

# **Freie Universität Berlin**

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

## **Entwicklung einer Kalender-App für Geeks: Ein Versuch wie man die Lücke zwischen Terminal und Smartphone überbrücken könnte**

**Thore Brehmer**

Matrikelnummer: 5216879

[thob97@zedat.fu-berlin.de](mailto:thob97@zedat.fu-berlin.de)

Betreuer/in: Prof. Dr.-Ing. Volker Roth

Eingereicht bei: Prof. Dr.-Ing. Volker Roth

Berlin, 8. Juli 2023

**Zusammenfassung**

*Ziele:* In dieser Arbeit wurde versucht, eine passende App für CLI-Terminkalender zu erstellen. „Passend“ bedeutet in diesem Sinne, dass die Stärken von PC und Handy berücksichtigt werden und somit in die zu erstellende Anwendung einfließen.

*Methoden:* Dazu wurde zunächst versucht herauszufinden, was die Stärken von Handys und PCs überhaupt sind, indem die wesentlichen Unterschiede zwischen ihnen verglichen wurden. Anschließend wurden mithilfe von Erhebungstechniken die Anforderungen an die zu erstellende App ermittelt und festgehalten. Auf Basis dieser Anforderungen wurden Gedanken zur Auswahl der passenden Technologien gemacht. Danach wurde das Design der Anwendung erstellt und überdacht, da dies ein wichtiger Punkt für die Anforderungen sowie eine mögliche Stärken von Handys ist. Da nun die Anforderungen und das Design der Anwendung bekannt sind sowie die Technologie, mit der sie erstellt werden soll, wurde sich anschließend mit der Implementierung befassen. Zum Schluss wurde die Anwendung mit einer Anforderungsverifizierung evaluiert, um festzustellen, ob das Erstellte auch das erfüllt, was sich zuvor vorgenommen wurde.

*Ergebnisse:* Eine wichtige Erkenntnis, die während der Erarbeitung der Stärken gewonnen wurde, ist, dass für die Intuitivität und Benutzerfreundlichkeit einer Handyanwendung das Design und dessen Richtlinien ein entscheidender Faktor sind. Weiterhin kam es zu dem Ergebnis, dass Handys für jene Aufgaben gut geeignet sind, die kurzweilig sind oder wenig Zeit benötigen, die unterwegs gelöst werden sollen und einfach sowie intuitiv sein sollen.

Während PCs gut für jene Aufgaben sind, die viel Leistung benötigen, schnelle, präzise oder vielfältige Eingaben erfordern, viele Informationen gleichzeitig darstellen oder benötigen sowie viele Optionen und Konfigurationen anbieten oder benötigen. Laut der Evaluation entstand während der Arbeit eine Anwendung, die in der Lage ist, genau diese Stärken umzusetzen.

*Schlussfolgerungen:* Es wird angenommen, dass das Ziel dadurch erfüllt werden konnte. Weiter wird sogar vermutet, dass es sich bei dieser App um eine nützliche Anwendung handeln könnte.

*Ausblick:* Falls sich die Anwendung als nützlich beweist, könnte dies bedeuten, dass es für Anwendungen durchaus lohnenswert sein kann, diese unter Berücksichtigung der Stärken sowohl von Handys als auch von PCs zu entwickeln. Möglicherweise könnte dies zeigen, dass Anwendungen nicht unbedingt als eigenständige Einheiten genutzt werden müssen, sondern voneinander abhängig sein und dennoch oder gerade deshalb nützlich sein können.  
Um jedoch die Nützlichkeit abschließend bewerten zu können, benötigt es weiterer Evaluation. Auch konnte die Anwendung noch nicht vollständig implementiert werden. So fehlen noch einige wenige Funktionen, Design Anpassungen und Ausprägungen von nicht funktionalen Anforderungen.

### **Eidesstattliche Erklärung**

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

8. Juli 2023

Thore Brehmer



# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>7</b>
1.1 Voraussetzungen . . . . .	7
1.2 Struktur . . . . .	7
1.3 Motivation . . . . .	7
1.4 Zielsetzung . . . . .	8
<b>2 Handy und PC Unterschiede</b>	<b>10</b>
2.1 Leistung . . . . .	10
2.2 Internet . . . . .	11
2.3 Speicher . . . . .	11
2.4 Eingabe . . . . .	11
2.5 Bildschirm . . . . .	12
2.6 Mobilität . . . . .	13
2.7 Benutzung . . . . .	13
2.8 Hardware . . . . .	14
2.9 Betriebssystem . . . . .	14
2.10 Software . . . . .	14
2.11 Auswertung & Erkenntnisse . . . . .	15
<b>3 Anforderungen</b>	<b>18</b>
3.1 Vorgehensweise . . . . .	18
3.1.1 Umfrage . . . . .	19
3.1.2 Vergleich . . . . .	20
3.2 Nicht Funktionale Anforderungen . . . . .	21
3.3 Funktionale Anforderungen . . . . .	23
<b>4 Technologische Überlegungen</b>	<b>26</b>
4.1 Auswahl des Frameworks . . . . .	26
4.2 CLI-Terminkalenderwahl . . . . .	28
4.3 Darstellung der Erinnerungen . . . . .	28
4.4 Konfigurationsdateiformat . . . . .	28
4.5 Entwicklungsumgebung . . . . .	29
4.6 Auswahl der Pakete . . . . .	30
<b>5 Design</b>	<b>32</b>
5.1 Konventionen . . . . .	32
5.1.1 Auswahl . . . . .	32
5.1.2 Auswertung & Erkenntnisse . . . . .	33
5.2 App Design . . . . .	34
5.2.1 Appleisten . . . . .	34
5.2.2 Kalenderseite . . . . .	35
5.2.3 Erinnerungenseite . . . . .	35
5.2.4 Einstellungsseite . . . . .	36
5.2.5 Tastatur . . . . .	36

<b>6</b>	<b>Implementierung</b>	<b>38</b>
6.1	Parser . . . . .	39
6.1.1	Modell . . . . .	39
6.1.2	Zeitangabe . . . . .	39
6.1.3	Variablen & Operationen . . . . .	40
6.1.4	Eingabe- & Formatfehler . . . . .	40
6.2	Verbindung zur Datenbank . . . . .	40
6.2.1	Authentikation & Berechtigungen . . . . .	40
6.2.2	Dateien herunterladen . . . . .	41
6.2.3	GitHub Issues . . . . .	41
6.3	Qualitätssicherung . . . . .	42
6.3.1	Tests . . . . .	42
6.3.2	Verständlichkeit . . . . .	43
6.4	Entwurfsmuster . . . . .	43
6.4.1	Fassade . . . . .	44
6.4.2	Singelton . . . . .	44
6.4.3	Strategie . . . . .	44
6.4.4	Proxy . . . . .	45
6.4.5	Datenflussnetz & Ablagebasiert . . . . .	45
<b>7</b>	<b>Evaluation</b>	<b>46</b>
7.1	Anforderungsverifizierung . . . . .	46
<b>8</b>	<b>Fazit</b>	<b>51</b>
<b>A</b>	<b>Anhang</b>	<b>60</b>

# 1 Einleitung

**Überblick:** Dieser Abschnitt behandelt die kongkrete Beschreibung des Problems sowie die Charakterisierung des Ziels. Dabei soll unter anderem auch der Nutzen der Arbeit deutlich werden.

Zusätzlich wird die Struktur des Dokuments erläutert und das erforderliche Vorwissen der angepeilten Leserschaft genannt. Dadurch soll verdeutlicht werden, ob und in welcher Weise die Ausarbeitung gelesen werden kann.

**Ergebnisse:** Für diese Ausarbeitung werden allgemeine Informatikkenntnisse vorausgesetzt. Zudem enthält jeder Abschnitt dieser Arbeit eine Übersicht und Zusammenfassung, um das Lesen erleichtern und Querlesen zu ermöglichen.

Während der Beschreibung des Problems wurden mehrere Erkenntnisse gewonnen. Zum einen erfreuen sich Handys Relevanz und Beliebtheit und dementsprechend lohnt es sich Anwendungen auch als App anzubieten. Zum anderen besitzen Handys und PCs verschiedene Stärken und Schwächen und CLI-Terminkalender sind diesbezüglich eine interessante Anwendung.

## 1.1 Voraussetzungen

Im Rahmen dieser Ausarbeitung werden wiederholt Begriffe aus dem Jargon der Informatik genutzt. Auf jene Begriffe, die mit einem Informatikstudium als informatisches Allgemeinwissen bezeichnet werden, wird nicht näher eingegangen. Das bedeutet, dass ein gewisses informatisches Vorwissen seitens des Lesers vorausgesetzt wird<sup>1</sup>.

Dies hat den Grund, ein Abschweifen während der Ausarbeitung und damit verbundene Störungen des Leseflusses zu verhindern.

## 1.2 Struktur

Jeder Abschnitt enthält eine Einleitung, in der der Inhalt und dessen Bedeutung für das Projekt erläutert werden sowie die verwendete Methodik beschrieben wird.

Darüber hinaus gibt es am Ende jedes Abschnitts eine kurze Zusammenfassung, die wichtige Entscheidungen und Ergebnisse des Kapitels zusammenfasst. Das soll dem Leser die Möglichkeit geben, Abschnitte schnell zu überfliegen und trotzdem die wichtigsten Aspekte und Ergebnisse der Arbeit zu verstehen. Zusammenfassend soll diese Strukturierung beim Querlesen helfen.

## 1.3 Motivation

Als Handys erstmals auf den Markt kamen, waren sie im weitesten Sinne in erster Linie als eine mobile Alternative zu den herkömmlichen Telefonsystemen gedacht [54]. Heutzutage scheinen Handys aber einen viel wichtigeren Teil unseres Lebens eingenommen zu haben. Anscheinend fühlt sich bereits die Mehrzahl an Leuten unwohl

---

<sup>1</sup>Beispiele Begriffe zur Orientierung:

- wird Vorausgesetzt: Framework, Package, API (Application Programming Interface)

## 1. Einleitung

das Haus ohne ihr Handy zu verlassen[87]. Des Weiteren werden Handys mittlerweile täglich 40 Minuten mehr als PCs benutzt[41, 42, 40]<sup>2</sup>. Das zeigt einen starken Wandel, wenn man bedenkt, dass Computer im Jahr 2011 noch rund 94% des Marktanteils ausmachten[80]. Diese Veränderung scheinen auch Entwickler und Unternehmen zu bemerken. So kündigte zum Beispiel Google bereits in 2017 den Umstieg auf Mobile-Indexing<sup>3</sup> an[63] und Unternehmen wie Facebook und Pinterest beziehen den Großteil ihrer Benutzerschaft aus Applikationen[77, 65].

Generell scheinen Handys also immer beliebter und wichtiger zu werden. Das spiegelt sich auch in den Nutzerzahlen wieder. Während laut einer Studie 96% der Befragten ein Handy besitzen ist die Anzahl bei PCs von 73% auf 59% innerhalb der letzten vier Jahre<sup>4</sup> gesunken[43]. Auch der Marktanteil des Handys zeigt ein Indiz darauf, denn dieser steht mit 53% knapp über den des Computers[80].

Basierend auf der Erkenntnis, dass das Handy einen wichtigen Stellenwert eingenommen hat, lassen sich für diese Arbeit zwei interessante Schlussfolgerungen ziehen.

Zum einen, dass es lohnt sich lohnt, Anwendungen nicht nur für den PC, sondern auch für Handys anzubieten.

Zum anderen, dass das Handy gewisse Vorteile und Stärken im Vergleich zu PCs bieten muss, was dazu führt, dass es an Beliebtheit und Nutzung gewinnt. Dies scheint jedoch beidseitig zu gelten. So gibt es Anwendungen, die trotz der steigenden Handy-Popularität besser auf dem Computer funktionieren. Beispielsweise möchten wahrscheinlich nur wenige Menschen eine wissenschaftliche Arbeit oder Steuererklärung auf einem Handy schreiben.

Eine weitere Gruppe von Anwendungen, die sich besser auf dem Computer als auf dem Handy nutzen lassen, sind die CLI-Terminkalender<sup>5</sup>. Das Kürzel „CLI“ steht hierbei für „Command Line Interface“, was gleichbedeutend mit einer Kommandozeile oder einem Terminal ist. Bei diesen Terminkalendern gibt es keine grafische Benutzeroberfläche, stattdessen wird über das Terminal interagiert. Die Anwendung erfordert es, unter anderem lange Texte, Befehle und Sonderzeichen zu schreiben. Durch die Tastatur ist dies auf dem Computer einfach auszuführen. Im Gegensatz dazu würde die begrenzte Softwaretastatur des Handys die Bedienung dieser Anwendung erschweren. Wahrscheinlich ist dies auch der Grund, warum es bisher keine erfolgreichen CLI-Terminkalender-Anwendungen für das Handy gibt.

### 1.4 Zielsetzung

Im letzten Abschnitt haben wir folgende Erkenntnisse gewonnen.

1. Handys erfreuen sich großer Relevanz und Beliebtheit
2. Daher lohnt es sich, Anwendungen auch als Apps anzubieten.
3. Handys und PCs haben unterschiedliche Stärken und Schwächen.

---

<sup>2</sup>Handy: 3.44 Stunden, PC: 2.59 Stunden, zusammen: 6.43 Stunden.

<sup>3</sup>Suchergebnisse werden besser bewertet, wenn Seiten eine mobile Ansicht anbieten.

<sup>4</sup>2018 bis 2022

<sup>5</sup>Beispiel CLI-Terminkalender: remind[27], khal[64], calcurse[22]

4. CLI-Terminkalender sind in dieser Hinsicht eine interessante Anwendung, da es noch keine passende App gibt und die Anwendung eher für den PC ausgelegt zu sein scheint.

Ziel ist es daher, eine passende App für die CLI-Terminkalender zu entwickeln. „Passend“ bedeutet in diesem Sinne, dass die Stärken von PC und Handy berücksichtigt werden und somit in die zu erstellende Anwendung einfließen.

Oder anders formuliert ist das Ziel die „Entwicklung einer Kalender-App für Geeks: Ein Versuch wie man die Lücke zwischen Terminal und Smartphone überbrücken könnte“.

Die simple Lösung einen CLI-Terminkalender auf das Handy zu portieren, ist also nicht zufriedenstellend. Stattdessen soll eine App entwickelt werden, welche auf existierende CLI-Terminkalender aufbaut, denn so könnten beide Geräte und dementsprechend auch dessen Stärken genutzt werden.

Es ist wichtig noch einmal zu betonen, dass es sich hierbei lediglich um einen Versuch handelt. Die in dieser Arbeit gewonnenen Informationen und Ergebnisse sollen weder als allgemeines Beispiel noch als Lösung dienen.

Da die Bearbeitungszeit für eine Bachelorarbeit begrenzt ist, ist es außerdem nicht das Ziel, ein vollständiges Produkt zu entwickeln. Stattdessen wird versucht, sich schrittweise dem Ziel zu nähern.

## 2 Handy und PC Unterschiede

Um das in **Unterabschnitt 1.4. Zielsetzung** genannte Ziel zu erreichen, muss zunächst festgestellt werden, was die Stärken und Schwächen von PCs und Handys sind. In diesem Abschnitt werden daher wesentliche Unterschiede zwischen PCs und Handys verglichen, um so zu versuchen Erkenntnisse über ihre Stärken zu gewinnen. Da es sich bei PCs jedoch um eine Kategorie von Systemen handelt<sup>6</sup>, wird sich auf einen davon beschränkt, da dieser Abschnitt sonst zu umfangreich würde. Dabei wurde der Desktop gewählt, da vermutet wird, dass die Unterschiede zwischen ihm und dem Handy am stärksten ausgeprägt sind.

**Überblick:** In diesem Abschnitt werden folgende Aspekte der Geräte verglichen: Leistung, Internetverbindung, Speicher, Eingabemöglichkeiten, Bildschirme, Mobilität, Nutzung, Hardware, Betriebssystem und Software, da diese als wesentliche Unterschiede betrachtet werden. Abschließend werden die Erkenntnisse zusammengefasst und Schlussfolgerungen daraus gezogen.

**Ergebnisse:** Beim Vergleich der Bildschirme wurde erkannt, dass für Handys das Design und dessen Richtlinien wichtig sind. Diese können sich auf die Intuitivität und Benutzerfreundlichkeit der Anwendung auswirken.

Zudem kam es durch den Vergleich der Unterschiede zu der Erkenntnis, dass PCs gut für Aufgaben funktionieren die viel Leistung benötigen, schnelle, präzise oder vielfältige Eingaben erfordern, viele Informationen gleichzeitig darstellen oder benötigen, viele Optionen und Konfigurationen anbieten oder benötigen sowie langwierig sind oder viel Zeit benötigen.

Handys hingegen funktionieren gut für Aufgaben die ressourcenschonend sind und nicht viel Leistung benötigen, keine schnelle, präzise und vielfältige Eingabe erfordern, nur wenige Informationen darstellen oder benötigen, ohne viele Optionen und Konfigurationen auskommen, kurzweilig sind oder wenig Zeit benötigen, lohnenswert sind, unterwegs zu lösen, sowie einfach und intuitiv sein sollen.

### 2.1 Leistung

In Bezug auf Leistung gelten PCs im Allgemeinen als schneller und stärker im Vergleich zu Smartphones. Das liegt an verschiedenen Faktoren. Zum einen haben die meisten PCs einen konstanten Zugang zu Strom, während Smartphones häufig nur über Batterien verfügen. Darüber hinaus können PCs aufgrund ihres größeren Volumens leistungsstärkere Hardware-Komponenten verbauen und größere Kühlsysteme nutzen, was die Verwendung dieser stärkeren Hardware überhaupt erst ermöglicht. Im Gegensatz dazu ist die Hardware von Smartphones in der Regel auf Portabilität und Energieeffizienz ausgelegt, um eine möglichst lange Akkulaufzeit zu ermöglichen.

---

<sup>6</sup>Beispielsweise gehören Laptops, Desktops oder Mini-PCs zu dieser Kategorie.

## 2.2 Internet

Was PCs betrifft, so sind sie oft mit schnellen und stabilen Ethernet-Anschlüssen verbunden, die jedoch stationär sind. Im Gegensatz dazu werden Handys in der Regel über WLAN oder mobile Daten genutzt. Diese Alternativen sind wesentlich mobiler, da sie keine Kabelverbindung erfordern. Stattdessen wird die Verbindung über die Luft hergestellt, was es ermöglicht, dass das Handy fast überall eine Internetverbindung herstellen kann. Allerdings ist zu beachten, dass diese Verbindungen oft langsamer und fehleranfälliger sind als kabelgebundene Verbindungen, da die Luft im Vergleich zum Kabel ein schwächeres Übertragungsmedium darstellt.

## 2.3 Speicher

Was den Speicher betrifft, so müssen die Speichermodule in Handys im Vergleich zu Festplatten für PCs kleiner sein, um in den Formfaktor des Geräts zu passen. Dem entsprechend bieten Handy-Speichermodule in der Regel auch eine geringere Kapazität als die größeren Festplatten von PCs. Laut einer Studie von Counterpoint besitzen Handys im Durchschnitt rund 118 Gigabyte Speicherkapazität[23]. Im Gegensatz dazu zeigt ein Report von Seagate, dass sie im vierten Quartal 2022 im Durchschnitt PC-Festplatten mit einer Größe von rund 8 Terabyte verkauft haben[75]<sup>7</sup>.

## 2.4 Eingabe

PCs werden in der Regel mithilfe einer Maus bedient. Mit ihr können Objekte auf der Oberfläche präzise ausgewählt werden. Außerdem verfügt die Maus über verschiedene Tasten, mit denen eine Interaktion mit den Objekten auf verschiedene Arten möglich ist.

Auf dem Handy wird hingegen der Touchscreen als Mausersatz benutzt. Durch Berühren des Bildschirms lassen sich Mausfunktionen simulieren. Anstatt beispielsweise das gewünschte Objekt mit der Maus auszuwählen, wird es mit dem Finger berührt. Weitere Funktionen wie Zoomen, Scrollen und Rechtsklick können durch Gesten ausgeführt werden.

Dadurch wird die Bedienung auf dem Handy intuitiver, insbesondere das Auswählen von Objekten fühlt sich natürlicher an.

Erfahrungsgemäß gilt das gleiche für die Gesten. Scheinbar fühlen sie sich intuitiv an und haben sich bereits als Standard in den Handymarkt eingegliedert, dass aus eigener Erfahrung viele Menschen lieber das Handy nutzen statt den PC um Bilder anzuschauen. Die Möglichkeiten mithilfe von Gesten zu wischen und vergrößern fühlt sich wohl angenehmer an, als über Verwendung von Maus und Tastatur die Bilder zu betrachten.

Dafür ist die Eingabe über den Touchscreen im Vergleich zur Maus ungenauer. Da der Finger deutlich größer ist als der Mauszeiger, lassen sich kleine Objekte nicht so konsistent auswählen. Um diesem Problem entgegenzuwirken, ist ein passendes Design erforderlich. Beispielsweise wird für Apps eine Mindestgröße für Buttons vorgeschrieben: „On a touchscreen, buttons need a hit target of at least 44x44 points to

---

<sup>7</sup>Die hohe Festplattengröße kommt vermutlich durch Server zustande. Der Speicher von Heimcomputern fällt dementsprechend wahrscheinlich etwas kleiner aus.

## 2. Handy und PC Unterschiede

accommodate a fingertip”<sup>[4]</sup>.

Zur Eingabe von Text auf einem PC wird üblicherweise eine physische Tastatur verwendet, während auf Handys eine Softwaretastatur zum Einsatz kommt. Aufgrund der unterschiedlichen Größe und Bedienung bieten beide Varianten verschiedene Vor- und Nachteile.

Eine Hardware-Tastatur ermöglicht durch ihre Größe und das Tippgefühl schnelleres und präziseres Schreiben im Vergleich zur Touchscreen-Tastatur auf Handys. Außerdem kann sie im Vergleich zur mobilen Variante ungefähr das Dreifache an Tasten darstellen.<sup>8</sup> Des Weiteren unterstützt Hardwaredataturen Tastenkombinationen. Dadurch können Programme, die lange Eingaben, viele verschiedene Symbole oder Tastenkombinationen erfordern, in der Regel auf einem PC besser bedient werden. Ein Beispiel dafür sind Programmiereditoren.

Die Handy-Tastatur hat den Vorteil der Mobilität und das Tippen fühlt sich oft intuitiver an, da sich die Tastatur je nach Anwendung variabel darstellen lässt. Ein Beispiel dafür ist die Taschenrechner-Anwendung, bei der nur die benötigten Zahlen und Symbole auf der Tastatur angezeigt werden und dementsprechend keine Buchstaben.

### 2.5 Bildschirm

PCs bieten die Möglichkeit, mehrere Monitore gleichzeitig zu nutzen und ihre Bildschirme sind in der Regel größer als die von Handys [82][69]<sup>9</sup>.

Dadurch kann der PC viele Informationen gleichzeitig darstellen.

Handys und PCs unterscheiden sich auch in der Ausrichtung ihrer Bildschirme. Während PC-Bildschirme meist im Landschaftsmodus betrieben werden, werden Handys meist im Porträtmodus verwendet.

Dies bedeutet, dass Anwendungen unterschiedlich gut auf den beiden Geräten funktionieren, da sie sich aufgrund der verschiedenen Ausrichtung und Bildschirmgröße unterschiedlich darstellen. Zum Beispiel ist die Darstellung einer großen, detaillierten Tabelle auf einem PC wahrscheinlich einfacher, während lange vertikale Listen mit einfachen Details sich besser auf Handys überfliegen lassen.

Dementsprechend wird angenommen, dass bei der Entwicklung von Anwendungen für Mobilgeräte besonderes Augenmerk auf das **Design** gelegt werden sollte. Diese Annahme wird weiter dadurch bestärkt, dass Apple und Google für die beiden beliebtesten Plattformen iOS und Android[84] ihre eigenen Designrichtlinien veröffentlicht haben[7, 49].

Interessant ist, dass durch die Anwendung dieser Richtlinien, die App intuitiver und einfacher werden kann. Dies liegt einerseits daran, dass diese Richtlinien Regeln enthalten, die speziell auf die Eigenschaften von Mobilgeräten abgestimmt sind. Eine Regel besagt beispielsweise, dass Schaltflächen eine Mindestgröße von 44x44 Pixel haben müssen, um die Größe der Finger zu berücksichtigen.<sup>[4]</sup>.

Andererseits spielt wahrscheinlich auch die Verbreitung dieser Richtlinien eine Rolle. So wirkt sich die Reichweite und der Einfluss von Apple und Google wahrschein-

---

<sup>8</sup>Full-size Tastaturen können bis zu 108 Tasten haben[91]. Die Standard deutsche Texttastatur auf iOS 16.1.2 hat hingegen rund 33 Tasten. Selbst nachgezählt und kontrolliert auf iPhone SE1, SE2 und 13 mini.

<sup>9</sup>Die Monitorgröße wurde anhand der Monitorauflösung abgeschätzt; die meisten Monitore sind zwischen 14 und 23 Zoll groß.

lich auch auf die Bekanntheit und Verbreitung ihrer Richtlinien aus. Durch eine hohe Bekanntheit und Verbreitung würden immer mehr Apps diesen Richtlinien folgen. Dadurch würden sich im Umkehrschluss verschiedene Apps ähnlich bedienen lassen und Nutzer müssten nicht für jede Anwendung eine neue Bedienung erlernen. Dementsprechend finden sich Nutzer in neuen Apps schneller zurecht. Ein Indiz für die Annahme, dass Richtlinien und deren Verbreitung Anwendungen einfacher und intuitiver machen, wäre die größere Popularität von Handy-Apps im Vergleich zu Webseiten<sup>[92]<sup>10</sup></sup>. Denn während für Apps zumindest einige der Regeln aus den Richtlinien, wie zum Beispiel die Knopfgröße, zwingend angewendet werden müssen<sup>[4]</sup>, sind für Webseiten diese Regeln nicht Vorschrift.

## 2.6 Mobilität

Eines der Hauptziele des Handys scheint die Mobilität zu sein. So scheinen alle zuvor erwähnten Unterschiede eine Auswirkung der von Handys gewünschten Eigenschaft: Mobilität. Wenn das Handy nicht mobil sein müsste, könnte es beispielsweise mehr Leistung, einen größeren Bildschirm und eine stabilere Internetverbindung besitzen. Im Vergleich zu Desktop-PCs sind Handys aufgrund ihrer kompakten Größe mobiler. Obwohl Laptops auch mobiler als Desktops sind, können sie in Bezug auf Mobilität nicht mit Handys konkurrieren, da Handys in die meisten Hosentaschen passen, während für Laptops oft eine größere Tasche oder ein Rucksack benötigt wird.

## 2.7 Benutzung

Um PCs zu nutzen, ist ein gewisser Einstiegsaufwand erforderlich. Einerseits muss sich, da Desktops stationär sind, zuerst zum Standort des PCs begeben werden. Darüber hinaus muss der PC vor jeder Nutzung eingeschaltet werden. Denn es wird davon ausgegangen, dass PCs normalerweise ausgeschaltet sind, wenn sie derzeit nicht in Benutzung sind. Das Anschalten dauert in der Regel ein paar Sekunden. Laut dem Benchmark „Startup Timer“ beträgt die schnellste aufgezeichnete Startzeit bis der PC nutzbar ist elf Sekunden<sup>[61]</sup>.

Bei Handys existiert dieser Aufwand hingegen nicht. So wird davon ausgegangen, dass die meisten Nutzer ihr Handy immer bei sich tragen. Das liegt einerseits an der zuvor erwähnten Mobilität, aber andererseits auch in der **Motivation** erwähnten Relevanz und Beliebtheit von Handys. Weiterhin wird davon ausgegangen, dass Handys normalerweise nicht ausgeschalten werden, um unter anderem für wichtige Anrufe oder Nachrichten erreichbar zu sein. Das Handy ist also immer an und benötigt dementsprechend keine Startzeit.

Ein weiterer Aspekt bei der Nutzung von Handys und PCs ist, wie Anwendungen auf den jeweiligen Geräten verwendet werden können. Dabei können Handys im Gegensatz zu PCs immer nur eine Anwendung gleichzeitig anzeigen. Dies könnte aufgrund von Limitationen des Betriebssystems, der Leistung oder des relativ kleinen Displays sein.

---

<sup>10</sup>Laut der Studie werden rund 88% der Nutzung von Handys für Apps und 12% für den Browser verwendet.

## *2. Handy und PC Unterschiede*

Währenddessen unterstützt der PC genau diese Funktion. Er kann mehrere Anwendungen gleichzeitig ausführen und darstellen.

### **2.8 Hardware**

Die meisten Desktop-Computer sind hinsichtlich ihrer Hardware modular aufgebaut. Dadurch haben Benutzer die Möglichkeit, viele Hardwarekomponenten nach Belieben auszutauschen. Allerdings muss das Gehäuse des PCs auch auf diese Anpassungen ausgelegt sein, da es Hardwarekomponenten in verschiedenen Größen und Formen gibt. Die Modularität hat also den Nachteil, dass das Gehäuse groß genug sein muss, um verschiedene Hardwareoptionen zu ermöglichen.

Das Handy unterstützt dementsprechend diese Konfigurierbarkeit nicht. Dafür bietet es sich als all-in-one-Gerät an. Dadurch entfällt im Gegensatz zum PC die Notwendigkeit, sich Gedanken über die Auswahl eines Monitors oder Eingabegeräts zu machen, da alles bereits in einem System integriert ist. Zudem bieten Handys Funktionen, die bei den meisten PCs nicht vorhanden sind, wie beispielsweise eine Kamera, GPS, Gyroskop und die Option für biometrisches Login-Verfahren.

### **2.9 Betriebssystem**

Bei Desktops gibt es eine Vielfalt an Betriebssystemen zur Auswahl und der Nutzer kann sich für eins oder mehrere davon entscheiden. Selbst mit älterer Hardware können oft noch aktuelle Betriebssysteme installiert werden. So unterstützt beispielsweise der fast zehn Jahre alte Prozessor Intel Pentium J1750 das weltweit meistgenutzte und immer noch aktuelle Betriebssystem Windows 10 [55, 81, 83].

Für Handys wird im Gegensatz zu Desktops ein festes Betriebssystem vorgegeben, das sich nicht ohne Weiteres ändern lässt. Zudem werden in der Regel Handy-Betriebssysteme nur für einen Zeitraum von drei bis fünf Jahren unterstützt [48, 85]. Obwohl das Handy nach dieser Zeitspanne noch funktioniert, leiden Sicherheit und Leistung des Geräts ohne weitere Software-Updates. Das hat den Nachteil, dass das Handy alle drei bis fünf Jahre ersetzt werden sollte.

Allerdings könnte die kürzere Unterstützungszeit der Handy-Betriebssysteme auch Vorteile bieten. Die Entwickler können sich auf eine kleinere Anzahl von Handys konzentrieren und die Software besser auf diese optimieren. Daher wird angenommen, dass Handys während des unterstützten Zeitraums beispielsweise mit weniger Defekten und Rucklern auskommen und eine flüssigere grafische Oberfläche bieten.

### **2.10 Software**

Die Software für den PC ist oft konfigurierbar. Bereits bei der Installation können verschiedene Optionen ausgewählt werden, wie beispielsweise der Speicherort, die automatische Aktualisierung, die Desktop-Verknüpfung und das automatische Starten.

In puncto Konfigurierbarkeit scheinen Handys eher auf Benutzerfreundlichkeit ausgelegt zu sein. Bei der Installation neuer Apps genügt oft ein Knopfdruck. Fragen wie

der Speicherort oder die Erstellung einer Verknüpfung werden vom Betriebssystem übernommen. Das Abarbeiten von Optionen entfällt hierbei. Dementsprechend wird angenommen, dass sich der Übergang von der Installation bis zum Nutzen der App für den Nutzer flüssiger gestaltet.

Dieses Verhalten scheint sich auch über die Installation hinaus fortzusetzen. Beispielsweise scheinen Anwendungen mit vielen Optionen generell besser auf dem PC zu funktionieren. So werden Aufgaben wie das Editieren von Videos, die Verwendung von Entwicklungsumgebungen oder das Erstellen von Steuererklärungen werden erfahrungsgemäß üblicherweise am Computer ausgeführt. Möglicherweise liegt das an den zuvor erwähnten Unterschieden und Limitierungen von Handys, wie beispielsweise der Displaygröße. Bei einem kleinen Display und ungenauer Eingabe könnten viele Optionen dazu führen, dass sie schwerer auswählbar und überschaubar werden. Aus Erfahrung spiegelt sich dieser Sachverhalt auch in Apps wieder. So werden für Handys meist nur die relevanten Optionen dargestellt. Das würde bedeuten, dass die restlichen eher nebensächlichen Optionen bereits vom Entwickler oder Betriebssystem getroffen worden.

Daher wird vermutet, dass sich Entwickler bei der Erstellung von Apps mehr Gedanken über passende Entscheidungen machen, da ihnen bewusst ist, dass die Nutzer die Optionen später nicht selbst anpassen können.

Im besten Fall führt das dazu, dass Handynutzer sich weniger oder gar keine Gedanken über die Auswahl von Optionen machen müssen und sich die App dementsprechend intuitiver anfühlt.

### 2.11 Auswertung & Erkenntnisse

Im Folgenden werden die Erkenntnisse und Schlussfolgerungen dieses Abschnittes aufgelistet.

PCs scheinen für jene Aufgaben gut zu funktionieren welche:

1. viel Leistung benötigen.

Durch den ständigen Zugang zur Stromversorgung können leistungsstarke Hardwarekomponenten genutzt werden. Zudem ermöglicht eine kabelgebundene Ethernet-Verbindung eine schnelle und stabile Internetverbindung. Aufgrund der Größe des PCs können auch große Hardwarekomponenten, wie etwa Festplatten mit einer hohen Kapazität, eingebaut werden.

2. schnelle, präzise oder vielfältige Eingaben erfordern.

Die Maus und die Tastatur ermöglichen aufgrund ihrer Größe und des haptischen Feedbacks eine schnellere und präzisere Bedienung. Die vielfältigen Eingaben werden durch die hohe Anzahl an Tasten und die Möglichkeit für Tastenkombinationen ermöglicht.

3. viele Informationen gleichzeitig darstellen oder benötigen.

Dank des großen Displays und der Möglichkeit, mehrere Displays zu verwenden, können viele Details gleichzeitig dargestellt werden. Zudem erlaubt Multitasking die gleichzeitige Anzeige von Informationen aus verschiedenen Anwendungen.

## 2. Handy und PC Unterschiede

4. viele Optionen und Konfigurationen anbieten oder benötigen.  
Die Hardware, das Betriebssystem und weitere Software können bei Desktops nach Belieben ausgetauscht und konfiguriert werden. Zudem ist ein großer Bildschirm von Vorteil, wenn viele Optionen dargestellt werden müssen.
5. langwierig sind oder viel Zeit benötigen.  
Einerseits benötigen Aufgaben auf dem PC generell mehr Startzeit, da der Einstiegsaufwand größer ist. Andererseits helfen die schnellen und präzisen Eingaben dabei lange Aufgaben schneller bewältigen zu können.

Handys scheinen hingegen für jene Aufgaben gut zu funktionieren welche:

1. Ressourcenschonend sind und nicht viel Leistung benötigen.  
Denn das Handy verfügt aufgrund der begrenzten Batteriekapazität über begrenzten Strom, weshalb die Komponenten auf Effizienz ausgelegt sind. Zusätzlich sind WLAN- oder mobile Datenverbindungen oft langsamer und instabiler im Vergleich zu einer kabelgebundenen Ethernet-Verbindung. Darüber hinaus ist aufgrund der begrenzten Größe des Geräts nur eine kleine Festplatte mit eingeschränkter Kapazität möglich.
2. keine schnelle, präzise und vielfältige Eingabe erfordern.  
Das relativ kleine Display kann nur wenige Tasten gleichzeitig darstellen. Der Finger ist größer als ein Mauszeiger und daher unpräziser beim Auswählen. Folglich ist die Eingabe oft langsamer und fehleranfälliger.
3. nur wenig Informationen darstellen oder benötigen.  
Das Display vom Handy ist relativ klein und Multitasking wird meistens nicht unterstützt.
4. ohne viele Optionen und Konfigurationen auskommen.  
Das Betriebssystem und die Hardware sind nicht anpassbar. Zusätzlich übernimmt das Betriebssystem viele eigentlich optionale Entscheidungen, wie beispielsweise das Installationsverzeichnis von Anwendungen. Außerdem können aufgrund des relativ kleinen Displays nicht viele Optionen gleichzeitig angezeigt werden.
5. kurzweilig sind oder wenig Zeit benötigen.  
Das Handy steht dem Nutzer jederzeit zur Verfügung und erfordert nur einen geringen Einstiegsaufwand. Des weiteren fällt das Abarbeiten langer Aufgaben auf dem Handy aufgrund der langsameren und unpräziseren Eingabe schwieriger.
6. lohnenswert sind unterwegs zu lösen.  
Da das Handy mobil ist und sich üblicherweise beim Besitzer befindet sowie in der Regel über eine Internetverbindung verfügt, können Aufgaben unterwegs bearbeitet werden.

7. einfach und intuitiv seien sollen.

Denn zum einen wird die Darstellung der App mithilfe von Richtlinien benutzerfreundlicher. Zum anderen ist die Bedienung des Handys durch unteranderem die Gesten intuitiver. Und schließlich muss sich der Nutzer keine Gedanken über die Konfiguration machen, da das Betriebssystem, die Software und die Hardware bereits passend für das Handy voreingestellt sind.

Diese gesammelten Erkenntnisse lassen sich auch durch Statistiken und Studien deuteln. Eine Studie zeigt zum Beispiel, dass Befragte Computer lieber für wichtige Aufgaben nutzen. Handys hingegen werden als einfacher zu bedienen bewertet, was zu der Erkenntnis der einfachen und intuitiven Aufgaben des Handys passt [58].

Aber auch das allgemeine Nutzerverhalten im Internet deutet darauf hin. Auf Google werden beispielsweise je nach Gerät unterschiedliche Suchanfragen gestellt. Auf dem PC werden eher aufwändige und komplexe Kategorien wie Computer, Elektronik, Arbeit, Ausbildung und Wissenschaft angefragt, während auf dem Handy oft eher einfache und kurze Themen wie Essen, Nachrichten und Sport gesucht werden [28]. Dass diese Aufgaben wirklich einfacher und schneller sind und Handys dafür bevorzugt werden, lässt sich anhand der Aufrufe und Dauer von Webseitenbesuchen beobachten. Obwohl 68% aller Webseitenaufrufe von Handys stammen, machen sie nur 33% der auf Webseiten verbrachten Zeit aus [28]. Das bedeutet, dass Handys hauptsächlich für viele kurze Suchanfragen genutzt werden. PCs werden hingegen eher für weniger, aber dafür längere Aufgaben genutzt.

Ein weiteres Indiz dafür, dass für Handys die Kürze und Einfachheit von Aufgaben wichtig sind, ist eine Messung von Google. Demnach verlassen etwa 53% der Handy-Nutzer eine Webseite, wenn sie länger als drei Sekunden benötigt, um zu laden [50]. Ein ähnliches Verhalten lässt sich auch bei E-Mails feststellen. Laut einer Umfrage von Adobe werden für berufliche E-Mails eher PCs genutzt, während für private E-Mails stattdessen zum Handy gegriffen wird[74]. Da angenommen wird, dass Arbeit eher mit langwierigen Aufgaben verbunden ist, die viele Informationen erfordern, während private Aufgaben eher kurzweilig und einfach sind, wie zum Beispiel das Aussuchen von Essen oder das Schreiben einer Nachricht, stimmt auch dieses Indiz mit den Erkenntnissen überein.

### 3 Anforderungen

Da wir nun wissen, welche Aufgaben auf dem PC und welche auf dem Handy gut funktionieren, kann als nächstes die Frage behandelt werden „was die zu erstellende App überhaupt können und leisten soll“.

**Überblick:** Zunächst wird untersucht, auf welche Arten und mit welchen Techniken versucht wird, diese Frage zu beantworten. Anschließend werden die gewünschten Eigenschaften und Funktionen der App aufgeführt, begründet und bewertet.

**Ergebnisse:** Für die Erhebung der Anforderungen wurden die Techniken der Introspektion, der Umfrage und des Vergleichs gewählt, da sie den besten Ausgleich zwischen Informationsgehalt und Zeitaufwand für diese Arbeit bieten. Bei den Anforderungen wurde stets versucht, Entscheidungen entsprechend der Stärken von Handys und PCs zu treffen. Dementsprechend wurden die nicht funktionalen Anforderungen „Stärken von PCs und Handys“ als am wichtigsten bewertet. Die Stärke des Handys „Benutzbarkeit“ wurde dabei separat betrachtet und als zweitwichtigste nicht funktionale Anforderung bewertet. Weitere gewünschte Eigenschaften sind „Wartbarkeit“, „Qualität“ und eine hohe „Reichweite“. Zuletzt wurden die Funktionen einer „Verbindung zum Backend“, einer „grafischen Darstellung für den Kalender“ sowie eines „Übersetzers für CLI-Terminkalender“ als unbedingt erforderlich eingeschätzt. Sie bilden also das Grundgerüst der App. Aber auch Funktionen wie das „Erstellen, Bearbeiten und Löschen von Erinnerungen“, „Benachrichtigungen“ und „Konfigurationen auf dem PC“ werden für diese Anwendung als wichtig erachtet.

#### 3.1 Vorgehensweise

Als Entwickler ist es oft schwierig zu durchschauen und zu verstehen, welche Funktionalitäten die Software in dem Anwendungsbereich erfüllen sollte.

Deshalb wird in diesem Unterabschnitt genau darüber nachgedacht. Es wird überlegt, wie die Anforderungen am besten ermittelt werden können. Dazu werden verschiedene Erhebungstechniken verglichen.

- **Introspektion:** Für die Introspektion versucht man, durch Nachdenken und Selbstreflexion Anforderungen zu erheben. Da jede Entscheidung in diesem Bereich gut durchdacht sein sollte, wird diese Erhebungstechnik durchgängig und fast immer angewendet. Diese Methode hat zwei Vorteile: Zum einen können Missverständnisse vermieden werden, da man auf Basis eigener Überlegungen arbeitet. Zum anderen ist kein großer Vorbereitungsaufwand nötig und man kann sofort loslegen. Allerdings erfordert die Introspektion ein Mindestmaß an Verständnis der Domäne, da man sonst keine Ideen entwickeln kann.
- **User Feedback:** Als einzelne Person ist es schwierig, alle Anforderungen und Wünsche der Nutzer zu erraten und zu überblicken. Daher sind Erhebungstechniken wie das User Feedback nützlich. Dabei geben Nutzer Feedback über die Software. Damit werden nicht nur existierende Funktionen bewertet, sondern es können auch neue Wünsche und Funktionen geäußert und entdeckt werden. Allerdings wird diese Technik nicht genutzt, da dafür eine lauffähige Software erforderlich ist und erwartet wird, dass eine solche erst zum Ende der Bearbeitungszeit verfügbar steht.

- **Umfragen:** Von daher wurde sich stattdessen für eine Umfrage entschieden. Dabei werden einige Fragen formuliert und einmalig an die Zielgruppe gestellt. Das hat den Vorteil, dass es vermutlich einfacher ist, freiwillige Nutzer für eine einmalige Frage als für einen dauerhaften Test zu finden. Außerdem können gezielte Fragen gestellt werden, um so Feedback zu gewünschten Themen zu erhalten. Jedoch hat diese Technik, genau wie die vorherige, den Nachteil, dass schriftliches Feedback aufgrund fehlender Kontextinformationen leicht missverstanden werden kann. Es muss auch darauf geachtet werden, dass einzelne Antworten nicht als zu wichtig interpretiert werden. Denn obwohl sie für einen Nutzer wichtig sein können, repräsentieren sie möglicherweise nicht die gesamte Zielgruppe.
- **Inspiration durch Vergleiche:** Es existiert zwar noch keine App, wie jene, welche in dieser Arbeit entwickelt werden soll, dennoch ist anzunehmen, dass es in dieser App ähnliche Funktionen und Anforderungen wie in konventionellen Kalender-Apps geben wird. Auch wenn die Anforderungen und Funktionen einer normalen Kalender-App zuerst einfach erscheinen mögen, sollte man sich nicht nur auf seine intuitiven Eindrücke verlassen. Es könnten beispielsweise Eigenheiten und etablierte Standards in Kalender-Apps geben, die man ohne Vergleiche nicht finden würde. Oder es gibt als Funktionen, die als selbstverständlich angesehen werden und deshalb von niemandem angesprochen werden, aber dennoch erwartet werden. Daher scheint es sinnvoll, sich zumindest für die allgemeinen und typischen Funktionen Inspiration zu suchen.
- **Domänenwissen:** Durch das Einarbeiten in die Domäne der CLI-Terminkalender könnte bewusst werden, welche Funktionen und Anforderungen sie besitzen. Dieses Wissen könnte zu Inspiration für neue Ideen und Anforderungen für die App führen. Allerdings wurde sich gegen das Einarbeiten in die Domäne entschieden. Einerseits wird die Einarbeitungszeit als zu hoch eingeschätzt, da es viele CLI-Terminkalender gibt und diese meist komplex und unterschiedlich zu bedienen sind und zudem verschiedene Eigenheiten bieten. Andererseits wird der Erwerb von Informationen als zu gering eingeschätzt. Es wird nämlich vermutet, dass die zu erstellende App sich aufgrund der Unterschiede zwischen PC und Handy sehr von CLI-Terminkalendern unterscheiden wird. Immerhin ist das Ziel nicht, ein solches Programm zu portieren, sondern die Stärken von PC und Handy zu nutzen.
- **Iterative Development:** Das iterative Arbeiten kann auch als Erhebungstechnik bezeichnet werden, da sich während jeder Iteration die Chance bietet, die Anforderungen zu überdenken und das zuvor neu Gelernte darauf anzuwenden. Da für diese Arbeit eine agile Arbeitsweise gewählt wurde, wird auch diese Technik angewendet.

### 3.1.1 Umfrage

Um möglichst wertvolle und aussagekräftige Ergebnisse aus der Umfrage zu erzielen, wurden die Durchführungsbedingungen und Fragen sorgfältig überdacht. Die Wahl

### 3. Anforderungen

des Standorts und die Formulierung der Fragen wurden dabei als entscheidende Faktoren für den Erfolg der Umfrage identifiziert.

**Standort:** Zur Auswahl stehen die folgenden Möglichkeiten: die Durchführung der Umfrage im Bekanntenkreis, an der Universität oder in Online-Foren.

Es wird vermutet, dass sich weder im Bekanntenkreis noch an der Universität viele Nutzer der Zielgruppe finden lassen.

Dementsprechend fällt die Entscheidung auf die Durchführung der Umfrage in Online-Foren. Jedoch ergibt sich nun die Frage, in welchen Foren die Umfrage veröffentlicht werden soll. Dadurch ergibt sich ein noch viel größeres Spektrum an Auswahlmöglichkeiten. Um nicht zu viel Zeit mit der Suche nach einem maßgeschneiderten Forum zu verschwenden, wurden ungefähr die ersten 20 Ergebnisse einer Google-Suche mit dem Suchbegriff „CLI-Calendar Forum“ betrachtet. Dabei wurden unter anderem folgende Foren vorgeschlagen: „reddit: r/commandline“<sup>[72]</sup> , „Stack Exchange: Unix & Linux“<sup>[79]</sup>, „archlinux: Forums“<sup>[20]</sup>, „Debian User Forums“<sup>[26]</sup>, „Linux Mint Forums“<sup>[60]</sup>, „Puppy Linux Discussion Forum“<sup>[67]</sup>. Viele der betrachteten Foren beschränken sich auf ein einzelnes Betriebssystem und haben daher vermutlich eine geringere Reichweite. Das einzige Forum, das dabei heraussticht, ist Reddit. Dementsprechend fiel auch die Wahl auf dieses Forum.

Da der zusätzliche Aufwand für das Veröffentlichen der Umfrage auf einem weiteren Reddit-Foreum als gering eingeschätzt wird, wird die gleiche Umfrage auch auf r/androidapps<sup>[71]</sup> und r/iosapps<sup>[73]</sup> veröffentlicht. Das Ziel ist es, so auch Einblicke in die Wünsche und Erwartungen der App-Nutzer zu erhalten.

**Fragen:** Für die zu stellenden Fragen wurde überlegt, ob sie vordefiniert oder als offenes Konstrukt gestaltet werden sollen. Vordefinierte Fragen haben den Vorteil, dass gezielt Antworten auf bestimmte Fragen erhalten werden können. Allerdings besteht hierbei die Möglichkeit, dass sich Nutzer zu sehr an den Fragen orientieren und dadurch wichtige und interessante Ideen nicht zum Vorschein kommen. Zudem wurde sich zu diesem Zeitpunkt noch keine Anforderungen bezüglich der Anwendung überlegt, was die Erstellung von Fragen erschwert. Daher wurde sich letztendlich für offene Freitextfragen und -antworten Konstrukt entschieden.

Die Umfragen können unter folgenden Links abgerufen werden:

[https://www.reddit.com/r/iosapps/comments/10k3d2c/developing\\_an\\_app\\_for\\_clicalendars\\_opinion\\_poll/](https://www.reddit.com/r/iosapps/comments/10k3d2c/developing_an_app_for_clicalendars_opinion_poll/)

[https://www.reddit.com/r/androidapps/comments/10k3k7w/developing\\_an\\_app\\_for\\_clicalendars\\_opinion\\_poll/](https://www.reddit.com/r/androidapps/comments/10k3k7w/developing_an_app_for_clicalendars_opinion_poll/)

[https://www.reddit.com/r/commandline/comments/10k38bc/developing\\_an\\_app\\_for\\_clicalendars\\_opinion\\_poll/](https://www.reddit.com/r/commandline/comments/10k38bc/developing_an_app_for_clicalendars_opinion_poll/)

#### 3.1.2 Vergleich

Ähnlich wie bei der **Umfrage** werden auch für diese Erhebungstechnik weitere Überlegungen angestellt. Das Ziel dabei ist es, möglichst wertvolle und aussagekräftige Ergebnisse zu erzielen und dabei zugleich zeiteffizient vorzugehen..

Durch den Vergleich sollen lediglich Inspiration sowie allgemeine und offensichtliche Anforderungen gesammelt werden. Das Ziel ist es jedoch nicht, Funktionen und Designs von anderen Apps zu kopieren oder sich von ihnen beeinflussen zu lassen. Aus diesem Grund werden nur wenige Apps zum Vergleich herangezogen und diese auch nur kurzzeitig getestet.

Bei der Auswahl der Apps wurde versucht, solche auszuwählen, die möglichst nützliche Informationen liefern können. Dazu wurden der native Apple iOS Kalender[2] und der Google Kalender[51] ausgewählt, da angenommen wird, dass diese Unternehmen aufgrund ihres Erfolgs, ihrer Größe und da sie eigenen Richtlinien für Apps haben [7, 49], besonders sorgfältig bei der Entwicklung dieser Apps vorgegangen sind.

Darüber hinaus wurde eine App aufgrund von Kundenbewertungen ausgewählt. Bei einer solchen App wäre es nämlich beispielsweise möglich, dass sie aufgrund von Funktionen, die in den anderen beiden Apps nicht vorhanden sind, so positiv bewertet wurde. Aus diesem Grund wurde sich für Calendars[70] entschieden.

## 3.2 Nicht Funktionale Anforderungen

In diesem Abschnitt werden die nicht funktionalen Anforderungen, das sind all jene Eigenschaften, welche die App besitzen soll, erläutert und in eine Aufzählung der Wichtigkeit nach bewertet.

Eine solche Priorisierung hilft dabei zu erkennen, welche Anforderungen für diese Arbeit am wichtigsten sind. Das ist hilfreich, da die begrenzte Arbeitszeit es nicht erlaubt, alle Anforderungen gleich intensiv zu berücksichtigen.

Dafür wird die MoSCoW-Priorisierung verwendet. Dabei werden die Anforderungen in die Kategorien „Must-have“ (unbedingt notwendig für das Projekt), „Should-have“ (wichtig, aber nicht notwendig für den Erfolg der Anwendung), „Could-have“ (wünschenswert, aber nicht unbedingt notwendig), und „Won’t-have“ (erwähnenswert, werden aber bewusst nicht umgesetzt) eingeteilt. Im Folgenden werden diese Kategorien mit M, S, C und W abgekürzt.

- **M Stärken von PCs und Handys:** Die wohl wichtigste Anforderung an die App besteht darin, die Vorteile von PCs und Handys optimal zu nutzen, da dies ein Hauptziel dieser Arbeit ist. Hierbei sollen alle Erkenntnisse berücksichtigt werden, die in [Abschnitt 2. Handy und PC Unterschiede](#) gesammelt wurden.
- **M Benutzbarkeit:** In dieser Arbeit wird unter „Benutzbarkeit“ verstanden, dass eine Anwendung intuitiv, einfach und effektiv nutzbar ist. Eine Erkenntnis aus [Abschnitt 2. Handy und PC Unterschiede](#) ist, dass genau das eine Stärke der Handys ist. Daher sollte diese Eigenschaft auch in der zu entwickelnden App sichtbar werden. Die Benutzbarkeit kann dabei helfen, dass die App nicht nur nützliches im Konzept, sondern auch nützlich für den Endnutzer ist sowie das möglichst viele Personen auch ohne große Einarbeitung und Vorwissen die App nutzen können. Um die Benutzbarkeit der App zu verbessern, scheint das Design der App eine wichtige Variable zu sein, da es das einzige ist, mit dem der Nutzer interagiert. Diese Erkenntnis wurde bereits in [Abschnitt 2. Handy und PC Unterschiede](#)

### 3. Anforderungen

festgestellt, zusammen mit der Feststellung, dass Richtlinien für gutes Design wichtig sind. Unter anderem wurde begründet, dass Handy-Nutzer Apps bevorzugen [92], da diese passende Richtlinien befolgen, während Websites eher für PCs ausgelegt sind und daher weniger auf für Handys passende Konventionen achten. Eine weitere Statistik, die auf die Bedeutung einer guten Benutzbarkeit hinweist, ist, dass Benutzer eher dazu neigen, eine Anwendung weiter zu nutzen, wenn sie intuitiv und einfach zu bedienen ist [89].

Dementsprechend wird die Benutzbarkeit als eine wichtige Anforderung bewertet.

- **S Wartbarkeit, Erweiterbarkeit, Verständlichkeit:** Da, wie zuvor erwähnt, die Bachelorarbeit nur eine begrenzte Bearbeitungszeit zulässt, muss damit gerechnet werden, dass nicht alle Funktionen bis zum Abgabetermin umgesetzt werden können. Daher sollte der Quellcode so gestaltet sein, dass er für zukünftige Entwicklungen und auch für Dritte leicht zugänglich ist. Um dies zu erreichen, muss der Programmcode möglichst wartbar sein, was bedeutet, dass er verständlich und erweiterbar sein sollte.
- **S Qualität & Korrektheit:** Mit dieser Anforderung soll sichergestellt werden, dass die App sich genau so verhält, wie zuvor spezifiziert. Anders ausgedrückt soll es zu keinen unerwarteten Situationen wie Fehlern oder Abstürzen kommen. Falls dies dennoch der Fall ist, würde dies nicht nur die Benutzbarkeit einschränken, sondern auch die Nutzer irritieren und möglicherweise dazu bewegen, die App nicht weiter zu nutzen. Laut einer Umfrage würden 88% der Nutzer die App bei einem Fehler verlassen[68]. Daten von Google bekräftigen diese Aussage, so handeln laut ihnen 54% aller 1-Sterne-Bewertungen im Play Store von Fehlern oder Stabilitätsproblemen [52]. Aus diesem Grund wird diese Anforderung als durchaus wichtig eingeschätzt.  
Um die Qualität sicherzustellen, darf man entweder beim Programmieren keine Fehler machen, was aus Erfahrung keine sinnvolle Annahme ist, oder man testet die Software ausgiebig genug, um nachzuweisen, dass die Software keine Defekte aufweist. Daher wird die zu erstellende Anwendung ausgiebig getestet.
- **C Leistung:** Eine Erkenntnis aus **Abschnitt 2. Handy und PC Unterschiede** ist, dass Anwendungen für das Handy nicht zu viel Leistung und Ressourcen benötigen sollten. Dementsprechend wird versucht, den Speicher-, Prozessor- und Netzwerkverbrauch zu minimieren. Ein zu großer Speicherbedarf könnte zum Beispiel dazu führen, dass die App aufgrund begrenzter Speicherkapazität nicht genutzt oder entfernt wird. Eine zu aufwändige App kann auch zu Ruckeln auf der grafischen Oberfläche und schneller entleerender Batterie führen. Beides davon würde zu einer Einschränkung der Benutzbarkeit und der Benutzererfahrung führen. Der als am wichtigsten bewertete Grund für die Beachtung der Performance ist jedoch, dass eine zu netzwerk- oder prozessorintensive App zu langen Ladezeiten führen kann. Diese langen Ladezeiten würden gegen die Stärke des Smartphones für kurzweilige Aufgaben widersprechen. Eine Messung von Google bekräftigt diese Annahme. So werden laut ihr 53% aller Webseiten

verlassen, wenn das Laden länger als drei Sekunden dauert[50].

- **C Reichweite:** Auch wichtig, aber im Vergleich zu den anderen Anforderungen eher zweitrangig, ist die Reichweite. Zwar soll es möglichst vielen Handy-Nutzern ermöglicht werden, die zu erstellende App zu nutzen, jedoch wird es sich bei der App wahrscheinlich eher um eine Nischenanwendung handeln. Denn bei den bereits existierenden CLI-Terminkalendern, auf welche die App aufbauen soll, scheint es sich, anhand der eher wenigen GitHub-Sterne, auch bereits um Nischenanwendungen zu handeln[59, 64]. Daher soll nicht zu viel Aufwand in diese Eigenschaft fließen.

Am wichtigsten für die Reichweite ist es, dass die App für Android- und iOS-Systeme verfügbar ist, da dies die beliebtesten Systeme für Handys sind[84]. Aber auch das Alter könnte dafür wichtig sein. Wenn die zu erstellende App eine zu hohe Android- oder Apple-Version voraussetzt, können ältere Handys diese nicht verwenden.

### 3.3 Funktionale Anforderungen

Dieser Abschnitt beschäftigt sich mit den funktionalen Anforderungen, also dem, was das System können soll. Wie bereits im **Unterabschnitt 3.2. Nicht Funktionale Anforderungen**, wird auch hier aus denselben Gründen die MoSCoW-Priorisierung verwendet.

- **M Verbindung mit Backend:** Damit das Handy und der PC zusammen genutzt werden können, müssen sich diese miteinander verbinden können. Dazu soll ein Backend verwendet werden. Die Wahl fiel auf GitHub, da es für diese Arbeit mehrere Vorteile bietet. Es bietet es einerseits viele interessante Funktionen, die in die App integriert werden könnten, wie beispielsweise die Versionsverwaltung, Zugriffskontrolle und die Möglichkeit von Entwicklungszweigen. Darüber hinaus wird vermutet, dass CLI-Terminkalender-Nutzer auch gerne Git nutzen, da es sich dabei um eine beliebte CLI-Anwendung handelt<sup>11</sup>. Dementsprechend müssten viele Nutzer keine neue Anwendung erlernen, um die App zu nutzen.
  - **M [+ C] Übersetzer für CLI-Terminkalender:** Damit Daten verarbeitet und ausgetauscht werden können, ist neben der Verbindung der Geräte auch eine Kommunikation erforderlich. Zur Ermöglichung dieser Kommunikation bieten sich zwei Optionen an: die Erstellung eines neuen CLI-Terminkalenders, der mit der App kommunizieren kann, oder die Erstellung eines Parsers bzw. Übersetzers für bereits existierende CLI-Terminkalender. Es wurde sich für die zweite Option entschieden, da sich dadurch einerseits auf die App konzentriert werden kann und andererseits mehrere verschiedene CLI-Terminkalender mit der App funktionieren können. Dadurch würde sich die Zielgruppe und der Nutzen der App erhöhen.
- Zunächst wird sich jedoch nur auf die Übersetzung für einen CLI-Terminkalender konzentriert, da einer ausreicht, um die restlichen Funktionen der App zu tes-

---

<sup>11</sup>Beispielsweise ist Git bereits standardmäßig in Ubuntu vorinstalliert.[88]

### 3. Anforderungen

ten und vorzuführen. Weitere CLI-Terminkalender können immer noch zu einem späteren Zeitpunkt hinzugefügt werden, um so die Zielgruppe zu erweitern.

- **M [+ C] Kalender Darstellung:** Eine Aufgabe, die gut zum Handy passt, da es eine kurzweilige und einfache Aufgabe ist, die man durchaus unterwegs lösen möchte, ist das Ansehen von anstehenden Terminen. Dafür benötigt es eine Darstellung für den Kalender. Dabei wurde sich von bereits existierenden und konventionellen Darstellungen inspiriert, um eine möglichst einfache und intuitive Benutzung zu ermöglichen. Die verglichenen Apps zeigten dabei oft mehrere verschiedene Darstellungen für den Kalender an. Um Zeit zu sparen und da nicht alle Darstellungen zum Benutzen des Kalenders notwendig sind, wurden vorerst nur zwei Darstellungen fokussiert. Es wurde sich für die Monats- und Tagesansicht entschieden, da sie zusammen einen guten Ausgleich zwischen Details und Übersicht bieten.  
Um verschiedene Darstellungen an Details und Übersicht zu ermöglichen können nachwirkend noch weitere Ansichten wie eine Listen-, Wochen- oder Jahresansicht hinzugefügt werden.
- **S Einträge erstellen, bearbeiten, löschen:** Eine weitere Aufgabe, die gut zum Handy passt, da es eine kurzweilige und einfache Aufgabe ist, die man durchaus unterwegs lösen möchte, ist das Erstellen, Bearbeiten und Löschen von Terminen. Da die Erstellung von CLI-Terminkalendereinträgen eine eigene Syntax mit vielen Sonderzeichen erfordert, sollte diese Funktion jedoch nicht genau wie auf dem PC über ein Terminal und Texteingabe gelöst werden. Stattdessen sollen neue Einträge auf dem Handy in einer separaten Erinnerungsliste erstellt werden. Für den PC sollen die Erinnerungen mit Hilfe von GitHub sichtbar gemacht werden. Zwar entsteht hierbei der Nachteil, dass eine gewisse Doppelarbeit entsteht, da die Einträge auf dem Handy lediglich als Erinnerung dienen und zu einem späteren Zeitpunkt auf dem PC vervollständigt werden müssen. Jedoch hat diese Methode den Vorteil, dass die Aufgabe passend den Stärken der Geräte aufgeteilt wird. Das Handy wird für einfache und kleine Einträge genutzt, während auf dem PC die Einträge vervollständigt werden können, um ausführliche Beschreibungen mit komplexen Zeichen und Syntax zu erstellen.
- **S Einschränkungen:** Im [Abschnitt 2. Handy und PC Unterschiede](#) wurden wiederholt Einschränkungen erwähnt, die dem Handy bewusst auferlegt wurden, um so positiven Eigenschaften wie Einfachheit zu betonen. Eine ähnliche Idee wird auch hier verfolgt. Einige Funktionen sollen begrenzt werden, um die Nutzung der App auf die Stärken des Handys zu beschränken, beispielsweise auf kurzweilige und einfache Aufgaben. So könnten zum Beispiel eine maximale Textlänge für neue Einträge auf dem Handy festgelegt werden, um zu vermeiden, dass zu viel auf dem Handy geschrieben wird. Außerdem könnte die maximale Anzahl der Erinnerungseinträge begrenzt werden, um so eine unübersichtlich lange Liste auf dem Handy zu vermeiden.
- **S Benachrichtigungen:** Eine Aufgabe, die sich besser für das Handy eignet als für den PC, sind Erinnerungsbenachrichtigungen für Termine. Das Handy kann den Nutzer fast immer benachrichtigen, da wie in [Abschnitt 2. Handy und PC](#)

**Unterschiede** erwähnt wurde, dass davon ausgegangen wird, dass das Handy fast immer beim Nutzer angeschaltet ist. Im Gegensatz dazu ist der PC stationär und wird in der Regel ausgeschaltet, wenn er nicht benutzt wird. Dadurch hat er weniger Möglichkeiten, den Nutzer zu benachrichtigen.

Daher wird in der App die Möglichkeit für Benachrichtigungen von Terminen angeboten. Allerdings wurde entschieden, diese Funktion nicht für die zuvor erwähnten Erinnerungseinträge anzubieten, da dies dazu führen könnte, dass Erinnerungen als Termine genutzt werden. Dies würde dem zuvor festgelegten Design widersprechen, das vorsieht, dass das Handy für kurze Aufgaben wie Erinnerungen genutzt wird, während auf dem PC komplexere Aufgaben erledigt werden.

- **S Konfiguration auf dem PC:** Für Optionen, die von möglicherweise Nutzern ändern wollen, wird eine Konfigurationsmöglichkeit bereitgestellt. Dazu gehören beispielsweise die Standardzeit für Erinnerungen oder Einstellungen zu den zuvor erwähnten Einschränkungen. Da in **Abschnitt 2. Handy und PC Unterschiede** festgestellt wurde, dass sich PCs eher als Handys zum Konfigurieren eignen, soll diese Funktion über den PC ermöglicht werden.
- **C Suchfunktion:** Um unterwegs schnell gezielte Informationen zu Terminen abrufen zu können, soll eine Suchfunktion implementiert werden. Da die Informationen auch über die grafische Oberfläche oder den PC abgerufen werden können, wird diese Aufgabe jedoch als zweitrangig betrachtet.
- **C Weitere Kalender Abonnieren & Teilen:** Die Funktionen, mehrere Terminkalender als einen darzustellen sowie eigene Terminkalender zu teilen und variable Zugriffsrechte zu bestimmen, sind in allen drei getesteten Apps wiederkehrende Funktionen und scheinen dementsprechend Standardfunktionen in Kalendern zu sein.
- **C Offline Funktionen:** Da die App lediglich zum Herunterladen und Hochladen von Dateien eine Internetverbindung benötigt, ist eine permanente Verbindung nicht zwingend erforderlich. Da jedoch fast alle Smartphones heutzutage über eine Internetverbindung verfügen, wird dieser Funktion vorerst weniger Aufmerksamkeit geschenkt.
- **W Anleitung:** Um die Nutzung der App zu vereinfachen, könnte es beim ersten Start eine Einführung mittels Anleitung geben. Noch besser wäre jedoch, wenn die App so intuitiv gestaltet ist, dass keine Anleitung benötigt wird.
- **W Commit-History:** Eine möglicherweise interessante Funktion, die durch GitHub ermöglicht wird, ist die Darstellung der Commit-History als Graph in der App. Dadurch können die letzten Veränderungen und Versionen betrachtet werden. Da unter anderem für die Darstellung lange Listen und viele Details erforderlich sind, wird diese Funktion jedoch nicht als besonders geeignet für die Nutzung auf dem Handy angesehen. Daher wird diese Funktion vorerst nicht implementiert.

## 4 Technologische Überlegungen

In diesem Abschnitt werden sich Gedanken zur Auswahl der Technologien gemacht, da diese Auswahl bereits Auswirkungen auf die funktionalen und nicht-funktionalen Anforderungen haben kann.

**Überblick:** Zunächst wird behandelt, mithilfe welchen Frameworks die App erstellt werden soll. Anschließend wird sich für einen CLI-Terminkalender, für die Darstellungen der Erinnerungen und für das Dateiformat der Konfigurationsdatei entschieden. Danach wird die Wahl der verwendeten Software und Entwicklungsumgebung begründet und schließlich werden für die Arbeit relevante Pakete ausgewählt.

**Ergebnisse:** Im ersten [Unterabschnitt 4.1. Auswahl des Frameworks](#) wurde behandelt, ob die Anwendung als App oder Webseite und mithilfe von Native oder Cross Platform Frameworks erstellt werden sollen. Die Wahl fiel dabei erstens auf eine App, da diese in der Regel intuitiver ausfallen können und beliebter sind, und zweitens auf Cross Platform Frameworks, da damit Apps für iOS und Android erstellt werden können. Zuletzt wurden noch die beiden Frameworks Flutter und React Native miteinander verglichen. Aufgrund von Beliebtheit, Performance und der Annahme, dass damit effizient gearbeitet werden kann, fiel die Wahl auf Flutter.

Die [CLI-Terminkalenderwahl](#) viel auf When und das [Konfigurationsdateiformat](#) auf JSON, da durch dieser Wahl erhofft wird, da durch diese Entscheidungen erhofft wird, während der Entwicklung ein evaluierbares Produkt zu erstellen. Weiter wurde für die [Darstellung der Erinnerungen](#) Issues gewählt, da diese ein passendes Format besitzen und zudem die GitHub-Webseite genutzt werden kann, um die Erinnerungen einzusehen.

Im darauf folgenden [Unterabschnitt 4.5. Entwicklungsumgebung](#) wurde entschieden, MacOS als Betriebssystem, Android Studio als IDE, GitLab zur Versionsverwaltung und Emulatoren sowie Handys zum Testen zu nutzen. Es wurde auch erwähnt, dass die Versionen dieser Software während der Arbeit nicht verändert werden, um so mögliche Komplikationen zu vermeiden.

Im letzten [Unterabschnitt 4.6. Auswahl der Pakete](#) wurden die für die Arbeit benötigten Pakete nach ihrem Alter, ihrer Beliebtheit und ihrem Update-Verlauf ausgewählt. Die Wahl fiel auf json\_serializable, tests und flutter\_lints, da sie die Lesbarkeit des Codes verbessern und nützliche Funktionen bieten. Weiter wurde das Paket github ausgewählt, da es das einzige verfügbare Paket ist, das eine Schnittstelle zum Backend bietet. Schließlich wurde das Paket syncfusion\_flutter\_calendar ausgewählt, da es eine Kalendarendarstellung bereitstellt und somit Zeit gespart wird, da diese nicht selbst implementiert werden muss.

### 4.1 Auswahl des Frameworks

In diesem Abschnitt wird die Frage behandelt, wie die Anwendung erstellt werden soll. Dabei bestehen die Optionen die Anwendung als eine Webseite oder als Applikation zu erstellen. Zwar bieten Webseiten einige Vorteile im Vergleich zu Applikationen, zum Beispiel die Plattformunabhängigkeit. Wie zuvor in [Abschnitt 2. Handy und PC Unterschiede](#) erwähnt, können Apps jedoch durch Richtlinien und Gesten aber um einiges intuitiver sein und Nutzer scheinen diese generell zu präferieren [92].

Daher soll die Anwendung als Applikation entwickelt werden.

Nun kann sich für ein Framework entschieden werden. Die erste Wahl liegt dabei zwischen einem Native-Framework oder Cross-Platform-Framework.

Generell scheinen Native-Frameworks eine bessere Performance als Cross-Platform-Frameworks zu bieten [57, 56]. Außerdem sehen und fühlen sich die nativ erstellte Apps einheitlich mit der Plattform an, da für diese Apps plattformspezifischen Funktionen und Komponenten benutzt werden. Das würde wahrscheinlich zu einer intuitiveren Benutzung für den Nutzer führen, da dies für ihm bereits bekannte Muster und Funktionen wären. Da Performance und Benutzbarkeit zwei Anforderungen für diese Arbeit sind scheinen native-SDKs eine gute Wahl für diese Arbeit zu sein.

Jedoch wurde sich für eine Cross-Platform-Framework entschieden. Einerseits wird der Performance-Verlust als marginal eingeschätzt, da es sich bei der zu erstellenden App wahrscheinlich um eine simple Anwendung ohne schwierige Berechnungen oder aufwändigen Animationen handeln wird. Andererseits kann die grafische Oberfläche auch versucht werden mit Cross-Platform-Frameworks passend für das System zu erstellen. Zwar wäre das ein größerer Aufwand als bei Native-Frameworks, aber dafür sind die Anwendungen von Cross-Platform-Frameworks mit iOS und Android kompatibel. Wie in [Abschnitt 3. Anforderungen](#) besprochen, ist dies der wichtigste Punkt, um eine große Reichweite zu ermöglichen. Zwar wäre dies auch möglich, indem für iOS und Android jeweils eine eigene Codebasis über native SDKs erstellt würde, jedoch müssten dann auch zwei Codebasen gepflegt werden. Durch die relativ kurze Bearbeitungszeit dieser Arbeit erscheint diese Idee weniger sinnvoll. Stattdessen wird versucht, mithilfe von Cross-Platform-Frameworks möglichst schnell und zeiteffizient eine evaluierbare Anwendung für beide Plattformen zu entwickeln. Falls sich die App später als nützlich und beliebt herausstellt, kann immer noch eine Native-Entwicklung gestartet werden.

Zuletzt muss eine Entscheidung für ein konkretes Framework getroffen werden. Wie in [Anforderungen](#) erwähnt, ist die Weiterentwickelbarkeit eine wichtige Anforderung. Daher ist es einerseits entscheidend, ein Framework auszuwählen, das möglichst lange unterstützt wird, um eine zukünftige Weiterentwicklung zu gewährleisten.

Andererseits muss auch berücksichtigt werden, dass das Framework beliebt ist und von vielen Personen genutzt wird, um die Chance zu erhöhen, dass Interessenten gefunden werden können.

Basierend auf der Update-Historie und der Sternbewertung auf Github sind React-Native und Flutter die beiden beliebtesten Frameworks für die Cross-Platform-App-Entwicklung[44, 29].

Wenn man die beiden Frameworks dementsprechend miteinander vergleicht, scheint Flutter beliebter zu sein. Es hat mit 150.000 Sternen auf GitHub etwa 38% mehr als React Native[44, 29]. Ein ähnliches Ergebnis zeigt sich auch bei Google Trends, da Flutter fast doppelt so viele Suchanfragen wie React Native hat[45].

Im Bereich der Performance verhält es sich ähnlich. Laut zwei Analysen von inVerita scheint Flutter ressourcensparender und schneller als React Native zu sein[57, 56].

Dafür bietet React Native den Vorteil, plattformspezifische Komponenten und Funktionen nutzen zu können. Wie zuvor erwähnt wurde, könnte dies zu einer verbessern-

## 4. Technologische Überlegungen

ten Benutzbarkeit führen.

Flutter bietet im Vergleich zu React-Native viele vorgefertigte Komponenten und Features, was letztendlich zur Entscheidung für dieses Framework geführt hat. Während React-Native nur 25 Core-Komponenten hat, besitzt Flutter allein für Animationen bereits 22 Komponenten[46, 66]. Diese vordefinierten Komponenten können dabei helfen, zeiteffizient vorzugehen, da sie sonst möglicherweise selbst implementiert werden müssten.

### 4.2 CLI-Terminkalenderwahl

Wie im [Unterabschnitt 3.3. Funktionale Anforderungen](#) erwähnt, soll ein Übersetzer für einen CLI-Terminkalender entwickelt werden. Dazu muss sich vorerst für einen entschieden werden.

Für die Wahl des CLI-Terminkalenders wurden calcurve[22], khal[64], remind[27], When[21] in betracht gezogen und sich [letztendlich] für When entschieden. Dieser beschreibt sich in seiner Dokumentation mit „minimalistic“ und „It's a very short and simple program“[21]. Dementsprechend wird angenommen, dass es für diesen Terminkalender am einfachsten ist, einen passenden Übersetzer zu entwickeln, und dies hilft wiederum dabei, in der Bearbeitungszeit ein evaluierbares Produkt zu erstellen. Falls sich die Anwendung nach der Evaluation als nützlich erweist, können immer noch weitere Übersetzer hinzugefügt werden.

### 4.3 Darstellung der Erinnerungen

Wie im [Unterabschnitt 3.3. Funktionale Anforderungen](#) erwähnt, soll es die Funktion geben Erinnerungen zu erstellen die mithilfe von GitHub zwischen dem Handy und dem PC übertragen werden.

Eine Möglichkeit hierfür wäre, die Erinnerungen als Dateien im GitHub-Repository zu speichern. Die jeweiligen Dateien können dann von beiden Geräten heruntergeladen werden.

Eine andere Möglichkeit, für welches sich letztendlich auch entschieden wurde, ist es die Erinnerungen als GitHub Issues darzustellen. Dies hätte den Vorteil, dass die Erinnerung neben der App und dem Terminal auch über die Webseite darstellbar sind. Darüber hinaus eignet sich das Format von Issues gut für Erinnerungen, da es die Möglichkeit bietet, einen Titel, eine Beschreibung, Dateien und verschiedene Status hinzuzufügen.

Ein Nachteil besteht jedoch darin, dass es nicht möglich ist, die GitHub Issues direkt über das Programm Git anzuzeigen. Daher wird eine weitere CLI-Anwendung benötigt, die diese Funktion unterstützt. Da GitHub jedoch bereits eine passende Anwendung anbietet[37], wird dies nicht als zu gravierender Nachteil betrachtet.

### 4.4 Konfigurationsdateiformat

Wie im [Unterabschnitt 3.3. Funktionale Anforderungen](#) wurde erwähnt, dass es möglich sein soll, Einstellungen für die Anwendung über den PC zu konfigurieren. Dazu soll die Konfigurationsdatei mithilfe von GitHub zwischen dem Handy und dem PC übertragen werden.

Nun wurde sich die Frage gestellt, in welchem Format die Datei vorliegen sollte. Es wäre wahrscheinlich passend, eine Textdatei zu verwenden, da diese zur Unix-Philosophie[90] und damit zu CLI-Programmen passen würde. Jedoch wurde zunächst entschieden, eine JSON-Datei zu verwenden, da sie einfacher mit Flutter verwendet werden kann und dadurch Arbeitszeit eingespart wird. Zudem kann das Dateiformat auch nach der Bachelorarbeit immer noch in eine Textdatei umgewandelt werden.

## 4.5 Entwicklungsumgebung

In diesem Abschnitt werden die verwendeten Software und deren Versionen aufgelistet. Die Auswahl der Software und ihrer Versionen kann bereits einen Einfluss auf das Endprodukt haben. So wäre es beispielsweise durch die Wahl von Windows nicht möglich, Anwendungen für iOS zu entwickeln.

Darüber hinaus wird durch die Nennung der Software sichergestellt, dass die Anwendung reproduzierbar ist. Durch die Verwendung unterschiedlicher Software oder anderer Versionen können erfahrungsgemäß Komplikationen auftreten. Zum Beispiel funktionierte Android Studio nach einem Update während der Arbeit nicht mehr ordnungsgemäß. Daher wurden alle verwendeten Software zu diesem Zeitpunkt auf die neueste Version aktualisiert und bis zum Ende der Arbeit auf dieser Version beibehalten.

1. Betriebssystem: Für das Betriebssystem standen Windows und MacOS zur Verfügung. Es wurde sich für MacOS entschieden, da auf diesem Betriebssystem die Entwicklung von iOS- und Android-Apps möglich ist. Version: macOS Ventura 13.2.1
2. IDE: Flutter empfiehlt unter anderem Visual Studio Code, Android Studio oder Emacs als Editor[30]. Da sowohl Flutter als auch Android Studio von Google entwickelt wurden, wird erwartet, dass der Editor besonders gut auf das Framework abgestimmt ist. Aus diesem Grund und weil Android Studio speziell für die App-Entwicklung ausgelegt ist, wurde er schlussendlich ausgewählt. Version: 2022.1.1
3. Framework: Flutter. Mehr dazu in [Unterabschnitt 4.1. Auswahl des Frameworks](#). Version: 3.7.3
4. Versionsverwaltung: Für die Versionsverwaltung wurde GitLab gewählt, da im Studium damit bereits Erfahrung gesammelt wurde.  
Um ein verständliches und leicht weiterentwickelbares Projekt zu erstellen, wurden einige Einstellungen in GitLab getroffen. So werden beispielsweise einheitliche und aussagekräftige Commit-Nachrichten verwendet, sodass sie auch für Dritte die Nachricht nachvollziehen können. Dabei wurde sich an das bereits existierende Regelwerk „Conventional Commits“[1] gehalten. Darüber hinaus wurde eine CI/CD-Pipeline erstellt, um Tests automatisch durchzuführen. Auf diese Weise werden Entwickler automatisch auf mögliche Fehler ihrer Änderungen aufmerksam gemacht. Version: git 2.37.1 (Apple Git-137.1)

#### 4. Technologische Überlegungen

5. Emulatoren: Zum Testen der App werden Emulatoren verwendet. Um sicherzustellen, dass die Anwendung auf Android und iOS sowie auf den neuesten und älteren Betriebssystemversionen läuft, werden sowohl ein iPhone- als auch ein Android-Emulator mit der jeweils neuesten und ältesten verfügbaren Version verwendet. Zusätzlich wurde ein echtes Handy für Tests verwendet, da vermutet wird, dass unter realen Bedingungen die Performance-Tests aussagekräftiger sind. Hierbei wurde das iPhone SE1 gezielt ausgewählt, da es sich um ein relativ altes Handy handelt. Dadurch sollte sich prüfen lassen, ob die Anwendung auch von älterer und schwächerer Hardware unterstützt wird. Außerdem hat das Handy im Gegensatz zu neueren Geräten eine relativ kleine Bildschirmdiagonale von vier Zoll. Dadurch lässt sich prüfen, ob das App-Design auch auf kleineren Bildschirmen funktioniert. Versionen: iPhone 14 iOS 16.2 & 13.7, Pixel 6 Android 13.0 & 5.0, iPhone SE1 iOS 15.7.3
6. CLI-Terminkalender: When. Mehr dazu im [Unterabschnitt 4.2. CLI-Terminkalenderwahl](#). Version 1.1.45
7. Weiteres: Die folgende Software wird von einigen der zuvor genannten Technologien standardmäßig benötigt. Daher wird hier lediglich die Version aufgeführt, ohne näher darauf einzugehen: Xcode 14.2, Android SDK Platform-Tools: 34.0.0, DevTools: 2.20.1, Dart 2.19.2

#### 4.6 Auswahl der Pakete

In diesem Abschnitt wird die Auswahl der Pakete behandelt. Zunächst wird erläutert, warum und nach welchen Kriterien die Pakete generell ausgewählt wurden. Anschließend werden die wichtigsten Pakete aufgelistet und ihre Verwendung begründet.

Die Pakete wurden nach einem ähnlichen Verfahren wie in [Unterabschnitt 4.1. Auswahl des Frameworks](#) ausgewählt. Es wurde dabei auf ihr Alter, ihre Beliebtheit und den Verlauf ihrer Aktualisierungen geachtet. Dabei wird nämlich angenommen, dass bei einer starken Ausprägung dieser drei Kriterien es wahrscheinlicher ist, dass ein Paket noch lange Zeit weitere Unterstützung und Aktualisierungen erhält. Das ist vorteilhaft, wenn die erstellte Anwendung auch in Zukunft relevant bleiben soll. Zudem kann eine gute Dokumentation und Bekanntheit der Pakete die Verständlichkeit erleichtern. Daher wurde, falls mehrere Pakete mit ähnlichen Funktionen zur Verfügung stehen, stets das ausgewählt, welches in diesen drei Punkten am besten abschneidet.

Pakete:

- json\_serializable[53]: Dieses Paket erstellt automatisch Klassen für JSON-Serialisierung und -Deserialisierung, die für die Übersetzung von Daten zu JSON-Objekten und umgekehrt benötigt werden. Einerseits wird durch die automatische Erstellung etwas Zeit gespart. Einerseits spart die automatische Erstellung Zeit, andererseits haben die automatisch erstellten Klassen alle das gleiche Muster, wodurch der Code und das Projekt einheitlicher werden und eine bessere Struktur und Übersichtlichkeit entsteht. Das Paket ist mit 2.550 Likes und 99% Popularität<sup>12</sup> das beliebteste in dieser Liste. Aufgrund dieser beiden Punkte wird erwar-

---

<sup>12</sup>Popularität ist eine eigene Bewertungskategorie auf dart.dev.

tet, dass andere Programmierer diese Klassen leichter erkennen und verstehen können, was letztendlich die Verständlichkeit erleichtern sollte.

- `tests`[25]: Aus demselben Grund wurde das Testpaket für Flutter ausgewählt. Es stellt einheitliche Muster für das Schreiben von Tests bereit, was zu einer übersichtlichen Struktur führt.
- `lint`[62]: Linter legen eine Reihe von Regeln für Programmierpraktiken fest, zum Beispiel, dass eine Textzeile nicht länger als 80 Zeichen sein darf oder dass die Benennung von Variablen stets mit einem kleinen Buchstaben beginnen muss. Durch die Nutzung eines solchen Pakets wird einerseits Zeit gespart, da sich der Entwickler weniger Gedanken um den Code-Style machen muss. Stattdessen können die vordefinierten Regeln des Paketes verwendet werden. Andererseits trägt es dazu bei, den Quellcode generell übersichtlicher zu gestalten, da durch das Paket in jeder Datei eine einheitliche Formatierung und Stil verwendet wird. Zunächst wurde das Paket `flutter_lints`[31] ausgewählt, da vermutet wurde, dass die Regeln gut durchdacht und der Programmiersprache und dem Framework angemessen sind, da es sich bei den Erstellern des Pakets um „`flutter.dev`“ handelt. Beim Programmieren stellte sich jedoch heraus, dass zwar viele Regeln angeboten werden, aber die meisten standardmäßig deaktiviert sind. Das Paket ist also darauf ausgelegt vom Entwickler angepasst zu werden. Das Ziel war jedoch, einen Linter mit vordefinierten Regeln zu verwenden. Daher wurde auf das Paket `lint` umgestellt. Dieses bietet eine Reihe von vordefinierten Regeln an und folgt dem Dart Style Guide[24]. Somit sollten die gleichen Vorteile wie bei `flutter_lints` erzielt werden.
- `github`[78]: Um mit dem Backend kommunizieren zu können, wird eine Schnittstelle benötigt, die die Programmiersprache mit dem Backend verbindet. Eine solche Schnittstelle zu erstellen ist eine umfangreiche und zeitaufwändige Aufgabe. Es würde also den Zeitrahmen dieser Arbeit sprengen, wenn versucht würde, diese Aufgabe eigenständig zu bewältigen. Daher wird stattdessen das Paket „`github`“ verwendet, das sich genau dieser Aufgabe widmet. Es stellt eine Verbindung zur GitHub-API bereit, damit über die Programmiersprache mit GitHub interagiert werden kann. Dieses Paket ist dabei das einzige, welches eine solche Schnittstelle für Flutter zur Verfügung stellt. Dies könnte sich als Nachteil herausstellen, da bei Komplikationen, Fehlern oder fehlender weiterer Unterstützung das Paket aufgrund fehlender Alternativen nicht ersetzt werden kann.
- `syncfusion_flutter_calendar`[86]: Um Zeit zu sparen und sicherzustellen, dass während der Bearbeitungszeit ein funktionsfähiges Produkt erstellt wird, wird vorerst ein vorhandenes Package für die Terminansichten verwendet. [Bei Bedarf kann dies jederzeit durch ein eigenes Design ersetzt werden.]

## 5 Design

Wie im [Abschnitt 3. Anforderungen](#) beschrieben wurde, ist eine gute Benutzbarkeit ein wichtiger Aspekt für die zu entwickelnde Anwendung. Dabei wurde insbesondere das Design und die Einhaltung von Designrichtlinien im [Abschnitt 2. Handy und PC Unterschiede](#) als eine entscheidende Rolle für eine gute Benutzbarkeit hervorgehoben.

**Überblick:** Im ersten Unterabschnitt werden daher diese Designrichtlinien genauer betrachtet. Hierbei wird zunächst eine Richtlinie ausgewählt und ihr Inhalt genauer erläutert, um die für die Arbeit relevanten Regeln zu identifizieren. Anschließend wird das konkrete Design der App vorgestellt und einige der getroffenen Entscheidungen begründet.

**Ergebnisse:** Es wurde sich zwischen den Richtlinien Apple und Google für Apps entschieden. Viele der Regeln in den Designrichtlinien stimmen mit den Stärken des Handys, beziehungsweise den Ergebnissen aus [Abschnitt 2. Handy und PC Unterschiede](#), überein. Beispielsweise wird empfohlen, dass die App und ihr Design folgende Merkmale aufweisen sollten: „sofort einsatzbereit sein“, „Aufgaben vereinfachen“, „Informationen reduzieren“, „Informationen indirekt vermitteln“, „konsistent innerhalb der App und Plattform sein“ und „intuitiv sein“. Abschließend wurden die Richtlinien und ihre Regeln sowie die Merkmale auf die verschiedenen Komponenten der App angewendet und erläutert, einschließlich der Kalenderseite, der Erinnerungsseite, der Einstellungsseite, der Applelisten und der Systemtastatur.

### 5.1 Konventionen

#### 5.1.1 Auswahl

Es stehen mehrere verschiedene Richtlinien für Apps zur Verfügung, z.B. eine für iOS von Apple und eine für Android von Google[[7](#), [49](#)]. Es wird jedoch eine einzige Richtlinie gewählt, da unterschiedliche Richtlinien unterschiedliche Regeln enthalten und sich gegebenenfalls sogar widersprechen können. Google bietet beispielsweise eigene Seiten und Regeln für „Floating Action Buttons“[[47](#)], während diese in den Apple Richtlinien nie erwähnt werden und daher wahrscheinlich auch nicht erwünscht sind. Da aufgrund persönlicher Präferenz die Apps von Apple als noch intuitiver und leichter zu bedienen empfunden werden und dies für die Anforderungen der App von Vorteil ist, wurde sich für die Richtlinien von Apple entschieden.

Die Richtlinien von Apple bestehen aus fünf Abschnitten, die unter anderem die Gestaltung einzelner Komponenten wie Textfelder und Knöpfe sowie allgemeine Regeln und Muster für die Erstellung von Apps behandeln. Insgesamt umfasst die Richtlinie etwa 148 Regelungen. Davon schienen anfangs etwas mehr als die Hälfte als nützlich für die Arbeit. Bei genauerer Betrachtung stellten sich 34 dieser Einträge als tatsächlich passend heraus, da sie Funktionen und Anforderungen behandeln, die in die Arbeit einfließen sollen. Die meisten restlichen Einträge behandeln Funktionen, die nicht in die Anwendung integriert werden sollen, wie zum Beispiel die Bedienung per Hardwaretastatur oder NFC-Funktionalität[[10](#), [15](#)]. Einige andere Einträge, die

eigentlich nützliche Funktionen für diese Arbeit beschreiben, wie zum Beispiel ein Nachtmodus oder Siri-Unterstützung, wurden aufgrund des begrenzten Zeitrahmens vorerst übersprungen[6, 17].

### 5.1.2 Auswertung & Erkenntnisse

In diesem Abschnitt werden Erkenntnisse und Muster beschrieben, die während der Ausarbeitung der Richtlinien ersichtlich wurden.

Diese Erkenntnisse haben einen allgemeineren Charakter als die Richtlinien und können dementsprechend auf die gesamte App und auch auf Bereiche angewendet werden, für die keine Regeln vorhanden sind.

Dabei ist zu beachten, dass einige der Erkenntnisse eng miteinander verknüpft sind und deshalb einen fließenden Übergang zwischen ihnen herrscht.

Das Design und die App sollten gemäß den Richtlinien folgende Merkmale aufweisen:

1. Sofort einsatzbereit sein.

Denn es wird unter anderem empfohlen:

- den Inhalt der App möglichst früh zu zeigen[13]
- die Notwendigkeit für eine Anmeldung möglichst lang zu verzögern[14]
- sowie initiale und unterbrechende Informationsanfragen, wie zum Beispiel nach Bewertungen oder Zugriffsrechten, nur wenn wirklich nötig anzufordern[11].

2. Aufgaben vereinfachen.

Denn es wird unter anderem empfohlen:

- Texte so klar wie möglich zu verfassen[19]
- wo möglich, Informationen automatisch vom System zu entnehmen, anstatt den Nutzer danach zu fragen[8, 7]
- sowie Alternativen zur Texteingabe, wie zum Beispiel „drag & drop“ oder eine Liste von Optionen, anzubieten[8].

3. Informationen reduzieren.

Denn es wird unter anderem empfohlen:

- Texte so kurz wie möglich zu verfassen[19]
- die Anzahl an Steuerelementen zu begrenzen[7]
- simplifizierte Designs für Icons zu verwenden[9]
- sowie wichtigen Informationen genügend Platz zu geben, indem zum Beispiel eher unwichtige Details ausgelassen werden[12].

4. Informationen indirekt vermitteln.

Denn es wird unter anderem empfohlen:

## 5. Design

- Farbe zu nutzen, um Bedeutung zu vermitteln (beispielsweise indem grün für bestätigende Aktionen genutzt wird) [5].
- Icons zu nutzen, da diese bei richtiger Verwendung ein Konzept sofort verständlich machen können[9]
- Textgröße, Schriftart und Farbe zu nutzen, um Wichtigkeit zu vermitteln[18].
- Platzierung von Objekten zu nutzen, um Wichtigkeit zu vermitteln[12]
- sowie zur Situation passende Sprache zu verwenden[19].

### 5. Konsistent innerhalb der App und Plattform sein.

Denn es wird unter anderem empfohlen:

- die Plattformfarben zu nutzen und bei einmal definierten Farben konsistent zu bleiben[5]
- die Plattformdefinierten Textstile zu nutzen[18]
- die Plattformdefinierten Gesten zu nutzen[3]
- sowie auf allen Seiten ein konsistentes Icondesign zu verwenden[9].

### 6. Intuitiv sein.

- Denn alle vorherigen Punkte haben dies als Ziel.

Interessant hierbei ist, dass sich diese Erkenntnisse mit den Ergebnissen aus **Ab schnitt 2. Handy und PC Unterschiede** überschneiden. So passt beispielsweise die Erkenntnis „Sofort einsatzbereit sein“ zur Stärke des Handys, gut für kurzweilige Aufgaben zu sein. Die Erkenntnisse „Aufgaben vereinfachen“, „Informationen indirekt vermitteln“ und „Intuitiv sein“ passen hingegen zum Ergebnis, dass einfache und intuitive Aufgaben gut zum Handy passen. „Informationen reduzieren“ passt hingegen zum Ergebnis, dass auf Handys Aufgaben besser funktionieren, die wenig Informationen darstellen oder benötigen.

Weiter gibt es in den Richtlinien auch einzelne Regeln, bei denen es zu einer ähnlichen Überschneidung kommt. So rät Apple beispielsweise sowohl von Texteingaben als auch von einer hohen Anzahl an Einstellungen ab[8, 16]. Dies überschneidet sich mit den Ergebnissen, dass Handys keine schnelle, präzise und vielfältige Eingabe erfordern sollten und ohne viele Optionen und Konfigurationen auskommen sollten.

## 5.2 App Design

In diesem Abschnitt werden einige konkrete Designentscheidungen vorgestellt. Generell wurde sich dabei an die Erkenntnisse aus **Unterunterabschnitt 5.1.2. Auswertung & Erkenntnisse** gehalten.

### 5.2.1 Appleisten

Bei den Appleisten wurde generell versucht, die Funktionen der Seite und der Tasten so klar wie möglich darzustellen. Dazu wurden für die Seiten und Tasten möglichst kurze und deutliche Wörter und Icons verwendet. Außerdem wurde versucht, die

Leisten möglichst konsistent mit bereits bestehenden Konventionen zu entwerfen. So befindet sich der Zurückknopf, wie man es erwarten würden, immer am oberen linken Rand. Zudem wurde vorerst ein „Tab-Design“ gewählt (siehe Abbildung 1), da dieses auch in vielen Apps vorkommt.

Jedoch wurde später beschlossen, von diesem ursprünglichen Design abzuweichen. Die Funktion, eine Erinnerung hinzuzufügen, wird als eine der häufig genutzten Hauptfunktionen der Anwendung angesehen. Allerdings ist sie in diesem Design relativ schwer zu erreichen (Plus-Icon oben rechts). Es wäre besser, diese Aktion unten oder in der Mitte anzubieten, da dieser Bereich für die meisten Nutzer besser erreichbar ist[7]. Zusätzlich dazu ist der Tab zu der Einstellungsseite relativ groß und immer sichtbar, obwohl davon ausgegangen wird, dass dieser Seite nur einmal oder sehr selten benötigt wird.

Dementsprechend wurde das Design abgeändert (siehe Abbildung 2). Die Hauptaktion eine Erinnerung zu erstellen ist nun verständlicher erklärt und besser erreichbar. Die Aktion zur Einstellungsseite ist nun kleiner und wurde aus dem meistgenutzten Bereich (unten und mittig) entfernt. Auch die Navigationstaste zur Kalenderseite wurde entfernt, da diese als Hauptseite gilt und dementsprechend immer über den Zurückknopf erreichbar ist.

### 5.2.2 Kalenderseite

Wie bereits im [Unterabschnitt 4.6. Auswahl der Pakete](#) erwähnt, wird für die Darstellung des Kalenders ein bestimmtes Paket genutzt. Daher ließ sich das Design dieser Seite nur begrenzt verändern. Allerdings verhält sich das Design des genutzten Pakets konsistent zu anderen Kalender-Apps und wird daher als passend für diese App bewertet.

Um die sofortige Nutzung der App zu ermöglichen, ist weder eine initiale Konfiguration noch ein Login erforderlich, um auf den Kalender oder andere Seiten zuzugreifen. Weiter wird während des Ladens von Daten aus der Datenbank eine Ladeanimation angezeigt, um so dem Benutzer visuelles Feedback zu geben und eine intuitivere Benutzererfahrung zu schaffen.

Für eine Beispieldarstellung siehe Abbildung 3 und 4.

### 5.2.3 Erinnerungenseite

Die Erinnerungsseite zeichnet sich hauptsächlich durch ihre Liste von Erinnerungen aus. Diese Liste nutzt Animationen, um einen fließenden Übergang beim Erstellen und Abschließen von Erinnerungen zu gewährleisten. Ohne diese Animationen könnten abrupte Änderungen in der Darstellung auftreten und möglicherweise zu Verwirrung führen.

Die Erinnerungseinträge bestehen aus einem Titel und enthalten optional eine Beschreibung sowie Dateien. Denn es wird angenommen, dass für einfache Erinnerungen wenige Wörter oder ein kurzer Satz ausreichen. Für komplexere Erinnerungen kann stattdessen die Beschreibung genutzt werden. In Fällen, in denen sich die Erinnerung hingegen leichter über eine Datei erklären lässt, soll auch dafür eine Möglichkeit bestehen.

Das dazu erstellte Design wurde versucht so kompakt wie möglich zu gestalten, um

## 5. Design

so auf dem relativ kleinen Bildschirm eines Handys alle notwendigen Informationen angemessen darzustellen. Des Weiteren wurde das Design so gestaltet, dass es ansprechend aussieht, unabhängig davon, ob die optionalen Eingaben vorhanden sind oder nicht.

Das Abschließen einer Erinnerung wird durch einen Knopf neben der entsprechenden Erinnerung ermöglicht. Da der Knopf immer sichtbar ist, lässt er sich deshalb mit minimalem Aufwand betätigen, ohne dass weitere Aktionen und Zwischenschritte erforderlich sind. Der Knopf wurde bewusst links von der Erinnerung platziert, da an dieser Position davon ausgegangen wird, dass Nutzer<sup>13</sup> ihn seltener versehentlich betätigen.

Außerdem wird, falls bei der Erstellung einer neuen Erinnerung kein Titel gewählt wurde, dieser automatisch erstellt, um dem Nutzer Zeit und Aufwand zu ersparen. Für eine Beispieldarstellung siehe Abbildung 5.

### 5.2.4 Einstellungsseite

Die Hauptmerkmale dieser Seite sind die Textfelder zum Einstellen der Login-Optionen. Dabei wurde versucht, sie so zu gestalten, dass sie möglichst konsistent mit den iOS-Plattformtextfeldern sind.

Um die Benutzerfreundlichkeit zu erhöhen, werden Abhängigkeiten zwischen den Textfeldern durch Farben und Fehlermeldungen und Hinweise zur Eingabe indirekt angezeigt. Zudem wird eine Ladeanimation eingeblendet, um den Fortschritt bei der Überprüfung der Eingaben zu signalisieren. Um Nutzern Arbeit abzunehmen, werden bei der Änderung eines Textfelds die davon abhängigen Textfelder automatisch überprüft. Andernfalls müsste der Benutzer beispielsweise nach der Änderung des Tokens auch das Repository und den Pfad zur Konfigurationsdatei neu festlegen.

Die Hauptaktion dieser Seite ist die „autoSetup-Aktion“ und ersetzt dementsprechend die Hauptaktion einer neuen Erinnerung zu erstellen. Diese Aktion erstellt ein neues Repository mit der benötigten Konfigurationsdatei sowie Vorschaueneinträgen und stellt die entsprechenden Optionen in der App automatisch ein. Um die Funktion dieser Aktion zu erläutern und da davon ausgegangen wird, dass diese Aktion durchaus aus Versehen betätigt werden kann, wird vorher mithilfe eines „Alerts“ die Aktion erklärt und nach Bestätigung gefragt.

Außerdem werden am Ende der Seite einige für den Nutzer möglicherweise interessante Informationen angezeigt.

Während Aktionen ausgeführt werden, die von der Datenbank abhängen, wie beispielsweise dem Erstellen, Beenden und Bearbeiten von Erinnerungen, werden kleine Ladeanimationen neben der ausgeführten Aktion angezeigt. Dadurch soll der Nutzer auch ein Feedback über den Status der Datenbank erhalten.

Für eine Beispieldarstellung siehe Abbildung 6 und 7.

### 5.2.5 Tastatur

Wie zuvor erwähnt, ist geplant, alternative Eingabemöglichkeiten für die Erstellung von Erinnerungen anzubieten. Es wurde entschieden, Fotos, Videos, Sprachnachrich-

---

<sup>13</sup>Diese Annahme gilt nur für Nutzer, die das Handy mit der rechten Hand bedienen.

ten oder andere auf dem Handy bereits vorhandene Dateien als Optionen anzubieten, da angenommen wird, dass diese Dateitypen nützlich für die Erinnerungen sein könnten und einfach über das Handy zugänglich sind.

Anstatt diese Alternativen statisch auf einer Seite anzuzeigen, sollten sie nur dann angezeigt werden, wenn sie benötigt werden. Dadurch werden die dargestellten Informationen reduziert und eine bessere Übersichtlichkeit gewährleistet. Gleichzeitig wird die Bedeutung der Aktionen indirekt vermittelt, da dadurch gezeigt wird, wann und wofür die Aktionen genutzt werden können. Der dafür passende Platz ist die Systemtastatur.

Um konsistent mit dem Betriebssystem zu bleiben, soll die Standardtastatur beibehalten werden. Anstatt eine eigene Tastatur zu entwerfen, werden die Alternativen stattdessen in die bereits existierende Standardtastatur integriert.

Die Aktionen werden wiederum als Icons dargestellt, um mit möglichst wenig Platz ihre Bedeutung zu vermitteln.

Für eine Beispieldarstellung siehe Abbildung 8.

## 6 Implementierung

Da nun die Anforderungen und das Design der Anwendung bekannt sind sowie die Technologie, mit der sie erstellt werden soll, kann sich der vorliegende Abschnitt mit der Implementierung befassen. Dabei wird nicht die gesamte Implementierung dargestellt, da dies den Umfang dieser Arbeit sprengen würde. Stattdessen werden die wichtigsten Komponenten der Anwendung sowie die während der Implementierung getroffenen Entscheidungen und Erkenntnisse behandelt.

### Überblick:

Zunächst werden einige Eigenheiten und Besonderheiten während der Implementation des Parsers und der Verbindung zur Datenbank genannt. Anschließend wird erläutert, wie die Qualität der Anwendung und des Programmcodes sichergestellt wurde. Zum Schluss werden einige gezielt genutzte Entwurfsmuster genannt und ihre Verwendung begründet.

### Erkenntnisse:

Für den Parser wurde generell versucht, sich möglichst immer an die Vorgaben der Dokumentation zu halten, damit Nutzer ihre bereits bestehenden Kalenderdateien nicht anpassen müssen. Dabei gab es jedoch zwei Fälle, in denen von der Dokumentation abgewichen wurde. Zum einen bei der Zeitangabe, da angenommen wurde, dass die Anwendung von dieser Änderung profitieren würde. So bietet die Anwendung zusätzlich zur Interpretation eines Startzeitpunkts auch die Funktion, den Endzeitpunkt zu verstehen und anzuzeigen. Zum anderen wurden Syntax und Fehleingaben der Kalenderdatei anders interpretiert, da sie beim When-Programm inkonsistent zu sein scheinen. Zuletzt wurde sich aufgrund der begrenzten Zeit vorerst darauf beschränkt, nur die wichtigsten Operationen und Variablen des CLI-Terminkalenders zu unterstützen.

Bei der Verbindung zur Datenbank mussten besondere Vorkehrungen für die Authentifizierung und das Herunterladen von Dateien getroffen werden, da die Endpunkte der API diese Funktionen nicht zufriedenstellend erfüllen. Zudem wurde bei der Darstellung der Erinnerungen in der Datenbank versucht, ein Format zu wählen, das sowohl für die WebView als auch das CLI geeignet ist.

Um die Stabilität der Anwendung sicherzustellen, wurden automatische Rückfalltests für den Parser und die Datenbankverbindung durchgeführt, während die grafische Oberfläche manuell getestet wurde. Zur Verbesserung der Lesbarkeit des Quellcodes wurden Modularisierung, Zugriffsmodifikatoren und eine eigene Kommentarstruktur angewendet.

Für die Implementierung der Datenbankverbindung und des Parsers wurden die Singleton- und Strategie-Patterns sowie eine ablagebasierte Struktur verwendet, um den Programmcode anpassungsfähiger und übersichtlicher zu gestalten. Darüber hinaus wurde das Proxy-Pattern eingesetzt, um geladene Dateien zwischenzuspeichern und somit das ständige Neuladen und Neuberechnen durch Datenbank und Parser zu verhindern.

## 6.1 Parser

### 6.1.1 Modell

Um einen Parser für den WhenKalendar zu erstellen, musste dieser zunächst verstanden werden. Dafür wurde die Dokumentation ausführlich [gelesen/studiert] und intensiv manuell getestet.

Dabei wurde festgestellt, dass ein Termin in When aus drei Teilen besteht: dem Datum, einer optionalen Zeitangabe und einer Beschreibung. Die Herausforderung beim Parsen der Termine liegt insbesondere im Datum. Während die Beschreibung sowie die Zeitangabe fast eins zu eins übernommen werden können, kann das Datum in variabler Darstellung vorliegen. So würde zum Beispiel „m=2“ bedeuten, dass der Termin jeden Tag im Februar stattfindet. Darüber hinaus können diese variablen Datumsangaben miteinander über Operationen kombiniert werden. Der Termin „m=2&(d=1 | d=25)“ würde zum Beispiel immer am ersten und am 25. Februar stattfinden.

Um diese Daten zu parsen, wird vorerst angenommen, dass Daten immer in Klammern eingeschlossen sind. Dadurch wird der Aufwand für die Implementierung des Parsers reduziert, da die verschiedenen Bindungsstärken der Operationen nicht berücksichtigt werden müssen. Anschließend wird das Datum rekursiv durchgegangen, bis ein einzelner Ausdruck ohne Operation gefunden wird. Aus diesem Ausdruck wird ein modelliertes WhenDate-Objekt erstellt. Dieses Objekt kann mit anderen Objekten seiner Art über die When-Operationen kombiniert werden. Dadurch kann die Rekursion schlussendlich ein Ergebnis liefern.

Nun verfügt der Parser über ein WhenDate-Objekt. Dieses kann jedoch immer noch variabel sein. Um Termine im Kalender anzuzeigen, benötigt es jedoch konkrete Daten. Dazu wird von einem gewählten Startdatum bis zu einem Enddatum überprüft, welche konkreten Daten das variable WhenDate-Objekt annehmen kann. Diese Funktion wird auch nützlich sein, um Termine dynamisch im Kalender nachzuladen. So werden zunächst nur wenige Monate im Voraus und in der Vergangenheit berechnet, um so eine möglichst schnelle Ladezeit zu gewährleisten. Wenn der Benutzer Termine über die initial geladenen Einträge hinaus ansehen möchte, werden diese automatisch nachgeladen und angezeigt.

Diese Funktion ist auch nützlich, um Termine dynamisch im Kalender nachzuladen und wird dementsprechend in der Kalenderimplementierung genutzt. So werden zunächst nur wenige Monate im Voraus und in der Vergangenheit berechnet, um eine möglichst schnelle Ladezeit zu gewährleisten. Wenn der Benutzer Termine über die initial geladenen Einträge hinaus ansehen möchte, werden diese automatisch nachgeladen und angezeigt.

Für eine modellierte Beispieldarstellung siehe Abbildung 9.

### 6.1.2 Zeitangabe

Dadurch, dass die App eine grafische Oberfläche hat, besteht die Möglichkeit, die Zeitspanne von Terminen anzuzeigen. Obwohl diese Option nicht von When unter-

## 6. Implementierung

stützt wird - bei Terminen wie „..., 17:00-18:00 ...“ wird immer nur die erste Zeitangabe berücksichtigt -, wird diese Funktion in die App integriert, da angenommen wird, dass sie eine nützliche grafische Information darstellt.

Für eine Beispieldarstellung siehe Abbildung 10.

### 6.1.3 Variablen & Operationen

When bietet eine Vielzahl von Variablen und Operationen an, um daraus Termine zu erstellen. Dazu gehören klassischen Vergleichsoperatoren wie „=,!=,>,<,>=,<=” sowie Logikoperatoren wie „|,&,!“. Aufgrund der begrenzten Zeit und der Vielzahl an Optionen wurde sich zunächst auf die als am wichtigsten und elementarsten eingeschätzten Variablen und Operationen konzentriert. Die Wahl viel dabei auf die Operationen „|,&“ und die Variablen „d,m,w,y“ für die Angabe eines Tages, eines Monats, eines Wochentages und eines Jahres.

### 6.1.4 Eingabe- & Formatfehler

Während des Testens von When wurden einige Inkonsistenzen festgestellt. Wenn ungültige Daten wie beispielsweise „2000 1 0“ als Konstanten eingegeben werden, wird ein Fehler angezeigt. Jedoch tritt dieser Fehler nicht auf, wenn das gleiche Datum über Variablen und Operationen eingegeben wird, z.B. „y=2000 & m=1 & d=0“.

Vorerst wurde angenommen, dass bei der Verwendung von Variablen keine Fehler auftreten, da auch bei ungültigen Eingaben ein gültiger Termin erzeugt werden kann. Zum Beispiel ist die Eingabe von „d=0“ ungültig, aber der folgende Ausdruck führt dennoch zu einem gültigen Termin: „d=0 | y=2000“. Jedoch hat sich herausgestellt, dass dies nicht immer der Fall ist, da Eingaben wie „d=-1“ oder „m=Marchz“ trotz Verwendung von Variablen zu Fehlermeldungen führen. Dieses Verhalten ist inkonsistent und scheint eher willkürlich zu sein. Es wäre besser, wenn Eingaben immer das gleiche Ergebnis liefern würden. Entweder sollten also alle genannten Beispiele zu Fehlern führen oder keines. Aus diesem Grund wurde sich im Sinne der Konsistenz leicht von der ursprünglichen Vorlage abgewendet. Es wurde entschieden, keines der oben genannten Daten als Fehler zu betrachten, da Variable Daten auch bei ungültigen Eingaben immer noch einen gültigen Termin erzeugen können.

## 6.2 Verbindung zur Datenbank

Wie bereits erwähnt, wurde für die Datenbankverbindung eine vorhandene API verwendet. Im Folgenden werden Schwierigkeiten und Besonderheiten bei der Verwendung dieser aufgeführt.

### 6.2.1 Authentikation & Berechtigungen

Die Authentizierung über einen Benutzernamen und ein Passwort wird von der Api nicht unterstützt[32]. Deshalb wird die Authentifizierung mit Tokens benutzt. Dies stellte aber kein Problem dar, da die Verwendung von Tokens ohnehin bevorzugt wird. Denn dadurch kann der Nutzer sicher sein, dass die Anwendung mit den gegebenen Zugriffsrechten nichts Schädliches ausführen kann.

Zwei der verwendeten Funktionen der API erfordern besondere Zugriffsrechte. Das Erstellen eines Repositorys erfordert die „repo“ Berechtigung[34], während das Erstellen und Aktualisieren von Dateien die „workflow“ Berechtigung erfordert[35]. Um den Benutzer über diese Informationen zu informieren, muss das Token auf diese Berechtigungen getestet werden. Jedoch unterstützt die API diese Funktion nicht mehr[38]. Stattdessen bietet GitHub jedoch die Möglichkeit, sich die Berechtigungen über die Antwort einer URL anzeigen zu lassen[39]. Um dies in der Anwendung zu nutzen, wird die URL mit curl aufgerufen und die Antwort nach den passenden Berechtigungen geparsst.

In der Dokumentation konnte zwar nichts dazu gefunden werden, jedoch hat sich beim Testen herausgestellt, dass diese Methode, um die Berechtigungen zu erlangen, nur von den „classic Tokens“ unterstützt wird. Für die neueren „beta Tokens“ sendet der Server keine Antwort zurück. Daher werden derzeit „beta Tokens“ in der App nicht unterstützt.

### 6.2.2 Dateien herunterladen

Die API bietet einen Endpunkt an, mit dem Dateien direkt aus einem Repository heruntergeladen werden können[36]. Allerdings fügt dieser Endpunkt in Flutter einige nicht konventionelle Zeichen hinzu, was zu ungültigen heruntergeladenen Dateien führt. Durch Tests wurde festgestellt, dass es sich dabei um Leerzeichen und ein \n-Zeichen handelt. Wenn diese entfernt werden, kann die Datei normal verwendet werden.

Beim weiteren Testen wurde festgestellt, dass dieser Endpunkt nur Dateien bis zu einer Größe von maximal 1 MB unterstützt. Diese Beschränkung ist aus Erfahrung zu klein, wenn Fotos, Videos und Audio heruntergeladen werden sollen. Dementsprechend wird der Endpunkt nicht mehr direkt genutzt. Stattdessen werden die Dateien über den GitHub-Link heruntergeladen. Dadurch können Dateien mit einer Größe von bis zu 100 MB heruntergeladen werden.

### 6.2.3 GitHub Issues

Wie zuvor erwähnt, sollen Erinnerungen über GitHub Issues ermöglicht werden. Dadurch ist es möglich, die Listen über die App, das CLI oder die Webview anzuzeigen und zu bearbeiten. Leider ist es nicht möglich, über die API Dateien zu Issues hinzuzufügen[76]. Um herauszufinden, wie Dateien dennoch zu Issues hinzugefügt werden können, wurde versucht herauszufinden wie Daten über die Webansicht zu Issues hinzugefügt werden. Es stellte sich heraus, dass die Dateien nicht direkt in den Issues gespeichert werden, sondern lediglich durch einen Link referenziert werden. Mit dieser Information konnte eine passende Funktion erstellt werden: Zunächst wird die GitHub API genutzt, um ein Issue zu erstellen. Anschließend wird die Datei, die dem Issue hinzugefügt werden soll, auf das entsprechende Repository hochgeladen. Schließlich wird ein Link zur Datei dem Issue hinzugefügt, wodurch die Datei im Webview als solche angezeigt wird.

Die GitHub-Webansicht unterstützt Dateien eingebettet darzustellen, anstatt sie nur als Link anzuzeigen. Beispielsweise wird bei einer PNG-Datei direkt das Bild angezeigt und bei einer MP4-Datei wird das entsprechende Video wiedergegeben. Dies

## 6. Implementierung

hat den Vorteil, dass es dem Nutzer Aufwand spart, da sie die Datei direkt in der Websicht betrachten können, ohne sie zuvor herunterladen zu müssen. Daher sollen auch die mit der App erstellten Issues von dieser Funktion profitieren. Um herauszufinden, wie die WebView Dateien einbetten kann, wurden zahlreiche Tests durchgeführt und die Dokumentation zurate gezogen[33]. Dabei stellte sich heraus, dass einige der in der Dokumentation erwähnten Funktionen für Issues nicht funktionieren, wie zum Beispiel „Relative Link“, die in Markdown-Dateien<sup>14</sup> verwendet werden können, aber nicht in Issues. Bei den Tests konnte erfolgreich die Einbindung von Bildern in die WebView reproduziert werden, jedoch war es nicht möglich, Videos einzubetten. Es scheint, dass Videos nur dann in die WebView eingebettet werden können, wenn sie über eine interne GitHub-URL der Form „<https://user-images.githubusercontent.com/...>“ verfügbar sind. Diese URLs können jedoch anscheinend nur generiert werden, wenn die Dateien über die WebView hinzugefügt wurden. Daher ist es derzeit nicht möglich, eingebettete Videos in der WebView mit der App zu erstellen.

Um die Issues auch im CLI ansprechend darzustellen, wurde versucht, bei deren Erstellung ein geeignetes Format zu verwenden.

Ein Issue besteht immer aus einem Titel und einer Beschreibung. Die Beschreibung wird so formatiert, dass die erste Zeile die Beschreibung der Erinnerung enthält und die folgenden Zeilen die Dateien.

Die Dateien müssen für die WebView im Format „[ ]{url}“ gekennzeichnet werden. Innerhalb der „[ ]“ wird der Dateipfad angegeben, um eine weitere Referenz auf die Datei für die CLI bereitzustellen. Um eine klare Struktur im Repository zu gewährleisten, werden die Dateien der Issues dabei in Ordnern der Form „issue{issueNummer}“ gespeichert.

Für eine Beispieldarstellung aller drei Ansichten siehe Abbildung 22, 23, 21.

## 6.3 Qualitätssicherung

### 6.3.1 Tests

Um sicherzustellen, dass die Anwendung stabil läuft und keine Defekte enthält, werden mehrere Tests durchgeführt. Dabei wird der Parsers und die Verbindung zur Datenbank am ausgiebigsten getestet, da sie die Grundfunktionen der Anwendung darstellen und somit als kritische Bereiche angesehen werden. Da diese beiden Funktionen viele Komponenten enthalten, die oft geändert werden, wäre das manuelle Testen zu zeitaufwendig und müsste nach jeder kleinen Änderung erneut durchgeführt werden. Deshalb wurde stattdessen beschlossen, automatisierte Rückfalltests durchzuführen. Obwohl die Erstellung von automatisierten Rückfalltests anfangs mehr Aufwand erfordert, lohnt es sich, sobald die Komponenten häufiger getestet werden müssen. Außerdem bieten die Tests eine Art von Dokumentation, da sie zeigen, wie die Komponenten funktionieren sollten. Deshalb werden die Rückfalltests auch als nützlich für die Weiterentwicklung der Anwendung eingeschätzt.

Um sicherzustellen, dass die Tests zeigen, ob ein Defekt von der getesteten Kom-

---

<sup>14</sup>Eine reine Textdatei, die auf GitHub Syntaxunterstützung bietet, um den Text zu formatieren.

ponente oder von einer von ihr aufgerufenen Funktion verursacht wird, werden die Tests nach dem Schema der Bottom-Up-Integrationstests geschrieben. Das bedeutet, dass zuerst die Module und Komponenten getestet werden, für die alles, was sie aufrufen, bereits getestet wurde.

Damit die Tests die Komponenten möglichst vollständig auf Defekte prüft, wurde bei der Erstellung der Tests versucht durch die Eingaben das Abdeckungskriterium der Bedingungsüberdeckung zu erreichen. Außerdem wurden stets Randfälle wie leere Eingaben oder völlig unsinnige Eingaben getestet, da dies aus Erfahrung oft zu Defekten führt.

Die Ausführung der Tests mit Code-Coverage-Analyse ergab, dass im Parser 70% und bei der Datenbankverbindung 90% aller Codezeilen getestet werden.

Die Funktionen der grafischen Oberfläche wurden hingegen manuell getestet, da sich diese besser für manuelle Tests eignen und somit zeiteffektiver sind als automatisierte Tests.

### 6.3.2 Verständlichkeit

Um den Quelltext möglichst verständlich zu gestalten, wurde generell versucht, Komponenten möglichst modular zu gestalten und so aufzuteilen, dass jede Komponente immer genau eine einzige einzigartige Aufgabe hat. Des Weiteren wurden Zugriffsmodifikatoren für Methoden gesetzt. Dadurch soll deutlich werden, welche Methoden der Komponente ausschließlich für interne Funktionen genutzt werden und welche dem Klienten zur Verfügung stehen. Außerdem wurden für wichtige Funktionen, wie beispielsweise alle Funktionen aus der Datenbankverbindung und dem Parser, Kommentare verfasst. Kommentare haben zwar den Nachteil, dass sie bei Änderungen des betreffenden Codes aktualisiert werden müssen, jedoch können sie die Verständlichkeit verbessern. Ein kurzer Satz kann beispielsweise die Funktion einer ansonsten großen und schwer lesbaren Methode erläutern. Um genau solche nützliche Kommentare zu schreiben, wurde ein selbst vorgegebenes Format verwendet, das sich während der Programmierung als hilfreich erwiesen hat. Das Format besteht aus den Feldern „def“, „purpose“, „assert“, „expect“, „return“ und „example“. „Def“ beschreibt die Funktion, „purpose“ gibt an, wofür die Funktion benötigt wird, „assert“ zeigt was die Funktion voraussetzt, „expect“ gibt an, was die Funktion erwartet, aber nicht unbedingt voraussetzt und „example“ enthält ein Beispiel. Die Felder werden nicht immer alle benutzt, sondern nur dann, wenn es als nützlich erscheint.

## 6.4 Entwurfsmuster

Während der Implementation wurden sich bewusst für die Verwendung einiger Entwurfsmuster entschieden, da sie in den jeweiligen Situationen nützlich erscheinen. Folgend werden die Entwurfsmuster aufgezählt und dessen Verwendungsgrund begründet.

## 6. Implementierung

### 6.4.1 Fassade

Durch das Umschließen der API mit einem Fassadenobjekt wird die Schnittstelle für den Nutzer der Datenbankverbindung vereinfacht und irrelevante Informationen werden verborgen. Dadurch konnten beispielsweise die erforderlichen Zwischenschritte zum Herunterladen einer Datei von GitHub verborgen werden.

Für eine modellierte Beispieldarstellung siehe Abbildung 11.

### 6.4.2 Singelton

Während der Implementierung wurde versucht, nicht auf das Singleton-Pattern zurückzugreifen. Obwohl es die Programmierung erleichtert, indem es den Zugriff auf ein Objekt von überall aus ermöglicht, hat es auch Nachteile. Durch Der globale Zugriff auf das Objekt erschwert die Nachvollziehbarkeit, welche Komponenten Zugriff auf das Objekt benötigen, und führt so unter anderem zu einer Verkomplizierung der Fehlerbehebung. Trotzdem wurde für die Datenbank das Singleton-Pattern verwendet, da die Datenbank sonst zwischen vielen Komponenten übergeben werden müsste und die Struktur des Programmcodes darunter leiden würde.

Es folgt die konkrete Situation welche während der Implementierung aufgetreten ist, als Beispiel. Siehe Abbildung 12, 13, 14 für die grafische Darstellung des Beispiels. Im Grunde benötigen nur drei Seiten Zugriff auf die Datenbank. Allerdings erstellt die Navigation in Flutter ein Seitenobjekt und benötigt daher ebenfalls Zugriff auf die Datenbank. Es gibt zwei Möglichkeiten: Entweder man gibt die Datenbank bis zum Navigationsknopf weiter oder man definiert die Navigationsfunktion bereits auf der Seite und gibt sie bis zum Knopf weiter. Beide Optionen würden jedoch dazu führen, dass viele Zwischenklassen die Datenbank oder Funktion als Parameter übergeben müssten, obwohl sie diese gar nicht benötigen. Das Singleton-Entwurfsmuster löst dieses Problem.

### 6.4.3 Strategie

Das Strategie Pattern würde für den Parser verwendet werden, da es in Zukunft geplant ist, dass die App auch weitere CLI-Terminkalender unterstützt. Das Strategie Pattern ermöglicht, dass der When-Parser einfach gegen einen anderen Parser ausgetauscht werden kann. Für die Datenbank wurde ebenfalls das Strategie Pattern verwendet. Die Idee dahinter war nicht nur, die Datenbank gegen eine alternative auszutauschen - obwohl das auch möglich wäre, beispielsweise durch eine GitLab-API anstelle der verwendeten GitHub-API. Vielmehr ermöglicht das Strategie Pattern auch den einfachen temporären Austausch einer Mock-Datenbank<sup>15</sup>. Eine simulierte Datenbank kann sich nämlich als hilfreich beim Testen herausstellen, da sie unabhängig von einer API ist. Entsprechend wurde auch für diese Anwendung eine solche Mock-Datenbank implementiert.

Für eine modellierte Beispieldarstellung siehe Abbildung 15 und 16.

---

<sup>15</sup>Eine Datenbank, welche die Funktionen einer echten simuliert.

#### 6.4.4 Proxy

Es wird angenommen, dass die Kalender- und Konfigurationsdateien sowie die Issues von GitHub während der Benutzung der App selten extern geändert werden. Deshalb ist es ausreichend, die Daten einmalig in der App zu laden und sie nur bei Bedarf, d.h. bei einer gezielten Anfrage nach neuen Dateien, erneut vom Server abzurufen. Derzeit lädt die App jedoch bei jeder Navigation auf eine Seite die Daten erneut von der Datenbank.

Dementsprechend wird für den Parser sowie der Datenbank das Remote Proxy Pattern verwendet. Dabei umschließe ein Proxy-Objekt das eigentliche Datenbank- bzw. Parser-Objekt und speichert die Ergebnisse in einem Cache. Wenn dieselben Daten erneut angefragt werden, werden sie aus dem Cache zurückgegeben. Bei einer gezielten Anfrage auf neue Dateien wird hingegen der Cache geleert und neue Dateien von der Datenbank geladen. Für eine modellierte Beispieldarstellung siehe Abbildung 17.

#### 6.4.5 Datenflussnetz & Ablagebasiert

Zu Beginn wurde aus Einfachheit für den Parser eine Struktur ähnlich einem Datenflussnetz erstellt. Im weiteren Verlauf der Entwicklung stellte sich jedoch heraus, dass diese Struktur zu unflexibel ist. Denn bei Änderungen an einzelnen Methoden mussten auch die unmittelbar vorherigen und nachfolgenden Methoden angepasst werden. Darüber hinaus wurde der Parser durch die lange Verkettung von Methoden immer unübersichtlicher. Deshalb wurde die Struktur des Parsers zu einer ablagebasierten Struktur geändert. Dadurch wird der Parser einerseits übersichtlicher und andererseits flexibler und änderungsfreundlicher, da die einzelnen Funktionen nicht direkt miteinander kommunizieren. Die Abbildungen 18, 19, 20 zeigen die Schritte des WhenParsers bei der Eingabe einer Datei bis zum erlangten Kalendareintrag in der zuvor genutzten datenflussnetz ähnlichen Struktur und in der verbesserten ablagebasierten Struktur.

## 7 Evaluation

In diesem Abschnitt wird versucht festzustellen, ob die erstellte Anwendung auch das erfüllt was sich zuvor vorgenommen wurde.

**Überblick:** Dazu wird eine Anforderungsverifizierung mit den funktionalen und nicht funktionalen Anforderungen aus dem [Abschnitt 3. Anforderungen](#) durchgeführt.

### Erkenntnisse:

Im Hinblick auf die funktionalen Anforderungen konnten alle „Must-haves“ und „Should-haves“ erfolgreich umgesetzt werden. Lediglich der Parser und die Funktion, Dateien zu Erinnerungen hinzuzufügen, blieben unvollständig, da andere Funktionen aufgrund von Zeitmangel priorisiert wurden. Die nicht funktionalen Eigenschaften „Wartbarkeit“ und „Reichweite“ konnten aufgrund derselben Einschränkungen nicht vollständig implementiert werden. Jedoch konnten die übrigen nicht funktionalen Anforderungen überwiegend bis vollständig erfüllt werden.

### 7.1 Anforderungsverifizierung

In diesem Abschnitt werden noch einmal alle Anforderungen aus [Abschnitt 3. Anforderungen](#) aufgelistet, um sie einzeln mit dem Endprodukt vergleichen und so verifizieren zu können. Dabei werden die funktionalen Anforderungen darauf überprüft, ob sie vollständig oder teilweise umgesetzt wurden, oder lediglich in der Konzeptionsphase verblieben sind. Bei vielen der nicht funktionalen Anforderungen ist es hingegen schwieriger zu bewerten, ob sie wie gewünscht umgesetzt wurden. Daraus könnten bei diesen Anforderungen lediglich Vermutungen angestellt werden. Es wäre wahrscheinlich besser gewesen, einen Nutzertest durchzuführen, um aussagekräftigere Schlüsse ziehen zu können. Insbesondere hätten die Nützlichkeit und Benutzerfreundlichkeit besser eingeschätzt werden können. Aber auch die Wartbarkeit und Leistung hätten mithilfe einer Änderungsfreundlichkeitsanalyse und Benchmarks besser beurteilt werden können. Leider blieb dafür jedoch keine Zeit mehr übrig.

#### Funktionale Anforderungen:

1. **M Verbindung mit Backend:** Vollständig implementiert. Alle benötigten Datenbankfunktionen stehen zur Verfügung.
2. **M (+C) Übersetzer für CLI-Terminkalender:** Teilweise implementiert. Wie in [Unterabschnitt 6.1. Parser](#) erwähnt versteht der Übersetzer derzeit nicht alle Operanden und setzt vorerst Klammersetzung voraus. Des Weiteren werden drei Terminaloptionen<sup>16</sup>, die „when“ bietet und die Interpretation von Terminen verändert, noch nicht unterstützt.
3. **M (+C) Kalender Darstellung:** Vollständig implementiert. Alle benötigten Kalenderansichten stehen zur Verfügung.
4. **S Einträge erstellen, bearbeiten, löschen:** Teilweise implementiert. Erinnerungseinträge können erstellt, bearbeitet und abgeschlossen werden. Jedoch können

---

<sup>16</sup>Die besagten when Terminaloptionen sind: monday\_first, ampm, auto\_pm[21].

noch keine Dateien wie Bilder, Audios oder Videos hinzugefügt werden. Außerdem wurde sich gegen das Löschen entschieden, da die Abschließen-Funktion bereits eine ähnliche Funktionalität bietet und daher angenommen wird, dass die Darstellung der Löschaktion die App nur unübersichtlicher machen würde.

5. **S Einschränkungen:** Vollständig implementiert. Die Anwendung hat mehrere Einschränkungen, die darauf abzielen, die Nutzung auf die Stärke des Handys zu lenken, wie etwa die kurzweiligen und einfachen Aufgaben. So werden unter anderem bei der Erstellung neuer Erinnerungen die Länge des Titels und der Beschreibung sowie die Anzahl der Dateien begrenzt. Zudem ist auch die Anzahl der in der App darstellbaren Erinnerungen begrenzt.
6. **S Benachrichtigungen:** Vollständig implementiert. Erinnerungen zu Terminen werden automatisch erstellt.
7. **S Konfiguration auf dem Pc:** Vollständig implementiert. Für die Anwendung existiert eine Konfigurationsdatei welche über den PC angepasst werden kann. Jedoch könnte diese noch weiter verbessert werden, indem das Dateiformat auf das Textformat umgewandelt wird und gegebenenfalls weitere Optionen hinzugefügt werden.
8. **C Suchfunktion:** Nicht implementiert.
9. **C Weitere Kalender Abonnieren & Teilen:** Nicht implementiert. Zurzeit kann nur ein Kalender gleichzeitig Angezeigt werden. Jedoch sollte es möglich sein, dadurch dass GitHub als Datenbank verwendet wird, diese zum teilen von Terminkalendern zu nutzen.
10. **C Offline Funktionen:** Nicht implementiert.
11. **W Anleitung:** Teilweise implementiert. Wie zuvor geplant wurde keine Anleitung implementiert, da es stattdessen besser wäre eine Anwendung zu erstellen, die so intuitiv ist, dass keine Anleitung benötigt wird. Jedoch würde sich für das erste aufsetzen des Repositories mit Kalendar sowie Konfigurationsdatei eine Anleitung lohnen, da dies ein relatives komplexes und aufwändiges Unterfangen darstellt. Statt jedoch eine Anleitung zu entwerfen, wurde beschlossen, eine Funktion bereitzustellen, welche dem Benutzer diesen Aufwand abnimmt. Daher verfügt die App über eine Auto-Setup-Taste, die automatisch ein Repository, eine Konfigurationsdatei und Beispieleinträge erstellt.
12. **W Commit-History:** Nicht implementiert.

#### Nicht funktionale Anforderungen:

1. **M Stärken von PCs und Handys:**

Umgesetzt. Da die Anwendung nach den Stärken des PCs und Handys aus **Ab schnitt 2. Handy und PC Unterschiede** erstellt wurde. So sind für diese Anwendung die Hauptaufgaben des PCs das Erstellen von Terminen und das Vollen-

## *7. Evaluation*

den von Erinnerungen sowie die Konfiguration der Anwendung und des Repositories. Im Gegensatz dazu sind die primären Aufgaben des Handys das Anzeigen von Terminen, Benachrichtigungen und Erinnerungen sowie das Erstellen von Erinnerungen.

Im Folgenden werden die Stärken des PCs und Handys erneut aufgelistet, um diese so einzeln [im Hinblick] auf die erstellte Anwendung zu bewerten.

Die Aufgaben des PCs:

1.1 benötigen viel Leistung.

Nicht umgesetzt. Die Anwendung beinhaltet grundsätzlich keine Aufgaben oder Funktionen, die viel Leistung benötigen.

1.2 erfordern schnelle, präzise oder vielfältige Eingaben.

Umgesetzt. Die Erstellung von Terminen über das Terminal profitiert von schneller und präziser Eingabe. Außerdem benötigt der CLI-Terminkalender spezielle Symbole, und dementsprechend erfordert diese Aufgabe auch eine vielfältige Eingabe.

1.3 stellen viele Informationen dar oder benötigen viele Informationen.

Umgesetzt. Es wird angenommen, dass es beim Erstellen von Termineinträgen durchaus hilfreich sein kann, Informationen aus anderen Quellen zu beziehen. Eine solche Informationsquelle wird beispielsweise durch die Erinnerungen bereitgestellt, die von dieser Anwendung erstellt werden.

1.4 bieten viele Optionen und Konfigurationen an.

Umgesetzt. Es besteht die Möglichkeit, die Anwendung, das Repository und gegebenenfalls den CLI-Terminkalender zu konfigurieren.

1.5 sind langwierig oder benötigen viel Zeit.

Überwiegend umgesetzt. Der Zeitaufwand für das Erstellen und Organisieren von Terminen auf dem Terminal ist nutzerabhängig. Einige Nutzer benötigen möglicherweise viel Zeit, da sie ihre Termine sorgfältig organisieren und komplexe Syntax für neue Termine verwenden, während andere Nutzer dafür nur den minimalen Aufwand aufbringen. Es wird jedoch vermutet, dass die Erstellung und Organisation von Terminen auf dem Terminal eher eine zeitaufwändige Aufgabe ist.

Die Aufgaben des Handys:

1.1 sind ressourcenschonend und benötigen nicht viel Leistung.

Überwiegend umgesetzt. Weder das Einsehen von Terminen und Benachrichtigungen noch das Erstellen von Erinnerungen sind leistungsaufwändige Aufgaben. Weiteres siehe: 5. C Leistung.

1.2 erfordern keine schnelle, präzise oder vielfältige Eingaben.

Umgesetzt. Lediglich das Erstellen von Erinnerungen erfordert wiederholte Eingaben und für diese werden alternative Eingabemöglichkeiten angeboten, um den Aufwand dabei zu verringern.

### 1.3 stellen weder viele Informationen dar noch benötigen sie viele.

Umgesetzt. Das [Einsehen/Ansehen] von Terminen und Benachrichtigungen benötigt keine weiteren Informationen. Auch das Erstellen von Erinnerungen sollte hingegen nur wenig weiteren Informationen benötigen, wie zum Beispiel Bilder, Videos, Text, Sprachnachrichten oder andere Dateien auf dem Handy. Darüberhinaus wurde bei der Gestaltung der grafischen Oberfläche darauf geachtet, dass die Anwendung generell übersichtlich ist und keine überflüssigen Informationen enthält.

### 1.4 kommen ohne viel Optionen und Konfigurationen aus.

Umgesetzt. Für das Handy wurden bereits passende Voreinstellungen getroffen. Für jene Einstellungen, bei denen angenommen wurde, dass Nutzer sie gegebenenfalls selbstständig anpassen möchten, wurden die Option zum Konfigurieren auf den PC verlegt.

### 1.5 sind kurzweilig oder benötigen wenig Zeit.

Umgesetzt. Einerseits wird eingeschätzt, dass die Aufgaben des Einsehens von Terminen und Benachrichtigungen sowie das Erstellen von Erinnerungen relativ kurzweilig sind. Darüber hinaus wurde der Zeitaufwand für die Erstellung von Erinnerungen durch die Bereitstellung alternativer Eingabemethoden und die Begrenzung der Textlänge und maximalen Anzahl von Erinnerungen reduziert. Andererseits erfordert die Anwendung wenig bis keinen Einstiegsaufwand, da sie schnell geladen wird (siehe Punkt 5. C Performance) und keine initiale Konfiguration erforderlich ist, um die Anwendung anzusehen. Um [jedoch] die Funktionen der Anwendung zu nutzen, sind nur wenige Konfigurationsschritte in Form von drei Feldern erforderlich. Diese wurde durch die Funktion des „Automatischen Setups“ noch weiter vereinfacht.

### 1.6 sind lohnenswert Unterwegs zu lösen.

Umgesetzt. Die Funktionen der Anwendung (Anzeigen von Terminen und Benachrichtigungen sowie Erstellen von Erinnerungen) wurden im **Abschnitt 3. Anforderungen** ausgewählt, da unter anderem davon ausgegangen wurde, dass es sich lohnt, diese Funktionen auch unterwegs nutzen zu können.

### 1.7 sind einfach und intuitiv.

Überwiegend umgesetzt. Siehe: 2. M Benutzbarkeit.

## 2. M Benutzbarkeit:

Überwiegend umgesetzt. Es wird angenommen, dass die App einfach, intuitiv und effektiv zu nutzen ist, da sich beim Design an die Regeln und Erkenntnisse der von Apple gegebenen Richtlinien gehalten wurde.

Jedoch gibt es beim Design noch einige wenige Entscheidungen, wie zum Beispiel die Navigationsleisten, welche noch ähnlicher zur iOS-Plattform aussehen könnten und daher verbesserungswürdig sind.

## 3. S Wartbarkeit, Erweiterbarkeit, Verständlichkeit:

Teilweise umgesetzt. Wie im **Abschnitt 6. Implementierung** erwähnt, sind ausreichend Rückfalltests und Dokumentation für die grundlegenden Komponen-

## *7. Evaluation*

ten vorhanden. Weiter steht eine Mocked-Datenbank zur Verfügung, mit der die Funktionalität unabhängig von der GitHub-API getestet werden kann. Darüber hinaus wurde versucht, möglichst so modular zu programmieren und an geeigneten Stellen wurden Entwurfsmuster verwendet. All dies sollte sich positiv auf die Wartbarkeit, Erweiterbarkeit und Verständlichkeit auswirken. Jedoch konnte dieser Qualitätsstandard nicht auf den gesamten Programmcode angewendet werden. Denn zum Ende der Bearbeitungszeit wurde der Schwerpunkt eher darauf gelegt, alle erforderlichen Funktionen zu in die Benutzeroberfläche zu integrieren, um so ein möglichst validierbares Produkt zu erhalten. Dadurch litt jedoch die Modularität und Dokumentation des Programmcodes etwas.

### **4. S Qualität & Korrektheit:**

Umgesetzt. Es wird mit großer Sicherheit vermutet, dass die Anwendung sich so verhält, wie zuvor spezifiziert wurde. Dies liegt daran, dass es, wie im **Unterabschnitt 6.3. Qualitätssicherung** erwähnt, für die Grundkomponenten der Anwendung reichlich Rückfalltests gibt und die grafische Oberfläche ausgiebig manuell getestet wurde.

### **5. C Leistung:**

Überwiegend umgesetzt: So lief die App während des Tests flüssig und es konnte keine hohe CPU-Auslastung festgestellt werden. Außerdem ist die App mit rund 73 MB relativ klein, und durch Funktionen wie den dynamisch nachladenden Kalender und den Proxy wurden Ladezeiten sowie Netzwerkauslastung reduziert. Die Ladezeit der Terminkalenderseite hängt jedoch von der Bearbeitungszeit des Parsers ab. Dieser ist wiederum abhängig vom Inhalt der CLI-Terminkalenderdatei und folglich auch vom Nutzer. Aus diesem Grund sollte der Parser zunächst einem Last-, Stress- und Leistungstest unterzogen werden, bevor die Ladezeit endgültig eingeschätzt werden kann. Es wird jedoch [erwartet/geschätzt], dass der Rechenaufwand des Parsers vernachlässigbar ist.

### **6. C Reichweite:**

Teilweise umgesetzt: Die App sollte einerseits auf recht alten Handys nutzbar sein, da sie verhältnismäßig alte Betriebssystemversionen wie iOS 11.0 (von 2017) und Android 5.0 (von 2014) unterstützt. Außerdem benötigt sie, wie zuvor erwähnt, relativ wenig Leistung. Jedoch lässt sich die Anwendung vorerst trotz der Verwendung des Cross-Frameworks Flutter nur auf iOS installieren. Aufgrund von zeitlichen Beschränkungen wurde sich zunächst eher auf eine Plattform konzentriert, in diesem Fall iOS. Dabei wurden einige Darstellungs-komponenten verwendet die inkompatibel mit Android sind. Jedoch wird der Aufwand, die Anwendung auch für Android zu ermöglichen, als relativ gering eingeschätzt. Es müssen lediglich die Komponenten durch eine Abfrage gegen die Android-Kompliment ersetzt werden.

## 8 Fazit

*Ziele:* In dieser Arbeit wurde versucht, eine passende App für CLI-Terminkalender zu erstellen. „Passend“ bedeutet in diesem Sinne, dass die Stärken von PC und Handy berücksichtigt werden und somit in die zu erstellende Anwendung einfließen.

*Methoden:* Dazu wurde zunächst versucht herauszufinden, was die Stärken von Handys und PCs überhaupt sind, indem die wesentlichen Unterschiede zwischen ihnen verglichen wurden. Anschließend wurden mithilfe von Erhebungstechniken die Anforderungen an die zu erstellende App ermittelt und festgehalten. Auf Basis dieser Anforderungen wurden Gedanken zur Auswahl der passenden Technologien gemacht. Danach wurde das Design der Anwendung erstellt und überdacht, da dies ein wichtiger Punkt für die Anforderungen sowie eine mögliche Stärken von Handys ist. Da nun die Anforderungen und das Design der Anwendung bekannt sind sowie die Technologie, mit der sie erstellt werden soll, wurde sich anschließend mit der Implementierung befassen. Zum Schluss wurde die Anwendung mit einer Anforderungsverifizierung evaluiert, um festzustellen, ob das Erstellte auch das erfüllt, was sich zuvor vorgenommen wurde.

*Ergebnisse:* Die wohl wichtigsten Ergebnisse dieser Arbeit sind die entstandene Anwendung und die Erkenntnisse über die Stärken von Handys und PCs.

Die Stärken sind dabei besonders bedeutend, da sie bei der Erstellung der Anwendung, die versucht, die Lücke zwischen Terminal und Smartphone zu überbrücken, genutzt wurden, um so das Ziel zu erreichen. Dabei kam es zu den Ergebnissen, dass Handys die Stärken haben, für jene Aufgaben gut zu funktionieren, die ressourcenschonend sind und nicht viel Leistung benötigen, keine schnelle, präzise und vielfältige Eingabe erfordern, nur wenige Informationen darstellen oder benötigen, ohne viele Optionen und Konfigurationen auskommen, kurzweilig sind oder wenig Zeit benötigen, lohnenswert sind, unterwegs zu lösen, sowie einfach und intuitiv sein sollen.

Während dessen besitzen PCs die Stärke für jene Aufgaben gut zu sein, die viel Leistung benötigen, schnelle, präzise oder vielfältige Eingaben erfordern, viele Informationen gleichzeitig darstellen oder benötigen, viele Optionen und Konfigurationen anbieten oder benötigen sowie langwierig sind oder viel Zeit benötigen.

Zudem viel während des Designs der Anwendung fiel dabei auf, dass viele der Regeln in den Designrichtlinien mit den Stärken des Handys übereinstimmen. Beispielsweise wird empfohlen, dass die App und ihr Design folgende Merkmale aufweisen sollten: „sofort einsatzbereit sein“, „Aufgaben vereinfachen“, „Informationen reduzieren“, „Informationen indirekt vermitteln“, „konsistent innerhalb der App und Plattform sein“ und „intuitiv sein“.

Die Stärken von PCs und Handys sowie des Designs flossen in die Anforderungen und damit in die Anwendung ein. So entstand laut der Evaluation eine Anwendung, die in der Lage ist, die Stärken von PCs und Handys umzusetzen, indem sie unter anderem Funktionen wie CLI-Terminkalender grafisch darzustellen, Erinnerungen zu erzeugen, diese mit dem PC zu teilen, Benachrichtigungen für Termine anzuzeigen und konfigurierbar über den PC zu sein, anbietet.

## *8. Fazit*

*Schlussfolgerungen:* Dadurch, dass die Stärken des Handys und PCs systematisch durch systematische Vergleiche erarbeitet und mithilfe einige Statistiken und Studien bekräftigt wurden, werden diese als aussagekräftig bewertet. Insbesondere, weil sie während des Designs durch Richtlinien und Regeln weiter bekräftigen ließen. Da weiter die Anforderungen der Anwendung auf diesen Stärken basieren und die Anwendung erfolgreich implementiert wurde, wird angenommen, dass das Ziel, eine Kalender-App zu entwickeln, die versucht, die Lücke zwischen Terminal und Smartphone zu überbrücken, erreicht wurde. Weiter wird sogar vermutet, dass es sich bei dieser App um eine nützliche Anwendung handeln könnte, da sie die Stärken von Handys und PCs nutzt und somit eine Anwendung für eine sonst schwer vorstellbare Kategorie, die CLI-Programme, auf einem Smartphone ermöglicht. Um jedoch aussagekräftigere Ergebnisse über die Nützlichkeit der App zu erzielen, sollten weitere Evaluationen durchgeführt werden.

*Schwierigkeiten:* Während der Arbeit traten einige Schwierigkeiten auf. Einerseits wurde der Aufwand für das Testen mithilfe von Rückfalltests für den Parser und die Datenbank unterschätzt. So mussten relativ häufig nach Änderungen an Funktionen viele Tests angepasst werden müssen. Andererseits war das Ergebnis der Nutzerumfrage, obwohl bereits erwartet wurde, dass dabei nicht allzu viele Informationen gewonnen werden können, dennoch enttäuschend. Der Nutzen stand in keinem Verhältnis zum Aufwand, der bei der Auswahl der Standorte und der Fragenstruktur betrieben wurde. Nachdem sich herausgestellt hatte, dass nur wenige Nutzer teilnahmen und dabei keine nützlichen Informationen erhoben werden konnten, wurde beschlossen, diese Erhebungstechnik nicht weiter zu verfolgen, um so weitere Zeit und Ressourcen zu sparen. Außerdem stellte sich heraus, dass obwohl absichtlich ein als „simpel“ betonter Parser ausgewählt wurde, die Implementierung des gewählten Parsers aufgrund seiner Komplexität doch unerwartet aufwändig war.

*Erfolge:* Während der Arbeit gab es auch einige Erfolge. Zum einen stellte sich beim Design der Anwendung heraus, dass viele der zuvor gezogenen Schlussfolgerungen über die Stärken von Handys mit den Erkenntnissen aus den Richtlinien und Regeln überschneideten. Somit konnten die Annahmen der Stärken des Handys und PCs weiter bestärkt werden. Zum anderen erwiesen sich die Rückfalltests zwar anfangs als sehr zeitaufwendig, doch sie waren letztendlich äußerst hilfreich. Insbesondere die Tests für die Datenbankverbindung trugen dazu bei, das Verständnis zu ihr zu verbessern und letztendlich viel Zeit zu sparen. Schließlich erwiesen sich auch die Entwurfsmuster als äußerst hilfreich, insbesondere das Zusammenspiel des Strategie-Entwurfsmusters mit einer simulierten Datenbank und vielen Rückfalltests für die echten Datenbankverbindung. Bis zur Fertigstellung aller Funktionen der Anwendung wurde ausschließlich die simulierte Datenbank für Tests verwendet, um mögliche API-Fehler auszuschließen. Als alles implementiert war, konnte die simulierte Datenbank durch die echte ersetzt werden, indem nur eine Codezeile geändert wurde, da das Strategie-Design-Pattern verwendet wurde. Die Anwendung funktionierte direkt mit der echten Datenbank und es traten keine Defekte auf, da anscheinend alle Fehler zuvor mithilfe der umfangreichen Rückfalltests beseitigt wurden.

*Ausblick:* Falls sich die Anwendung als nützlich beweist, könnte dies bedeuten, dass es für Anwendungen durchaus lohnenswert sein kann, diese unter Berücksichtigung der Stärken sowohl von Handys als auch von PCs zu entwickeln. Möglicherweise könnte dies zeigen, dass Anwendungen nicht unbedingt als eigenständige Einheiten genutzt werden müssen, sondern voneinander abhängig sein und dennoch oder gerade deshalb nützlich sein können.

Um jedoch die Nützlichkeit sowie einige der nicht funktionalen Anforderungen abschließend bewerten zu können, benötigt es weiterer Evaluation. Insbesondere ein Nutzertest könnte hierbei nützliche Erkenntnisse liefern.

Bevor ein Nutzertest durchgeführt wird, empfiehlt es sich jedoch, die Anwendung zunächst zu vervollständigen, um durch den Test auch aussagekräftigere Schlüsse ziehen zu können. So sollte die Anwendung zunächst „refactored“ werden, da zum Ende hin versucht wurde, möglichst viele Funktionen noch zu implementieren, was zu einer Einschränkung der Qualität des Programmcodes führte. Des Weiteren bestehen noch zu kleine Designverbesserungen. So könnte beispielsweise die Appliste noch einheitlicher zur iOS-Plattform aussehen. Außerdem existieren noch einige funktionale Anforderungen, die noch nicht umgesetzt wurden und entsprechend lohnenswert wären, zu implementieren.

Ab diesem Punkt wäre ein Nutzertest vorstellbar. Es gibt jedoch immer noch weitere Anforderungen, die verbessert werden könnten. So wäre es zum Beispiel nützlich, die Reichweite der Anwendung zu erhöhen, indem weitere Parser hinzugefügt werden. Es wäre auch sinnvoll, die Anwendung für Android zu ermöglichen. Dafür müssten einige wenige Komponenten durch eine Abfrage gegen das Android-Komplement ausgetauscht werden. Danach wäre es ratsam, das Design passend für die Android-Anwendung anzupassen, da die jetzige Anwendung im iOS-Design gestaltet ist und somit für Android die Vorteile eines einheitlichen Plattform-Designs entfallen würden.

## Literatur

- [1] Creative Commons CC BY 3.0. *Conventional Commits*. [Online; accessed Mar 3, 2023]. o. D. URL: <https://www.conventionalcommits.org/en/v1.0.0/>.
- [2] Apple Inc. *Calendar*. [Online; accessed Feb 28, 2023] mai. 2016. URL: <https://apps.apple.com/us/app/calendar/id1108185179>.
- [3] Apple Inc. *Accessibility*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/foundations/accessibility>.
- [4] Apple Inc. *Buttons*. [Online; accessed Feb 5, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/components/menus-and-actions/buttons/>.
- [5] Apple Inc. *Color*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/foundations/color>.
- [6] Apple Inc. *Dark Mode*. [Online; accessed Mar 10, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/foundations/dark-mode>.
- [7] Apple Inc. *Designing for iOS*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/platforms/designing-for-ios>.
- [8] Apple Inc. *Entering data*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/patterns/entering-data>.
- [9] Apple Inc. *Icons*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/foundations/icons>.
- [10] Apple Inc. *Keyboards*. [Online; accessed Mar 10, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/inputs/keyboards>.
- [11] Apple Inc. *Launching*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/patterns/launching>.
- [12] Apple Inc. *Layout*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/foundations/layout>.
- [13] Apple Inc. *Loading*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/patterns/loading>.
- [14] Apple Inc. *Managing accounts*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/patterns/managing-accounts>.
- [15] Apple Inc. *NFC*. [Online; accessed Mar 10, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/technologies/nfc>.
- [16] Apple Inc. *Settings*. [Online; accessed Mar 10, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/patterns/settings>.

- [17] Apple Inc. *Siri*. [Online; accessed Mar 10, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/technologies/siri/introduction>.
- [18] Apple Inc. *Typography*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/foundations/typography>.
- [19] Apple Inc. *Writing*. [Online; accessed Apr 3, 2023]. o. D. URL: <https://developer.apple.com/design/human-interface-guidelines/foundations/writing>.
- [20] archlinux Forums. [Online; accessed Jan 24, 2023]. o. D. URL: <https://bbs.archlinux.org>.
- [21] Benjamin Crowell. *When*. [Online; accessed Apr 2, 2023] dec. 2011. URL: <http://www.lightandmatter.com/when/when.html>.
- [22] calcuse Development Team. *calcuse*. [Online; accessed Feb 2, 2023]. 2012. URL: <https://calcuse.org>.
- [23] Counterpoint. *Average Smartphone NAND Flash Capacity Crossed 100GB in 2020*. [Online; accessed Jan 26, 2023] mar. März 2021. URL: [https://report.counterpointresearch.com/posts/report\\_view/Individual/2292](https://report.counterpointresearch.com/posts/report_view/Individual/2292).
- [24] Dart. *Effective Dart: Style*. [Online; accessed Apr 2, 2023]. o. D. URL: <https://dart.dev/guides/language/effective-dart/style>.
- [25] dart.dev. *test 1.23.1*. [Online; accessed Feb 15, 2023] apr. 2015. URL: <https://pub.dev/packages/test>.
- [26] Debian User Forums. [Online; accessed Jan 24, 2023]. o. D. URL: <https://forums.debian.net/index.php>.
- [27] Dianne Skoll. *remind*. [Online; accessed Feb 2, 2023] mar. 1996. URL: <https://dianne.skoll.ca/projects/remind/>.
- [28] Eric Enge. *Mobile vs. Desktop Usage in 2020*. [Online; accessed Jan 28, 2023] mar. März 2021. URL: <https://www.perficient.com/insights/research-hub/mobile-vs-desktop-usage>.
- [29] Facebook. *react-native*. [Online; accessed Feb 13, 2023]. o. D. URL: <https://github.com/facebook/react-native>.
- [30] flutter. *Set up an editor*. [Online; accessed Feb 15, 2023]. o. D. URL: <https://docs.flutter.dev/get-started/editor?tab=androidstudio>.
- [31] flutter.dev. *flutter\_lints 2.0.1*. [Online; accessed Feb 15, 2023] may. 2021. URL: [https://pub.dev/packages/flutter\\_lints](https://pub.dev/packages/flutter_lints).
- [32] GitHub. *Authenticating with username and password*. [Online; accessed Mar 11, 2023]. o. D. URL: <https://docs.github.com/en/rest/overview/authenticating-to-the-rest-api?apiVersion=2022-11-28#authenticating-with-username-and-password>.
- [33] GitHub. *Basic writing and formatting syntax*. [Online; accessed Mar 11, 2023]. o. D. URL: <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax#section-links>.

- [34] GitHub. *Create a repository for the authenticated user*. [Online; accessed Mar 11, 2023]. o. D. URL: <https://docs.github.com/en/rest/repos/repos?apiVersion=2022-11-28#create-a-repository-for-the-authenticated-user>.
- [35] GitHub. *Create or update file contents*. [Online; accessed Mar 11, 2023]. o. D. URL: <https://docs.github.com/en/rest/repos/contents?apiVersion=2022-11-28#create-or-update-file-contents>.
- [36] GitHub. *Get repository content*. [Online; accessed Mar 11, 2023]. o. D. URL: <https://docs.github.com/en/rest/repos/contents?apiVersion=2022-11-28#get-repository-content>.
- [37] GitHub. *GitHub CLI manual*. [Online; accessed Apr 2, 2023]. o. D. URL: <https://cli.github.com/manual/>.
- [38] GitHub. *OAuth authorizations*. [Online; accessed Mar 11, 2023]. o. D. URL: <https://docs.github.com/en/enterprise-server@3.8/rest/oauth-authorizations#about-oauth-authorizations>.
- [39] GitHub. *Scopes for OAuth Apps*. [Online; accessed Mar 11, 2023]. o. D. URL: <https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/scopes-for-oauth-apps>.
- [40] GlobalWebIndex. *Connecting the dots*. [Online; accessed Jan 30, 2023] S. 8, 2022. URL: <https://mpost.io/wp-content/uploads/GWI-Connecting-The-Dots-2023-Global-Trends.pdf>.
- [41] GlobalWebIndex. *The global media landscape*. [Online; accessed Jan 31, 2023] S. 15, 2022. URL: <https://mpost.io/wp-content/uploads/GWI-Global-Media-Landscape-2023.pdf>.
- [42] GlobalWebIndex. *The global media landscape*. [Online; accessed Jan 31, 2023] S. 18, 2022. URL: <https://mpost.io/wp-content/uploads/GWI-Global-Media-Landscape-2023.pdf>.
- [43] GlobalWebIndex. *The global media landscape*. [Online; accessed Jan 31, 2023] S. 9, 2022. URL: <https://mpost.io/wp-content/uploads/GWI-Global-Media-Landscape-2023.pdf>.
- [44] Google. *flutter*. [Online; accessed Feb 13, 2023]. 2023. URL: <https://github.com/flutter/flutter>.
- [45] Google. *Google Trends Vergleich: Flutter vs React Native*. [Online; accessed Feb 13, 2023]. 2023. URL: [https://trends.google.de/trends/explore?date=2021-12-01%202023-02-13&q=%2Fg%2F11f03\\_rzbg,%2Fg%2F11h03gfxy9](https://trends.google.de/trends/explore?date=2021-12-01%202023-02-13&q=%2Fg%2F11f03_rzbg,%2Fg%2F11h03gfxy9).
- [46] Google. *Animation and motion widgets*. [Online; accessed Feb 14, 2023]. o. D. URL: <https://docs.flutter.dev/development/ui/widgets/animation>.
- [47] Google. *Floating action buttons*. [Online; accessed Mar 10, 2023]. o. D. URL: <https://m3.material.io/components/floating-action-button/overview>.
- [48] Google. *Learn when you'll get software updates on Google Pixel phones*. [Online; accessed Feb 6, 2023]. o. D. URL: <https://support.google.com/nexus/answer/4457705?hl=en#zippy=>.

- [49] Google. *Material Design*. [Online; accessed Feb 5, 2023]. o. D. URL: <https://m3.material.io>.
- [50] Google Data. *53% of visits are abandoned if a mobile site takes longer than 3 seconds to load*. [Online; accessed Jan 27, 2023] mar. März 2016. URL: <https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/mobile-site-load-time-statistics/>.
- [51] Google LLC. *Google Kalender: Terminplaner*. [Online; accessed Feb 28, 2023] feb. 2023. URL: <https://apps.apple.com/de/app/google-kalender-terminplaner/id909319292>.
- [52] Google Play internal data. [Online; accessed Apr 13, 2023] may. 2021.
- [53] google.dev. *json\_serializable* 6.6.1. [Online; accessed Feb 15, 2023] jul. 2017. URL: [https://pub.dev/packages/json\\_serializable](https://pub.dev/packages/json_serializable).
- [54] Hofbauer, W. *Ein Tag schreibt Geschichte: Das erste Mobiltelefon*. [Online; accessed Apr 13, 2023] aug. o. D. URL: <https://www.terramatteMagazin.com/a/te/das-erste-mobiltelefon-ein-datum-schreibt-geschichte>.
- [55] intel. *Unterstützt mein Intel Prozessor Microsoft Windows 10?* [Online; accessed Feb 21, 2023] feb. 2023. URL: <https://www.intel.de/content/www/de/de/support/articles/000006105/processors.html>.
- [56] inVerita. *Flutter vs Native vs React-Native: Examining performance*. [Online; accessed Feb 14, 2023] mar. 2020. URL: <https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980>.
- [57] inVerita. *Flutter vs React Native vs Native: Deep Performance Comparison*. [Online; accessed Feb 14, 2023] jun. 2020. URL: <https://medium.com/swlh/flutter-vs-react-native-vs-native-deep-performance-comparison-990b90c11433>.
- [58] Kate Moran & Kim Salazar. *Large Devices Preferred for Important Tasks*. [Online; accessed Jan 31, 2023] aug. Aug. 2019. URL: <https://www.nngroup.com/articles/large-devices-important-tasks/>.
- [59] lfos. *calcurse*. [Online; accessed Apr 2, 2023] sep. 2013. URL: <https://github.com/lfos/calcurse>.
- [60] Linux Mint Forums. [Online; accessed Jan 24, 2023]. o. D. URL: <https://forums.linuxmint.com/index.php?sid=a358cbf3d4a2eb882e8ec10c36f507a4>.
- [61] Macecraft Software. *Fastest Starting Windows*. [Online; accessed Feb 24, 2023]. o. D. URL: <https://startuptimer.com/#leaderboard>.
- [62] pascalwelsch.com. *lint* 2.0.1. [Online; accessed Apr 2, 2023] mar. 2020. URL: <https://pub.dev/packages/lint>.
- [63] Doantam Phan. *Mobile-first indexing*. [Online; accessed Jan 30, 2023] nov. Nov. 2016. URL: <https://developers.google.com/search/blog/2016/11/mobile-first-indexing>.
- [64] pimutils. *khal*. [Online; accessed Feb 2, 2023] aug. 2013. URL: <https://github.com/pimutils/khal>.
- [65] pinterest. *Company*. [Online; accessed Jan 30, 2023]. 2023. URL: <https://newsroom.pinterest.com/en/company>.

## Literatur

- [66] Meta Platforms. *Core Components and APIs*. [Online; accessed Feb 14, 2023]. o. D. URL: <https://reactnative.dev/docs/components-and-apis>.
- [67] Puppy Linux Discussion Forum. [Online; accessed Jan 24, 2023]. o. D. URL: <https://forums.linuxmint.com/index.php?sid=a358cbf3d4a2eb882e8ec10c36f507a4>.
- [68] Qualitest. *Survey: 88% of App Users Will Abandon Apps Based on Bugs and Glitches*. [Online; accessed Feb 8, 2023] may. 2017. URL: <https://qualitestgroup.com/news/survey-88-of-app-users-will-abandon-apps-based-on-bugs-and-glitches/>.
- [69] RapidTables. *Screen Resolution Statistics*. [Online; accessed Feb 6, 2023]. 2014. URL: <https://www.rapidtables.com/web/dev/screen-resolution-statistics.html>.
- [70] Readdle Technologies Limited. *Calendars - Planner & Kalender*. [Online; accessed Feb 28, 2023] feb. 2023. URL: <https://apps.apple.com/de/app/calendars-planner-kalender/id608834326>.
- [71] Reddit. *Apps for Android!* [Online; accessed Jan 24, 2023] dec. Dez. 2009. URL: <https://www.reddit.com/r/androidapps/>.
- [72] Reddit. *Command Line*. [Online; accessed Jan 24, 2023] sep. Sep. 2010. URL: <https://www.reddit.com/r/commandline/>.
- [73] Reddit. *iOS Apps*. [Online; accessed Jan 24, 2023] oct. Okt. 2010. URL: <https://www.reddit.com/r/iosapps/>.
- [74] Ryan Dietzen. *Email Use 2017 - US report*. [Online; accessed Jan 30, 2023] S. 16, 2017. URL: <https://www.slideshare.net/adobe/adobe-consumer-email-survey-report-2017>.
- [75] Seagate. *Supplemental financial information*. [Online; accessed Feb 3, 2023] S. 9, 2022. URL: [https://s24.q4cdn.com/101481333/files/doc\\_financials/2022/q4/STX-Supplemental-FQ4'22.pdf](https://s24.q4cdn.com/101481333/files/doc_financials/2022/q4/STX-Supplemental-FQ4'22.pdf).
- [76] ShahimEssaid. *Allow issue comment attachments through API #1133*. [Online; accessed Mar 11, 2023] nov. 2017. URL: <https://github.com/isaacs/github/issues/1133>.
- [77] Simon Kemp. *Digital 2022: global overview report*. [Online; accessed Jan 31, 2023] S. 127, jan. Jan. 2022. URL: [https://datareportal.com/reports/digital-2022-global-overview-report?utm\\_source=Global\\_Digital\\_Reports&utm\\_medium=Article&utm\\_campaign=Digital\\_2022](https://datareportal.com/reports/digital-2022-global-overview-report?utm_source=Global_Digital_Reports&utm_medium=Article&utm_campaign=Digital_2022).
- [78] spinlock.sh. *github 9.9.0*. [Online; accessed Feb 15, 2023] jul. 2014. URL: <https://pub.dev/packages/github>.
- [79] Stack Exchange. *Unix & Linux*. [Online; accessed Jan 24, 2023]. o. D. URL: <https://unix.stackexchange.com>.
- [80] statcounter. *Desktop vs Mobile Market Share Worldwide*. [Online; accessed Jan 30, 2023]. 2022. URL: <https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide/#yearly-2011-2022>.

- [81] statcounter. *Desktop Operating System Market Share Worldwide*. [Online; accessed Feb 21, 2023]. 2023. URL: <https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/#monthly-202201-202301>.
- [82] statcounter. *Desktop Screen Resolution Stats Worldwide*. [Online; accessed Feb 6, 2023]. 2023. URL: <https://gs.statcounter.com/screen-resolution-stats/desktop/worldwide/#yearly-2022-2022-bar>.
- [83] statcounter. *Desktop Windows Version Market Share Worldwide*. [Online; accessed Feb 21, 2023]. 2023. URL: <https://gs.statcounter.com/windows-version-market-share/desktop/worldwide/#monthly-202201-202301>.
- [84] statcounter. *Mobile Operating System Market Share Worldwide*. [Online; accessed Feb 6, 2023]. 2023. URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide>.
- [85] Statista. *How Long Does Apple Support Older iPhone Models?* [Online; accessed Feb 6, 2023] sep. 2022. URL: <https://www.statista.com/chart/5824/ios-iphone-compatibility/>.
- [86] syncfusion.com. *syncfusion\_flutter\_calendar*. [Online; accessed Apr 2, 2023] dec. 2019. URL: [https://pub.dev/packages/syncfusion\\_flutter\\_calendar](https://pub.dev/packages/syncfusion_flutter_calendar).
- [87] Trevor Wheelwright. *2022 Cell Phone Usage Statistics: How Obsessed Are We?* [Online; accessed Jan 30, 2023] jan. Jan. 2022. URL: <https://www.reviews.org/mobile/cell-phone-addiction/>.
- [88] Ubuntu. *Ubuntu Manifest*. [Online; accessed Apr 2, 2023] mar. 2023. URL: <https://releases.ubuntu.com/focal/ubuntu-20.04.6-live-server-amd64.manifest>.
- [89] 79% of people say they're more likely to revisit and/or share a mobile site if it is easy to use. *Getting Things Done on Mobile*. [Online; accessed Jan 30, 2023] dec. Dez. 2017. URL: <https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/smartphone-productivity-tools-revisit-mobile/>.
- [90] Christian Weisgerber. *Die Unix-Philosophie*. [Online; accessed Apr 2, 2023] jun. 1998. URL: <http://sites.inka.de/mips/unix/unixphil.html>.
- [91] William Judd. *Full-size, TKL, 60% and more: a guide to mechanical keyboard sizes*. [Online; accessed Feb 18, 2023] aug. 2017. URL: <https://www.keyboardco.com/blog/index.php/2017/08/full-size-tkl-60-and-more-a-guide-to-mechanical-keyboard-sizes/>.
- [92] Yoram Wurmser. *The Majority of Americans' Mobile Time Spent Takes Place in Apps*. [Online; accessed Jan 30, 2023] jul. Juli 2020. URL: <https://www.insiderintelligence.com/content/the-majority-of-americans-mobile-time-spent-takes-place-in-apps>.

## **A Anhang**

Link zum Repository: [https://git.imp.fu-berlin.de/thob97/bachelor-cli\\_calendar\\_app](https://git.imp.fu-berlin.de/thob97/bachelor-cli_calendar_app)

Commit-Hash: ab2bf74a810a521c61af575b9af284b2dc07eca0

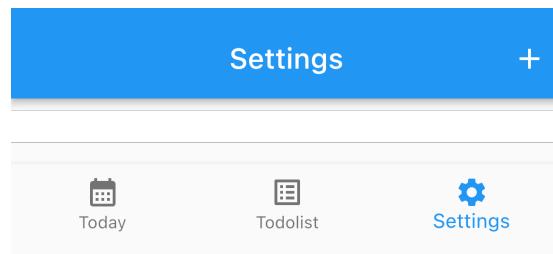


Abbildung 1: Erstes Appleisten Design

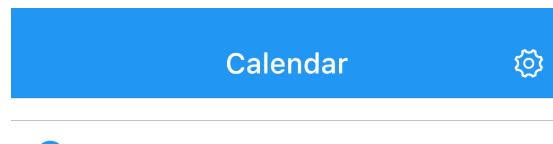


Abbildung 2: Aktuelles Appleisten Design

## A. Anhang

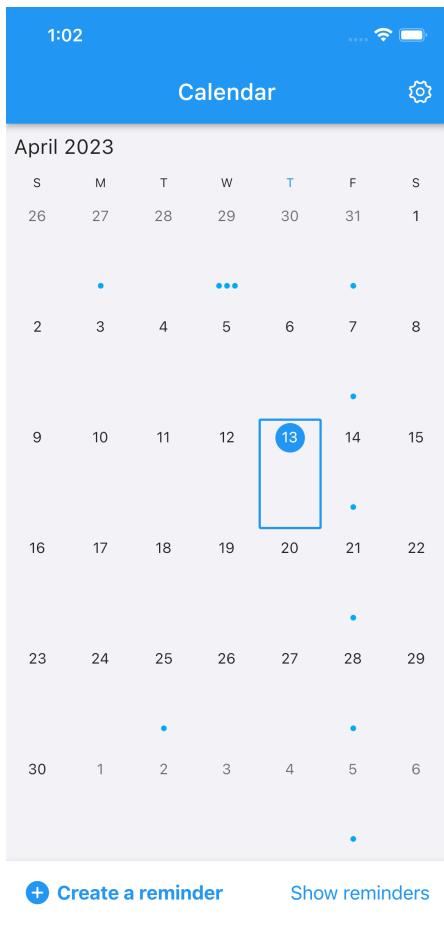


Abbildung 3: Monatsansicht

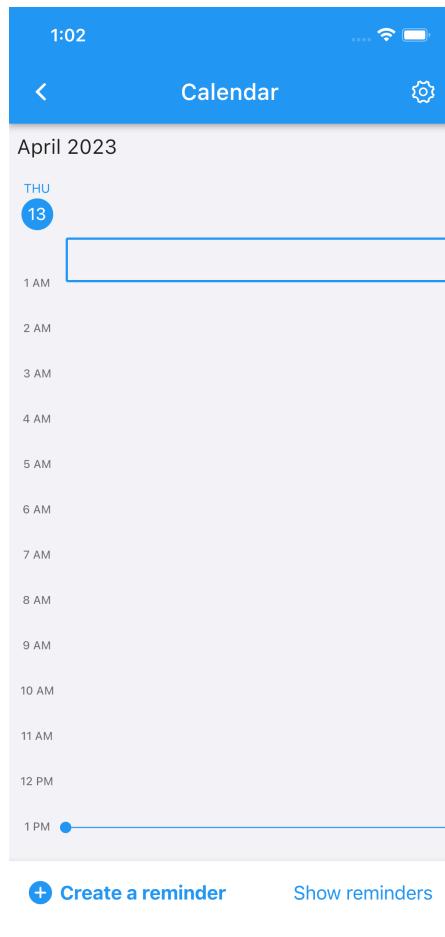


Abbildung 4: Tagesansicht

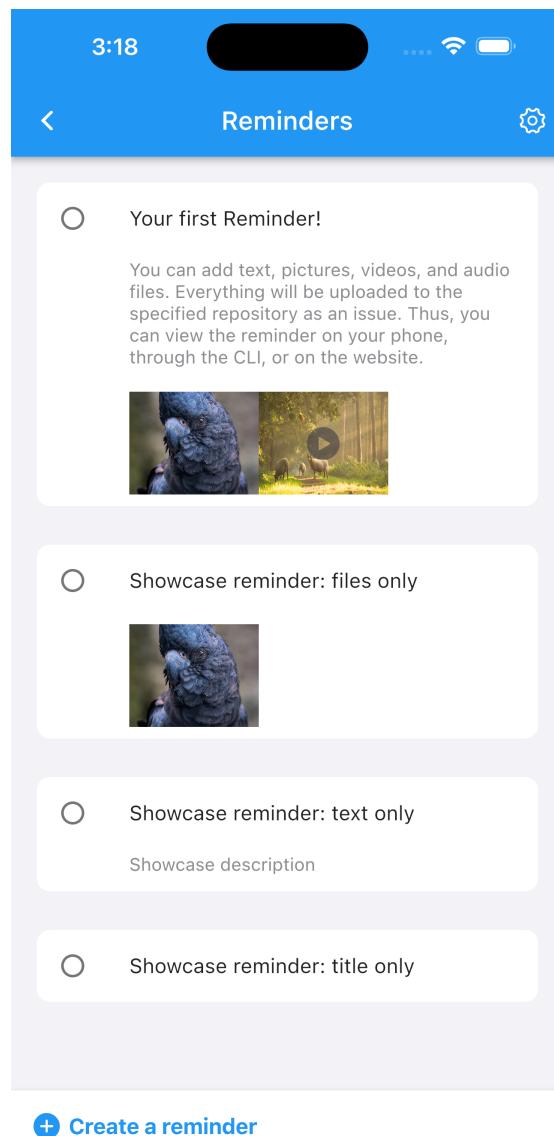


Abbildung 5: Erinnerungsseite mit Beispiel Erinnerungen

## A. Anhang

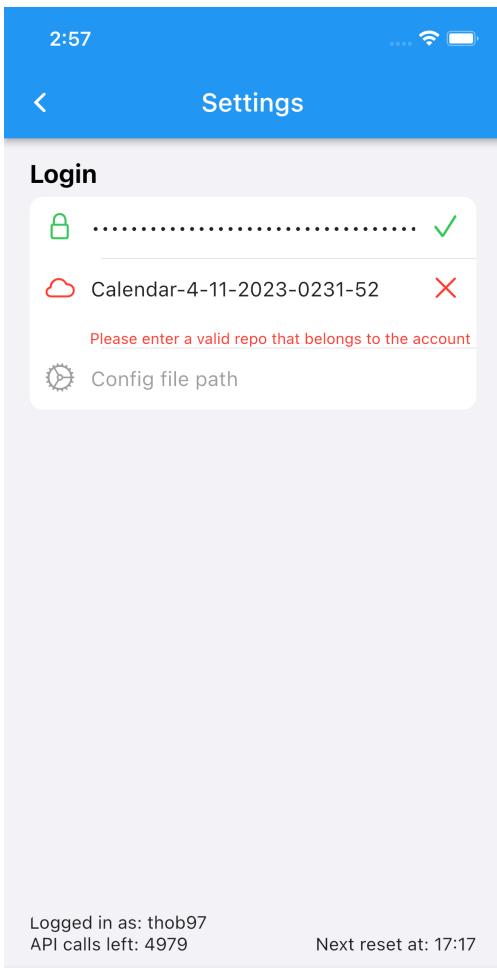


Abbildung 6: Loginseite

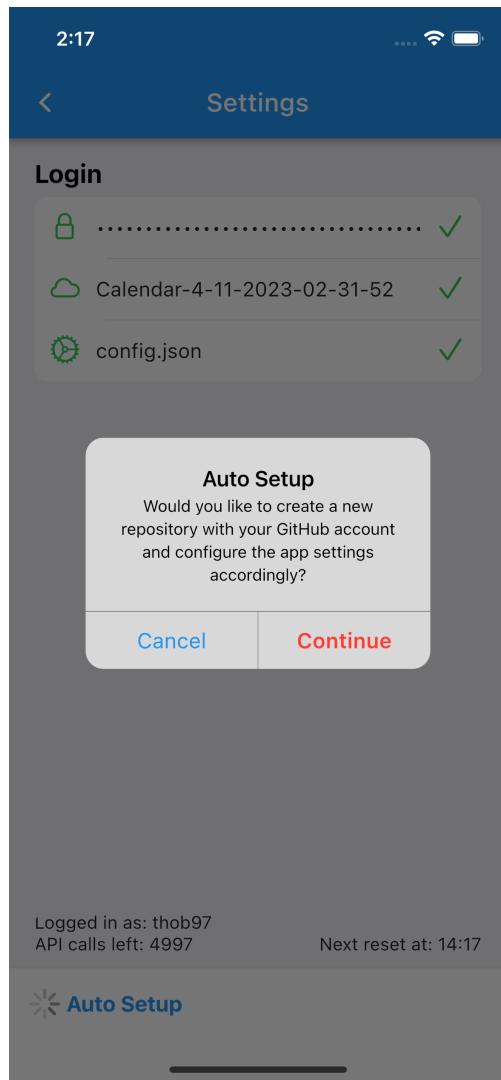


Abbildung 7: AutoSetup Alert



Abbildung 8: Systemtastatur mit Icons

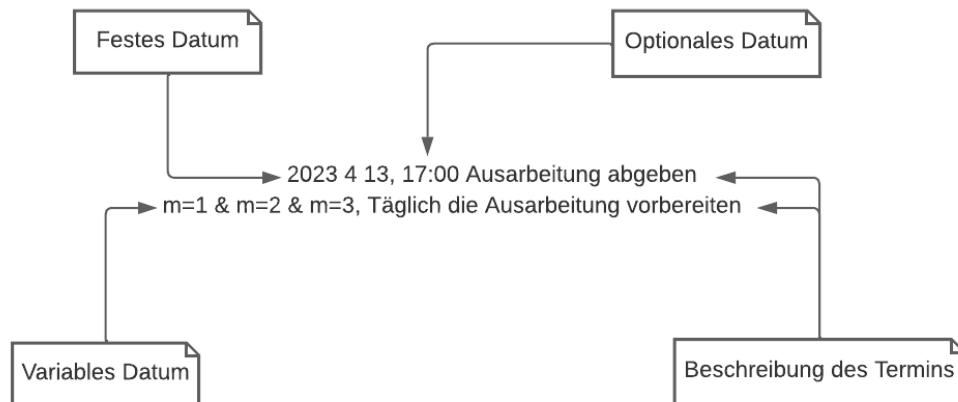


Abbildung 9: Modellierter WhenTermin

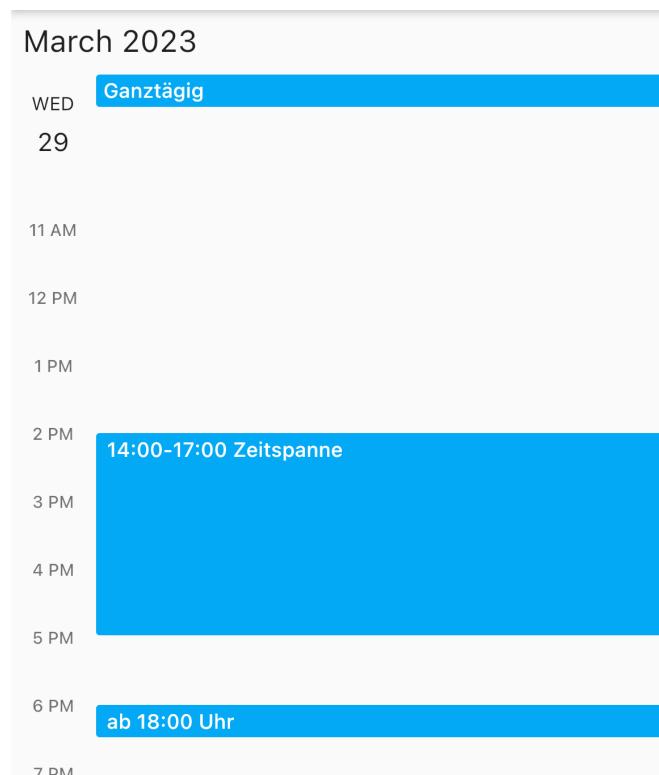


Abbildung 10: Beispiel Darstellung einer Zeitangabe

## A. Anhang

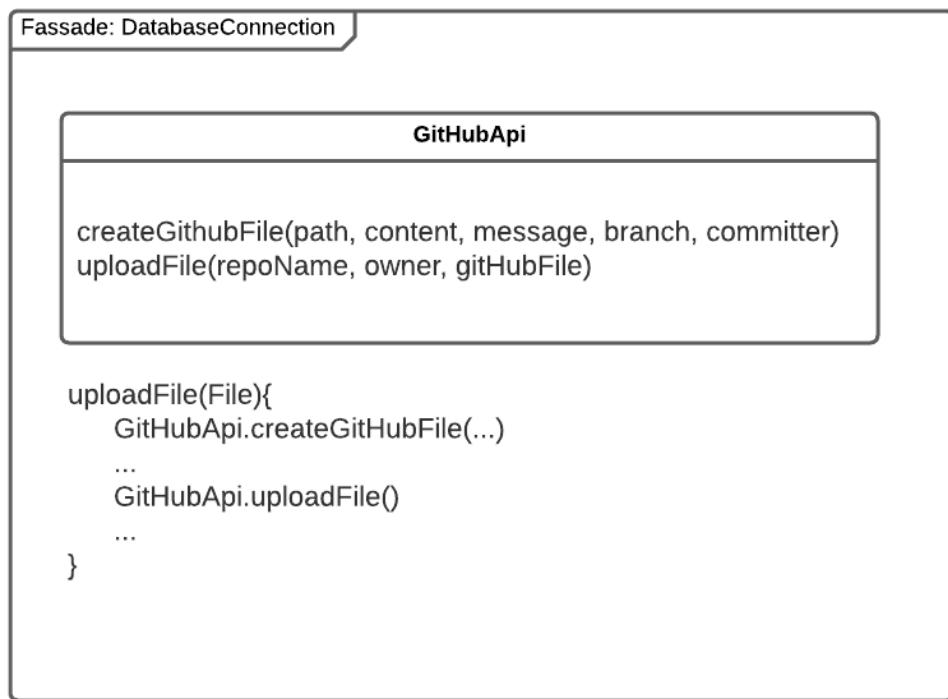


Abbildung 11: Vereinfachte Darstellung der Fassade

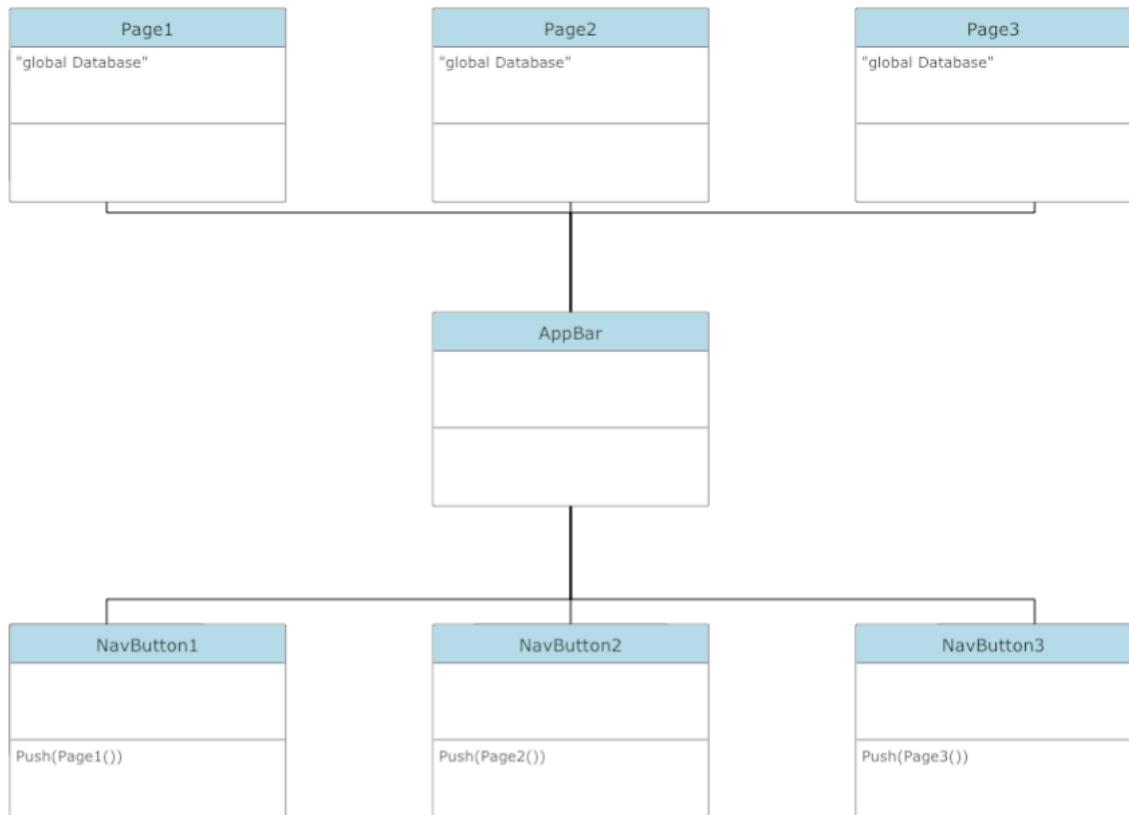


Abbildung 12: Globale Datenbank durch Singelton

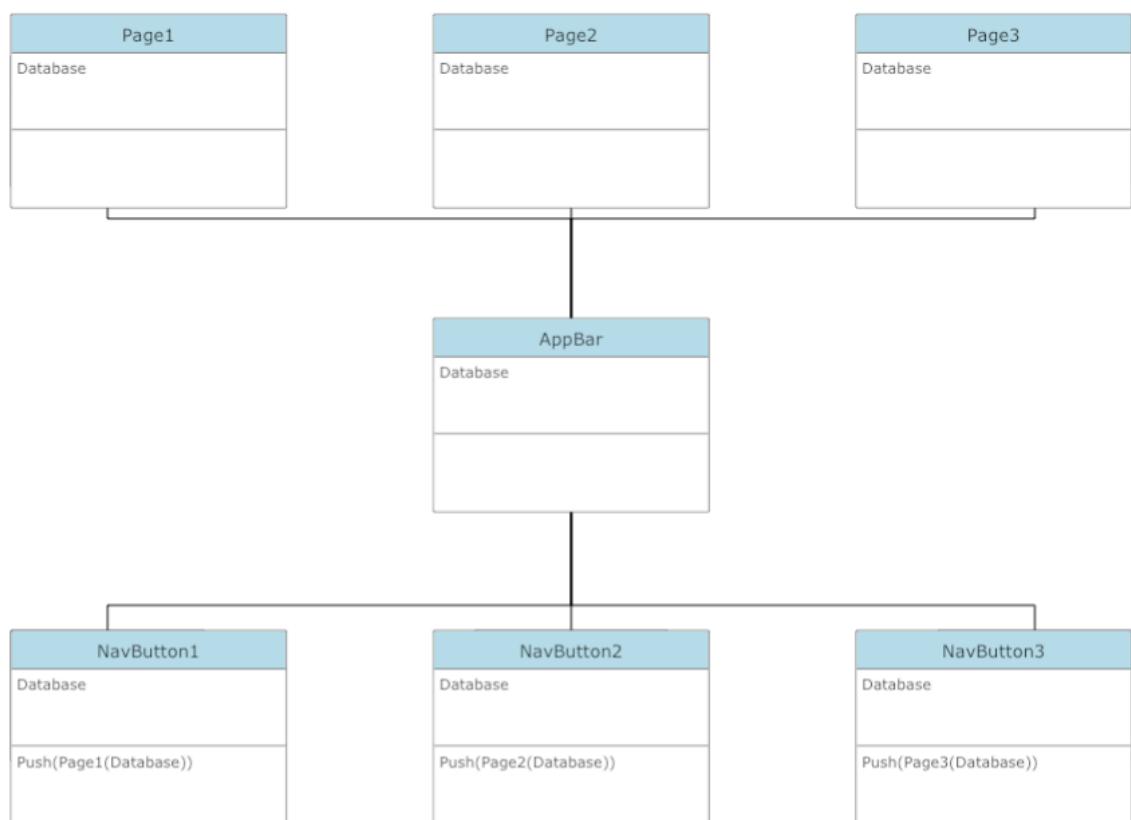


Abbildung 13: Weitergereichte Datenbank

## A. Anhang

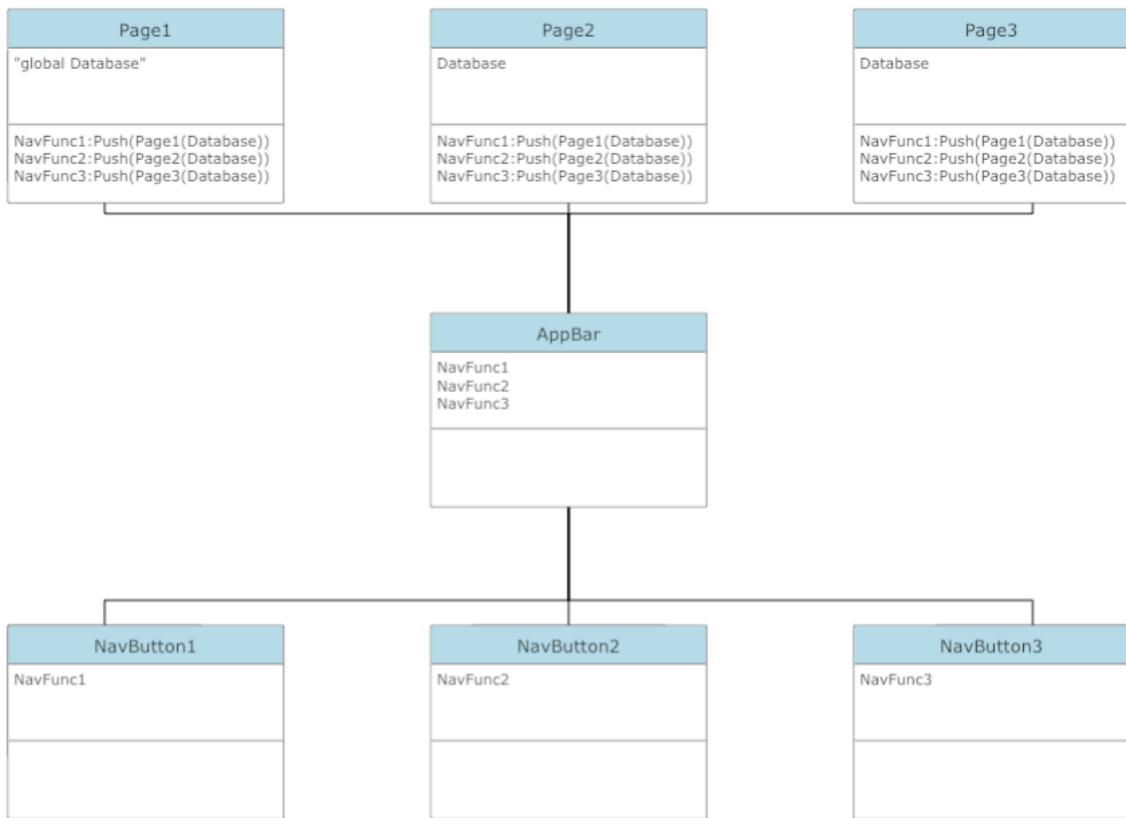


Abbildung 14: Nach oben geschobene Navigationsfunktionen

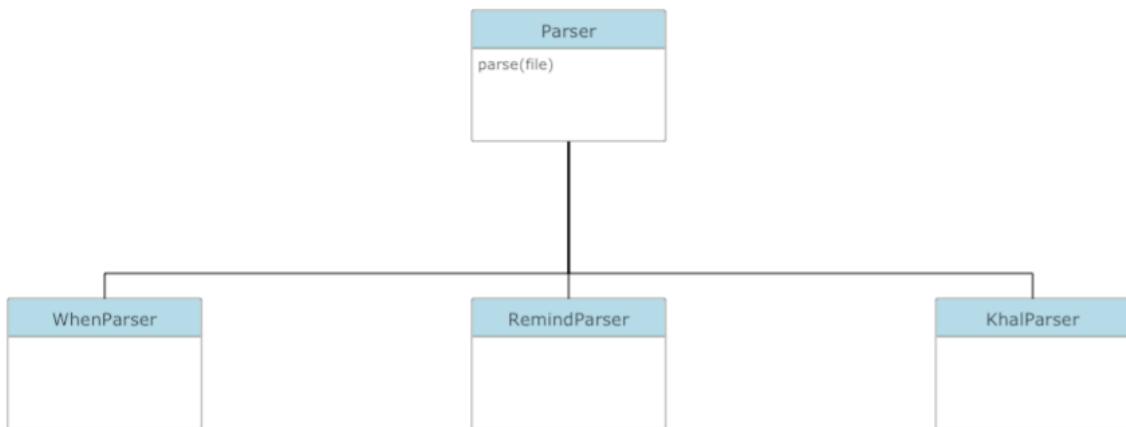


Abbildung 15: Beispiel der Strategiedarstellung für den Parser

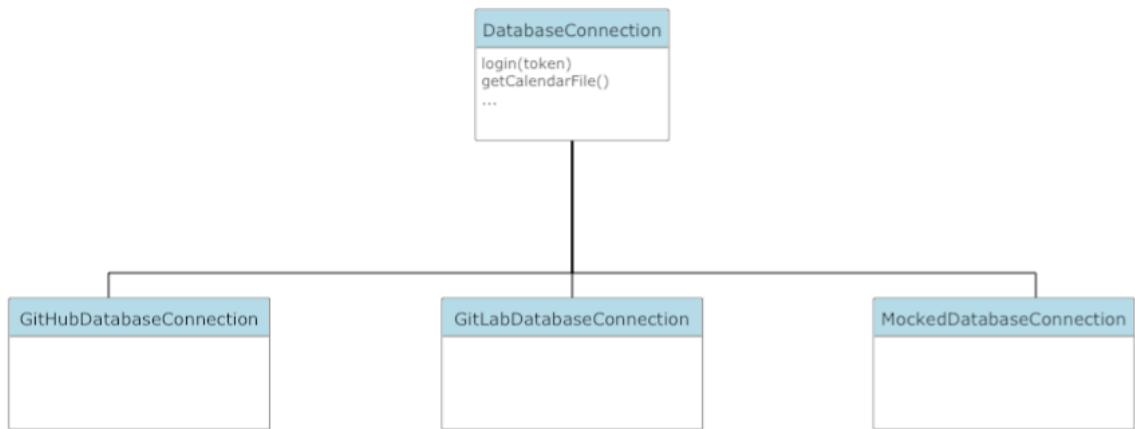


Abbildung 16: Beispiel der Strategiedarstellung für die Datenbank

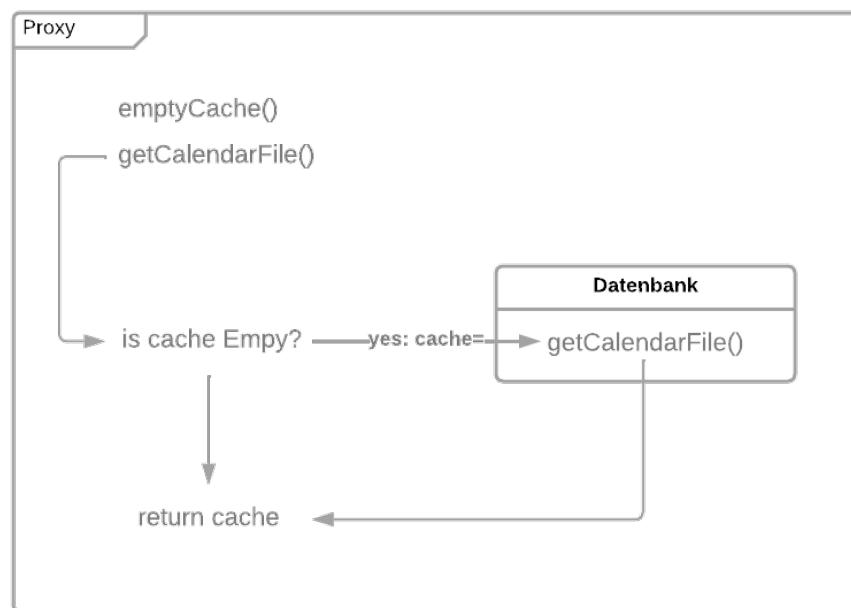


Abbildung 17: Vereinfachte Darstellung der Proxydatenbank

## A. Anhang

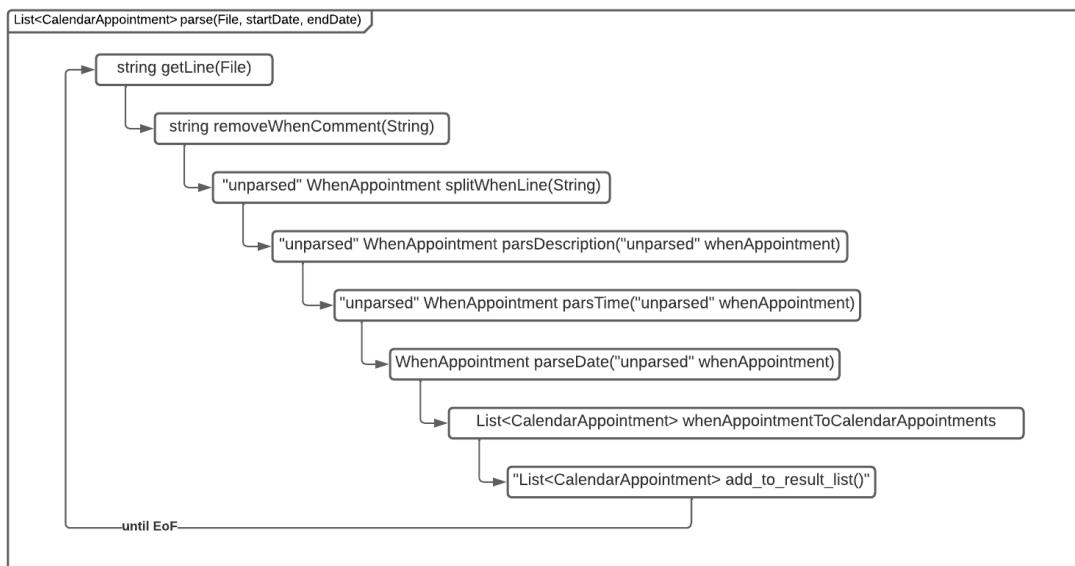


Abbildung 18: Ablauf mit der datenflussnetz ähnlichen Struktur

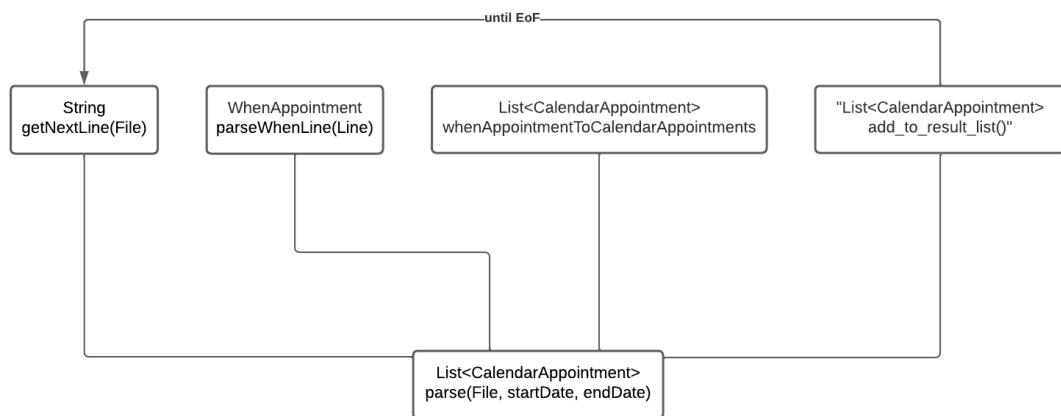


Abbildung 19: Ablauf mit der ablagebasierten ähnlichen Struktur Teil 1

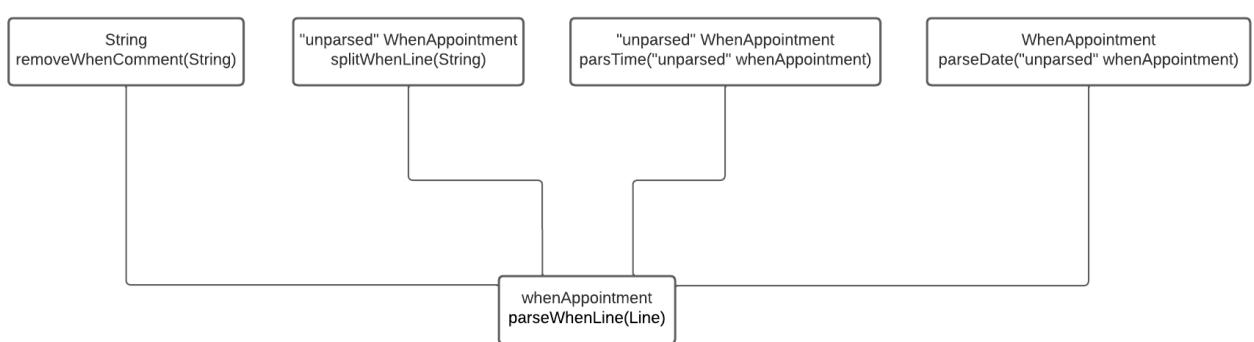


Abbildung 20: Ablauf mit der ablagebasierten ähnlichen Struktur Teil 2

*A. Anhang*

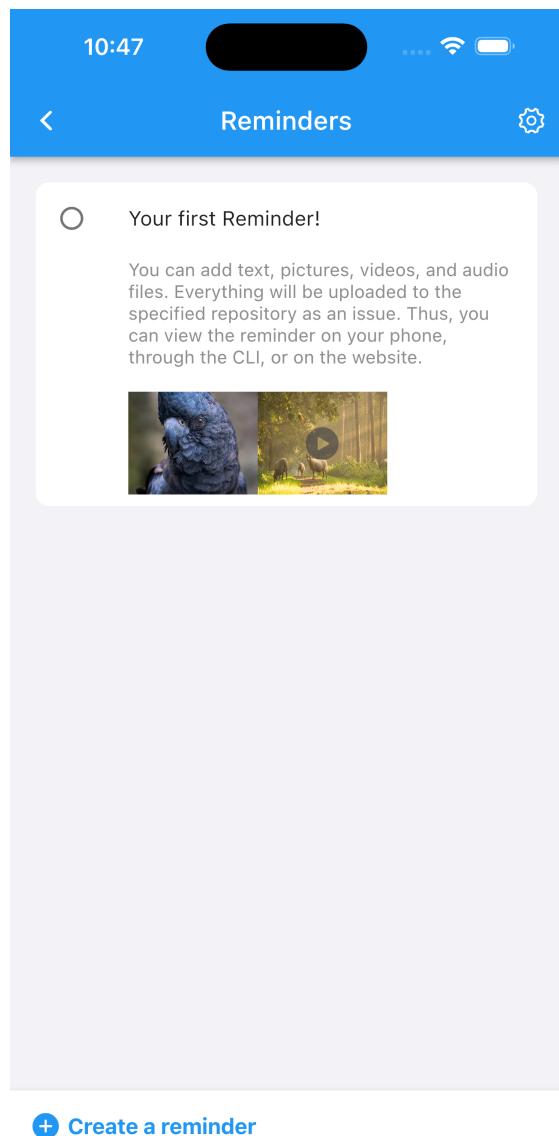


Abbildung 21: Handyansicht der Erinnerungen

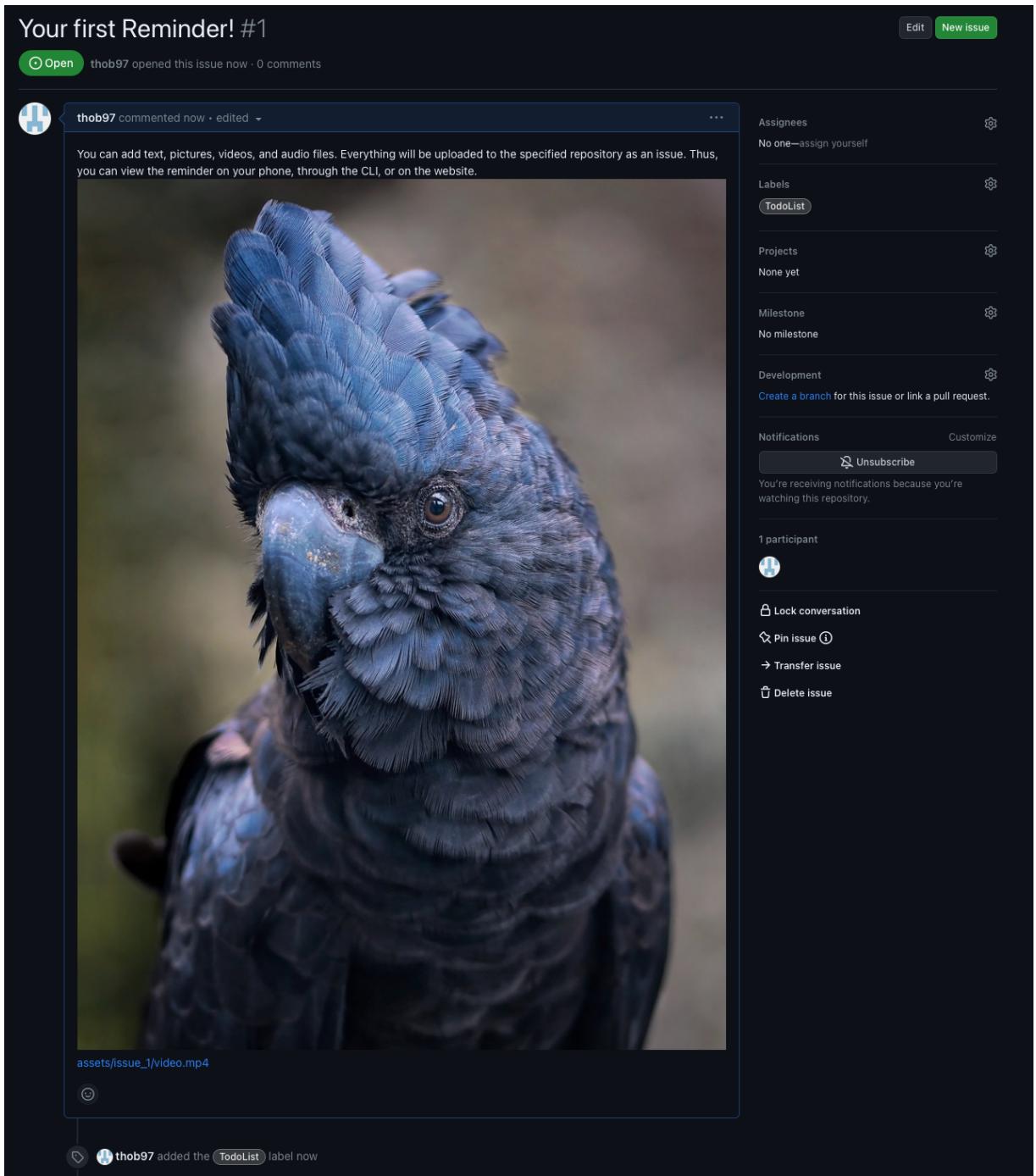
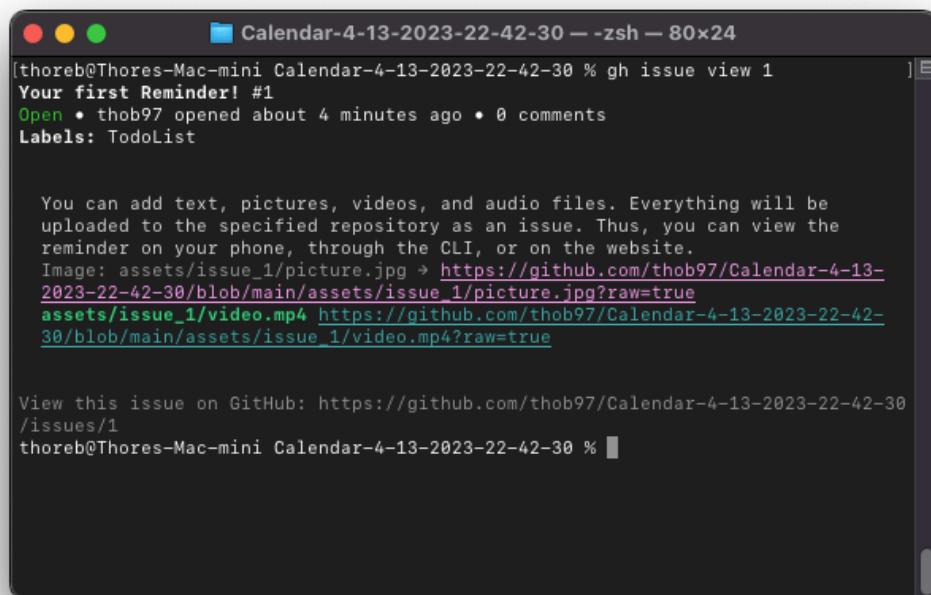


Abbildung 22: Webseitenansicht der Erinnerungen

## A. Anhang



The screenshot shows a macOS terminal window with the title bar 'Calendar-4-13-2023-22-42-30 — zsh — 80x24'. The terminal content is as follows:

```
[thoreb@Thores-Mac-mini Calendar-4-13-2023-22-42-30 % gh issue view 1
Your first Reminder! #1
Open • thob97 opened about 4 minutes ago • 0 comments
Labels: TodoList

You can add text, pictures, videos, and audio files. Everything will be
uploaded to the specified repository as an issue. Thus, you can view the
reminder on your phone, through the CLI, or on the website.
Image: assets/issue_1/picture.jpg → https://github.com/thob97/Calendar-4-13-2023-22-42-30/blob/main/assets/issue\_1/picture.jpg?raw=true
assets/issue_1/video.mp4 https://github.com/thob97/Calendar-4-13-2023-22-42-30/blob/main/assets/issue\_1/video.mp4?raw=true

View this issue on GitHub: https://github.com/thob97/Calendar-4-13-2023-22-42-30/issues/1
thoreb@Thores-Mac-mini Calendar-4-13-2023-22-42-30 %
```

Abbildung 23: CLI-Ansicht der Erinnerungen