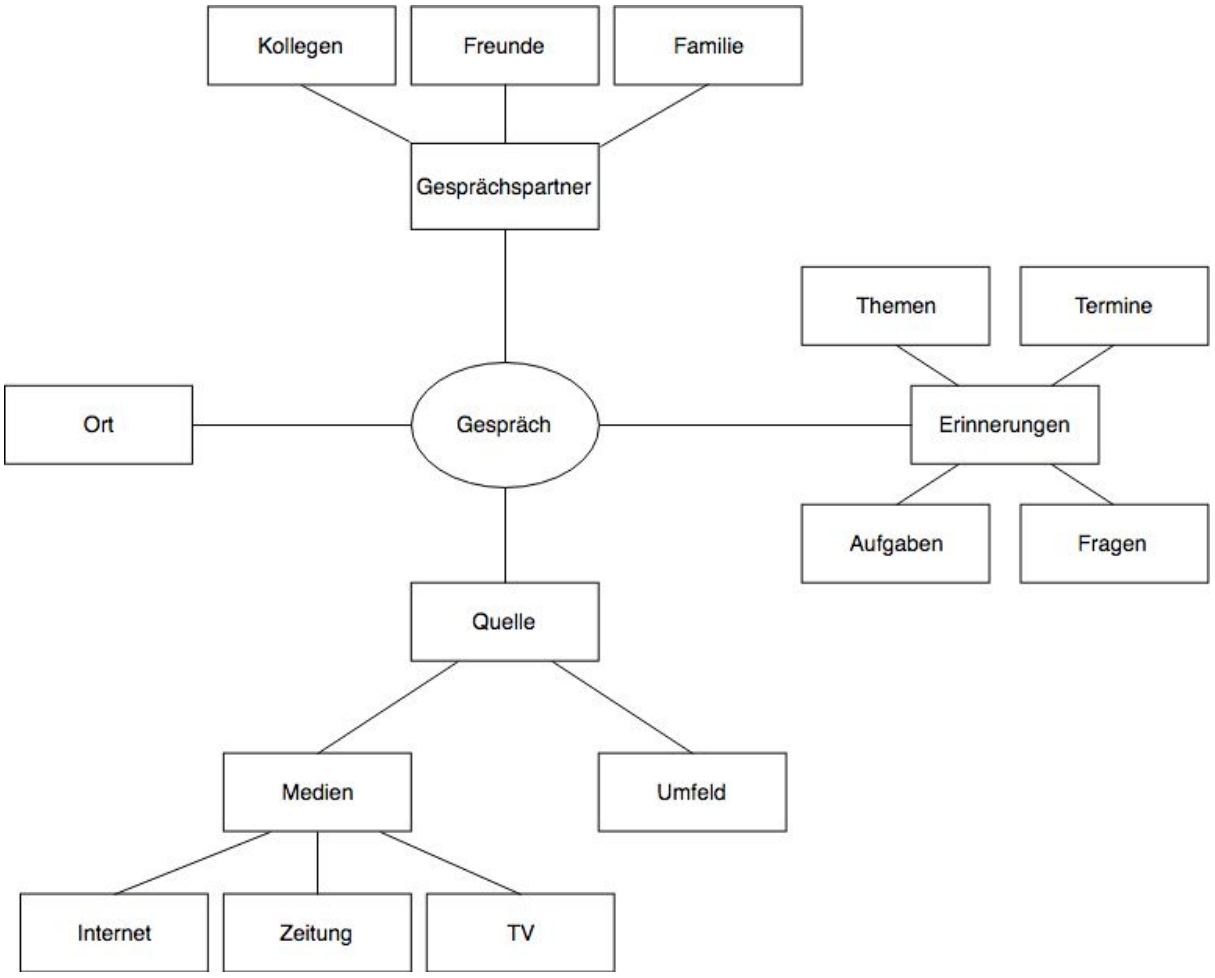


Abb. 3: Domänenmodell zur Domäne “Gespräch”

Aus diesem Domänenmodell können bereits die ersten Schlüsse für ein fachliches Datenmodell gezogen werden. Diese Schlüsse werden in den weiteren Schritten des Vorgehensmodell zur Modellierung mit einbezogen und in der Begründung der Systemkomponenten und Datenhaltung mit einbezogen. Es bildet also den gemeinsamen Wortschatz für den weiteren Verlauf und erste Überlegungen für die Systemstruktur.

5.2. Stakeholderanalyse

Desweiteren bringt uns die Stakeholderanalyse aus dem Konzept Einblicke in die Rollen des Systems. Zusammen mit den Anforderungen und dem Domänenmodell sind hiermit die Grundkenntnisse über die Domäne und die Beziehung zum System vorbereitet um sie im Role Model auszuarbeiten.

5.2.1. Tabelle: Stakeholderanalyse

Bezeichnung	Beschreibung	Beziehung zum System	Priorität für das Projekt
Menschen im Arbeitsumfeld	Benutzer die viel unterwegs sind und dadurch oft Kontakten	Interesse	sehr hoch

	begegnen. Sowohl innerhalb eines Unternehmens, als auch außerhalb (Außendienstmitarbeiter, Berater).		
Menschen mit vielen Kontakten	Benutzer, die durch einen gewissen Umstand (Vermögen, Berühmtheit) viele Kontakte haben und diese Kontakte pflegen wollen.	Interesse	niedrig
Gestresste Menschen	Benutzer, die aufgrund von privatem Umfeld (Eltern, Alleinstehende) oder hohen Verantwortungen (Geschäftsführer, Ärzte) im Alltag viele Dinge im Kopf haben müssen.	Interesse	sehr hoch
Vergessliche Menschen	Benutzer, die wegen ihrem Alter und einer Erkrankung (Demenz, Alzheimer)	Interesse	hoch
Schüchterne Menschen	Benutzer, denen es schwer fällt Gespräche aufzubauen, da sie keinen Einstieg, zum Beispiel in Form von Themengebieten kennen.	Interesse	mittel
Ersteller von Medienprodukten (Wissenschaftler, Journalisten usw.)	Personen, die durch das Veröffentlichen von Medien (z.B. Zeitungsartikel) Themen für Gespräche liefern.	Interesse	mittel
Alle	Datensicherheit: Daten müssen abgesichert sein. Wie Daten gehalten und benutzt werden muss transparent sein.	Anrecht	hoch

5.3. Fazit aus Domain Model

// Wenn uns das zur Argumentation reicht, genügt das. Also hier aufschreiben, dass unsere Recherche aus dem Konzept ausreicht und wir sie ja auch noch über die weitere Modellierungen erweitern.

6. Role Model

Für das Usage Centered Design liegt der Fokus eher auf dem Task Model, aber die Perspektive der Benutzer spielt trotzdem eine Rolle. Es ist wichtig, dass Entwickler und Benutzer in Interaktion treten, damit beide Perspektiven berücksichtigt werden. Nur so kann

ein gutes System gewährleistet werden. Diese Perspektiven werden mit dem Role Model analysiert.

Das Role Model ist in simpler Form eine Liste von Benutzerrollen, beschrieben in Bedürfnisse, Interessen, Erwartungen, Benehmen und Verantwortlichkeiten.
Eine Benutzerrolle ist eine abstrakte Klasse, definiert durch eine bestimmte Beziehung zum System. Es wäre zu grob gesagt, dass sie eine Gruppe von Benutzern zusammenfassen. Denn User Roles sind eine Abstraktion und stellen keine echten Benutzer dar. [QUELLE]

6.1. User Roles

6.1.1. User Roles Vorgehen

Das Vorgehen bei User Roles sieht folgendermaßen aus:

1. Compile: Zuerst wird Brainstorming verwendet, wobei dabei nicht diskutiert werden darf. Alle Ideen sind akzeptiert und es werden Aspekte der User Roles gesammelt, also einzelne Eigenschaften, Bedürfnisse etc.
2. Organize: Der erste Schritt wurde sorgfältig überprüft und nun geht es darum die gesammelten Aspekte zu sortieren und mit aussagekräftigen Titeln zu gruppieren.
3. Detail: Da es nun eine Gruppierung gibt und die entsprechenden Merkmale zugeordnet wurden, können diese nun verfeinert werden und Lücken geschlossen werden.
4. Refine: Nun wird das organisierte und detaillierte Model verfeinert und komplettiert. Hier wird es genauestens überprüft und Kritik geäußert.

Bei den User Roles sollten folgende Fragen beantwortet werden:

1. Wer würde oder könnte das System benutzen?
2. Welcher Klasse oder Gruppe gehören sie an?
3. Was zeichnet sie aus, wie sie das System benutzen?
4. Was charakterisiert ihre Beziehung zum System?
5. Was brauchen sie typischerweise vom System?
6. Wie verhalten sie sich gegenüber dem System und was erwarten sie vom Verhalten des Systems?

6.1.2. User Role Tabelle

Abstrakte Klasse und Wer? (1., 2.)	Wie ist die Benutzung des Systems? (3.)	Beziehung zum System (4.)	Bedürfnisse (5.)	Verhalten gegenüber System (6.)	Erwartetes Verhalten vom System (6.)

Durch Alltagsstress Vergessliche (Berufstätige, Familien, Freunde)	Schnelle und häufige Benutzung, viele Kontakte, viele Erinnerungen	Funktion der Erstellung der Erinnerungen sowie die Funktion der Themen-vorschläge wichtig	Effiziente Bedienung, lieber Symbole als Text	schnell, oft, kurzlebig	Zuverlässigkeit, Schnelligkeit, Gebrauchs-tauglichkeit
Durch Arbeitsstress Vergessliche	Schnelle und häufige Benutzung, Arbeitskollegen als Kontakte, viele Erinnerungen	Funktion der Erstellung der Erinnerungen wichtig, Themenvorschläge unwichtig, da sonst nur Themen über die Arbeit	Effiziente Bedienung	schnell, oft, kurzlebig	Zuverlässigkeit, Schnelligkeit, Gebrauchstauglichkeit
Durch Haushalt Vergessliche	Häufige Benutzung, eher Familie als Kontakte, viele Erinnerungen, zu Hause in Ruhe Erinnerungen erstellen	Funktion der Erstellung der Erinnerungen sowie die Funktion der Themen-vorschläge wichtig	Effiziente Bedienung	in Ruhe, oft, kurzlebig	Zuverlässigkeit, Gebrauchstauglichkeit
Organisatorische Benutzer (Außendienst)	Wohl überlegte Benutzung, ausgewählte Kontakte, Erinnerungen spezifisch.	Funktion der Erstellung der Erinnerungen wichtig, Themenvorschläge eher nicht	Effiziente Bedienung	in Ruhe, wohl überlegt, oft überprüfend	Zuverlässigkeit, Gebrauchstauglichkeit
Krankheits-bedingt Vergessliche (Alte Menschen)	langsame Benutzung, wenig Kontakte, simple Erinnerungen	Nur Funktion der Erstellung der Erinnerungen wichtig.	Viel Text, wenig Symbole, große, gut lesbare Buchstaben mit hohem Kontrast.	langsam, ausgewählte Erinnerungen	Zuverlässigkeit, Einfachheit

			Verständliche, einfach zu erklärende Bedienung.		
Introvertierte Menschen (Schüchterne, ruhige Personen)	Ausgewählte Kontakte, wenige aber ausführliche Erinnerungen	Funktion der Themen-vorschläge besonders hilfreich, da gemeinsame Interessen schnell gefunden werden können.	Fachliche Informationen der Themenvorschläge, keine kritischen “Fake News”	schnell, oft, kurzlebig	Zuverlässigkeit, Schnelligkeit, Gebrauchs-tauglichkeit
Extrovertierte Menschen (viele Freunde, viele Gesprächs-themen)	Viele Kontakte, schnelle und häufige Benutzung, viele Erinnerungen	Funktion der Erstellung der Erinnerungen sowie die Funktion der Themen-vorschläge wichtig, da sie so neue interessante Themen ansprechen, die dem Gegenüber auch wirklich interessieren	Gute Übersicht aller Kontakte, keine Belästigung wegen zu häufigen wieder-kehrenden Erinnerungen, effiziente Bedienung, lieber Symbole als Text	schnell, oft, kurzlebig	Zuverlässigkeit, Schnelligkeit, Gebrauchs-tauglichkeit

6.2. User Role Maps

Die User Role Map verbildlicht die Beziehungen der einzelnen User Roles zueinander. Somit ist auf einem Blick zu erkennen, wer die Benutzer des Systems sind und wie sie dieses nutzen. Die User Roles können auf 3 unterschiedliche Weisen verbunden sein [QUELLE]:

6.2.1. Affinity

Die User Roles besitzen viele Gemeinsamkeiten. Sie besitzen ähnliche Interaktionen, Erwartungen und Charakteristiken. (Dargestellt durch eine gestrichelte Linie und dem Stichwort “resembles”) .

6.2.2. Classification

Manche User Roles stellen eine Unterklasse einer generellen User Role dar. Diese sind eine spezialisierte Version der Oberklasse und besitzen spezifische Eigenschaften, Interaktionen oder Bedürfnisse etc. (Dargestellt durch Pfeil von Unterklasse zur Oberklasse und dem Stichwort: “specializes”).

6.2.3. Composition

Manche User Roles kombinieren die Eigenschaften und Charakteristiken von zwei oder mehr anderen Rollen. Das heißt, dass ein User die Rolle von zwei anderen User Roles einnehmen kann, auch wenn sich diese grundlegend unterscheiden. (Dargestellt durch Pfeil von kombinierter Klasse zu den speziellen Klassen und dem Stichwort “includes”).

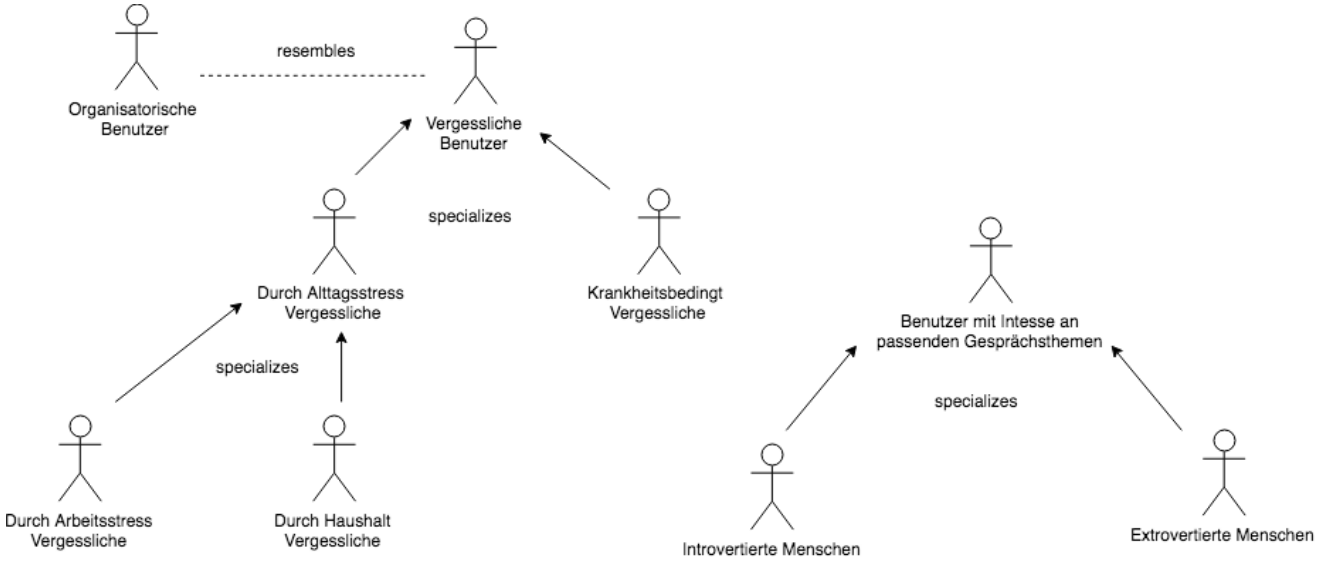


Abb 2: User Role Map

6.3. Fazit aus User Roles

Die User Roles ähneln sich alle dahingehend, dass es normale Benutzer des Systems sind. Es gibt keine Personen mit besonderen Rechten oder Pflichten, wie bspw. bei Planungssystemen.

Grundlegende Unterscheidungen gibt es bei den Benutzern, die vergesslich sind und denjenigen, für die das Gesprächsthema eine wichtige Rolle spielt. Wobei letztere Benutzer erst eine Erinnerung erstellen müssen, damit sie die Funktion der Themenvorschläge nutzen können. Somit ergibt sich ein wesentlicher Use Case, nämlich das Erstellen der Erinnerung und ein weiterer wichtiger Use Case: Themenvorschläge nutzen. Auch dieser basiert aber auf dem Use Case der Erstellung einer Erinnerung.

Ein besonderer Fokus muss auf die krankheitsbedingten Vergesslichen Benutzern gesetzt werden. Diese haben als einzige Benutzerrolle wichtige zu berücksichtigende Bedürfnisse. Das System muss für ältere Personen, die nicht oft mit Technik umgehen, verständlich sein und das User Interface muss für Personen mit Sehschwächen angepasst sein.

Allgemein profitiert das System stark vom Netzwerkeffekt [QUELLE Netzwerkeffekt]. Je mehr Benutzer das System nutzen und Erinnerungen anlegen, desto höher ist der Nutzen für die Benutzer. Dies kann sowohl einen positiven Effekt bewirken, wenn viele Benutzer das System nutzen und somit auch gute Themenvorschläge geliefert werden, als auch einen negativen Effekt geben, falls wenige Benutzer das System für Erinnerungen nutzen oder nur eine der beiden Kommunikationspartner, sodass das System dieser Person keine guten Vorschläge liefern kann.

7. Task Model

Aufbauend auf den Rollen der Benutzer kann die Modellierung über das Task Model vorgenommen werden. Hierin wird beschrieben, wie der Benutzer vorgeht um seine Aufgaben zu erledigen. Bei komplexen Aufgaben (“Tasks”) werden sie in kleine “Subtasks” zerlegt. Dadurch ist es möglich auch große und komplizierte Aufgaben verständlich abbilden zu können. In den folgenden Abschnitten werden diese Aufgaben mittels Essential Use Cases modelliert und sie zueinander in Verbindung gesetzt. [QUELLE Buch Seite 99-100]

7.1. Use Cases

Zunächst werden in diesem Modellierungsschritt Use Cases erarbeitet, anders als bei Szenarien, die die Probleme der Benutzer erzählend umschreiben, geht es bei bei den Use Cases um die konkrete Verwendung des Systems und der Interaktionen mit diesem. “Jeder Use Case beschreibt eine Interaktion, die vollständig, klar definiert und für einige Benutzer von Bedeutung ist” [QUELLE Seite 101]. Hierbei wird der komplette Verlauf der Interaktionen mithilfe einer Liste beschrieben. Folgende Use Cases gibt es:

7.1.1. Kontakt anlegen

User Action	System Responsibility
Suche Geräte in der Nähe	
	Nahe Geräte werden aufgelistet
Gerät auswählen und Kopplung starten	

	Kopplung wird vollzogen
Kopplung bestätigen	
	Geräte verbunden
Name für Kontakt eintippen	
Kontakt speichern	

- Kontakt anlegen
- Erinnerung hinzufügen
- Benachrichtigung zu einer Erinnerung erhalten
- Themenvorschläge nutzen

7.2. Essential Use Cases

Bei den Essential Use Cases geht es dabei noch mehr um den Zweck und Absichten des Benutzers und weniger um die sequentielle Reihenfolge der Aufgabenlösung [QUELLE Seite 103].
Essential Use Cases werden mit hilfe einer Tabelle dargestellt. Über der Tabelle befindet sich der Name des Essential Use Case. Die linke Spalte zeigt die User Intentions, also die Absichten, die der Benutzer bei der Handlung hat. In der rechten Spalte befinden sich die System Responsibilities, die die Antwort vom System aufzeigt, die der Benutzer erwartet [QUELLE Seite 104].

7.2.1. Kontakt anlegen

User Intention	System Responsibility
Ein Gerät in der Nähe finden	
	Nahe Geräte werden aufgelistet
Gerät auswählen	
Mit diesem Gerät koppeln	
	Kopplung wird vollzogen
Name für Kontakt eintragen	
Kontakt speichern	

7.2.2. Erinnerung hinzufügen

User intention	System responsibility

Kontaktliste einsehen	
	Kontaktliste ausgeben
Kontakt auswählen	
	Kontaktdetailseite (Erinnerungen und Themenvorschläge) anzeigen
Erinnerung hinzufügen	
Erinnerungstext einfügen	
Themen-Label setzen	
speichern	

7.2.3. Benachrichtigung zu einer Erinnerung erhalten

User intention	System responsibility
Zwei Kontakte nähern sich	
	Benachrichtigung senden
Benachrichtigung öffnen	
Benachrichtigung lesen	
Mit Kontakt darüber sprechen	
Erinnerung als erledigt markieren	
	Benachrichtigung wird nicht erneut versendet

7.2.4. Themenvorschläge nutzen

User intention	System responsibility
App vor einem Treffen öffnen	
	Kontaktliste wird angezeigt
Kontakt auswählen	
	Kontaktdetailseite (Erinnerungen und Themenvorschläge) anzeigen
Themenvorschlag ansehen	

7.3. Use Case Maps

// Beziehungen der Elemente untereinander als Karte modelliert

7.4. Fazit aus Use Cases

Aus den Use Cases und Essential Use Cases gehen die Anforderungen an das System hervor. Durch die erarbeiteten Einsatzmöglichkeiten von Meet & Remind und die Ziele, die der Nutzer bei der Benutzung hat, können zusammen mit den Anforderungen dann Erkenntnisse für den Content Model und Implementational Model gezogen werden.
Hierbei ist auffällig, dass es nur wenige Use Cases gibt. Dies konnte bereits im Role Model erahnt werden, da sich die Rollen der Benutzer sehr ähneln.
Mit der Modellierung des Task Model wurde die Wahl des Usage Centered Design als Vorgehensmodell zudem bestätigt, da die Benutzung und Benutzbarkeit bei einem so kompakten System und viele Benutzerrollen im Vordergrund steht. Der Fokus kann also weiterhin auf diese Aspekte gesetzt werden.

8. Anforderungen

Die Anforderungen werden von den Erfordernissen und den Use Cases abgeleitet und werden im späteren Verlauf eingesetzt um bei der Evaluierung des System zu prüfen, ob die gestellten Anforderungen auch erfüllt wurden.
Anforderungen können bspw. in funktionale, organisationale, qualitative und technische Anforderungen unterteilt werden. Dabei ist bei den funktionalen Anforderungen das folgende Schema zu beachten:

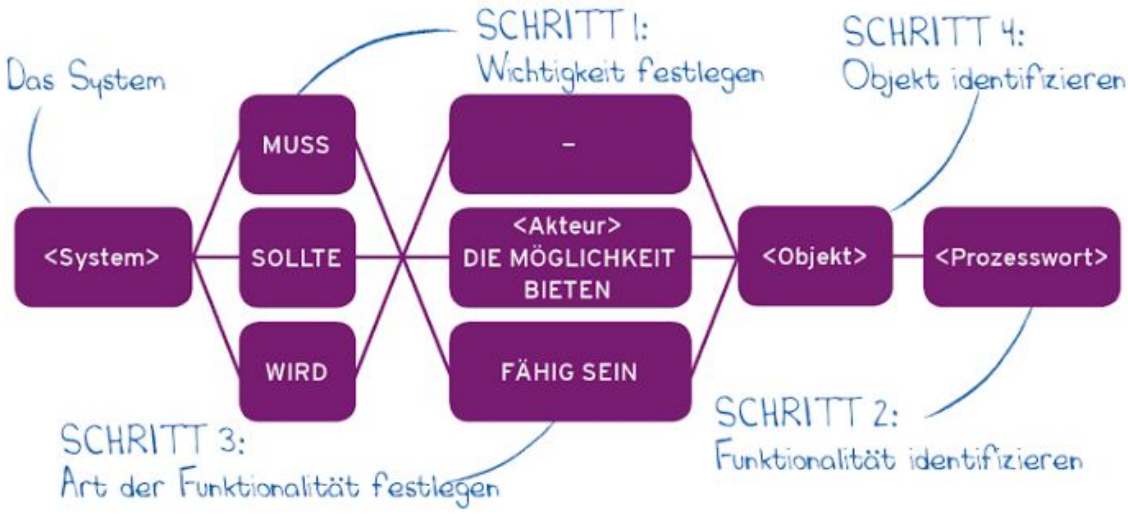


Abb. 3: Anforderungsschablone

[QUELLEN von Abbildung und Schema]

Hierbei ist die Unterscheidung von Muss, Sollte und Wird wichtig. “Muss” sind Mindestanforderungen an das System, “Soll” bedeutet, dass die Funktion nicht zwingend notwendig ist und “Wird” stellt einen Ausblick dar.

Es wurde eine Listenansicht für die Anforderungen verwendet, um die Übersicht zu gewährleisten und bei der Evaluierung die Anforderungen der Reihe nach abhaken zu können. Dabei wurde jedoch darauf geachtet vollständig ausformulierte Sätze zu verwenden.

Außerdem müssen bei den Anforderungen die Qualitätsanforderungen beachtet werden:
Vollständig, wenn sie umfangreich Information liefern.
Notwendig ist eine Anforderung, wenn sie zur Zielerreichung beiträgt.
Atomar, wenn eine Anforderungsformulierung ein Vollverb besitzt.
Verfolgbar, wenn die Anforderungen nummeriert sind.
Technisch lösungsneutral, damit im Vorfeld keine Einschränkungen gemacht werden.
Realisierbar, also innerhalb der möglichen Grenzen.
Konsistent, also widerspruchsfrei.
Eindeutig, also nicht missverständlich.
Prüfbar, also messbar oder es können Tests durchgeführt werden.

[QUELLEN von diesen begriffen]

8.0.1. Funktionale Anforderungen

1. Das System muss dem Benutzer die Möglichkeit bieten sich mit einem anderen Benutzer koppeln zu können.
    1.1. Das System muss eine Liste aller verfügbaren Geräte zum Koppeln anzeigen.
    1.2. Das System muss dem Benutzer die Möglichkeit bieten ein Gerät zum Koppeln auszuwählen.
    1.3. Das System muss auf beiden Geräten einen Dialog starten, ob die Kopplung erwünscht ist.
2. Das System muss fähig sein zu erkennen, wenn sich zwei bereits gekoppelte Geräte in der Nähe befinden.
3. Das System muss eine Übersicht aller bereits gekoppelten Geräte bieten.
4. Das System muss dem Benutzer die Möglichkeit bieten ein bereits gekoppeltes Gerät auszuwählen, um für dieses Gerät eine neue Erinnerung zu erstellen.
5. Das System muss dem Benutzer die Möglichkeit bieten Erinnerungen zu erstellen.
    5.1. Das System muss mindestens ein Textfeld zur Erstellung bieten.
    5.2. Das System muss die Möglichkeit bieten, Themen-Labels zu der jeweiligen Erinnerung auszuwählen.
6. Das System muss dem Benutzer die Möglichkeit bieten eine Erinnerung zu bearbeiten.
7. Das System muss fähig sein dem Ersteller die jeweilige Erinnerung als Benachrichtigung anzuzeigen, sobald er sich der zur Erinnerung zugehörigen Person genähert hat.
8. Das System muss erkennen, wenn die Gesprächspartner in einem gewissen Zeitraum dieselben Themen-Labels gesetzt haben.
9. Das System muss bei gleichen Themen-Labels beiden Gesprächspartnern denselben informativen Text (Artikel etc.) zur Verfügung stellen.
10. Das System muss dem Benutzer die Möglichkeit bieten eine Erinnerung als “Erledigt” zu markieren.

10.1. Das System soll bei nicht-erledigten Erinnerungen die Benachrichtigung beim nächsten Aufeinandertreffen erneut schicken.

8.0.2. Organisationale Anforderungen

1. Das System muss im Play Store für alle Android Geräte ab Version 7.0 verfügbar sein.
2. Das System muss sowohl auf Tablet als auch auf Smartphones nutzbar sein.

8.0.3. Qualitative Anforderungen

1. Das System muss problemlos das Koppeln ermöglichen.
2. Das System muss in jeder Situation Reaktionszeiten von unter einer Sekunde haben.
3. Das System darf nicht unerwartet abstürzen oder einfrieren.
4. Das System muss die Datensicherheit beachten.
5. Das System soll gebrauchstauglich im Sinne der Benutzer sein.

8.0.4. Anforderungen an die Benutzungsoberfläche

1. Das System muss auch für die Zielgruppe der älteren vergesslichen Leute gut lesbar sein.

8.0.5. Technische Anforderungen

1. Das System muss einen Zuweisungs-Algorithmus der Themen-Labels auf Serverseite besitzen.
2. Der Server soll eine Schnittstelle besitzen, die der Client nutzen kann.
3. Der Server soll Zugriff auf eine Datenhaltung besitzen.
4. Die Datenhaltung soll unabhängig vom Client sein, sodass bei Umstellung des Datensystems nur die Serverseite angepasst werden muss.
5. Die Kommunikation zwischen Client-App und Server, sowie zwischen Server und Datenhaltung, muss sicher sein.

8.1. Fazit aus Anforderungen

Die Anforderungen geben an, was vorausgesetzt wird, damit der Benutzer seine Ziele mit dem System bewerkstelligen kann. Sie bilden die Grundlage für die weiteren Modellierungen und liefern einen tieferen Einblick über den Nutzungskontext des Systems.
Aus den Anforderungen können zudem Schlüsse für Systemarchitektur und Datenstruktur gezogen werden, die durch die weiteren Schritte des Vorgehensmodells überprüft werden. Hier können die Anforderungen bei der Evaluierung genutzt werden, um die modellierten Bestandteile des User Interfaces zu prüfen.

9. Einbeziehung der Technologie

9.1. Begründung für das Einbeziehen

Für den Schritt des Content Model wird ein Einbeziehen der Technologie notwendig. Aus den Anforderungen und den Proofs of Concept aus dem Konzept geht hervor, dass

Bluetooth die notwendige Grundlage für die personenbezogenen Erinnerungen von Meet & Remind liefert. Auch aus dem Task Model geht hervor, dass wir die Funktionalität benötigen, die im Rapid Prototype bereits überprüft wurde. Als Alleinstellungsmerkmal fungiert Bluetooth für die personenbezogene Erinnerung über die Kernfunktion des Systems. Um diese Funktionalität bei der Modellierung des User Interfaces einbeziehen zu können, setzen wir sie als Grundlage für die weiteren Begründungen. Dabei werden diese iterativ im Abschnitt der Systemkomponenten geprüft und optimiert.

9.2. Schlüsse aus der Modellierung der Systemkomponenten

Aus der Modellierung der Systemkomponenten und der genannten Ergebnisse aus dem Rapid Prototype geht hervor, dass ein Client-Server-System implementiert wird. Da als Client eine Android App fungiert, muss dies im weiteren Prozess des Vorgehensmodell bedacht werden. Einerseits wird der Prozess dadurch eingeschränkt, dass die einzige Benutzerschnittstelle eine Android App darstellt, andererseits können damit die Betriebssystem-Spezifischen Gegebenheiten berücksichtigt werden.

10. Content Model

Das Content Model zielt darauf ab eine abstrakte Repräsentation der Inhalte der verschiedenen Interaktionsräumen zu erstellen. In diesen Interaktionsräumen geschieht die Ausführung der Benutzeraufgaben.
Die Navigation Map bildet ab, wie sich der Benutzer zwischen den einzelnen Interaktionsräumen im späteren System bewegen können soll.

Im Buch “Software for use” werden die Begriffe Content (Inhalt) und Context (Zusammenhang) gleichgestellt. Der Nutzungskontext ist also der Interaktionsraum und in diesem befindet sich der Content, also Funktionen und Materialien.

10.1. Content Model

Ein abstrakter Prototyp unterstützt die Designer zuerst darüber nachzudenken, was im User Interface benötigt wird, bevor sich damit auseinandersetzt wird, wie es aussehen oder wie es sich Verhalten wird.

Die Technik die von Constantine und Lockwood vorgeschlagen wird, umfasst ein Blatt Papier sowie Post-Its. Somit ist gewährleistet, dass sich der Designer noch keine bildlichen Vorstellungen macht. Lediglich die Funktionen und Materialien werden abgebildet, sodass Designentscheidungen erst später getroffen werden müssen. Dies hilft dabei das User Interface einfach zu halten und sicherzustellen, dass wirklich nur die erforderlichen Inhalte Bestandteil sind.

Das Vorgehen beginnt mit einem leeren Papier. Hier wird der Interaktionsraum aussagekräftig benannt. Namen wie “Main Screen” sollten vermieden werden.
Danach werden die verschiedenfarbigen Post-Its beschriftet. Die Funktionen (Tools) sind die aktiven, vom Benutzer steuerbaren Elemente. Diese können, wenn gewünscht, nach der im Buch beschriebenen Konvention in warmen Farben, wie pink, orange und gelb dargestellt

werden. Dies ist jedoch keine Vorschrift, sondern wird vorgeschlagen, um ein einheitliches Schema zu haben.
Die Materialien (Materials) sind Daten, Container oder simple Anzeigen, mit denen der Benutzer passiv handelt. Diese können in kalten Farben wie blau oder grün dargestellt werden.

Mit diesem Vorgehen werden die verschiedenen Interaktionsräume, die sich aus den Use Cases ergeben, abstrakt dargestellt. Das Content Model bietet die Grundlage für spätere Designentscheidungen und stellt einen gewissen Grad der Gebrauchstauglichkeit sicher.

10.2. Content Models für Meet & Remind

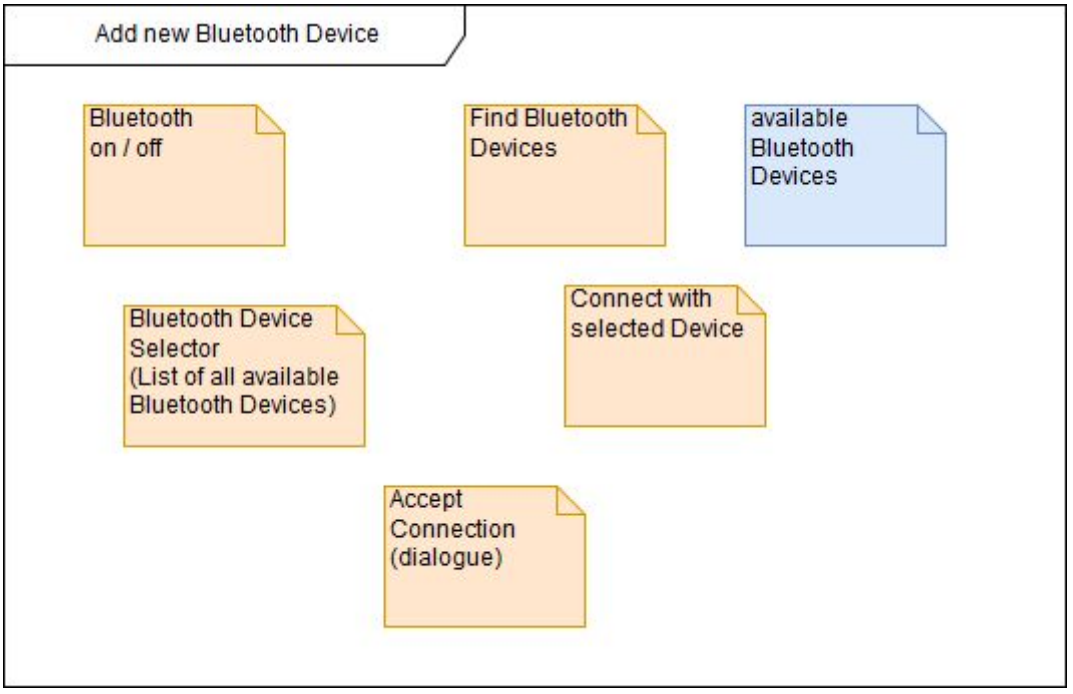


Abb. X

Dieser Interaktionsraum wurde bereits im Proof of Concept implementiert. Dabei ging es nur um die Funktionalität der Bluetooth Kopplung. Im PoC gab es eine Funktion, die die Bluetooth Sichtbarkeit des eigenen Geräts ein- oder ausschaltet. Beim Gestalten des Models ist aufgefallen, dass diese Funktion nicht zwangsweise benötigt wird, da der Benutzer beim Einschalten von Bluetooth davon ausgehen wird, dass sein Gerät für Andere sichtbar ist. In der Praxis ist dies nicht automatisch gewährleistet und es kann sein, dass diese Funktion technisch bedingt doch notwendig ist.

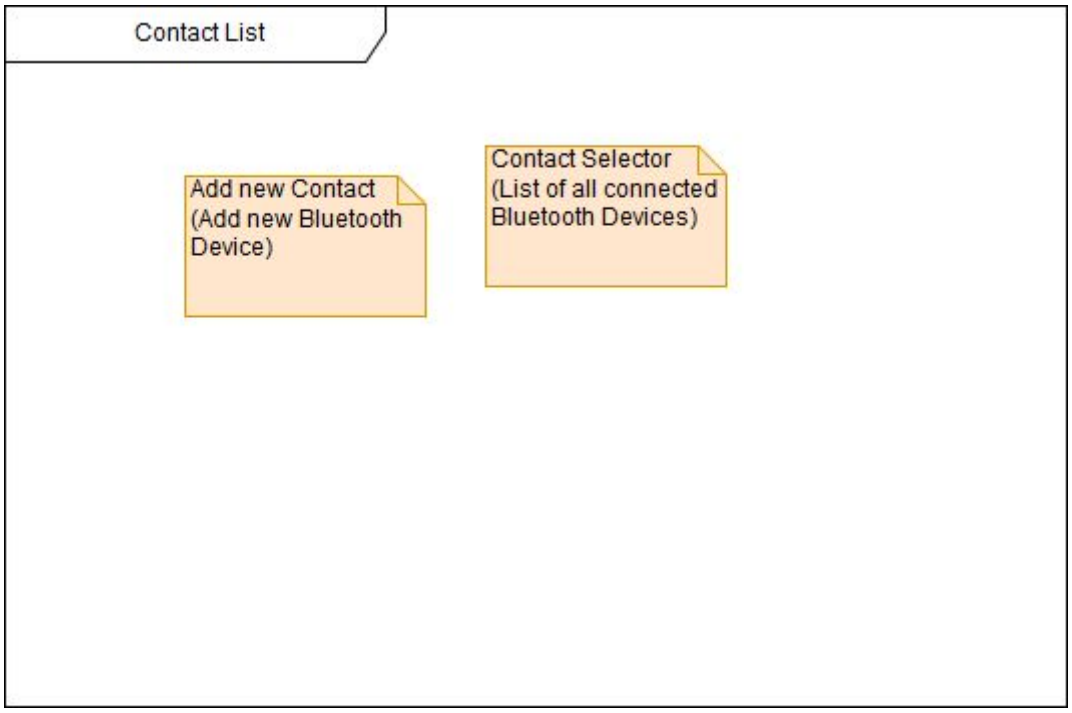


Abb. X

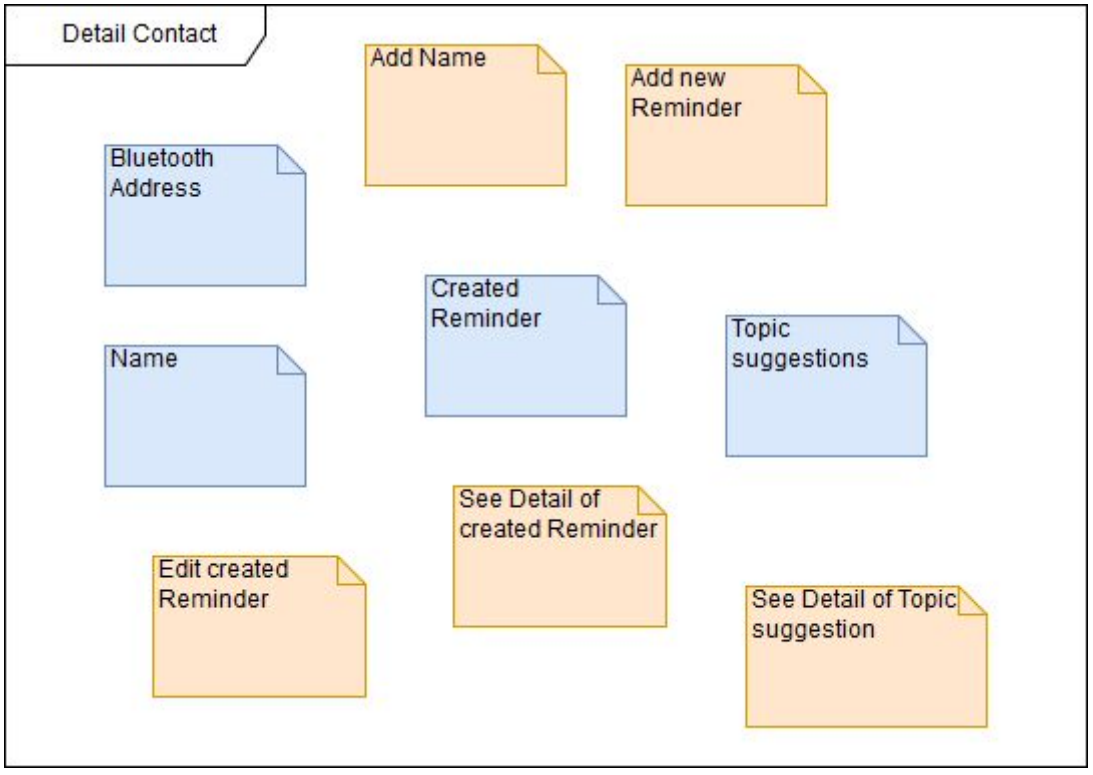


Abb. X

// Detail Suggestion begründen extra Frame ja nein

Hier ist eine neue Anforderung hinzugekommen: Das System muss dem Benutzer die Möglichkeit bieten der Bluetooth Adresse einen Namen hinzuzufügen. Andernfalls müsste sich der Benutzer merken, welche Person zu welcher kryptischen Adresse gehört.

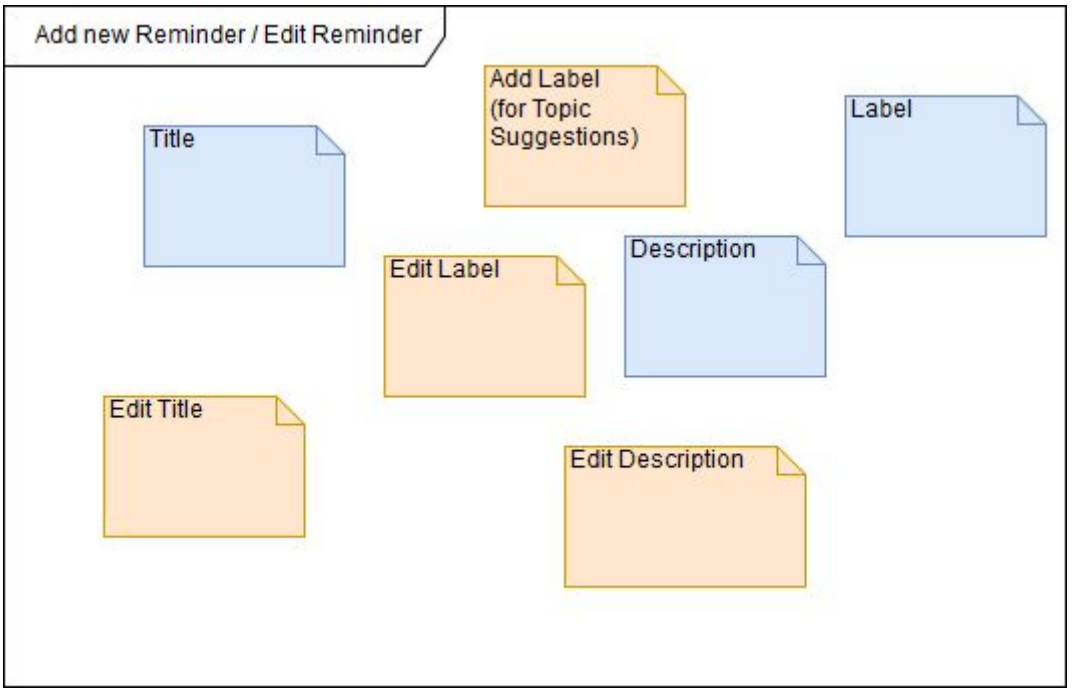


Abb. X

Erst bei der Erstellung des Models ist aufgefallen, dass in den Anforderungen auch noch die Funktionen des Editierens hinzugefügt werden müssen.

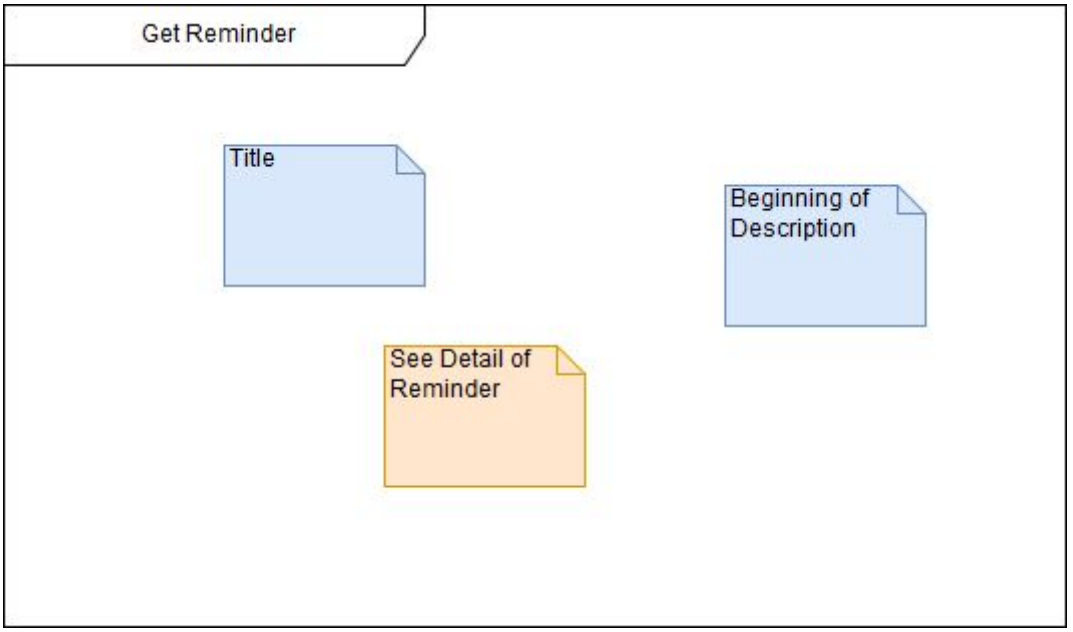


Abb. X

Dieser Interaktionsraum befindet sich nicht direkt in der Applikation, sondern als Benachrichtigungsfenster auf dem Smartphone. Von dort aus kann jedoch direkt auf die App zugegriffen werden.

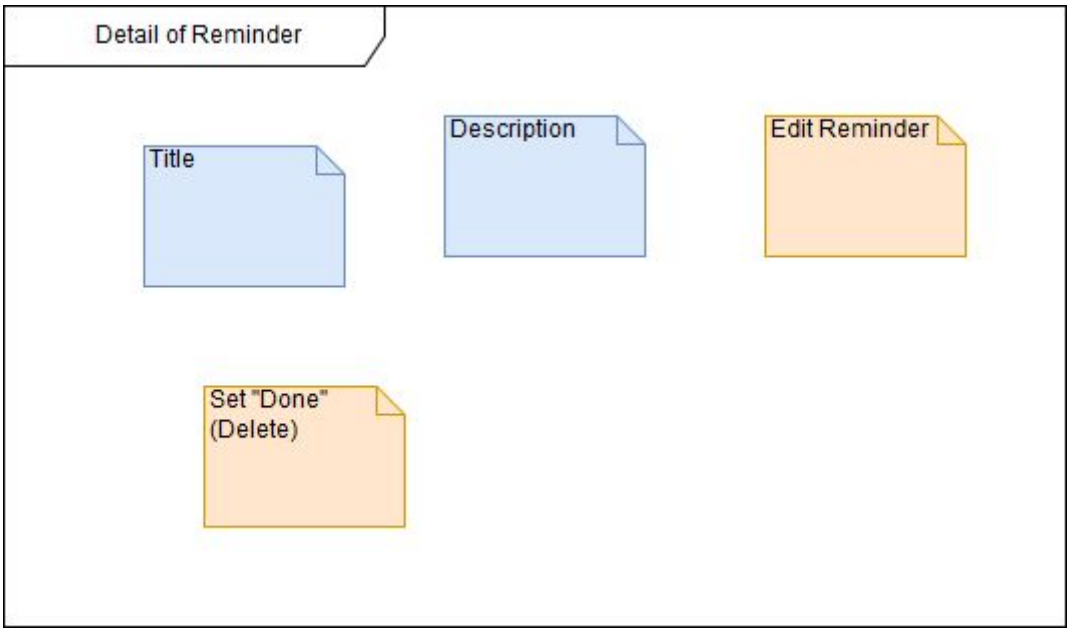


Abb. X

Hier gibt es die zusätzliche Funktion die Erinnerungen als “Erledigt” zu markieren. Damit wird die Erinnerung gelöscht. Wird dies nicht markiert, setzt die Anwendungslogik ein, die Erinnerung beim nächsten Mal erneut zu schicken.

10.3. Context Navigation Map

Das User Interface sollte einen einfachen und effizienten Workflow bieten. Daher wird in der Context Navigation Map modelliert, wie sich die Benutzer von einem Interaktionsraum zum anderen navigieren, um ihre Aufgaben, die in den Use Cases beschrieben sind, zu erfüllen. Die komplette Struktur der User Interface Architektur soll abstrakt dargestellt werden.

Bei den Interaktionsräumen soll beachtet werden, dass der Kontext zwar klein und einfach gestaltet sein sollte, dies aber im Endeffekt komplex werden kann, da es mehr Interaktionsräume gibt, zwischen denen navigiert werden muss.
Andererseits sind wenige, komplexe Interaktionsräume auch nicht zielführend.
Hier muss ein gutes Mittelmaß gefunden werden.

Für die Context Navigation Map wird ein Diagramm erstellt, in dem Rechtecke die Kontexte abbilden und Pfeile die Beziehung zwischen ihnen, also die möglichen Übergänge, die ein Benutzer (oder das System) auslösen kann.

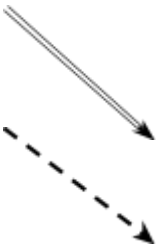
Für unser System sind folgende Konventionen, die zum Teil von uns ergänzt wurden, zu gebrauchen:


Jeder Interaktionsraum


Screen oder Anzeige


10.4. Dialogfenster oder Nachricht

Übergang zwischen Interaktionsräumen ausgelöst durch “Aktion”.
Unterscheidungen: Menü Auswahl: View | Toolbars
Button oder Listenauswahl: [Apply], [Kontakt]
Icon oder Tool Auswahl: <page width>
Vom System ausgelöst: “Erfolg”


Übergang mit impliziertem “Return”.
Kurzschreibweise, damit nicht zwei Pfeile gemacht werden müssen, wenn eine Rückkehr durch Return oder Ereignisse wie OK oder Cancel möglich sind.
Aus technischen Gründen ist im untenstehendem Diagramm der Pfeil gestrichelt.

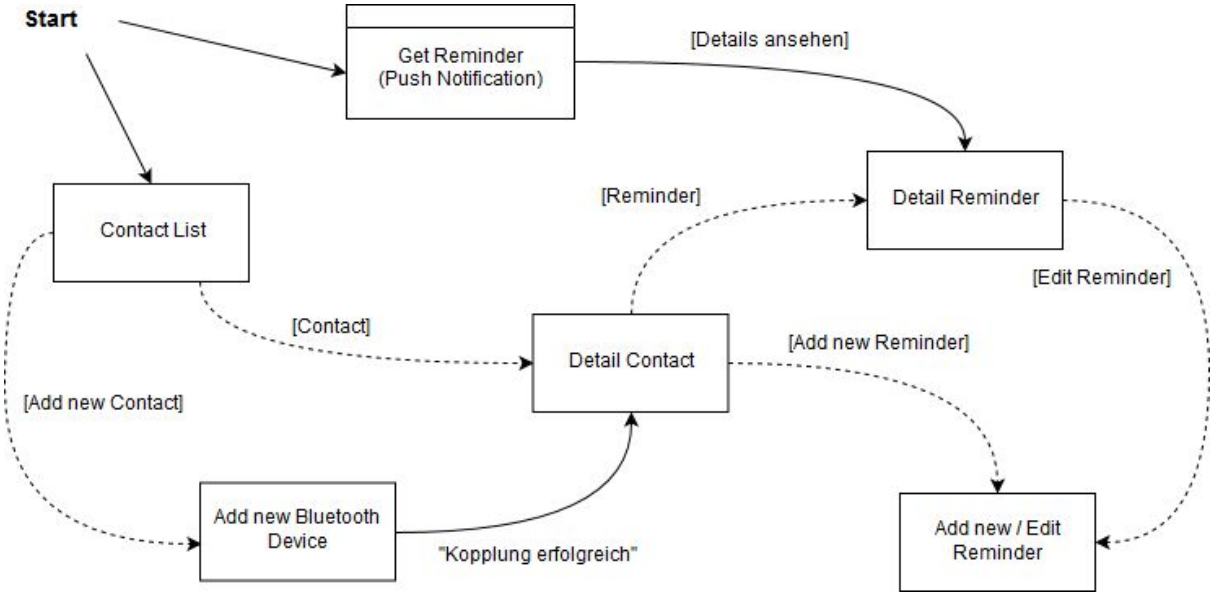


Abb. X:

10.5. Exkurs Mobile Computing

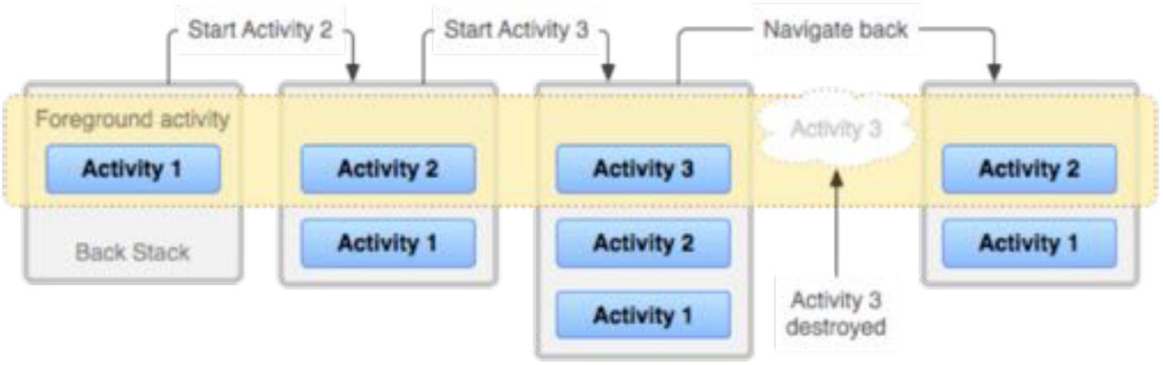


Abb. X:

In der Abbildung ist der Navigation Stack von Android abgebildet. Dieser bewirkt, dass bei einem Return der Benutzer automatisch zur vorherigen Activity geleitet wird.
Die Activities in der Android Entwicklung sind übertragen auf das Content Model die Interaktionsräume.
Die Navigation Map stellt durch die Übergänge mit impliziten Return bereits den standardmäßigen Navigation Stack dar.
Bei den Übergängen ohne impliziten Return muss in der späteren Implementierung der Navigation Stack überschreiben werden, sodass der Benutzer zu einer anderen Activity geleitet wird.

10.6. Fazit aus Content Model

Das Content Model war sehr hilfreich, da die Anwendungsfälle genauer betrachtet wurden und sich darüber Gedanken gemacht wurde, welche Funktionen in den einzelnen Interaktionsräumen notwendig sind. Bei der Navigation Map kann gut eingesehen werden, wie sich später der Benutzer durch das System navigieren wird.

Insgesamt ist das Content Model, sowie die Navigation Map, sehr gut übertragbar für Android. Die Interaktionsräume werden am Ende die Activities der App und die Übergänge müssen später in der Implementierung etwas angepasst werden.
Mit dem Content Model wurden diese Erkenntnisse relativ früh erfasst und können nun in der Implementierung einfach beachtet werden.

11. Implementation Model

In diesem Schritt soll das Content Model, in dem die Funktionen des Systems abstrakt beschrieben wurden, nun in das Implementation Model transformiert werden. Das Implementation Model ist eine Repräsentation wie das finale User Interface aussehen und funktionieren wird. Dabei wird jedoch im Buch “Software for Use” betont, dass dies “Creative Interface Engineering” ist und nicht bloß Grafikdesign. Denn beim Usage Centered Design geht es darum, die Funktionen gebrauchstauglich zu gestalten. Und dies passt eher zu den Ingenieurtätigkeiten, da es darauf ankommt praktische Ziele zu erreichen und die Leistung zu gewährleisten.

Bevor dieser Prototyp begonnen werden kann, müssen zwei Fragen beantwortet werden:

11.1. Frage 1: Wie werden die Interaktionsräume repräsentiert?

Im Content Model wurden die abstrakten Interaktionsräume erstellt, in denen der Benutzer handeln kann. In der Context Navigation Map wurde bereits festgelegt, dass nur der Interaktionsraum “Erinnerung erhalten” von den anderen abweicht. Dieser stellt in Android eine Push Benachrichtigung dar und liegt sozusagen außerhalb des “Application Context”. Alle anderen Interaktionsräume sind die einzelnen Activities in der App - Also Screens.

Da im Content Model das Design und die Funktionen auf abstrakter Ebene dargestellt wurden, war noch nicht klar, wie “Detail des Themenvorschlags” später repräsentiert würde. Hier gab es mehrere Optionen: Entweder ein Dialogfenster, das in den Vordergrund rückt, oder innerhalb der Detail Kontaktseite, oder der Einfachheit halber ein externer Link, oder ein separater Screen. Es wurde sich für einen weiteren Interaktionsraum entschieden, damit das Design konsistent ist und der Benutzer nicht aus der App geleitet wird.
Außerdem soll der Benutzer direkt von der gesamten Kontaktliste einsehen können, wo ein Themen-Matching erfolgreich stattgefunden hat, damit er nicht bei jedem Kontakt einzeln danach suchen muss.

Des Weiteren wurden bereits im Content Model die Interaktionsräume “Erinnerung bearbeiten” und “neue Erinnerung erstellen” zusammengefasst. Bei genauer Betrachtung ist aufgefallen, dass “Detail Erinnerung” kein zusätzlicher Interaktionsraum sein muss, da man ihn mit “Erinnerung bearbeiten” ebenfalls zusammenfassen kann.
Somit hat sich dann ergeben die Funktion “Als Erledigt markieren (löschen)” innerhalb des Interaktionsraumes “Detail Kontakt” zu der jeweiligen Erinnerung zu legen.

11.2. Frage 2: Wie werden die Komponenten repräsentiert?

Jede abstrakte Komponente innerhalb jeden Interaktionsraumes muss in eine visuelle Komponente überführt werden.

Listen:
Listen jeglicher Art, ob für die Kontakte, die Bluetooth Geräte oder die Erinnerungen, sollen Listen sein, in denen die Elemente anklickbar sind. Wenn auf diese geklickt werden, soll der

Benutzer zum nächsten Screen geleitet werden, bzw. bei Bluetooth die Kopplung eingeleitet werden.

Neuer Kontakt/ Neue Erinnerung:
Diese Funktionen sollen gut sichtbar und erreichbar für den Benutzer sein, weshalb sich Floating Buttons gut dafür eignen. Sie schweben über den eigentlich Inhalt des Screens und heben sich damit ab.
Um den Unterschied schneller zu erkennen sollen hier zwei verschiedene Icons verwendet werden.

Name:
Der Standard-Name des Bluetooth Gerät soll auf der Kontakt Detailseite in der Toolbar zu sehen sein. Sozusagen als Überschrift. Daneben soll ein Tool, in Form eines Icons, zum Bearbeiten des Namens bereitgestellt werden.

Neue Erinnerung/ Erinnerung bearbeiten:
Hier müssen für Titel und Beschreibung Eingabefelder eingefügt werden. Der Benutzer muss seine Erinnerungen in textlicher Form anlegen können.
Bei der Auswahl eines Themen-Labels gab es die Diskussion, ob nur eins oder mehrere ausgewählt werden können. Für das System und der damit verbundenen Anwendungslogik, sowie für die Praxisrelevanz, reicht es aus, wenn der Benutzer ein Label auswählt. Somit soll die Auswahl mit Radio Buttons markiert werden. Eine Alternative wäre ein Dialogfenster, aber mit dieser Variante müsste der Benutzer einen zusätzlichen Klick machen. Dies wäre etwas unvorteilhaft, da viele Benutzer die Themenvorschläge nutzen sollen, damit häufige Matches zustande kommen.

Bluetooth on/off:
Hier bietet sich, wie gewohnt von den Bluetooth Einstellungen, ein Switch Button an.

Nach verfügbaren Bluetooth Geräten suchen:
In den normalen Bluetooth Einstellungen, geschieht dies automatisch, aber anhand technischer Schwierigkeiten, wird es einfacher sein, einen Button bereitzustellen, der die Suche triggert.

11.3. Visual Design

Der Prototyp schließt die Lücke zwischen Content Model und Implementierung. Anhand eines Prototypen wissen Entwickler eher was zu tun ist und haben genaue Angaben. Außerdem liefern Prototypen in der Regel bessere, gebrauchstaugliche Systeme.

Es gibt drei Kategorien mit je zwei Arten bei Prototypen: Action, Fidelity und Orientation. Aus jeder Kategorie ist eine Art auszuwählen.

Action:
Ein **passiver** Prototyp ist eine simple Zeichnung auf Papier oder digital. Es ist ein Mock-up, das keine Funktionalitäten bietet.

Aktive Prototypen bieten in einem bestimmten Grad gewisse Funktionen. Sie Verhalten sich wie das implementiere Endergebnis.

Fidelity:
High-fidelity Prototypen sehen dem später implementierten User Interface sehr ähnlich und viele Designentscheidungen wurden bereits getroffen.
Low-fidelity Prototypen sehen nur ungefähr so aus, wie das Endprodukt. Sie geben nur eine grobe Vorstellung.

Orientation:
Horizontale Prototypen sind eine oberflächliche Repräsentation des User Interface. Das heißt, dass Funktionen keine Rolle spielen.
In Kombination mit einem aktiven Prototypen heißt dies, dass ein Dialog zwar gestartet wird und auch wieder geschlossen werden kann, aber im Hintergrund keine Funktion ausgeführt wird.
Vertikale Prototypen spiegeln das komplette Design eines Segments wider, sowie all seine benötigten Funktionen. Als PoC wurde bspw. ein Prototyp entwickelt, der die Funktion der Bluetooth Kopplung dargestellt hat.

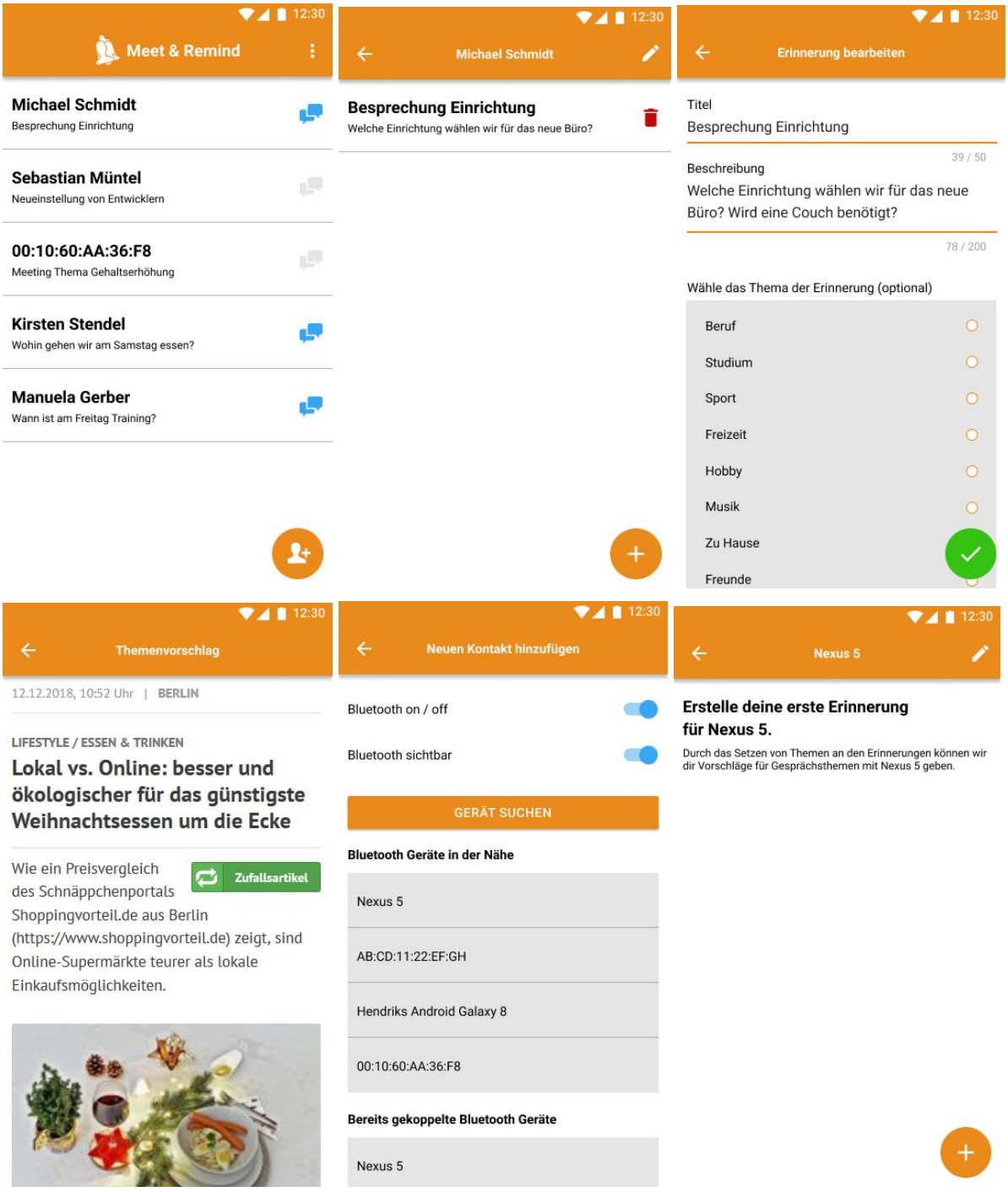
Fazit:
Der zu entwickelnde Prototyp soll ein aktiver Prototyp sein, in dem Sinne, dass der Benutzer zwischen den Screens navigieren kann und ggf. Dialoge geöffnet werden. Dies kann mit Figma gut realisiert werden, da auch Overlays dargestellt werden können.
Außerdem soll der Prototyp high-fidelity sein, damit wir ein fertiges Design haben, an dem wir uns in der Implementierung halten können, ohne ständig nachfragen zu müssen oder eigene Designentscheidungen, ohne Abstimmung, machen zu müssen.
Er wird eine oberflächliche Repräsentation des User Interface sein, da hier erstmal nur das Design der Funktionen eine Rolle spielt und nicht die Funktionalität selbst.

11.3.1. Grundlegende Designentscheidungen

Bevor mit dem Prototypen angefangen wird, haben wir uns auf grundlegende Designentscheidungen geeinigt, die einzuhalten sind. Einige davon werden auch im Buch “Software for use” erläutert.

1. Das Design soll insgesamt modern und zeitgemäß sein.
2. Genug Weißraum lassen, damit das Design aufgeräumt wirkt.
3. Nicht mehr als zwei verschiedene Schriftarten auf einem Screen.
4. Vermeidung von “under-forty fonts”. Also die Schrift muss für alte Menschen gut lesbar sein. (Siehe User Roles)
5. Nicht mehr als drei verschiedene Farben auf einem Screen.
6. Die wichtigsten Navigationselemente befinden sich immer an der selben Stelle.
7. Icons tragen dazu bei, dass das Design aufgeräumt wirkt; müssen jedoch für alle Benutzer verständlich sein. Es werden nur vorgefertigte Icons benutzt, die allgemein bekannt sind und einheitlich passen.
8. Ein Hamburger Menu ist nicht notwendig, da es keine Menüpunkte gibt, außer Impressum und Datenschutz.

11.4. Prototypen



https://www.figma.com/proto/U7WRXg3duUYbBRXB06qqsIuE/Prototype?node-id=1%3A2&scaling=scale-down

//Designerläuterungen
//Warum Orange? Warum Blau? Komplementärkontrast.
//Warum grünes Häkchen nochmal andere Farbe? Aufmerksamkeit wegen Abspeichern.

//Icons von Google

12. Evaluierung

12.1. Usability Inspection

// Bewertungsbogen Seite 550

12.2. Redesign

// gegebenenfalls
// schwächen erläutern

13. Systemkomponenten

13.1. Architektur

Das System Meet & Remind basiert sowohl auf einer Client-Server Architektur als auch auf einer Peer-to-Peer Architektur. Die Grundlage für das System bildet die Client-Server Architektur, bei der der Client aus einer Android App besteht und der Server mit der Datenhaltung dazu beiträgt, dass eine Verteiltheit des Systems möglich ist. Diese Verteiltheit wird dadurch erweitert, dass die verschiedenen Clients in einer Peer-to-Peer Architektur zueinander stehen. Außerdem befindet sich auf dem Server eine Anwendungslogik, um Daten anzureichern.

13.1.1. Client-Server-Architekturdiagramm

13.1.2. Peer-to-Peer-Architekturdiagram

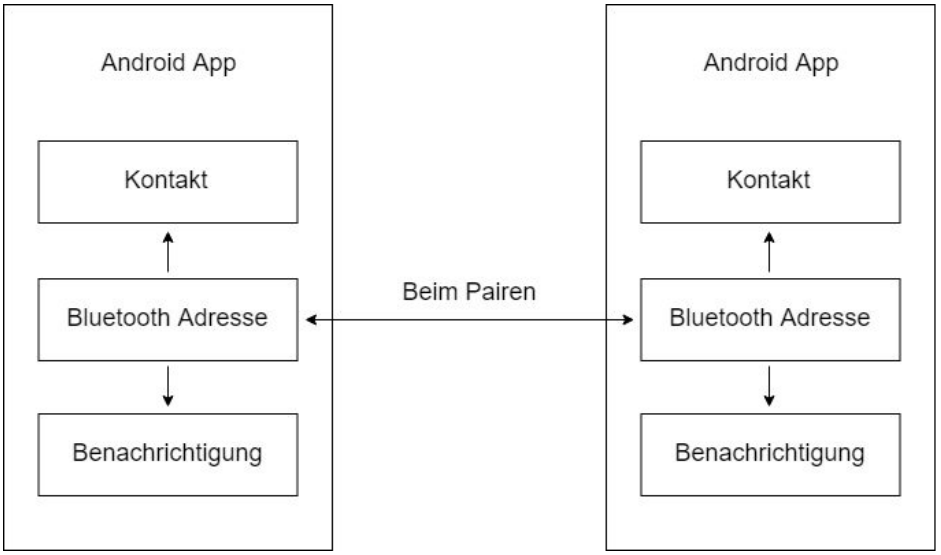


Abb: Peer-to-Peer-Architekturdiagramm für die Verteiltheit der Clients

Das System wird bei ihrer Benutzung in einer Peer-to-Peer Form verwendet. Jeder Client steht gleichwertig zueinander und hat die gleichen Funktionen und Rollen. Die Kopplung (Pairen) der Clients zum Erstellen von Kontakten geschieht synchron. Es bildet die Grundlage für die Kontakte und die gespeicherten Erinnerungen, die dann für Benachrichtigungen sorgen.

13.1.3. Client

Der Client bildet das einzige User Interface von Meet & Remind und wird ausschließlich für Android Geräten zur Verfügung gestellt. Wie bereits im vorhergegangenen Konzept beschrieben, weist der Client die Präsentationslogik, Bluetooth-Logik, die Funktionalität der Benachrichtigungen und die clientseitige Anwendungslogik auf. Letzte wird in einem folgenden Abschnitt genauer erläutert. Die Gestaltung der Funktionen im User Interface wurde im MCI Vorgehensmodell des Usage Centered Design modelliert und begründet.

13.1.4. Model-View-Controller

Zur Unterteilung der Bestandteile innerhalb des Clients nutzen wir das Model-View-Controller (kurz: MVC) Muster. Dieses Muster zur Softwareentwicklung wird verwendet um die Bestandteile aufgrund ihrer Verwendung aufzuteilen. Sie bestehen aus den drei Teilen Model, View und Controller. In folgenden Abschnitten werden sie erklärt und auf unser System bezogen. [QUELLE http://www.datenbanken-verstehen.de/lexikon/model-view-controller-pattern/] Am Ende wird der Nutzen von MVC für Meet & Remind erläutert.

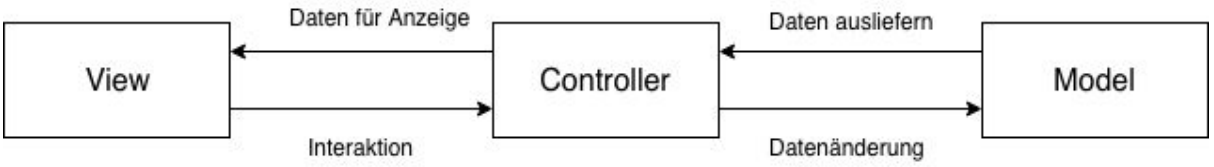


Abb. 3: Model-View-Controller Muster

Model
Das Bestandteil Model verwaltet die Daten der Anwendungen. Es kümmert sich darum, dass Controller und dadurch auch der View mit den Daten ausgestattet werden. Dies bedeutet nicht unbedingt, dass die Datenhaltung im Model implementiert ist. Im Fall von Meet & Remind sorgt das Model für die Übergabe und das Anfordern von Daten vom Server. Es dient also als Schnittstelle, die den Client mit den aktuellen Daten versorgt oder die Daten vom Client aus an den Server schickt. Um einen möglichst performante Datenaustausch bereitzustellen, kann zudem im Model ein Caching implementiert werden, sodass es zu keiner großen Wartezeit bei geringer Netzwerkverbindung kommt.

Controller
Der Controller stellt die logischen Operationen zur Verfügung und kümmert sich darum, dass die View mit den gerade benötigten Daten versorgt wird. Er informiert die View über Veränderungen, die Auswirkungen auf das User Interface haben. Im Controller befindet sich somit auch unsere Anwendungslogik, Präsentationslogik, Bluetooth-Logik und Logik zum Versenden der Benachrichtigungen.

View
Unter View versteht man das User Interface, also den Bestandteil, der die Interaktion mit dem System bereitstellt. Bei Meet & Remind handelt es sich dabei um die Oberfläche, die im Vorgehensmodell des Usage Centered Design modelliert wurde. Die View erhält die Daten vom Model über den Controller zugeteilt und stellt nur die grundlegende Anzeige zur Verfügung. Hier wird keine Logik implementiert oder die Daten verändert, da sich darum der Controller kümmert.

Vorteile bei der Nutzung von MVC
Der Vorteil bei der Nutzung von MVC besteht darin, dass die Bestandteile gekapselt implementiert werden. Dadurch können zum Einen Änderungen in einem der drei Teile durchgeführt werden, ohne große Anpassungen an den anderen vornehmen zu müssen. Eine neue Oberfläche, die im View umgesetzt wird, kann also unberührt gelassen werden, falls sich die Anwendungslogik verändert. Das System wird also zusätzlich zum objektorientierten Vorgehen weiter in Komponenten aufgeteilt, die die Wartbarkeit und Übersicht optimieren.
Zum Anderen kann die Arbeitsteilung der Entwickler bei der Implementierung erleichtert werden, da zum Beispiel in verschiedenen Klassen gearbeitet wird. Damit wird die Kommunikation zwischen den Entwickler erleichtert, weil weniger Absprachen benötigt werden und parallel gearbeitet werden kann.

13.1.5. Server

Der Server stellt das Datenhandling und die serverseitige Anwendungslogik zur Verfügung. Er interagiert als Schnittstelle zur Datenhaltung und liefert dadurch die Daten an die Clients aus. Es besteht somit für die Clients eine einheitliche Datenhaltung auf dem Server.
In den ersten Schritten der Implementierung wird der Server lokal genutzt und später auf Heroku gehostet [QUELLE https://www.heroku.com/].

13.1.6. Datenhaltung

Bei der Datenhaltung gilt es zu berücksichtigen, ob ein externer Service genutzt wird oder die Datenhaltung auf dem Server eingebunden wird. Hier wurde im Konzept angedacht auf den Web-Service von Google Firebase zurückzugreifen. Firebase bietet einige Vorteile in der Erweiterung von Android Apps, die genutzt werden könnten um das System zum Beispiel mit einer Authentifizierung auszustatten. Es gilt jedoch zu beachten, dass bei einem externen Service Datenschutzrichtlinien einzuhalten sind und die Datenhaltung abhängig von einem Dienstleister ist. Falls dieser Dienstleister Änderungen vornimmt oder den Dienst einstellt, müssen Änderungen am System vorgenommen werden und es würden im schlimmsten Fall Daten verloren gehen.

In Bezug auf Firebase überwiegen die Vorteile jedoch die Nachteile. Der Datenschutz ist durch die DSGVO abgesichert, an die sich Google bei Firebase hält [QUELLE https://support.google.com/firebase/answer/9019185?hl=de]. Außerdem hat Google sich freiwillig dem US-EU Privacy Shield unterworfen [QUELLE https://www.privacyshield.gov/participant?id=a2zt000000001L5AAI&status=Active]. Somit

gibt es aus datenschutzrechtlicher Sicht keine Bedenken bei den Google Cloud Diensten, zu denen auch Googles Firebase gehört. Um diesen Datenschutz und diese Datensicherheit zu gewährleisten, wäre ein Experte im Datenschutz und Datenbanken notwendig.
Ein weiterer Vorteil ist die Skalierbarkeit, da sich die Datenhaltung auf einem Cloud Dienst befindet und nicht auf einem eigenen Server. Dadurch gibt eine keine Einschränkung durch die Hardware des Servers, die die Skalierbarkeit einschränkt. Auch hier wäre Expertenwissen nötig, falls das System von einer großen Menge an Usern genutzt wird.
Dabei ist auch der Betrieb bei dem Cloud Dienst sichergestellt, da keine Einrichtung, Pflege, Updates oder Datensicherungen vorgenommen werden müssen, für die das Team bei einer eigenen Datenhaltung zuständig wäre.
Auch aus wirtschaftlicher Sicht ist Google Firebase insoweit sinnvoll, dass keine Investition in Form einer Bindung von Kapital vorgenommen werden muss, wenn Hardware für Datenhaltung und Skalierbarkeit gekauft werden muss.

13.2. Kommunikation zwischen den Komponenten

Nachdem die Kommunikation innerhalb der Komponenten selbst beschrieben wurde, kann nun die Kommunikation der Komponenten zueinander beschrieben werden.
// asynchron /synchron in Bezug auf das Architekturdiagramm
// Clients untereinander sind Peer to peer und das System selber ist Client Server
// Notifications nutzen für Erinnerungen

13.3. Programmiersprachen

Die Auswahl der Programmiersprache ist ein wichtiger Schritt bei der Konzeption eines interaktiven Systems. Hier gibt es mehrere Kriterien, die für die Auswahl ausschlaggebend sind, zu beachten. Zunächst sollte geprüft werden, welche Programmiersprache bereits bekannt ist und ob die geforderten Systemkomponenten mit dieser umsetzbar sind. Hier ergeben sich die Vorteile durch Kompaktheit des Codes und bereits vorhandener Frameworks oder Libraries, die für die Umsetzung genutzt werden können. Die Erfahrung der Entwickler spielt bei der Wahl der Programmiersprache auch eine große Rolle. Die Einarbeitung in eine unbekannte Sprache kann aufgrund des Syntax und der Kompaktheit viel Zeit in Anspruch nehmen.

13.3.1. Clientseitige Programmiersprache

Da für den Client eine native mobile Anwendung in Form einer Android App implementiert wird, ist die Wahl der Programmiersprache stark eingegrenzt. Die Unterstützung von Java ist hier am größten [QUELLE für ANDROID APP Sprachen].
In den Modulen Algorithmen und Programmierung 1 und 2 wurde der Umgang mit der objektorientierten Entwicklung mit Java gelernt und in Praktika vertieft. Auch in den weiteren Modulen haben die Entwickler Java mit importierten Libraries und Frameworks genutzt, um Projekte zu bearbeiten.

13.3.2. Serverseitige Programmiersprache

Serverseitig wird die Programmiersprache JavaScript mit dem Framework NodeJS genutzt. Die Wahl ist auf diese Kombination gefallen, da das Framework NodeJS eine besonders

einfachen und kompakten Aufbau für Implementierung eines Servers bietet. NodeJS wurde vor allem auf Performance und erweiterbare Module ausgelegt. Die Basis ist schlank und kann mit den benötigten Modulen so erweitert werden, dass der Umfang des Codes klein gehalten werden kann [QUELLE Node JS].
Zudem wurde im Modul Web-basierte Anwendungen 2 bereits ein Server mit diesem Framework implementiert. Außerdem wurde bei diesem Projekt bereits mit einer Anbindung von Firebase als Datenhaltung gearbeitet, sodass auch hier Erfahrung mit dem Syntax und möglichen Problemstellungen in NodeJS gesammelt wurde.

13.4. Entwicklungsumgebung

Aufbauend auf der Wahl der Programmiersprache wird die Entwicklungsumgebung (auch “Integrierte Entwicklungsumgebung”, kurz “IDE” [QUELLE]) ausgewählt. Auch hier gibt es wie bei bei der Programmiersprache mehrere Kriterien, die zu beachten sind. Auf der einen Seite stehen die Kriterien der Umgebung selbst. Ein Kriterium kann die Funktionalität mit der gewählten Programmiersprache genannt werden. Die IDE sollte zum Beispiel Code-Autovervollständigung und ein gutes Syntax-Highlighting für die Programmiersprache besitzen. Mit diesen Hilfsmitteln wird es für den Entwickler leichter den Code zu schreiben und zu verstehen.
Dadurch wird auch die andere Seite der Kriterien klar. Hier stehen die Erfahrungen und Vorlieben der Entwickler mit der IDE selbst. Kennen die Entwickler sich bereits mit einer Umgebung aus, die die ersten Kriterien erfüllen? Haben sie bereits in anderen Projekten gute Erfahrungen machen können?

13.4.1. Clientseitige Umgebung

Unser Client besteht aus einer Android App, die mit Java entwickelt wird und für das Layout der Interaktionselemente XML verwendet. Google bietet für ihre Android App eine offizielle IDE Namens Android Studio [QUELLE] an.
Betreffend der im oberen Abschnitt genannten Kriterien bietet es eine sehr gute Code-Autovervollständigung und ein gutes Syntax-Highlighting an. Da sie genau für die Entwicklung von Android Applikationen aufgebaut ist, ist auch die Integration von anderen Java Klassen unkompliziert. Gerade bei der Entwicklung der Bluetooth Komponenten bringt dies viele Vorteile mit sich. Ein weiterer Vorteil von Android Studio ist die Möglichkeit, die Layouts des User Interfaces per Drag-and-Drop zusammenstellen zu können, sodass eine erste rudimentäre Code-Struktur für das Layout ohne viel Aufwand aufbaubar ist.
Des weiteren sprechen die Kriterien der Entwickler für den Einsatz dieser IDE. Im Wahlpflichtfach Mobile Computing haben beide Teammitglieder den Umgang mit Android Studio gelernt und bereits in einem Projekt angewendet.
Für die clientseitige Entwicklung kann theoretisch jede Umgebung genutzt werden, die Java-fähig ist, also auch IntelliJ. Jedoch bringt IntelliJ standardmäßig nur wenige Funktionen mit, die sich direkt auf Android Entwicklung beziehen, wie zum Beispiel die direkte Drag-and-Drop Implementierung des User Interfaces.

13.4.2. Serverseitig Umgebung

Da Serverseitig zur Entwicklung NodeJS genutzt wird, wird eine IDE benötigt, die Javascript-fähig ist. Hier bietet die IDE IntelliJ von Jetbrains [QUELLE] gutes

Syntax-Highlighting und eine gute Code-Autovervollständigung an. Zudem gibt es für NodeJS Plugins, die genutzt werden können, um den kompletten Entwicklungsprozess über diese IDE handhaben zu können.
Auch hier sprechen die Kriterien der Entwickler für IntelliJ. Im Modul Web-basierte Anwendungen 2 wurde im Rahmes eines Projektes zum Aufsetzen eines Servers IntelliJ genutzt. Die Vorgänge zum Kompilieren und Ausführen des Servers sind also bereits bekannt.

13.4.3. Zusammenspiel der beiden Umgebungen

Da Android Studio auf Jetbrains IntelliJ basiert, spielen die beiden Umgebungen sehr gut zusammen und weisen ein sehr ähnliches Interface auf. Eine Umstellung bei der Entwicklung von Server und Client also nicht nötig. Auch die Integration von Git ist in beiden standardmäßig vorhanden und verhält sich identisch, sodass es zu keinen Konflikten bei der Versionsverwaltung kommt.

13.5. Arten von Endgeräten

Da Meet & Remind ein System ist, das rein über eine Android App benutzt wird, ist der Umfang der Gerätevarianten klein. Es gibt weder eine Webanwendungen, die über den Browser benutzt wird, noch gibt es einen Desktop Client für Windows oder Apple Rechner. Trotzdem ist zu beachten, dass Auflösung und Formfaktor des Gerätes eine Rolle spielen.

13.5.1. Begründung für Android

Bei den mobilen Geräten ist Android das am meisten genutzte Betriebssystem. Der Markt für Android Apps ist somit der größte und bietet daher die besten Voraussetzungen für die Vermarktung der App.

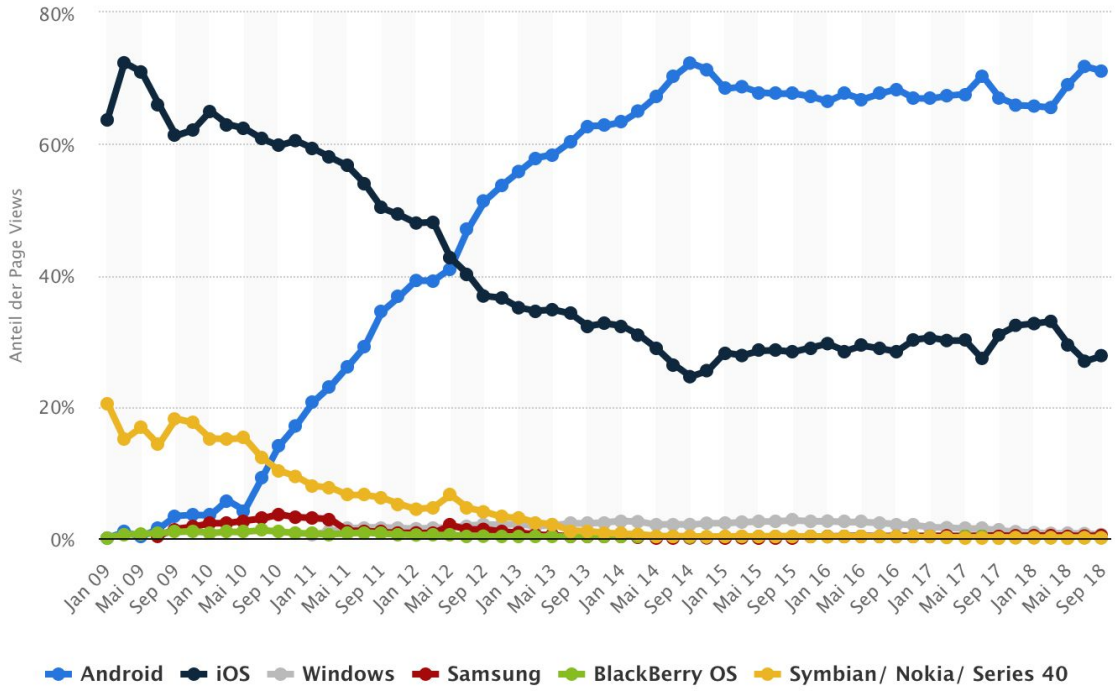


Abb. 3: Marktanteil der mobilen Betriebssysteme an der Internetnutzung in Deutschland seit 2009

QUELLE
https://de.statista.com/statistik/daten/studie/184332/umfrage/marktanteil-der-mobilen-betriebssysteme-in-deutschland-seit-2009/

Des Weiteren ist die Entwicklung von Android Apps im Gegensatz zu iOS Apps nicht an das Betriebssystem gebunden, dass zur Entwicklung eingesetzt wird. Android Studio gibt es sowohl für Windows als auch für Apple Rechner. Das für iOS Apps konzipierte Xcode gibt es jedoch nur für das Apple Betriebssystem MacOS.

13.5.2. Verfügbarkeit

Das Interface von Meet & Remind wird sowohl auf Smartphones als auch auf Tablets nutzbar sein. Die Implementierung des User Interfaces ist in Android über den Layout-XML Editor so aufbaubar, dass er responsive und somit unabhängig von der Gerätegröße oder Auflösung ist.
Für die Nutzung ist mindestens Android in der Version 7 nötig, da viele der Funktionen erst ab dieser Android Version über die App implementierbar sind. Da die Verteilung der Android Geräte dieser Version den größten Teil aufweist, ist diese Voraussetzung zudem begründet.

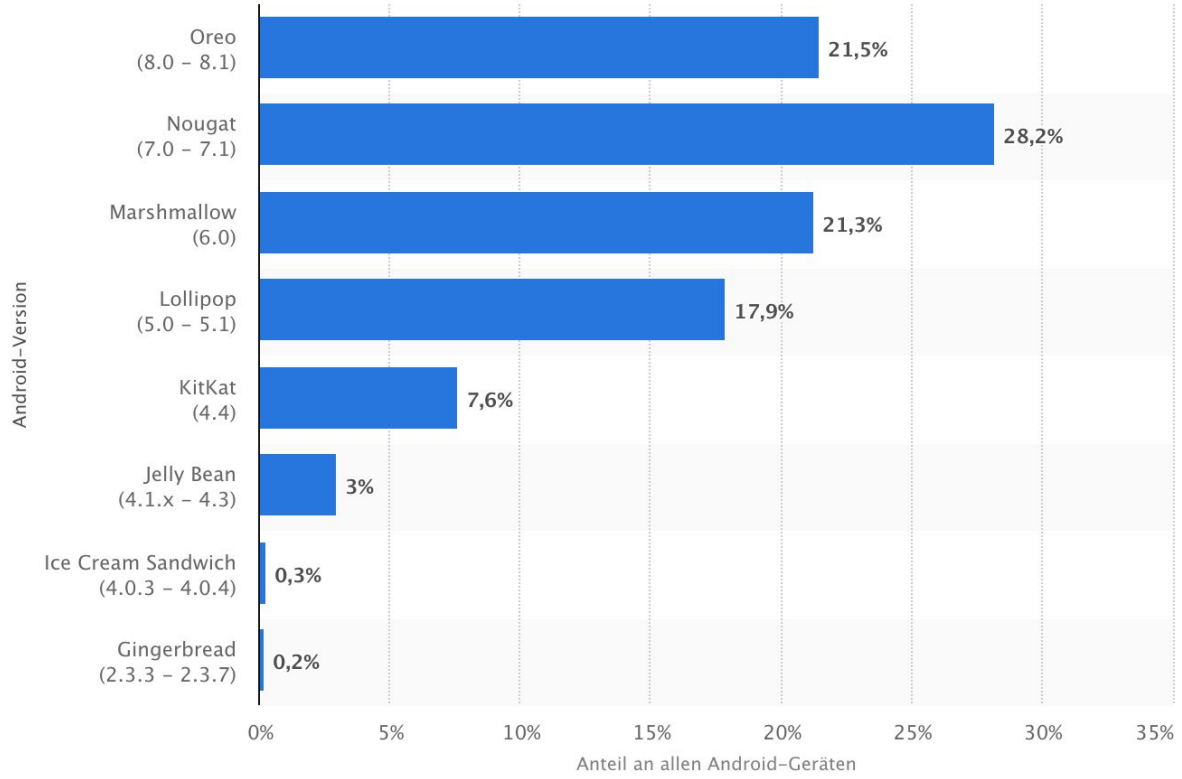


Abb. 3: Anteil der Android Versionen weltweit im Zeitraum 20. bis 26. Oktober 2018

https://de.statista.com/statistik/daten/studie/180113/umfrage/anteil-der-verschiedenen-android-versionen-auf-geraeten-mit-android-os/

14. Verteilte Anwendungslogik

Mit der verteilten Anwendungslogik wird das System ohne Nutzerinteraktion angereichert. Sowohl auf dem Client als auch auf dem Server wird eine Logik implementiert, die Haupt-Funktionalitäten des System darstellen. Um einen klareren Überblick über die beiden Anwendungslogiken zu erhalten, werden sie in folgenden Abschnitten aufbauend auf dem Konzept beschrieben und begründet. Außerdem wird die technische Implementierung in Betracht gezogen, sodass Erkenntnisse für die Systemkomponenten und die Datenstruktur gezogen werden können.

14.1. Clientseitige Anwendungslogik

14.1.1. Beschreibung

Aus dem Konzept ging folgende Beschreibung der Clientseitige Anwendungslogik hervor:

“Das System gibt eine selbstdefinierte Erinnerung aus, wenn sich zwei Personen gleichzeitig an einem Ort befinden. Es handelt sich um eine nicht-interaktionsgetriebene Logik, da der Benutzer das System nicht direkt benutzt. Das System reichert Daten an, indem es sich merkt, wenn eine Erinnerung nicht abgehakt wurde. Dadurch wird die Erinnerung höher priorisiert und bei der nächsten Begegnung der Personen wieder angezeigt.”

// weiter ausführen der Logik

14.1.2. Entwurf zur Umsetzung

// Pseudocode oder diagramm, zeichnung

14.2. Serverseitige Anwendungslogik

14.2.1. Beschreibung

Aus dem Konzept ging folgende Beschreibung der Serverseitige Anwendungslogik hervor:

“Die selbstdefinierten Erinnerungen können durch vorgegebenen Labels mit Themen versehen werden. Das System lernt daraus, welche Gesprächsthemen die beiden Personen haben. Gleiche Labels werden auf der Serverseite von beiden Personen summiert. Ab einem gewissen Wert hat das System so weit gelernt, dass es Themen für ein Gespräch der beiden Personen eigenständig vorschlagen kann.”

Der Begriff “Lernen” muss hier kritisch betrachtet werden. Das Matching in dieser Anwendungslogik kann nicht direkt als lernen bezeichnet werden, da keine ähnlichen Themen erkannt werden, sondern nur die vorgegebenen Themen analysiert werden.

// weiter ausführen der Logik

14.2.2. Entwurf zur Umsetzung

// Pseudocode oder diagramm, zeichnung

14.2.3. API für Themenvorschläge

// Was macht die Themen API und welche nehmen wir

15. Datenstruktur

15.1. Datenbank

// nosql -> document database
// firebase bietet dazu möglichkeit
// nicht dateibasiert

15.2. Datenformat

// json
// machen keine validierung notwendig ist wie bei xml
// JSON flexibel und leichter erweiterbar da weniger
// JSON weniger bandbreite
https://www.infragistics.com/community/blogs/b/torrey-betts/posts/mobile-performance-testing-json-vs-xml

16. PoCs

//kritisch reflektieren und verbessern.

17. Fazit

18. Quellen

1. Beispielquelle: https://www.paulwatzlawick.de/axiome.html (abgerufen am 08.11.2018)
2. https://www.hanser-elibrary.com/doi/book/10.3139/9783446443136 Alternative: https://books.google.de/books?id=pJKqBAAAQBAJ&pg=PA1&dq=requirements+engineering&hl=de&sa=X&ved=0ahUKEwiHrOaOjonfAhVJUlAKHaNJBm4Q6AEIZDAI#v=onepage&q&f=false
3. Software for use: ….