TIMEAID

Andre Rivera Arboleda, Johannes Fessler, Bjorna Kalaja, Miel Satrapa

Inhaltsverzeichnis

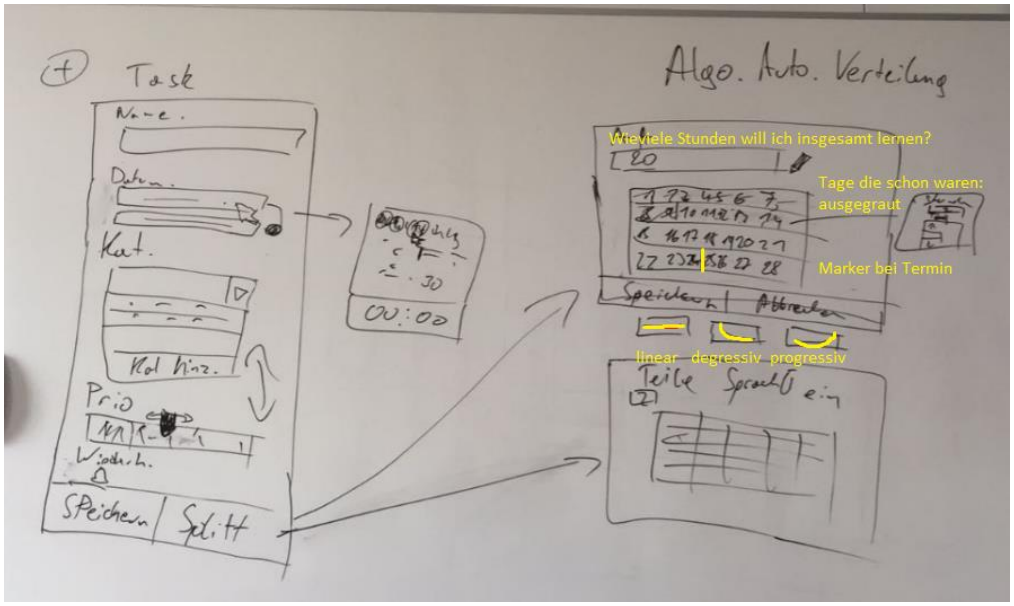
1. Spezifikation

1.1 Kurze Projektbeschreibung

Die Vision des Projektes „TimeAid“ ist es, eine Kalendermanagement-App für Android zu erstellen. Dabei wird ein vom Benutzer schon existierender Google Kalender importiert und synchronisiert.

Das ausschlaggebende Feature unserer App soll die dynamische und automatische Termineinteilung sein.

Die App definiert sich grob als eine externe Zusatzfunktion des Google Kalenders. Der Kalender wird importiert, die Termine werden eingetragen/dynamisch eingeteilt und wieder mit Google synchronisiert, sodass der User auf die mit unserer App eingetragenen Termine auch im Google Kalender Zugriff hat.



Die dynamische Termineinteilung soll wie folgt funktionieren: Der User trägt z.B. einen Prüfungstermin ein und wählt aus, dass er für die Prüfung insgesamt 10 Stunden lernen will. Dann kann er auswählen ob er lieber am Anfang mehr lernt und dann nur noch wiederholt (degressiv), ob er gleichverteilt also z.B. jeden Tag 1 Stunde lernen will (linear) oder ob er mehr lernen will, je näher der Test kommt (progressiv).

Unser System schaut nun nach, zu welchen Zeiten Platz fürs Lernen ist. Natürlich werden Dinge wie Schlafenszeit (also dass der Termin nicht um 3:00 in der Früh eingetragen wird), Userpräferenzen (User kann Zeitfenster einstellen z.B. von 13:00-21:00) und kollidierende Termine beachtet.

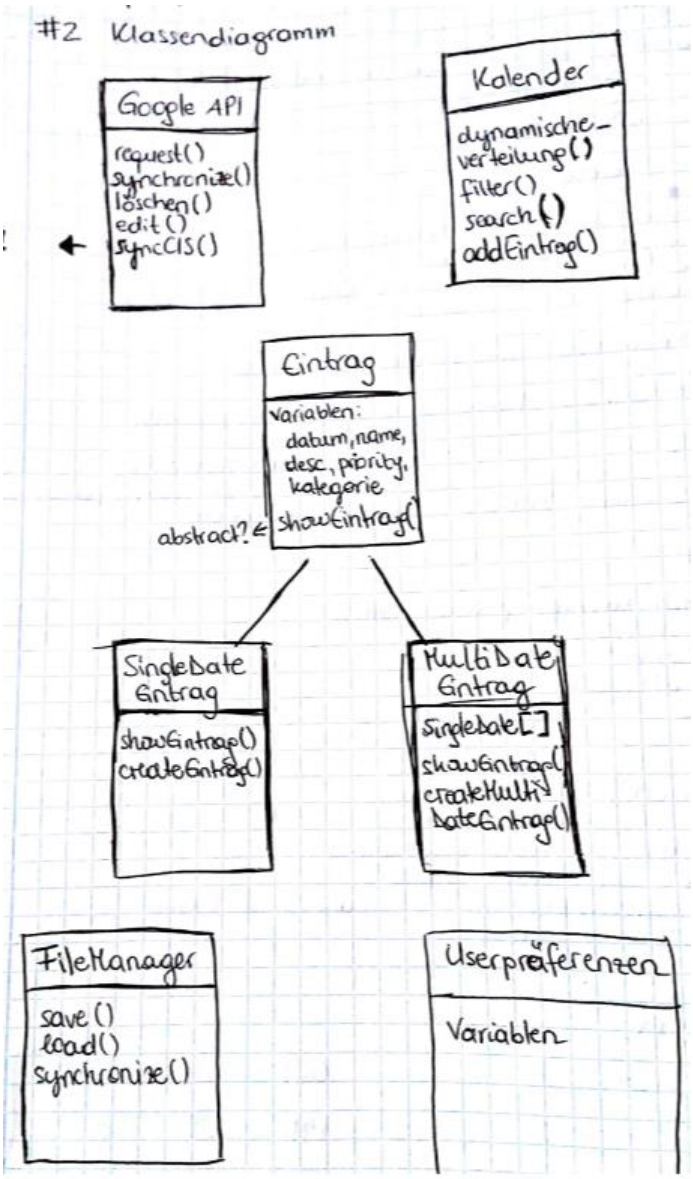
Gefällt dem User einer oder mehrere der vorgeschlagenen Termine nicht, kann er diese auch manuell löschen bzw. verschieben.

Zusätzlich zu diesem Termineinteilungsalgorithmus übernehmen wir den Großteil der Funktionen die auch der Google Kalender hat wie z.B. einmalige und wiederholte Termine erstellen, Alerts und Erinnerungen für Termine oder von Tages- auf Wochen- und Monatsansichten ändern.

Der CIS-Kalender wird natürlich auch mitberücksichtigt, vorausgesetzt man hat diesen schon im Google-Kalender importiert.

Unsere App richtet sich hauptsächlich an vielbeschäftigte Personen, die ein komfortables Zeitmanagement-System schätzen und bietet somit einen riesigen Mehrwert für alle, die ihre Zeit gern für Anderes nutzen, als für die nie enden wollende Gestaltung des Kalenders.

1.2 Technische Spezifikation



1.2.1 Google API Klasse

Funktion siehe „Ziele: Google API“

1.2.2 Kalenderklasse

Die Kalenderklasse wird die Logik hinter der dynamischen Terminverteilung beinhalten. Außerdem werden hier verschiedene Filteroptionen und die Fähigkeit, nach bestimmten Terminen zu suchen implementiert. Die Kalenderklasse wird außerdem Funktionen zum Löschen und Bearbeiten von Terminen beinhalten.

1.2.3 Eintrag

Ein Eintrag ist eine allgemein gehaltene Überklasse für Single Date Eintrag und Multi Date Eintrag. Hierbei wird es sich entweder um eine Klasse oder eine Schnittstelle handeln. (Genauere Details werden wir beim Programmieren herausstellen) Außerdem wird sie Basisfunktionen wie zum Beispiel show_Eintrag() beinhalten.

1.2.4 Single Date Eintrag

Eine Klasse die einen klassischen Termin beschreibt und Funktionalitäten ihrer Elternklasse zur Verfügung stellt. Hier werden Informationen gespeichert wie das Start und End-Datum, die Dauer und die Priorität sowie die Kategorie. Die letzten 2 Parameter sind jeweils Filteroptionen und Informationen für die dynamische Verteilung.

1.2.5 Multi Date Eintrag

Ein Multi Date Eintrag speichert einerseits den Single Date Eintrag der ursprünglich erstellt wurde und andererseits eine Liste von Single Date Einträgen die zusätzlich beim aufsplitten des Termins entstanden sind.

1.2.6 File Manager

Der File Manager kümmert sich um alles was mit Serialisierung zu tun hat. Zugriffe auf Daten auf dem Handy passieren nur über diese Klasse.

1.2.7 Userpräferenzen

Speichern der Präferenzen eines Users sowie seinen Namen für personalisierte Pop-Ups und Nachrichten.

1.3 Verwendete Technologien

1.3.1 Programmiersprache

- JAVA

1.3.2 IDE

- Android Studio (IntelliJ)

1.3.3 Projektmanagement

- Trello (Kanban)
- GitHub (Dokumente/Code)

1.4 Teamaufteilung

Wir waren uns von Anfang an einig, dass wir alle programmieren wollen. Also haben wir uns die Hauptaufgabenbereiche angesehen und jedem, nach dem Vier-Augen-System einen Haupt- und einen Zweitverantwortlichen zugeteilt. Das heißt, dass jeder bei jeder Aufgabe mitwirken kann, aber der Hauptverantwortliche den Überblick haben muss.

	Kalender App	Google API	Statisches Layout/Design	Projektmanagement
1.Verantwortliche/r	Andre	Bjorna	Johannes	Miel
2.Verantwortliche/r	Johannes	Miel	Bjorna	Andre

2. Ziele & Nicht-Ziele

2.1 Ziele

2.1.1 Google Schnittstelle

Die Google Schnittstelle soll es einem ermöglichen eine Verbindung zur Google Kalender API aufzubauen und an diese verschiedene Requests schicken zu können.

Möglich sein soll: das Erstellen, Bearbeiten und Löschen von Terminen aus seinem Google Konto.

Da die App auch im Offlinemodus funktionieren soll, werden Anfragen an die API periodisch abgewickelt. Das passiert natürlich nur im Onlinemodus.

Optional: Synchronisierung wahlweise nur mit WLAN nicht mit mobilem Internet.

2.1.2 Termine Editieren/Erstellen/Löschen

Termine sollen bearbeitet, erstellt und gelöscht werden können. Da der Mehrwert unserer App in der dynamischen Terminverteilung liegt, werden wir der Terminverwaltung einen großen Teil unserer Aufmerksamkeit schenken (Siehe Dynamische Terminverteilung). Diese Funktionen sollen sowohl online als auch offline verfügbar sein (Siehe Google Schnittstelle).

Es sollte möglich sein einen Termin zusammen mit den zugehörigen Unterterminen löschen zu können. Es sollte außerdem die Möglichkeit bestehen, die Einteilung von Unterterminen auch nachträglich zu ändern.

Die Suche und das Filtern von Terminen (z.B. nach Kategorie/Priorität) soll für den User möglich sein.

Beim Erstellen bzw. Bearbeiten von Terminen soll man auch einstellen können, ob man eine Benachrichtigung bzw. ein Alert bekommen will.

2.1.3 Dynamische Terminverteilung

Die dynamische Terminverteilung ist das Kernstück unserer App. Hier soll es möglich sein, bei der Erstellung eines Termines anzugeben, ob man „Untertermine“ haben will (wie z.B. 10 Std. bis zum Termin lernen). Untertermine werden entweder automatisch oder manuell verteilt. Im zweiten Fall werden Termine automatisch vorgeschlagen. Ein Anwendungsbeispiel könnte so aussehen:

Johannes hat in einem Monat einen Test und möchte 20 Stunden dafür lernen. Er tippt den Termin des Tests ein und wählt die Möglichkeit der dynamischen Terminverteilung. Diese Untertermine benennt er „Für Test lernen“. Johannes bekommt eine Vorschau wie die App jetzt dynamisch seine Termine verteilen würde. Dabei hat er verschiedene Optionen wie er diese Termine verteilt haben möchte. Er kann sich für verschiedene Stundenblockgrößen sowie für ein Termineinteilungsmodell (linear, degressiv, progressiv -> siehe kurze Projektbeschreibung) entscheiden.

2.1.4 User Interface

Da der User wahrscheinlich schon primär die Google-Kalender App benutzt hat, wollen wir das Design unserer GUI stark an die Google Kalender Oberfläche anlehnen.
Optional: Verschiedene Themen implementieren.

2.1.5 Online & Offlinemodus

Wie bei der Google Schnittstelle schon erklärt wird das Programm, obwohl es mit dem Google Kalender verbunden ist, auch offline verfügbar sein. Dafür werden wir eine lokale Datenstruktur auf dem Handy anlegen. Die Serialisierung von Objekten werden wir mit einer externen Library abdecken.

2.2 Falls Zeit ist

Sofern Zeit ist, könnten zusätzlich folgende Features implementiert werden:

Lerntypenanalyse: Mithilfe einer Lerntypen-Analyse soll festgestellt werden, zu welcher Tageszeit Lerneinheiten sinnvoll sind.

Pomodoro Uhr: Nach dem klassischen 25 Minuten Modell soll die Pomodoro Uhr für eine möglichst effiziente Lernzeit sorgen.

Statistiken: Statistiken sollen Aufschluss über diverse Verhaltensweisen und/oder Daten wie z.B. Betriebszeit der Software bieten.

2.3 Nicht - Ziele

Kalender ganz neu programmieren

Unser Hauptaugenmerk liegt auf der dynamischen Terminverteilung und nicht der Entwicklung eines neuen Kalenders.

2.3.1 Responsive Design

Wir werden nicht auf verschiedene Displaygrößen Rücksicht nehmen, oder die Möglichkeit der Anpassung des Kalenders bei Drehung des Handys.

2.3.2 Kontoverwaltung

Die Präferenzen eines Users werden zwar in einer separaten Datei auslagerbar sein, und es wird möglich sein sie auch wieder einzulesen, aber eine Online Kontoverwaltung mit Passwort und Benutzername wird es nicht geben.

2.3.3 Datenbank

Da wir mit der Google API arbeiten sehen wir keinen Sinn darin eine Datenbank für unser Projekt zu verwenden. Alle Daten liegen entweder serialisiert bis zu einem gewissen Ausmaß auf dem Smartphone oder Online auf dem Google Kalender.

2.3.4 Coole Effekte

Es werden außer Standardanimationen keine weiteren Animationen in unser Programm eingebaut.

2.3.5 Wegzeiten

Wegzeiten werden bei unserem Terminverteilungssystem nicht beachtet, da wir dies als nicht sehr sinnvoll erachten. Es kann ja z.B. sein, dass man um 12:00 die erste LV hat aber schon davor in die FH fährt um zu lernen.

3. Arbeitspakete & Aufwandsschätzung ~280h

3.1 Vorprojekt/Spezifikation ~22h

3.1.1 Ideen Sammeln (zählt zu Meeting) ✓

Entwicklung der Projektidee und Einigung auf grobe Grundsätze. Android? Apple? Beide? Welche Technologien werden verwendet? Etc.

3.1.2 Arbeitsaufteilung (zählt zu Meeting) ✓

Zuweisung der Hauptverantwortlichen für die jeweiligen Aufgaben.

3.1.3 Technische Details überlegen (zählt zu Meeting) ✓

Überlegungen zum Aufbau der App. Überlegungen zu den verwendeten Klassen aufstellen.

3.1.4 Ziele / Nichtziele setzen (zählt zu Meeting) ✓

Spezifizieren was geplant ist und was definitiv nicht geplant ist.

3.1.5 Fertige Spezifikation ~22h ✓

Dieses Dokument 😊

3.2 Umsetzung ~136h

3.2.1 Logo Design ~3h ✓

Erstellung eines fertigen Logos für unsere App.

3.2.2 Java Crashkurs ~5x4=20h ✓

Java als Programmiersprache erlernen. Tutorials für Einsteiger sehen bzw. lesen und dann spezifische Recherche machen, je nach dem was für unsere Applikation notwendig ist, z.B. Serialisierung, Implementierung von Schnittstellen usw.

3.2.3 Google Schnittstelle Testen ~6h

Testprogramm schreiben um Grundfunktionalitäten auszuprobieren.

3.2.4 Meilenstein: Graphical User Interface ~20h

GUI für die App entwickeln. Als Basis dient das Layout von Google Kalender API. Es wird eine Navigationsleiste implementiert, wo die verschiedenen Buttons zur Funktionalität der App zu finden sind.

3.2.5 Meilenstein: Google API Einbindung ~15h

Google bietet eine Schnittstelle zum Darstellen und Erstellen von Kalender-Ereignissen. Zusätzlich kann man den Google Kalender mit anderen Kalendern synchronisieren und Ereignisse bearbeiten. Der Code zur Implementierung der Schnittstelle wird entwickelt und die Methoden der Schnittstelle getestet. Eine funktionierende Implementierung der Google API bietet die Basis für die Weiterentwicklung der Applikation.

Hier gibt es die Methoden: request(), synchronise(), edit() und syncCis() implementiert.

3.2.6 Kalender Klasse ~40h

Die Grundfunktionalität der Applikation wird hier entwickelt. Die Klasse Kalender enthält die Methoden: dynamischeVerteilung(), filter() und suche().

dynamischeVerteilung() - siehe Ziele (dynamische Terminverteilung).

filter() – Filteroptionen werden integriert, z.B. man will nur Ereignisse, die mit der FH zu tun haben.

suche() – Suche nach einem bestimmten Ereignis.

3.2.7 Eintrag Klasse ~5h

Code zu Einträgen (Ereignisse) hinzufügen, bearbeiten bzw. löschen wird entwickelt. Hierbei wird zwischen SingleDate und MultiDate Ereignissen unterschieden. Danach folgt die Implementierung der Klassen Eintrag, SingleDate und MultiDate.

SingleDate: Sind Ereignisse, die einzelne Termine darstellen, z.B. ein Test. Hier gibt es die Methoden showEvent(), createEvent(), editEvent().

MultiDate: Sind Ereignisse die ein übergeordnetes SingleDate Objekt haben dem dann einzelne kleinere Events zugeordnet werden, z.B. das Lernen für einen Test das auf mehrere Tage aufgeteilt werden muss. Technisch gesehen ist ein Multidate Eintrag ein SingleDate Objekt mit einer Liste an zugeordneten SingleDate Objekten. Die so entstandene Struktur ist leichter zu bearbeiten.

3.2.8 Userpräferenzen Klasse ~3h

Die Klasse Userpräferenzen ermöglicht die Änderung von Benutzereinstellungen. Für die dynamische Terminverteilung werden die Lernzeiten bzw. Lerntyp bestimmt, z.B. ich lerne von 08:00 bis 16:00 jeden Tag.

3.2.9 File Manager Klasse ~8h

Hier wird die Serialisierung von Objekten implementiert. Dafür wird eine externe Bibliothek verwendet. Kalender-Ereignisse werden in eine lokale Datei gespeichert und beim Starten der App wieder von der Datei abgerufen und in der App zur Verfügung gestellt.

Hier werden die Methoden: save(), load() und synchronize() implementiert.

3.2.10 Meilenstein: Beta-Version ~4x4=16h

Am Ende der Umsetzung sollten wir eine fertige komplett funktionale zusammengesetzte Beta-version haben, die nur noch umgehen getestet werden muss.

3.3 Dokumentation ~19h

3.3.1 GIT erlernen ~1x4=4h ✓

Das Konzept hinten GIT erlernen. GIT installieren, GIT Repository erstellen und GIT GUI erlernen.

3.3.2 Projektbeschreibung ~5h

Komplette Projektbeschreibung für das Abschlussdokument.

3.3.3 Projektablauf Dokumentation ~2h

FH-Template Dokumentation ausfüllen

3.3.4 Handbuch ~2x4=8h

Benutzer oder Entwicklerhandbuch (ca. 8 Seiten)

3.4 Testen ~33h

3.4.1 Google API testen ~5h

Testen der Google Schnittstelle & Google API

3.4.2 Kalender Klasse testen ~5h

Testen der Kalender Klasse

3.4.3 GUI testen ~5h

Testen des Graphical User Interface (Usability)

3.4.4 Save and Load Funktionen Testen ~5h

Testen der Save und Load Funktionen

3.4.5 Externer User-Test ~5h

Um auch eine Meinung von außen zu bekommen, werden wir unsere App auch extern testen lassen.

3.4.6 Meilenstein: Endversion ~8h

Am Ende der Testphase soll die App komplett fertig sein.

3.5 Evaluierung ~8h

3.5.1 Soll-Ist Vergleich ~1x4=4h

Vergleich der Stunden und der erreichten/geplanten Ziele

3.5.2 Lessons Learned ~1x4=4h

Was haben wir durch dieses Projekt gelernt?

3.6 Meetings ~62h

3.6.1 Team-Meetings ~10x4=40h

Wöchentliche Team-Meetings und Kontrolle der in der vergangenen Woche geleisteten Arbeitspakete.

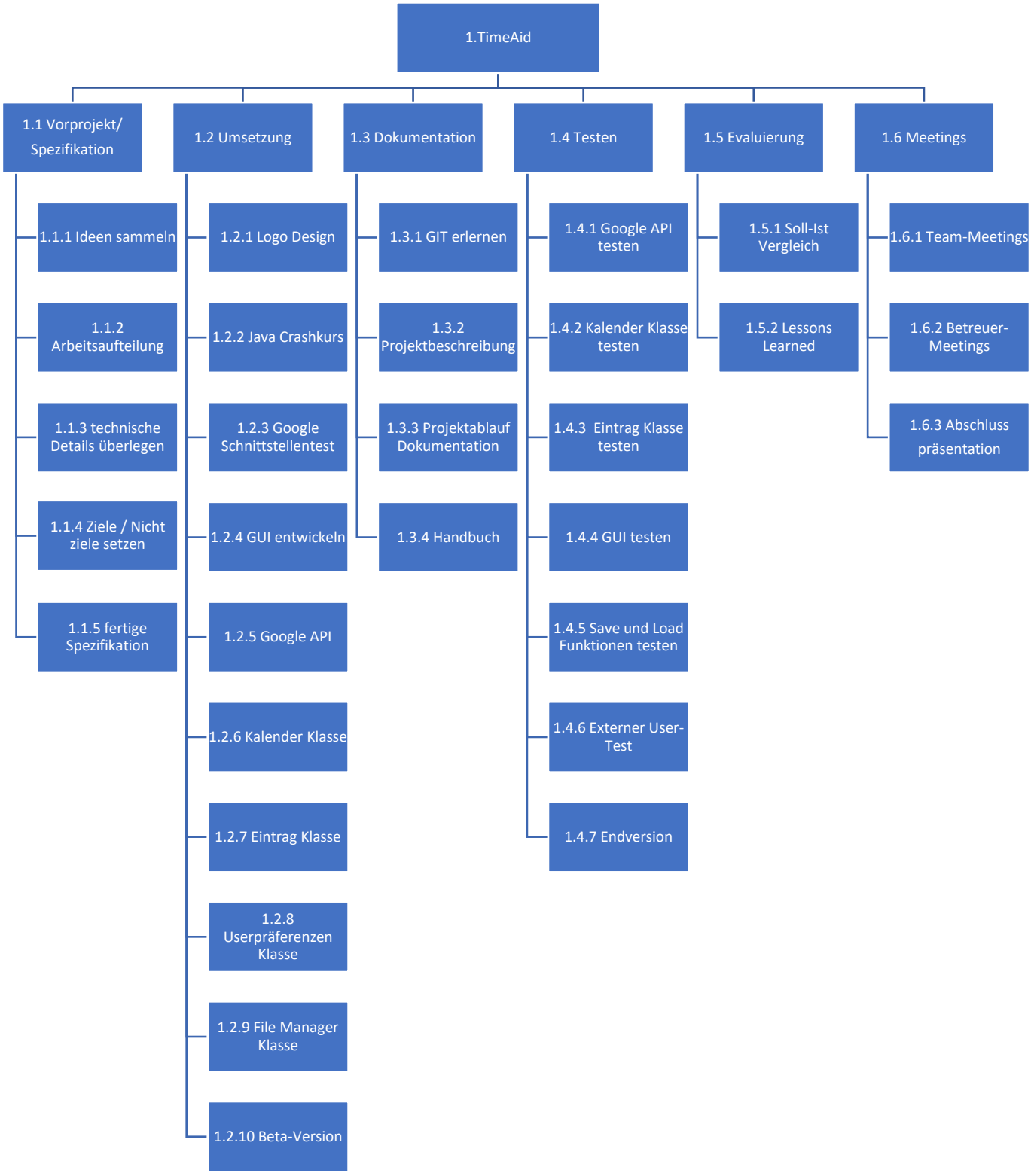
3.6.2 Betreuer-Meetings ~1,5x4=6h

Meetings mit dem Betreuer und Besprechung der Statusberichte und geleisteten Arbeitspakete der vergangenen Wochen.

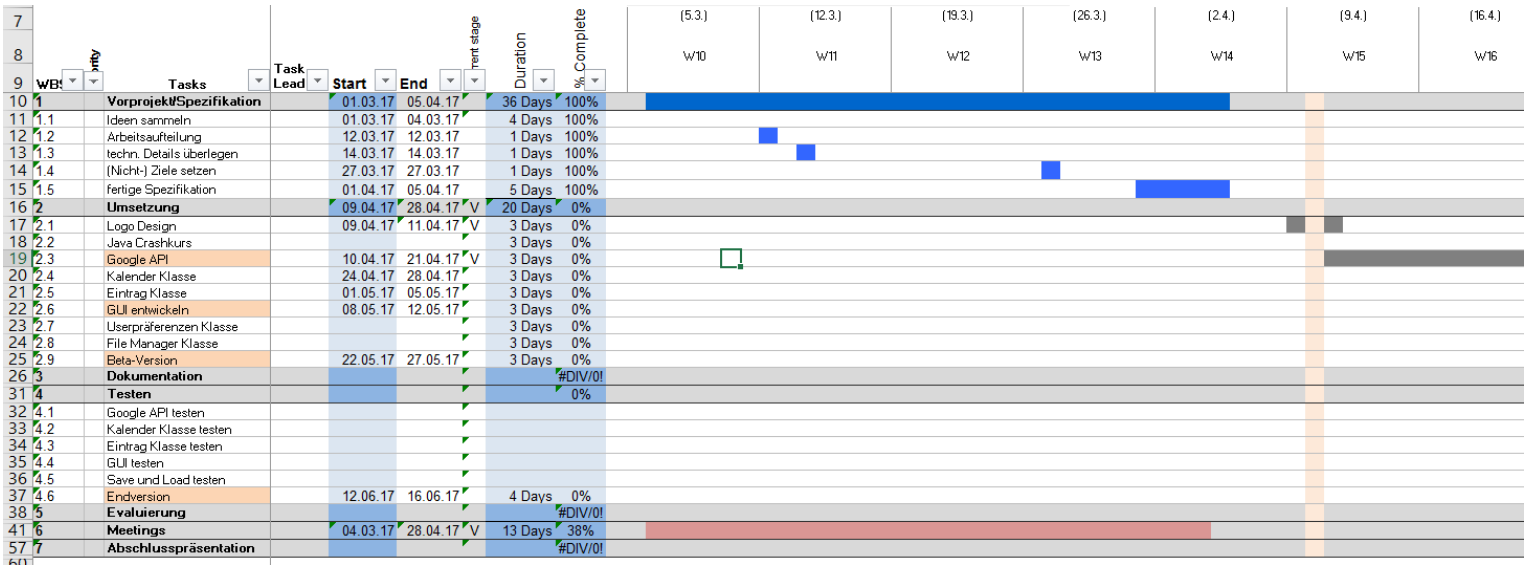
3.6.3 Abschlusspräsentation ~4x4=16h

Vorbereitung und Erstellung einer Präsentation für die Abschlusspräsentation

4. Projektstrukturplan



5. Projektablaufplan



WBS	Tasks	Task Lead	Start	End	Duration	% Complete
1	**Vorprojekt/Spezifikation**		01.03.17	05.04.17	36 Days	100%
1.1	Ideen sammeln		01.03.17	04.03.17	4 Days	100%
1.2	Arbeitsaufteilung		12.03.17	12.03.17	1 Days	100%
1.3	techn. Details überlegen		14.03.17	14.03.17	1 Days	100%
1.4	(Nicht-) Ziele setzen		27.03.17	27.03.17	1 Days	100%
1.5	fertige Spezifikation		01.04.17	05.04.17	5 Days	100%
2	**Umsetzung**		09.04.17	28.04.17 V	20 Days	0%
2.1	Logo Design		09.04.17	11.04.17 V	3 Days	0%
2.2	Java Crashkurs				3 Days	0%
2.3	Google API		10.04.17	21.04.17 V	3 Days	0%
2.4	Kalender Klasse		24.04.17	28.04.17	3 Days	0%
2.5	Eintrag Klasse		01.05.17	05.05.17	3 Days	0%
2.6	GUI entwickeln		08.05.17	12.05.17	3 Days	0%
2.7	Userpräferenzen Klasse				3 Days	0%
2.8	File Manager Klasse				3 Days	0%
2.9	Beta-Version		22.05.17	27.05.17	3 Days	0%
3	**Dokumentation**					#DIV/0!
4	**Testen**					0%
4.1	Google API testen					
4.2	Kalender Klasse testen					
4.3	Eintrag Klasse testen					
4.4	GUI testen					
4.5	Save und Load testen					
4.6	Endversion		12.06.17	16.06.17	4 Days	0%
5	**Evaluierung**					#DIV/0!
6	**Meetings**		04.03.17	28.04.17 V	13 Days	38%
7	**Abschlusspräsentation**					#DIV/0!

(5.3.) W10 | (12.3.) W11 | (19.3.) W12 | (26.3.) W13 | (2.4.) W14 | (9.4.) W15 | (16.4.) W16

Da wir ein agiles Projektmanagement verfolgen, werden diverse Daten fortführend eingefügt, wie z.B. „Google API testen“ oder „Java Crashkurs“.

Selbiges gilt für die Meetings, deren Anzahl uns bereits bekannt ist, jedoch noch keine Details.

Für die interne Kommunikation und Synchronisation verwenden wir TRELLO.