



# Schriftliche Ausarbeitung

Erstellung einer Applikation zur Verwaltung von  
Todos, Events und Notizen

Erstellt von den **AngryNerds**:

Fabia Schmid

Florian Rath

Jan Beilfuß

Joscha Nassenstein

Robin Menzel

Ruthild Gilles

Sertan Cetin

Yannick Rüttgers

Prüfer:

Prof. Dr. Seifert

Eingereicht am:

07.02.2019

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>IV</b>
<b>Listingverzeichnis</b>	<b>V</b>
<b>1 Das Team: Angry Nerds</b>	<b>1</b>
<b>2 Ziele des Projektes (Fabia Schmid)</b>	<b>2</b>
<b>3 Projektplanung</b>	<b>4</b>
3.1 Beschreibung des Funktionsumfangs (Florian Rath) . . . . .	4
3.2 Projektlaufplan (Fabia Schmid) . . . . .	4
3.3 Planung der Software . . . . .	7
3.3.1 Planung des Mock-Ups (Robin Menzel) . . . . .	7
3.3.2 Planung der Datenstruktur und Schnittstellen (Ruthild Gilles) .	11
3.3.3 Planung der Activities und Layouts (Florian Rath) . . . . .	15
3.3.4 Planung der Navigation zwischen den Activities (Yannick Rüttgers)	15
3.4 Geplante Aufgabenverteilung im Team (Fabia Schmid) . . . . .	15
<b>4 Beschreibung des Projektverlaufs</b>	<b>16</b>
4.1 Tatsächliche Aufgabenverteilung im Team (Fabia Schmid) . . . . .	16
4.2 Teammeetingprotokolle . . . . .	16
4.3 Projekttagebücher aller Teammitglieder . . . . .	16
4.3.1 Projekttagebuch Jan Beilfuß . . . . .	16
4.3.2 Projekttagebuch Joscha Nassenstein . . . . .	17
4.3.3 Projekttagebuch Fabia Schmid . . . . .	18
4.3.4 Projekttagebuch Florian Rath . . . . .	20
4.3.5 Projekttagebuch Robin Menzel . . . . .	21
4.3.6 Projekttagebuch Ruthild Gilles . . . . .	22
4.3.7 Projekttagebuch Sertan Cetin . . . . .	24
4.3.8 Projekttagebuch Yannick Rüttgers . . . . .	25
4.4 Beschreibung von Problemen . . . . .	26
<b>5 Dokumentation der Software</b>	<b>27</b>
5.1 Dokumentation der Paketstruktur (Sertan Cetin) . . . . .	27
5.2 Dokumentation der Activities . . . . .	27

5.2.1	Main Activity . . . . .	27
5.3	Dokumentation der Navigation zwischen Activities (Yannick Rüttgers)	37
5.4	Dokumentation der Activity-übergreifenden, persistenten Datenhaltung (Jan Beilfuß) . . . . .	37
5.5	Dokumentation der Activity-übergreifenden Klassen (Ruthild Gilles) . .	38
5.5.1	Models-Klassen . . . . .	38
5.5.2	Service-Klassen . . . . .	38
5.6	Kapitel mit Abbildung . . . . .	39
<b>6</b>	<b>Fazit der Teammitglieder</b>	<b>41</b>
6.1	Fazit von Fabia Schmid . . . . .	41
6.2	Fazit von Florian Rath . . . . .	42
6.3	Fazit von Jan Beilfuß . . . . .	42
6.4	Fazit von Joscha Nassenstein . . . . .	42
6.5	Fazit von Robin Menzel . . . . .	42
6.6	Fazit von Ruthild Gilles . . . . .	42
6.7	Fazit von Sertan Cetin . . . . .	43
6.8	Fazit von Yannick Rüttgers . . . . .	44
<b>7</b>	<b>Quellenverzeichnis</b>	<b>45</b>
7.1	Unterkapitelüberschrift . . . . .	45
7.1.1	Unterunterkapitelüberschrift . . . . .	45
<b>8</b>	<b>Anhang - Quelltexte</b>	<b>46</b>
8.1	Activities . . . . .	46
8.2	Data . . . . .	46
8.2.1	Service Klassen (Ruthild Gilles) . . . . .	46
8.3	Overview . . . . .	48
8.4	Main Activity . . . . .	48
<b>9</b>	<b>Anhang - Verwendete Tools und Hilfsmittel (Robin Menzel)</b>	<b>49</b>

**Ehrenwörtliche Erklärung**

## Abbildungsverzeichnis

Abbildung 1: Die Angry Nerds . . . . .	1
Abbildung 2: Projektstrukturplan . . . . .	5
Abbildung 3: Meilensteinplan . . . . .	5
Abbildung 4: GANTT-Diagramm . . . . .	6
Abbildung 5: Geplante Aufgabenverteilung . . . . .	6
Abbildung 6: Mockups - Übersicht über alle TENs . . . . .	7
Abbildung 7: Mockups - Übersicht über ein vorhandenes und ein neues Event	8
Abbildung 8: Mockups - Übersicht über ein vorhandene, eine leere und eine Bild-Notiz . . . . .	9
Abbildung 9: Mockups - Übersicht über vorhandene und neue Todos . . . . .	10
Abbildung 10: Klassendiagramm . . . . .	12
Abbildung 11: Systemkontextdiagramm . . . . .	13
Abbildung 12: CRUD-Klassen . . . . .	14
Abbildung 13: Abbildungsbeschriftung . . . . .	15
Abbildung 14: Tatsächliche Aufgabenverteilung . . . . .	16
Abbildung 15: Overview Activity . . . . .	28
Abbildung 16: Landscape Ansicht - Overview Activity . . . . .	29
Abbildung 17: Note Fragments . . . . .	30
Abbildung 18: Todo Fragments . . . . .	31
Abbildung 19: Event Fragments . . . . .	32
Abbildung 20: Overview Activity - Löschen . . . . .	34
Abbildung 21: Overview Activity - Suchen . . . . .	36
Abbildung 22: Abbildungsbeschriftung . . . . .	40

## **Listingverzeichnis**

Listing 1: Create Klasse (Ruthild Gilles) . . . . .	46
Listing 2: Read Klasse (Ruthild Gilles) . . . . .	47
Listing 3: Update Klasse (Ruthild Gilles) . . . . .	48
Listing 4: Delete Klasse (Ruthild Gilles) . . . . .	48

## 1 Das Team: Angry Nerds

**Abbildung 1:** Die Angry Nerds



Von links: Florian Rath, Sertan Cetin, Jan Beilfuß, Joscha Nassenstein, Ruthild Gilles, Yannick Rüttgers, Fabia Schmid, Robin Menzel

## 2 Ziele des Projektes (Fabia Schmid)

Das Projekt „TEN Manger“ umfasst die Planung und Entwicklung einer Applikation zur Erstellung und Verwaltung von ToDos, Events und Notes. Diese Applikation stellt die Prüfungsleistung für das Modul „Projekte der Wirtschaftsinformatik“ dar.

Die Umsetzung erfolgt durch das Team „Angry Nerds“, welches sich aus Ruthild Gilles, Fabia Schmid, Jan Beilfuß, Yannick Rüttgers, Robin Menzel, Florian Rath, Joscha Nassenstein und Sertan Cetin zusammensetzt.

Der Zeitrahmen für die Realisierung des Projektes erstreckt sich vom 05.09.2017 bis zum 07.02.2018. Die Organisation, wie die Vereinbarung von Meetings und die konkrete Aufgabenverteilung erfolgen selbstständig innerhalb des Teams.

Die Grundlage für das Projekt bilden sowohl Vorkenntnisse aus vorherigen Vorlesungen von den Teammitgliedern erworben wurden, sowie die Einführung in die Android Programmierung im Rahmen der Vorlesung "Projekte der Wirtschaftsinformatik".

Ziel des Projektes ist die Entwicklung einer Applikation in der Programmiersprache Java. Die Applikation soll auf einem Tablet mit dem Betriebssystem Android laufen und die TEN Verwaltung ermöglichen. Die Verwaltung soll die Erstellung von ToDos, Events und Notes ermöglichen, sowie die Verwaltung dieser. Die Verwaltung soll aus dem Anzeigen, Filtern, Bearbeiten und Löschen bestehen.

Neben der Applikation soll im Rahmen des Projektes noch eine ausführliche Dokumentation angefertigt werden, welche den ganzen Projektverlauf und das Endergebnis dokumentiert und erklärt. Diese muss fristgerecht mit der Applikation abgegeben werden.

Um das Ziel des Projektes zu erreichen wurden verschiedene Termine festgelegt, welche im Folgenden aufgelistet werden:

- 12.09.2018 – Abgabe Projekttagebuch
- 31.10.2018 – Abgabe Projekttagebuch
- 19.12.2018 – Abgabe Projekttagebuch
- 07.02.2019 – Abgabe Dokumentation per Mail
- 09.02.2019 – Vorstellung der Applikation und Abgabe der Applikation

Das Projekt kann nur als erfolgreich bezeichnet werden, wenn alle Termine fristgerecht erfüllt wurden.

## 3 Projektplanung

### 3.1 Beschreibung des Funktionsumfangs (Florian Rath)

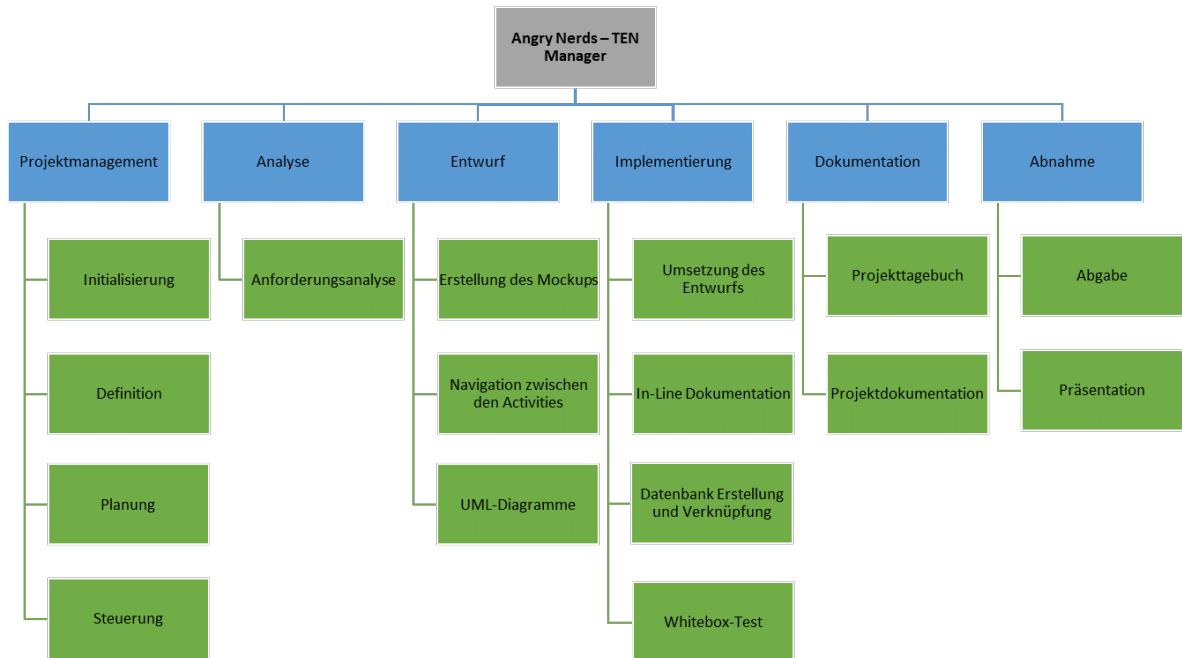
Hier den Text einfach hin kopieren.

### 3.2 Projektablaufplan (Fabia Schmid)

Als Projektleiterin wählte ich als Vorgehensmodell für die Entwicklung das erweiterte Wasserfallmodell fest. Dafür sprach die klare Struktur des Modelles und der geringe Managementaufwand. Zusätzlich war die Übersichtlichkeit, sowie die leichte Verständlichkeit ein klarer Vorteil. Außerdem sprach für das erweiterte Wasserfallmodell, dass es für kleine Projekte mit festgelegten Umfang ausgelegt und geeignet ist.

Auf diesem Vorgehensmodell basierte die Projektstrukturplanung, die Meilensteinplanung und die Zeitplanung, welche in einem GANTT-Diagramm visualisiert wurde.

Abbildung 2: Projektstrukturplan



Quelle: Erstellt von Fabia Schmid

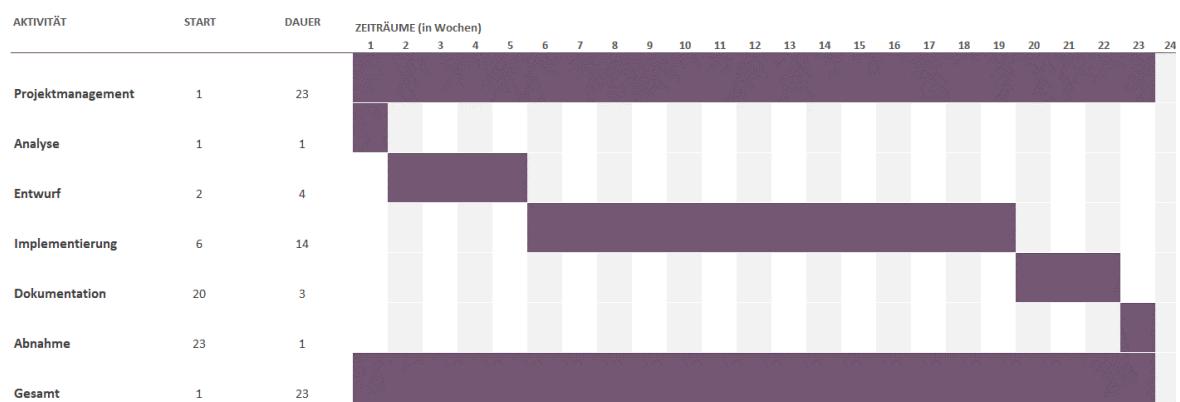
Abbildung 3: Meilensteinplan



Quelle: Erstellt von Fabia Schmid

Abbildung 4: GANTT-Diagramm

### Angry Nerds - GANTT Diagramm



Quelle: Erstellt von Fabia Schmid

Abbildung 5: Geplante Aufgabenverteilung



Quelle: Erstellt von Fabia Schmid

### 3.3 Planung der Software

#### 3.3.1 Planung des Mock-Ups (Robin Menzel)

##### Übersicht

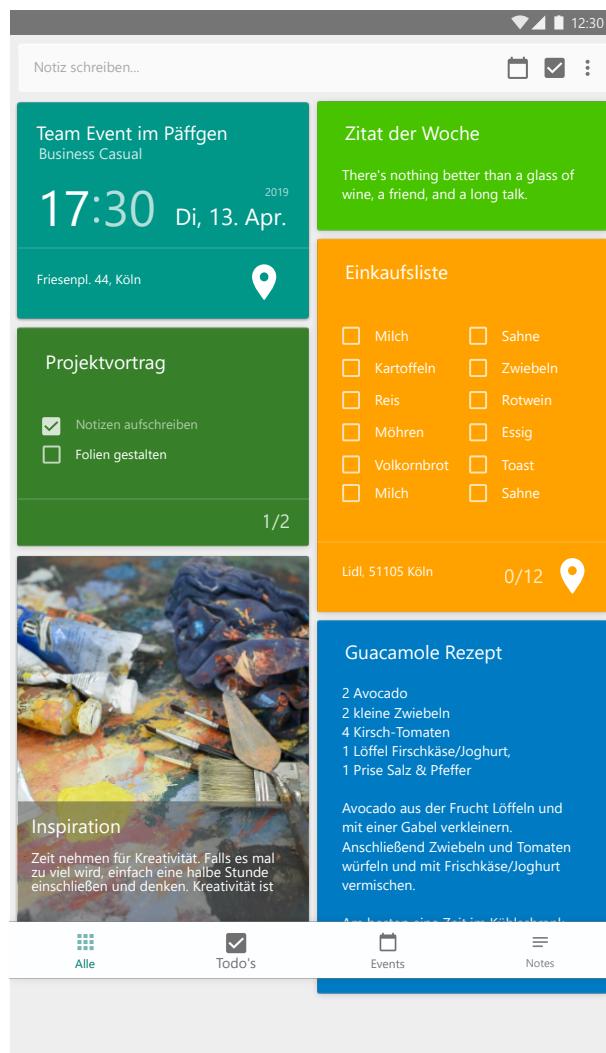
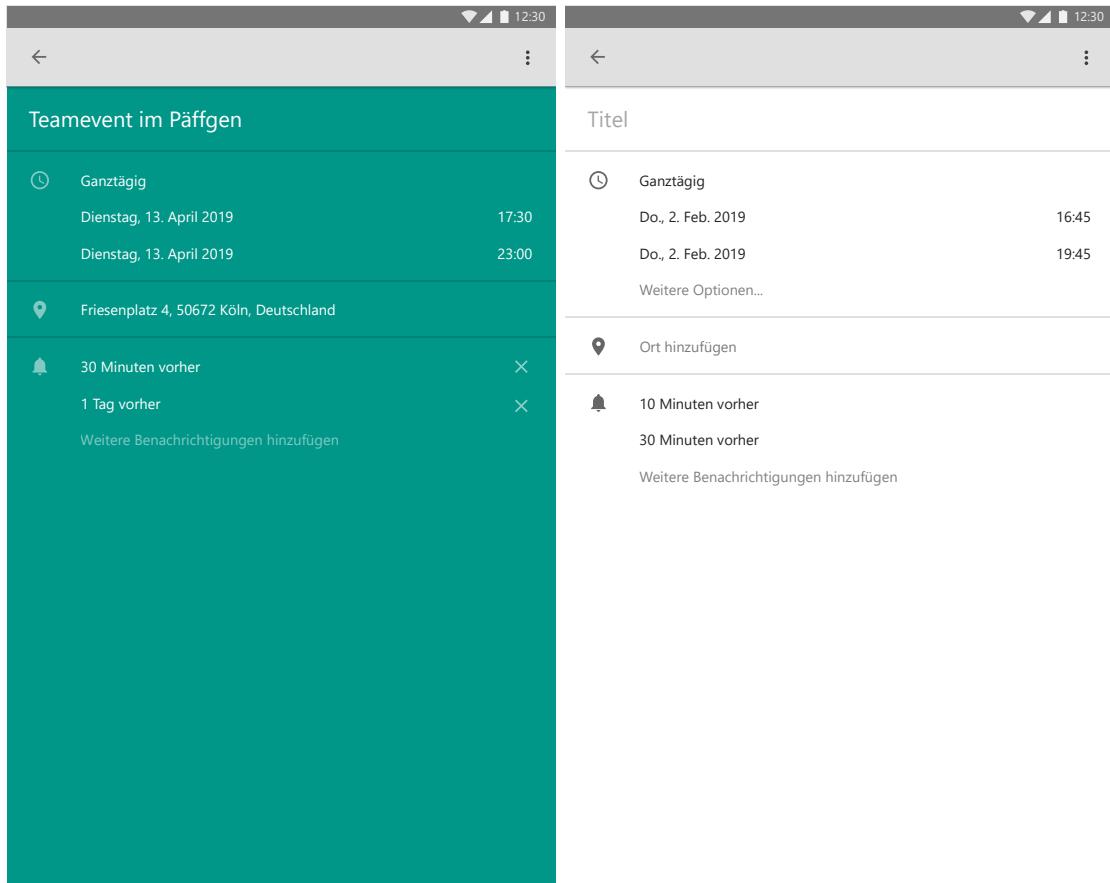


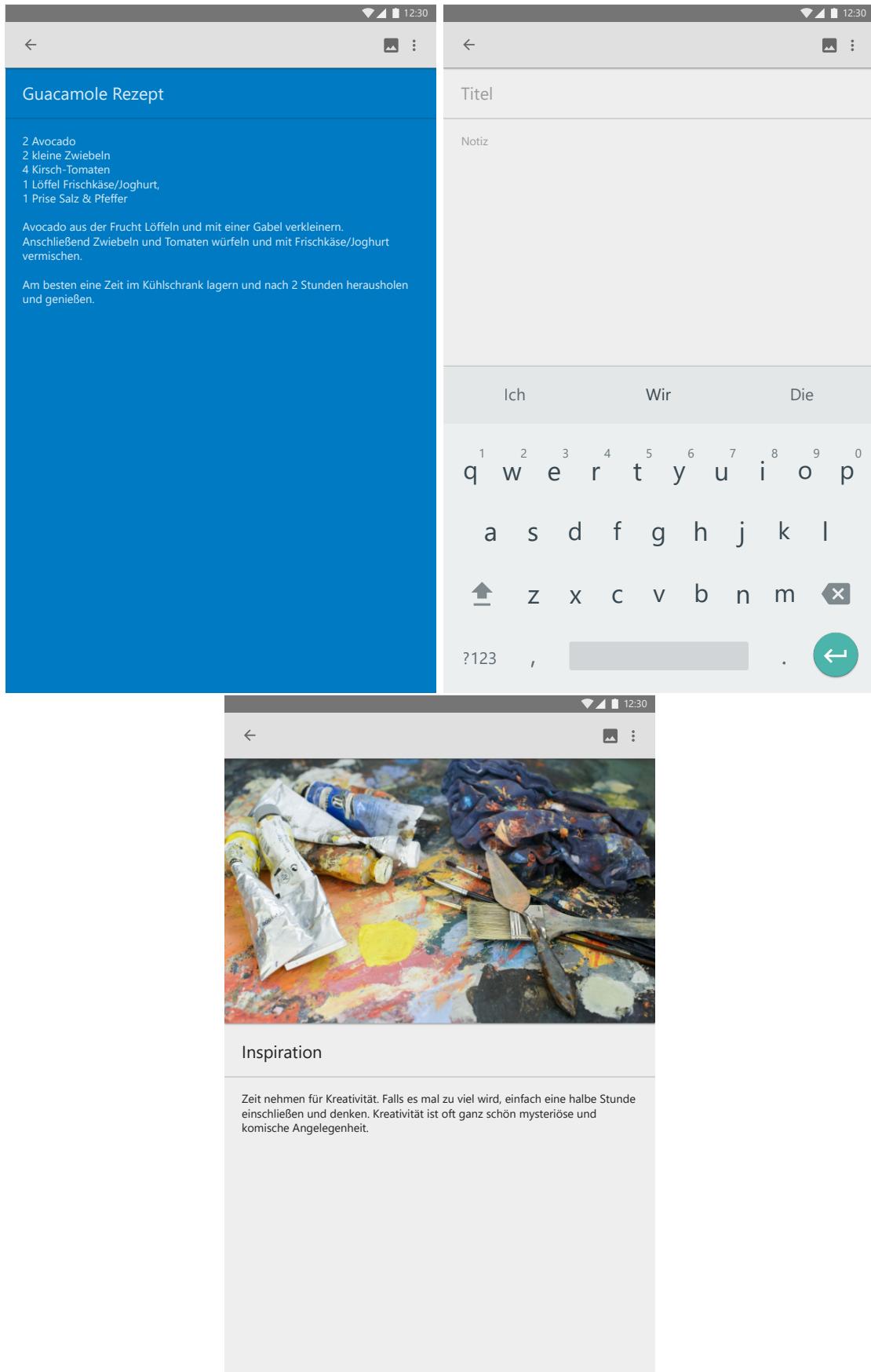
Abbildung 6: Mockups - Übersicht über alle TENs

## Event Activity - Unsere Events



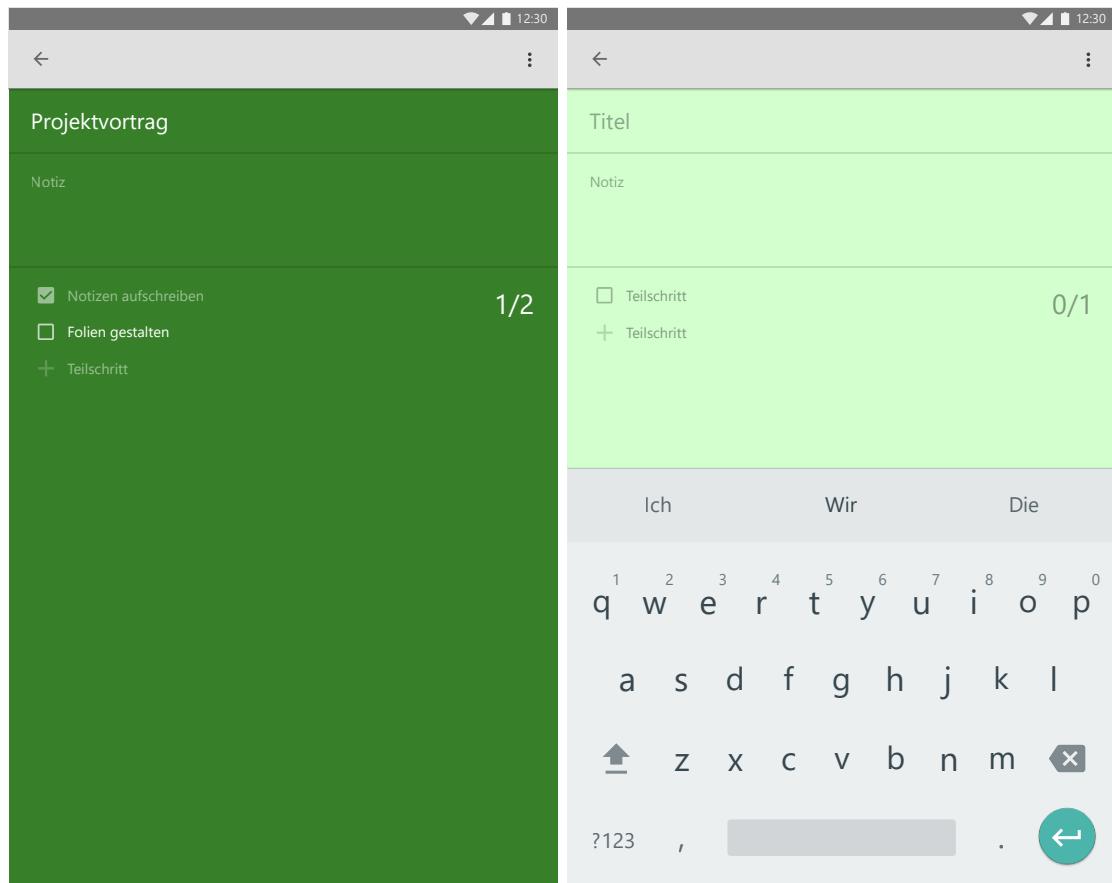
**Abbildung 7:** Mockups - Übersicht über ein vorhandenes und ein neues Event

## Note Activity - Unsere Notizen



**Abbildung 8:** Mockups - Übersicht über ein vorhandene, eine leere und eine Bild-Notiz

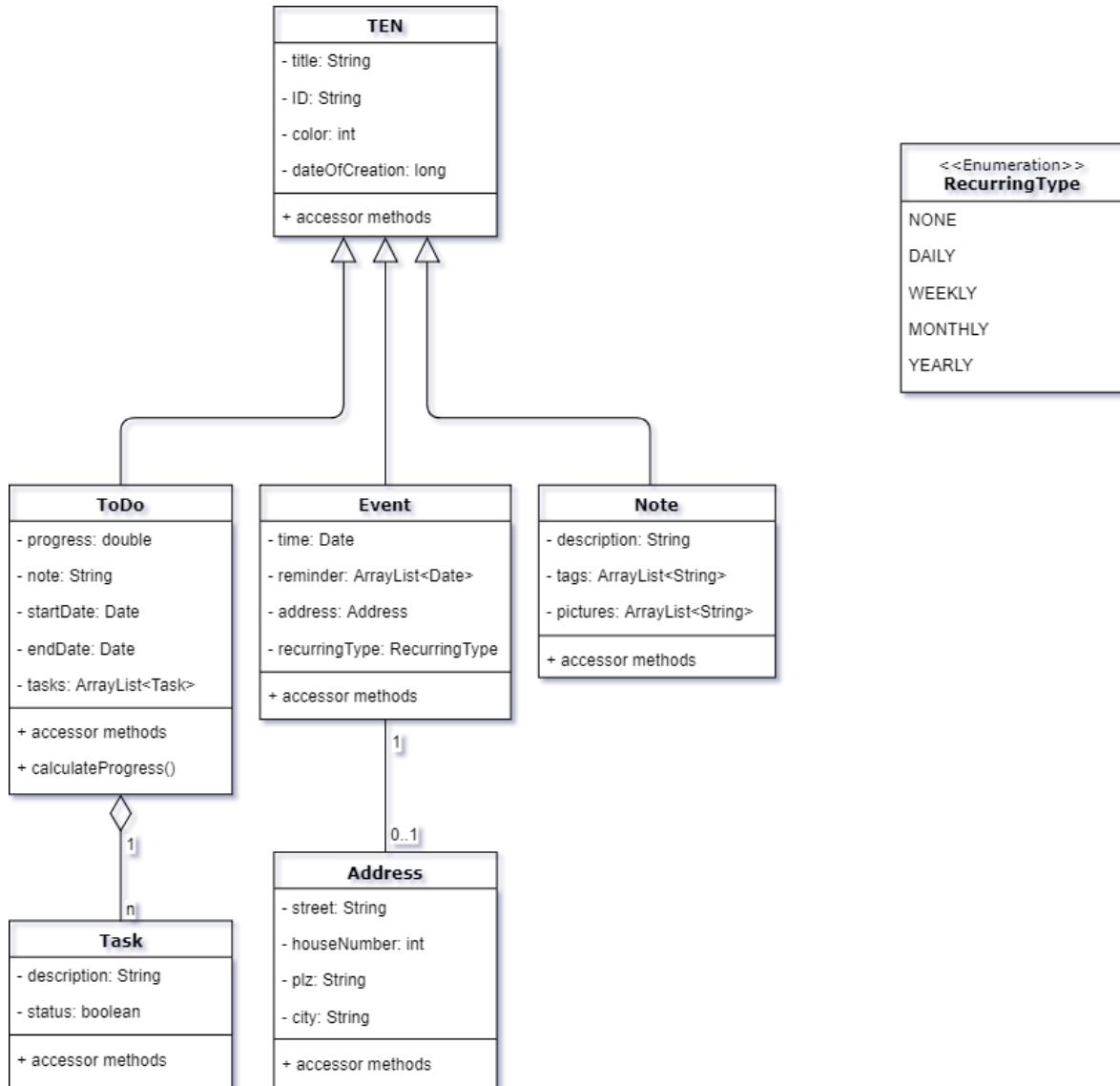
## Todo Activity - Unsere ToDos



**Abbildung 9:** Mockups - Übersicht über vorhandene und neue Todos

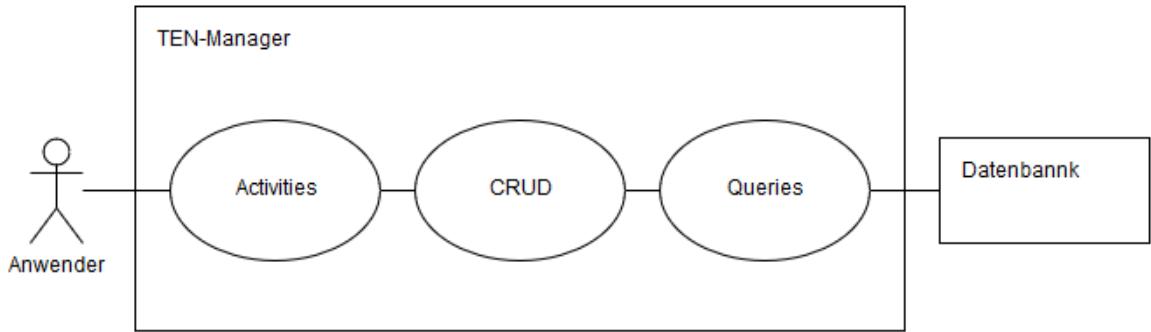
### **3.3.2 Planung der Datenstruktur und Schnittstellen (Ruthild Gilles)**

Die gewünschte Applikation soll das Managen von Todos, Events und Notes vereinfachen. Anhand der Anforderungen an die Applikation überlegte sich das Datenteam, welche Daten beziehungsweise Informationen in der Datenstruktur der Applikation abgebildet werden sollen. Da sowohl Todo-, Event-, als auch Note-Objekte einheitlich aufgebaut sein sollen, wurde sich dazu entschieden, dass die jeweiligen Klassen von einer TEN-Klasse erben. Alle Todo-, Event- und Note-Objekte benötigen eine ID zu eindeutigen Identifikation des Objektes auf der Datenbank und in der Applikation. Außerdem könne die Objekte jeweils einen Titel haben. Zudem sollen die verschiedenen Objekte weitere Informationen enthalten. In folgendem Klassendiagramm sind alle geplanten Attribute der Klassen aufgelistet.

**Abbildung 10:** Klassendiagramm

**Quelle:** Erstellt von Joscha Nassenstein

Zusätzlich zu der Struktur der Daten in Form von Klassen mit entsprechenden Attributten wurde ebenfalls die Struktur der Applikation vom Datenteam definiert. Diese ist in nachfolgender Abbildung in einem Systemkontextdiagramm dargestellt.

**Abbildung 11:** Systemkontextdiagramm

**Quelle:** Erstellt von Ruthild Gilles

Um die Daten auch nach Beendigung der Applikation bei erneutem Starten wieder anzeigen zu können, wurde eine dokumentenbasierte Datenbank an die Applikation angebunden. Auf diese Weise kann eine persistente Datenhaltung erzielt werden. Die einzelnen Activities, welche als Schnittstelle zu den Anwendern dienen, sollen die vom Benutzer eingegebenen Informationen auf der Datenbank speichern können. Dazu sollen Activity-übergreifende Klassen verwendet werden.

Das Datenteam plante die Activity-übergreifenden Klassen und deren Methoden anhand der Anforderungen, der einzelnen Activities. Es sollte möglich sein, einzelne oder auch alle TEN-Objekte von der Datenbank zu erhalten. Auch sollte das Löschen und das Speichern von einzelnen TEN-Objekten möglich sein. Während der Planungsphase wurden hier verschiedene Ansätze in Erwägung gezogen, um diese Anforderungen umzusetzen. Zur Übersichtlichkeit entschied sich das Datenteam letztendlich dafür, einzelne Klassen für jede der vier CRUD-Operationen zu erstellen. Die CRUD-Operationen beinhalten das Erstellen (Create), das Lesen (Read), das Aktualisieren (Update) und das Löschen (Delete) von einzelnen Objekten. Die einzelnen Methoden der CRUD-Klassen sind in folgender Abbildung dargestellt.

**Abbildung 12:** CRUD-Klassen

Create	Read	Update	Delete
<ul style="list-style-type: none"> <li>+ newTodo(): Todo</li> <li>+ newEvent(): Event</li> <li>+ newNote(): Note</li> </ul>	<ul style="list-style-type: none"> <li>+ getAllTENs(): ArrayTENs</li> <li>+ getTodoByID(String): Todo</li> <li>+ getEventByID(String): Event</li> <li>+ getNoteByID(String): Note</li> </ul>	<ul style="list-style-type: none"> <li>+ saveTEN(TEN):</li> </ul>	<ul style="list-style-type: none"> <li>+ deleteTEN(TEN):</li> <li>+ deleteMultipleTENs(ArrayTENs):</li> </ul>

**Quelle:** Erstellt von Ruthild Gilles

Da der Aufwand für die Umsetzung aller geforderten Anforderungen zu Projektstart lediglich grob geschätzt werden konnte, definierte das Datenteam abgesehen von der Schnittstelle zur Datenbank noch einige weitere Schnittstellen. Die Implementierung dieser weiteren Schnittstellen wurde nicht in den Anforderungen gefordert und würde nur bei genug Zeitüberschuss umgesetzt werden. Zu den weiteren optionalen Schnittstellen gehören das Exportieren von Todos, Events und Notes in die Zwischenablage oder auch in andere Applikationen, die auf dem entsprechenden Endgerät installiert sind. Für ein Event soll es die Möglichkeit geben eine Adresse hinzuzufügen. Hier wäre eine weitere optionale Schnittstelle die Verknüpfung mit Google Maps. Auch könnte eine Schnittstelle zu einer anderen Kalender App implementiert werden, in die ein Event exportiert werden könnte.

### 3.3.3 Planung der Activities und Layouts (Florian Rath)

Hier den Text einfach hin kopieren.

### 3.3.4 Planung der Navigation zwischen den Activities (Yannick Rüttgers)

Hier den Text einfach hin kopieren.

## 3.4 Geplante Aufgabenverteilung im Team (Fabia Schmid)

Hier den Text einfach hin kopieren.

So kann man Abbildungen einfügen:

**Abbildung 13:** Abbildungsbeschriftung



Quelle: <http://dominique-fleury.com/?p=302>

## 4 Beschreibung des Projektverlaufs

### 4.1 Tatsächliche Aufgabenverteilung im Team (Fabia Schmid)

Abbildung 14: Tatsächliche Aufgabenverteilung



Quelle: Erstellt von Fabia Schmid

Hier den Text einfach hin kopieren.

### 4.2 Teammeetingprotokolle

Hier den Text einfach hin kopieren.

### 4.3 Projekttagebücher aller Teammitglieder

#### 4.3.1 Projekttagebuch Jan Beilfuß

Beschreibung	Dauer	Datum
--------------	-------	-------

#### 4.3.2 Projekttagebuch Joscha Nassenstein

Beschreibung	Dauer	Datum
--------------	-------	-------

### 4.3.3 Projekttagebuch Fabia Schmid

Beschreibung	Dauer	Datum
Aufgabe lesen und verstehen	30 min	05.09.2018
Kick-Off Meeting zur Besprechung der Aufgabe, Verteilung der Rollen	50 min	05.09.2018
Erstellung eines Meilensteinplans	50 min	05.09.2018
Vorstellung des Meilensteinplans und Besprechung der anderen Ergebnisse	40 min	05.09.2018
Aufarbeitung der Abgabetermine und Information der Gruppenteilnehmer	20 min	10.09.2018
Installation von Latex	120 min	10.09.2018
Festlegung des Vorgehensmodells, durch Abwägung von Vor- und Nachteilen der verschiedenen Modelle (Ergebnis: Erweitertes Wasserfallmodell)	40 min	11.09.2018
Teammeeting	60 min	22.09.2018
Github Anmeldung und Einbindung des Projektes	60 min	13.10.2018
Entwicklung des Layouts für das „Event“	70 min	20.10.2018
Entwicklung des Layouts für das „Note“	60 min	21.10.2018
Recherche über Listen, Checkboxen und Verwendung von mehreren Layouts	50 min	21.10.2018
Besprechung der Layouts (Aufbau, etc.)	50 min	27.10.2018
Entwicklung des Layouts für das „ToDo“	40 min	27.10.2018
Erstellung des Projekttagebuch mit Latex	40 min	30.10.2018
Zusammentragung der momentanen Projektstände und Abschätzung, ob die Meilensteine erreicht werden können	30 min	17.11.2018
Neuplanung eines Meilensteins und Abstimmung mit dem Team	20 min	26.11.2018
Teammeeting	50 min	04.12.2018
Koordination der anzufertigen Diagramme und Entwicklungsstand überprüfen	10 min	04.12.2018
Entwicklung des Layouts für das „Image“	20 min	12.12.2018
Entwicklung des Layouts für die Overview	60 min	02.01.2019
Aufteilung der Ausarbeitung	20 min	02.01.2019
Entwicklung des Layouts für das Bedienleisten-Fragment	20 min	04.01.2019

## Beschreibung des Projektverlaufs

---

Meilenstein Umplanung	10 min	15.01.2019
Erinnerung an die Erstellung der Kapitel der Ausarbeitung	10 min	23.01.2019
Dokumentation	240 min	03.02.2019

Summe in Minuten: 1270

#### 4.3.4 Projekttagebuch Florian Rath

Beschreibung	Dauer	Datum
--------------	-------	-------

#### 4.3.5 Projekttagebuch Robin Menzel

Beschreibung	Dauer	Datum
--------------	-------	-------

#### 4.3.6 Projekttagebuch Ruthild Gilles

Beschreibung	Dauer	Datum
Aufgabe lesen und verstehen	30 min	05.09.2018
Kick-Off Meeting zur Besprechung der Aufgabe, Verteilung der Rollen	50 min	05.09.2018
Festlegung des Umfangs (Muss/Kann Kriterien)	50 min	05.09.2018
Besprechung der in Aufgabenteilung entstandenen Ergebnisse	40 min	05.09.2018
Installation von LaTeX	120 min	10.09.2018
Projekttagebuch ausfüllen	10 min	11.09.2018
Erstellung des Datenmodells	50 min	15.09.2018
Teammeeting	60 min	22.09.2018
Überlegung Aufgabenteilung, Aufgabenvergabe und Erklärung dieser bezogen auf die MainActivity	40 min	09.10.2018
Installation und Einrichtung von GIT	30 min	13.10.2018
Teammeeting Data-Team	60 min	26.10.2018
Klonen des Data-Branches von GIT	120 min	27.10.2018
Deklaration von gettern und settern für Datenobjekte	120 min	28.10.2018
Projekttagebuch ausfüllen	20 min	31.10.2018
Entwicklung von SetterService Klasse	180 min	16.11.2018
Teammeeting Data-Team	100 min	20.11.2018
Entwicklung von Service Klassen	120 min	21.11.2018
Überlegung neuer Struktur für Services	90 min	22.11.2018
Implementierung neuer Service-Struktur	120 min	30.11.2018
Einbindung der Mockdaten	80 min	01.12.2018
Implementierung weiterer Teile neuer Struktur	90 min	02.12.2018
Änderungen an neuer Service-Struktur	40 min	03.12.2018
Teammeeting	50 min	04.12.2018
Entwicklung von Create-Klasse	60 min	04.12.2018
Änderungen an Update-Klasse	40 min	17.12.2018
Projekttagebuch ausfüllen	20 min	18.12.2018
Bugfixing in Update- und Read-Klasse	90 min	02.01.2019
Mockdaten durch Zugriff auf Datenbank ersetzt	60 min	02.01.2019

Ergänzung einer Methode in Delete-Klasse	30 min	04.01.2019
Neue Farben für GUI festlegen	30 min	07.01.2019
Latex - Vorlage anpassen	120 min	14.01.2019
Latex - Struktur für Ausarbeitung erstellen	120 min	18.01.2019
Meinen Teil der Ausarbeitung schreiben	180 min	28.01.2019
Latex - Meetingprotokolle einfügen	180 min	01.02.2019
Latex - Quellcode einfügen	120 min	03.02.2019
Latex - Ausarbeitungen der anderen einfügen	180 min	03.02.2019
Projekttagebuch ausfüllen	20 min	03.02.2019
Latex - Projekttagebücher einfügen	120 min	03.02.2019

Summe in Minuten: 3090

#### 4.3.7 Projekttagebuch Sertan Cetin

Beschreibung	Dauer	Datum
--------------	-------	-------

#### 4.3.8 Projekttagebuch Yannick Rüttgers

Beschreibung	Dauer	Datum
--------------	-------	-------

#### **4.4 Beschreibung von Problemen**

Im Lauf des Projektes konnten zwei Meilensteine nicht fristgerecht erfüllt werden. Einmal die Fertigstellung der Activities und die Fertigstellung der Dokumentation.

Die Fertigstellung der Activities und der Layouts war für den 01.12.2018 geplant, konnte jedoch nicht fristgerecht fertiggestellt werden, da projektfremde Tätigkeiten die Umsetzung verzögerten. Beispielsweise schrieben wir eine Klausur, die bei der Projektplanung noch nicht bekannt war. Als Ergebnis wurde der Meilenstein auf ein späteres Datum geplant und konnte danach fristgerecht erfüllt werden. Diese Verschiebung hatte jedoch keinen negativen Einfluss auf die Fertigstellung des Projektes.

Auch der Meilenstein „Dokumentation fertig“ konnte nicht fristgerecht zum 01.02.2019 erreicht werden. Auch in diesem Fall waren projektfremde Tätigkeiten die Ursache für den Verzug. Jedoch musste der Meilenstein nur um 3 Tage verschoben werden, wodurch das Projektergebnis nicht gefährdet wurde.

Weitere Probleme oder Verzögerungen traten nicht auf und das Projekt konnte erfolgreich am 09.02.2019 vorgestellt werden.

## 5 Dokumentation der Software

### 5.1 Dokumentation der Paketstruktur (Sertan Cetin)

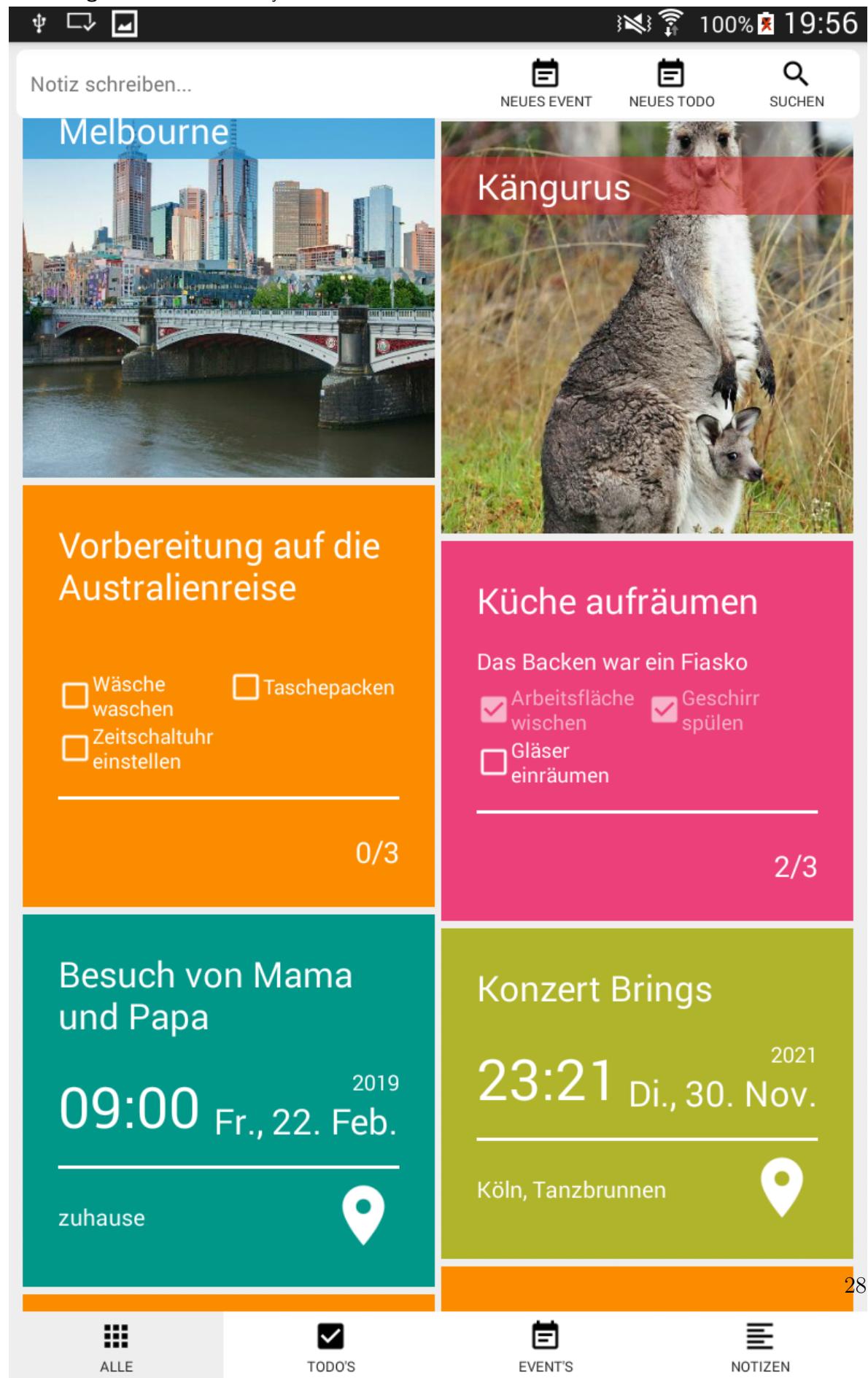
### 5.2 Dokumentation der Activities

#### 5.2.1 Main Activity

##### Layouts (Fabia Schmid)

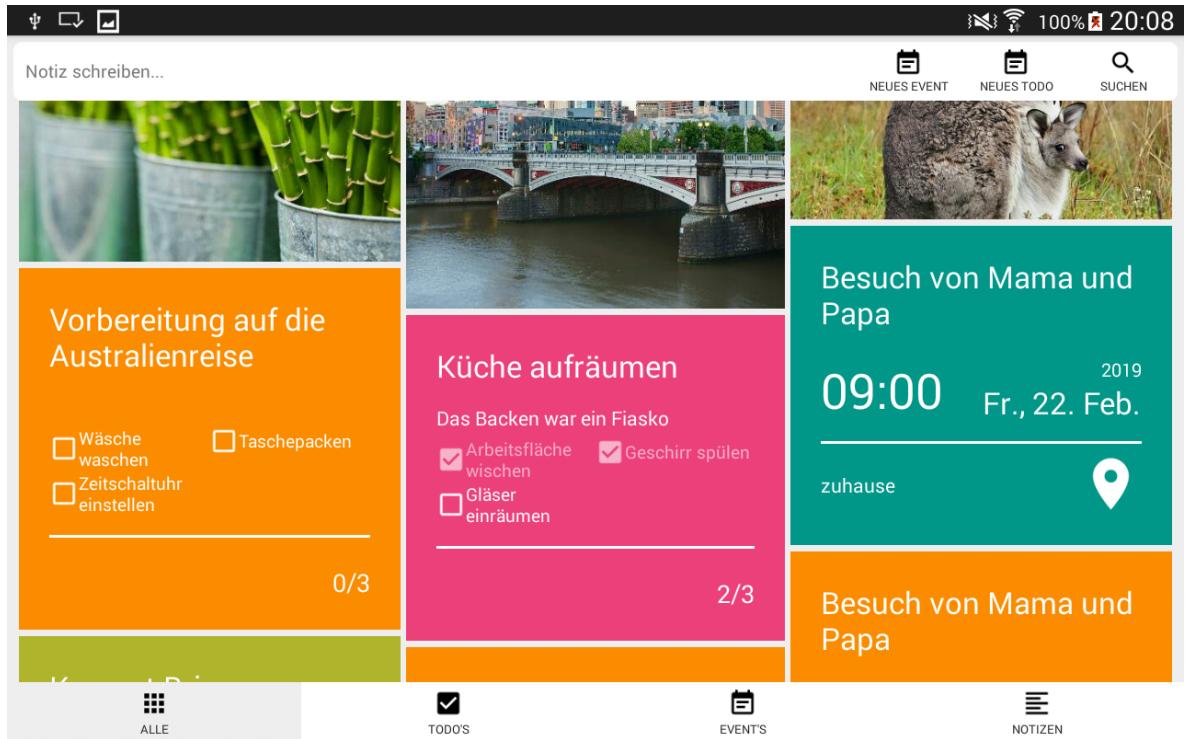
Das Layout der Overview Activity orientiert sich an dem vorher erstellten Mockup und besteht aus drei Grundkomponenten. Es gibt am oberen Bildschirmrand eine Leiste zum Erstellen von Notes, Events und ToDos, sowie ein Button zum Suchen. Unter dieser Leiste befindet sich ein Container, welcher die verschiedenen Fragments der vorhandenen Notes, Events und ToDos beinhaltet. Am unteren Bildschirmrand befindet sich noch eine Leiste mit vier Buttons zum Filtern der Fragments.

Abbildung 15: Overview Activity



Die Activity Overview zeigt normal 2 Spalten mit Fragments, wenn jedoch das Tablet gedreht wird zeigt die Landscape-Ansicht drei Spalten mit Fragments, um den Bildschirm optimal zu nutzen.

**Abbildung 16:** Landscape Ansicht - Overview Activity

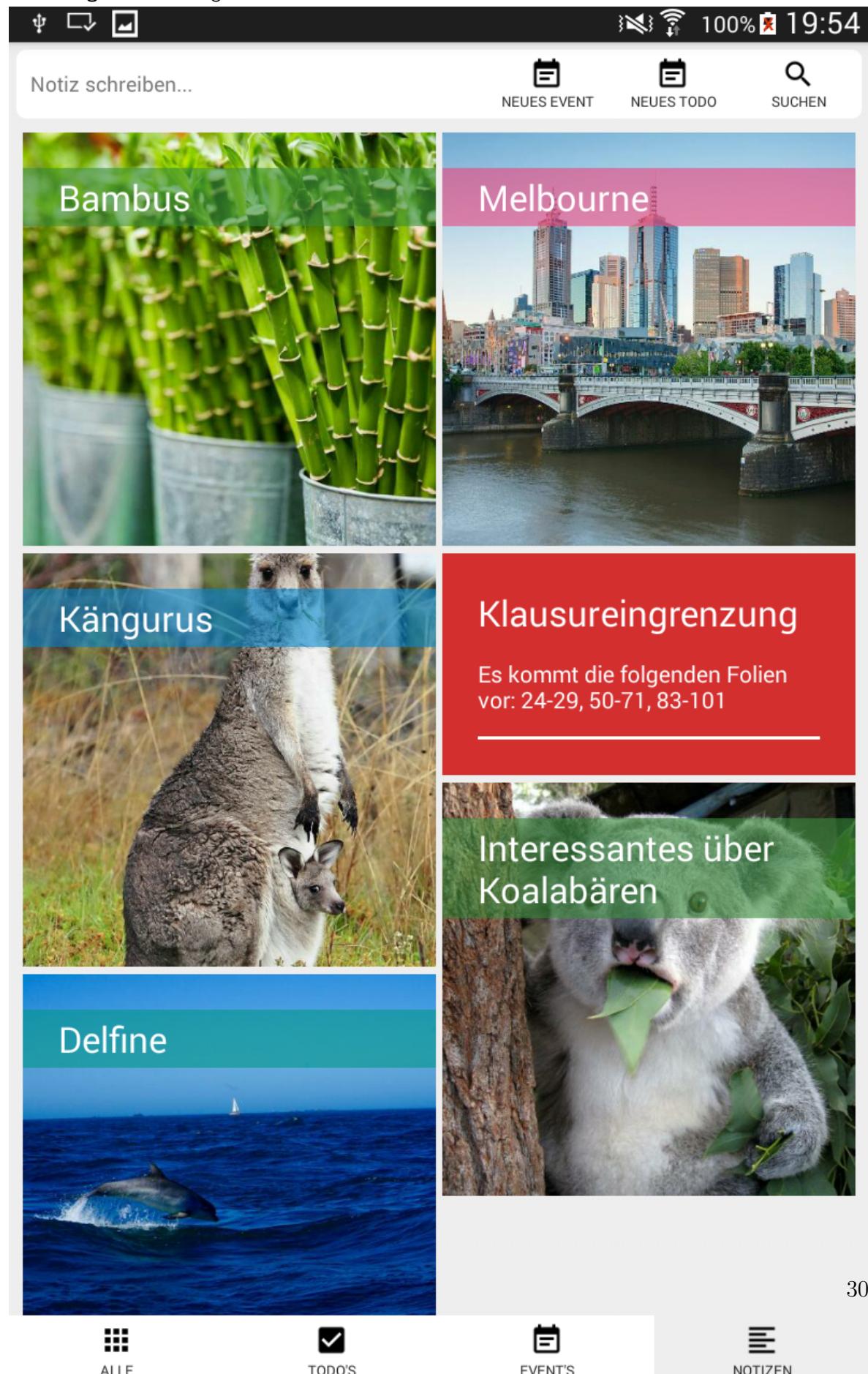


**Quelle:** Screenshot aus der Benutzeroberfläche

Die drei verschiedenen Arten von Fragments sind Notes, ToDo und Event.

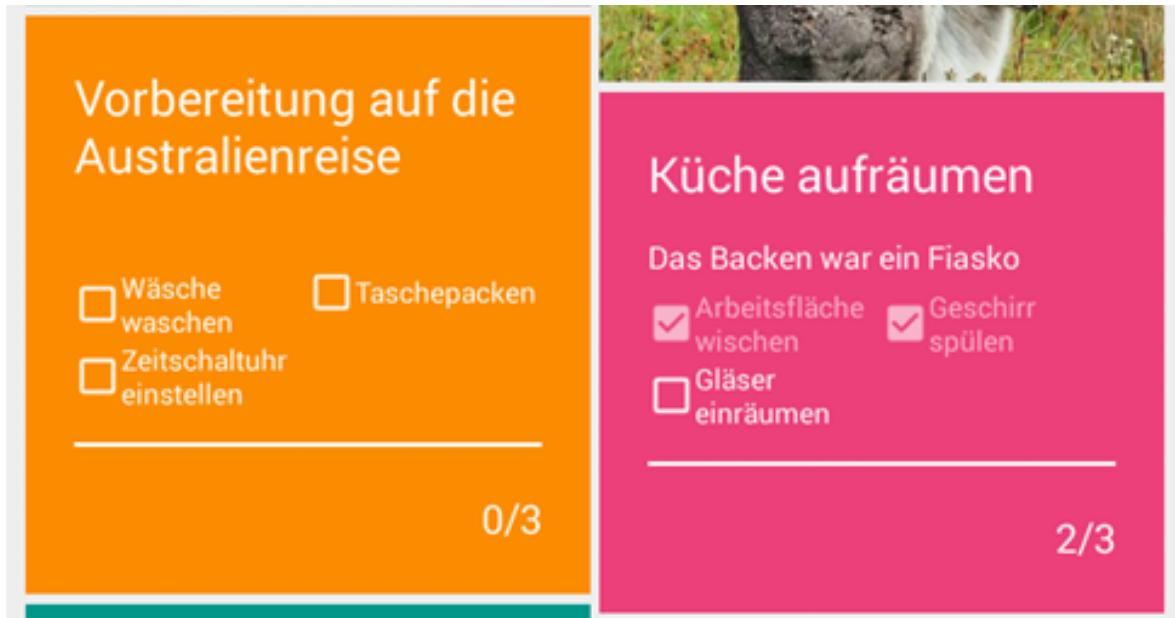
Bei einer Note wird die Überschrift und ein Textauszug angezeigt. Wenn jedoch ein Bild in der Notiz vorhanden ist wird dieses mit der Überschrift in der Overview Activity angezeigt.

Abbildung 17: Note Fragments



Die Fragments für die ToDos, zeugen auch den Titel an und die einzelnen Aufgaben mit Kästchen, die mit einem Hacken gefüllt sind, wenn sie erledigt sind. Zusätzlich wird Angezeigt, wie viele Aufgaben schon erfüllt wurden.

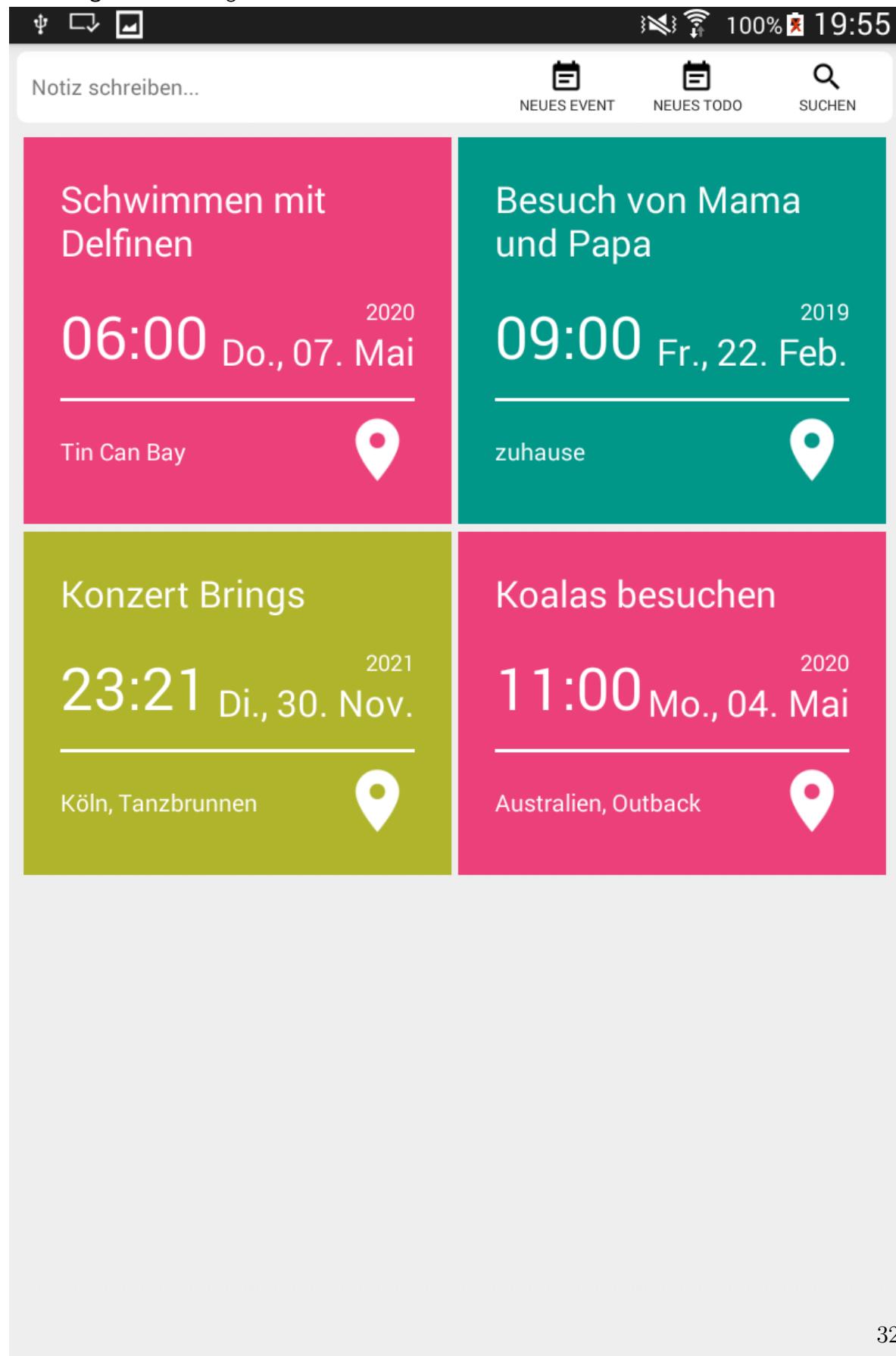
**Abbildung 18:** Todo Fragments



**Quelle:** Screenshot aus der Benutzeroberfläche

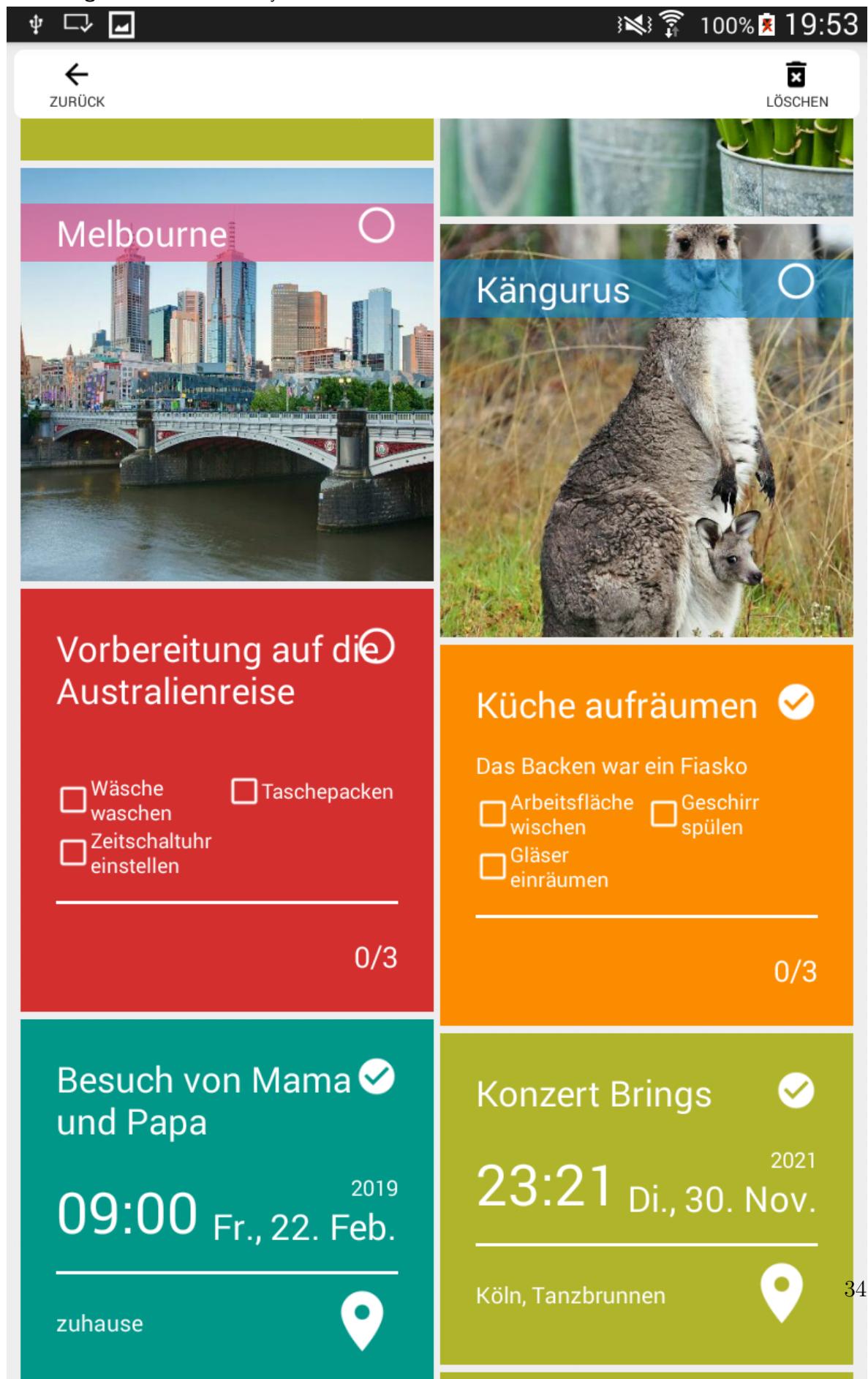
Die Event Fragmentes bestehen aus der Überschrift, dem Datum und der Uhrzeit, sowie aus einem Ort, der angegeben werden kann.

Abbildung 19: Event Fragments



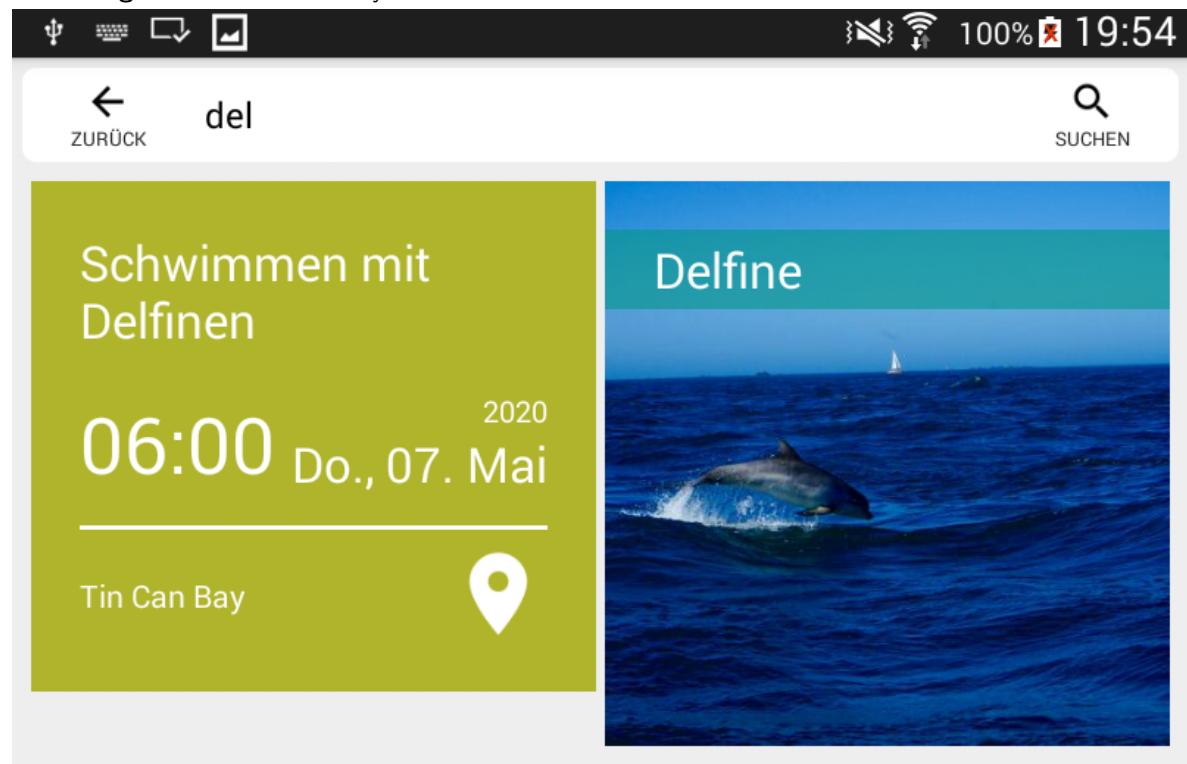
In der Overview Activity kann zusätzlich gesucht und gelöscht werden. Wenn man einen langen Click auf ein Fragment macht, wird die obere Leiste verändert und man bekommt einen Button zum Löschen. Zusätzlich kann in dieser Ansicht nun auf die verschiedenen Fragments geklickt werden, um diese zu markieren und mehrere gleichzeitig zu löschen. Diese Ansicht kann durch den Zurück-Pfeil verlassen werden.

Abbildung 20: Overview Activity - Löschen



Auch bei der Suchfunktion wird die obere Leiste angepasst. Dabei kann nun ein beliebiges Wort eingegeben werden, welches dann in den verschiedenen Einträgen gesucht wird. Die gefundenen Einträge werden daraufhin angezeigt. Um die Suche zu beenden kann der Zurück-Pfeil verwendet werden.

Abbildung 21: Overview Activity - Suchen



### **5.3 Dokumentation der Navigation zwischen Activities (Yannick Rüttgers)**

Hier den Text einfach hin kopieren.

### **5.4 Dokumentation der Activity-übergreifenden, persistenten Datenhaltung (Jan Beilfuß)**

Hier den Text einfach hin kopieren.

## 5.5 Dokumentation der Activity-übergreifenden Klassen (Ruthild Gilles)

Zu den Activity-übergreifenden Klassen gehören abgesehen von den Query-Klassen für die persistente Datenhaltung auch die Service-Klassen und die Models-Klassen. Die Models-Klassen enthalten die TEN-Klasse und die einzelnen Todo-, Event- und Note-Klassen. Hier sind auch weitere Util-Klassen untergeordnet. In den Service-Klassen sind Methoden enthalten, die die Schnittstelle zwischen Datenbank und Activities darstellen. Im Folgendem wird genauer auf die einzelnen Klassen eingegangen.

### 5.5.1 Models-Klassen

### 5.5.2 Service-Klassen

Damit die Activities die Daten der TEN-Objekte auf der dokumentenbasierten Datenbank speichern können, wurden Service Klassen implementiert. Hierzu gehörten hauptsächlich Klassen zum Erzeugen neuer TEN-Objekte (Create), zum Erhalten bereits gespeicherter TEN-Objekte von der Datenbank (Read), zum Speichern veränderter TEN-Objekte (Update) und zum Löschen von TEN-Objekten von der Datenbank (Delete). Diese sogenannte CRUD-Operationen wurden in den entsprechenden Klassen teilweise für alle drei verschiedenen Objekttypen einzeln eingefügt, teilweise aber auch für TEN-Objekte im Allgemeinen. Dank Polymorphie können die jeweils einzelnen Objekttypen ebenfalls an die entsprechenden Methoden übergeben werden.

Diese Struktur wurde während der Planungsphase vom Datenteam überlegt und während der Implementierung angepasst. Wie auch schon in der Planungsphase definiert enthält die Create-Klasse Methoden, die ein neues leeres Todo, Event oder Note zurückgeben. Diese Methode ruft den Konstruktor der jeweiligen TEN-Klasse auf. Eine Interaktion mit der Datenbank ist hier noch nicht nötig.

Die Read-Klasse hingegen muss auf die Datenbank zugreifen, um entweder alle TEN-Objekte an die Main-Activity in einem ListArray zu übergeben oder aber ein spezielles Todo, Event oder Note, welches von den einzelnen Todo-, Event- oder Note-Activities aufgerufen werden kann. Damit das gewünschte TEN-Objekt in der Datenbank gefunden werden kann, benötigen die Read-Methoden die ID des gewünschten Objektes. Dieses wird als String beim Aufruf der jeweiligen Methode übergeben.

Die Update-Klasse dient zum Speichern von Änderungen an Todo-, Event- und Note-Objekten. Dazu überprüft die Methode, der ein TEN-Objekt übergeben wurde, ob dieses bereits auf der Datenbank existiert. Ist dies der Fall, wird eine Methode zum Ausführen des Update-Befehls auf der Datenbank aufgerufen. Ist dies nicht der Fall, wird eine Methode zum Ausführen des Insert-Befehls auf der Datenbank aufgerufen. Da die Todo-, Event- und Note-Klassen von der TEN-Klasse erben, kann hier Polymorphie angewandt werden. Es ist nur eine Methode zum Speichern notwendig.

Für das Löschen von TEN-Objekten sind in der Delete-Klasse zwei Methode vorhanden. Die eine löscht nur ein übergebenes TEN-Objekt aus der Datenbank, indem es eine entsprechende Methode aus einer der Repository-Klassen aufruft und dieser die ID des TEN-Objektes übergibt. Sollen jedoch mehrere TEN-Objekte auf einmal gelöscht werden, kann der zweiten Methode eine Array List, die mehrere zu löschen Objekte enthält, übergeben werden. Dabei müssen die Objekte nicht alle von dem gleichen Datentyp sein, sondern können Todo-, Event- und Note-Objekte enthalten. Die Methode iteriert durch die übergebene Array List und ruft für jeden Eintrag die Methode zum Löschen eines Objektes auf.

## 6 Fazit der Teammitglieder

### 6.1 Fazit von Fabia Schmid

Mein Fazit wird sich in ein generelles Fazit zu der Applikation, ein Fazit bezüglich der Gruppe und ein Fazit zu meiner eigenen Arbeit unterteilen.

Die Applikation, welche von uns, dem Team Angry Nerds, entwickelt wurde, entspricht den Vorgeben, die von dem Professor vorgegeben wurden und erweitert diese mit verschiedenen Funktionen. Zusätzlich ist die Oberfläche farbenfroh und benutzerfreundlich. Zusätzlich achteten wir auf eine einfache und intuitive Bedienung, wodurch auch eine hohe User-Akzeptanz erwartet wird.

Somit kann in Bezug auf die Applikation, das Fazit gezogen werden, dass die Applikation erfolgreich und sehr zufriedenstellend entwickelt wurde.

Auch das Fazit bezüglich der Gruppenarbeit fällt sehr positiv aus. Alle Gruppenmitglieder waren stets motiviert und haben sich an allen Schritten beteiligt. Bei Problemen probierten alle zu helfen und eine Lösung zu finden. Zusätzlich waren alle Teammitglieder sehr zuverlässig und zielorientiert. Durch die durchgängige Motivation war das ganze Team auf einer Wellenlänge und konnte gut zusammenarbeiten und das Projekt erfolgreich abschließen. Die Kommunikation wurde hier sehr vereinfacht, durch die Nutzung von Microsoft Teams und einer WhatsApp-Gruppe, in der schnell kleinere Fragen beantwortet werden konnten.

Die Projektsteuerung, welche meine Hauptaufgabe war, verlief auch erfolgreich. Die geplanten Aktivitäten konnten bis auf zwei Meilensteine fristgerecht erfüllt werden. Die beiden Meilensteine konnte ich jedoch durch verschieben trotzdem abschließen, wodurch das Projektziel nicht gefährdet wurde. Zusätzlich sorgte ich durch regelmäßige Erinnerungen für die Erfüllung der Aufgaben und koordinierte verschiedene Aufgaben im Team erfolgreich. Neben der Koordination, wozu auch die Zeitplanung des Projektes gehörte, erstellte ich in der Overview Activity alle Layouts und programmierte drei Klassen. Auch diese Aufgabe vollendete ich erfolgreich. Die Layouts passen zu den vorher festgelegten Vorgaben im Mockup und die Funktionen funktionieren wie geplant.

Somit kann auch in Bezug auf meine Arbeit, ein positives Fazit gezogen werden. Ich erfüllte die Aufgabe der Projektleiterin zufriedenstellend und beendete auch die programmatischen Aufgaben erfolgreich.

## **6.2 Fazit von Florian Rath**

Fazit

## **6.3 Fazit von Jan Beilfuß**

Fazit

## **6.4 Fazit von Joscha Nassenstein**

Fazit

## **6.5 Fazit von Robin Menzel**

Fazit

## **6.6 Fazit von Ruthild Gilles**

Das Modul WIP enthält nicht nur die Programmierung einer Applikation, sondern für die erfolgreiche Implementierung wurde auch Projektmanagement benötigt. Diese Kombination finde ich realitätsnäher als es in anderen Modulen der Fall ist. Ein Projekt von vorne bis hinten zusammen in einem Team durchzuführen hat mir viele neue Erkenntnisse und Erfahrungen gebracht.

Zusätzlich wurden während des Projektes allerdings auch einige andere Fähigkeiten gefragt, welche ich noch nicht beherrschte. Dazu gehörte die Versionierung, welche wir mit Git Hub realisiert haben. Zu Beginn war es für mich eine Herausforderung die Funktionsweise von Git zu verstehen. Nach einiger Einarbeitungszeit beherrschte ich jedoch die Grundfunktionen, sodass ich diese Fähigkeit jetzt auch in meinem restlichen

Leben einsetze kann. Abgesehen von der Versionierung, welche nicht für das Projekt gefordert war, stand mir die Aufgabe zu, mich mit dem Textverarbeitungsprogramm von Latex zu beschäftigen. Da uns hier ebenfalls jegliche Einweisung fehlte, benötigte ich auch hier zusätzliche Zeit zum Erlernen der benötigten Grundkenntnisse für Latex. Jedoch werde ich auch diese neuen Kenntnisse außerhalb von dem Modul einsetzen können.

In unserem Projektteam gab es einige sehr gute Entwickler und ich fand es eine Herausforderung mit dieser Leistung mitzuhalten. Wenn ich entwickle benötige ich deutlich mehr Zeit, um mich in die Logik hinein zu denken. Deswegen kam es dazu, dass die Implementierung von Logik von anderen Teammitgliedern übernommen wurden, da es für diese ein höherer Zeitaufwand gewesen wäre, mir die Logik der umgebenden Klassen und Methoden zu erklären, als es eben selbst schnell zu machen.

Ich war dem Datenteam zugeordnet und zu Beginn hatten wir Schwierigkeiten mit der Planung der persistenten Datenhaltung. Die Schnittstelle zwischen den Activities und den Datenbank-Klassen, die von mir entwickelt wurden, konnten wir zu Beginn nicht im Detail planen, da wir nicht genau wussten, welche Anforderungen die Activities an die Datenbank stellen würden. Dies lag an einer nicht ausgereiften Kommunikation zu Beginn des Projektes. Aus diesem Grund erstellte ich Klassen und musste diese später ändern und anpassen. Es war mehr ein Ausprobieren als ein strukturiert geplantes Entwickeln. Bei einem nächsten Projekt würde ich besonders zu Beginn, häufiger Teammeetings einplanen um gemeinsam das Grundgerüst der Applikation zu planen.

Zusätzlich habe ich den Zeitaufwand für die Fertigstellung der Ausarbeitung in Latex unterschätzt. Besonders da im Zeitraum von November bis Februar ausgesprochen viele Prüfungsleistungen und auch der Abschluss unserer Ausbildung anstanden, war es für alle Teammitglieder eine Herausforderung ihren Teil der Ausarbeitung bis zur Deadline fertig zu stellen. Letztendlich hat es jedoch noch alles geklappt.

## 6.7 Fazit von Sertan Cetin

Fazit

## 6.8 Fazit von Yannick Rüttgers

Fazit

## 7 Quellenverzeichnis

### 7.1 Unterkapitelüberschrift

Unterkapitel

#### 7.1.1 Unterunterkapitelüberschrift

Unterunterkapitel

## 8 Anhang - Quelltexte

### 8.1 Activities

### 8.2 Data

#### 8.2.1 Service Klassen (Ruthild Gilles)

**Listing 1:** Create Klasse (Ruthild Gilles)

```
public class Create {  
    /* Ruthild Gilles  
     * Class Create contains methods to create new empty TEN objects.  
     * This class only exists to give a consistent form to the CRUD methods.  
     */  
  
    public static Todo newTodo() {  
        return new Todo();  
    }  
  
    public static Event newEvent() {  
        return new Event();  
    }  
  
    public static Note newNote() {  
        return new Note();  
    }  
}
```

**Listing 2:** Read Klasse (Ruthild Gilles)

```

public class Read {
    /* Ruthild Gilles
    Class Read contains methods to get all or one specific TEN object.
    */

    /**
     *-----*
     * Method to get all TEN objects in an arraylist
     *-----*/
    public static ArrayList<TEN> getAllTENs() {
        DatabaseRepository databaseRepository = new DatabaseRepository();
        ArrayList<TEN> allTEN;
        allTEN = databaseRepository.getAllTENs();
        Log.i("Mainfix", "Number Of TENs: " + allTEN.size());
        for (TEN ten : allTEN) {
            Log.i("Mainfix", "ID: " + ten.getID() + ", Titel: " + ten.getTitle());
        }
        return allTEN;
    }

    /**
     *-----*
     * Methods to get one TEN object by ID
     *-----*/
    public static Todo getTodoByID(String id) {
        DatabaseRepository databaseRepository = new DatabaseRepository();
        Todo todo = databaseRepository.getTodoByID(id);
        return todo;
    }

    public static Event getEventByID(String id) {
        DatabaseRepository databaseRepository = new DatabaseRepository();
        Event event = databaseRepository.getEventByID(id);
        return event;
    }

    public static Note getNoteByID(String id) {
        DatabaseRepository databaseRepository = new DatabaseRepository();
        Note note = databaseRepository.getNoteByID(id);
        return note;
    }

    public static int[] getColors(String tenID) {
        DatabaseRepository databaseRepository = new DatabaseRepository();
        int[] colors = databaseRepository.getTENColors(tenID);
        return colors;
    }
}

```

**Listing 3:** Update Klasse (Ruthild Gilles)

```

public class Update {
    /* Ruthild Gilles
       Class Update contains methods to save information on a changed or newly created TEN object */
    /**
     *-----*
     * Methods for saving a TEN object
     *-----*/
    public static void saveTEN(TEN newTen) {
        DatabaseRepository databaseRepository = new DatabaseRepository();
        if (newTen.getID() == null) {
            databaseRepository.insertTEN(newTen);
        } else databaseRepository.updateTEN(newTen);
    }
}

```

**Listing 4:** Delete Klasse (Ruthild Gilles)

```

public class Delete {
    /* Ruthild Gilles
       Class Delete contains methods to delete the given TEN object.*/
    public static void deleteTEN(String tenID) {
        DatabaseRepository databaseRepository = new DatabaseRepository();
        databaseRepository.deleteTEN(tenID);
    }

    public static void deleteMultipleTENs(ArrayList<String> tenIDs) {
        DatabaseRepository databaseRepository = new DatabaseRepository();
        for (String tenID : tenIDs) {
            databaseRepository.deleteTEN(tenID);
        }
    }
}

```

## 8.3 Overview

## 8.4 Main Activity

## 9 Anhang - Verwendete Tools und Hilfsmittel (Robin Menzel)

Tool / Programm	Einsatz
Adobe Xd	Erstellung des Mock-Ups
Android Studio	Entwicklungsumgebung (IDE)
Draw.io	Erstellung der UML-Diagramme
Excel	Durchführung der Aufwandsschätzung
GitHub	Versionsverwaltung des Quelcodes
LaTeX	Dokumentation des Projektes (Struktur)
OneDrive	Dateiablage
OneNote	Protokollierung, Notizen und Absprachen
PowerPoint	Erstellung der Planungsdokumente
Word	Erstellung von Dokumenten

## **Ehrenwörtliche Erklärung**

Hiermit erkläre ich, dass ich die vorliegende Schriftliche Ausarbeitung selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

---

Ort, Datum

---

Unterschrift