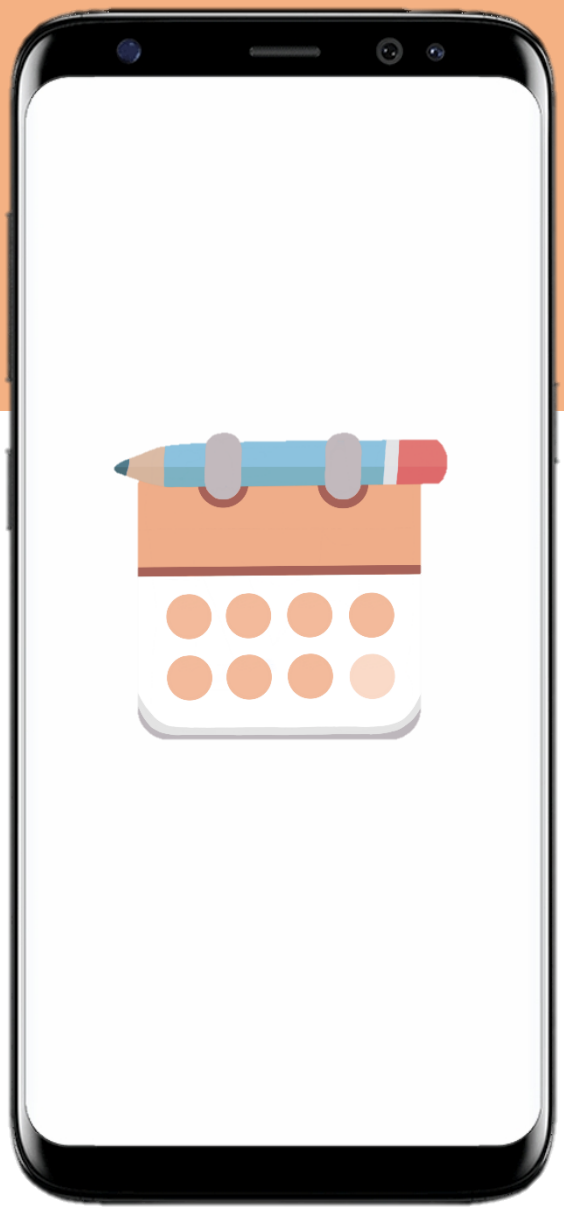
Selbständige Projektarbeit
Informatikmittelschule Frauenfeld

Planner

Autor: Yasmin Häberli
Betreuer: Matthias Bolli
Datum: 28. Februar 2020
Version: 1.0



INHALTSVERZEICHNIS

1 Projekt

1.1 Management Summary

Im Rahmen der Semesterarbeit 2019/2020 wurde die Android App «Planner» entwickelt, welches die Verwaltung von Terminen und Aufgaben vereinfacht. Nutzer können mittels dieser Applikation bevorstehende Pläne eintragen und erhalten nach Ablauf des Termins eine Meldung. Falls gewünscht, kann eine frühere Meldung hinzugefügt werden. Die Aufgaben können mithilfe von Tags (Kennzeichnung der Aufgaben mit Begriffen) beliebig sortiert werden, um einen klaren Überblick zu erschaffen.

1.2 Projektorganisation

Die Realisierung des Projektes wurde von mir, Yasmin Häberli durchgeführt und Matthias Bolli hat dies betreut.

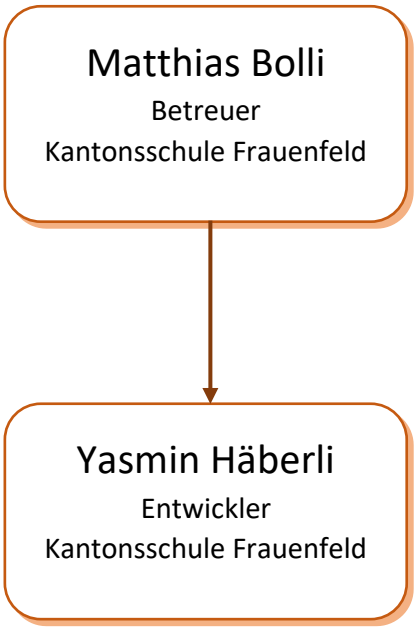


Abbildung 1: Projektorganisation

2 INFORMIEREN

2.1 Aufgabenstellung

Als Semesterarbeit 2019/2020 sollte eine App erstellt werden, welche eine «To-Do-List» ähnelt.

In der App können Nutzer ein Konto erstellen, in dem sie sich auf diversen mobilen Geräten des gleichen Betriebssystems anmelden können. Die Hauptfunktion dieser Applikation ist die Eintragung von Terminen und Aufgaben in einer Liste und in einem Kalender. Zu jeder Aufgabe können folgende Informationen gespeichert werden:

- Name
- Datum
- Zeit
- Beschreibung
- Tags (Kennzeichnung des Termins mit Begriffen)

Diese Daten werden in einer Datenbank gespeichert, welche jederzeit aufgerufen werden kann.

Falls nötig, können Aufgaben bearbeitet oder gelöscht werden.

Automatisch werden Erinnerungen in Form einer Notifikation erzeugt, welche am Datum des Termins erscheinen. Weiterhin können frühere Meldungen eingestellt werden, die vor dem Termin erscheinen.

Ein weiterer Feature ist, dass die Liste nach Wunsch des Benutzers gefiltert werden kann mittels der Tags.

2.2 App-Entwicklung

2.2.1 Bauarten von Apps

Es gibt drei verschiedene Möglichkeiten eine App zu realisieren. Diese sind Native Apps, Web Apps und Hybrid Apps. Jedes dieser Varianten haben ihre Vorteile und Nachteile. Um die passende Bauart zu finden, verglich ich alle miteinander und berücksichtige meine eigenen Anforderungen. Wichtig war vor allem die Komplexität der Bauart und ob meine Funktionen mit dieser Art realisiert werden können. Um einen Überblick zu jeder Variante zu erschaffen, sind kurze Informationen in den nächsten Teilen enthalten.

	Native Apps	Hybrid (PhoneGap)	(Web Apps)
Performance	🍌🍌🍌	🍌	🍌
User Experience	🍌🍌🍌	🍌	🍌
Plattformunabhängige Entwicklung	❌	✅	✅
*Kosten ***	💰💰💰	💰💰	💰
*Entwicklungszeit ***	🕐🕐🕐	🕐🕐	🕐
Push Notifications	✅	✅	✅
Gerätefunktionen	🍌🍌🍌	🍌🍌	🍌
App Store Upload möglich	✅	✅	❌
Sicherheit	🍌🍌🍌	🍌	🍌
Offline Fähigkeit	🍌🍌🍌	🍌	🍌
Etabliert (mehr Libraries, Nachhaltigkeit)	🍌🍌🍌	🍌	🍌🍌🍌

Abbildung 2: Übersicht Vergleich von Bauarten von Apps

2.2.2 Native App

Für die Entwicklung einer Native App ist das Betriebssystem die Basis, denn es wird für ein bestimmtes Betriebssystem wie Windows, iOS oder Android bereitgestellt. Sie werden in der Programmiersprache geschrieben, welche die Plattform akzeptiert. Zum Beispiel wird Android Applikationen mit Java und iOS Applikationen in Swift oder Objective-C geschrieben.

Wenn man eine Native App entwickelt, kann einfacher auf die Hardware zugegriffen werden und so können diese optimal genutzt werden.

2.2.2.1 Vorteile

- Performance
- Hohe Bedienungsfreundlichkeit
- Einfache Installation
- Native Apps erhalten öfters gute Bewertungen in den App Stores und werden deswegen häufiger gekauft

2.2.2.2 Nachteile

Der grösste Nachteil von Native Apps ist, dass die App nur auf einem Betriebssystem laufen kann. Verbunden mit dieser Kenntnis, entstehen folgende Nachteile, wenn man sich dafür entscheidet für mehr als nur ein Betriebssystem eine App zu entwickeln.

- Hohe Kosten
- Hoher Aufwand

2.2.3 Web Apps

Eine Web App ist ein vollständiger Browser basierte Anwendung. Sie wird über den Browser aufgerufen und nutzt ausschließlich Webtechnologien. Die Web App ist für die Nutzung mobiler Endgeräte optimiert und kann somit überall und plattformunabhängig über einen Browser aufgerufen werden.

2.2.3.1 Vorteile

- Bei Web Apps werden Sprachen genutzt, die mir bereits bekannt sind
- Plattformunabhängig
- Installation wird nicht benötigt

- Tiefe Kosten
- Kürzere Entwicklungszeit

2.2.3.2 Nachteile

- verliert Sichtbarkeit, da man sie nicht in den App Stores zu finden kann
- Einschränkungen beim Zugang zu bestimmter Hardware
- Anpassung der App, sodass es auf jedem Endgerät so erscheint wie erwartet
- Zugang zum Internet wird benötigt

2.2.4 Hybrid Apps

Eine Hybride App ist im engeren Sinne eine Browser Anwendung in einer nativen Hülle, also eine Webseite, die in einen nativen Container gepackt wurde, um sie als App auf einem mobilen Endgerät bereitstellen zu können. Die Web-App wird einmal entwickelt und in einem nativen Container als Hybride App für das jeweilige Betriebssystem “exportiert”.

2.2.4.1 Vorteile

- Enthält die wichtigsten Vorteile und Nachteile der Web- & Native-Apps
- Plattformunabhängig
- Kürzere Entwicklungszeit
- Tiefere Kosten

2.2.4.2 Nachteile

- Schlechtere Performance
- Diverse Beschränkungen

3 PLANUNG

3.1 Deklaration der Vorkenntnisse

Die Entwicklung von Apps ist mir bereits ein wenig bekannt, jedoch kenne ich mich nur bei Web-Apps aus und nicht bei Native-Apps. Android-Applikationen werden ausschliesslich mit Kotlin oder Java programmiert. Zwischen beiden Sprachen entschied ich mich für Java, denn diese habe ich im zweiten Semester bei der Realisierung eines Spiels kennengelernt. Nebst dem habe ich keine Vorkenntnisse.

3.2 Must-Haves

Am Ende des Projektes muss die App folgende Must-Haves besitzen:

- Verknüpfung App mit Datenbank
- Events speichern (Datum, Zeit & Text)
- Events löschen (Datum, Zeit & Text)
- Events bearbeiten (Datum, Zeit & Text)
- Meldungen falls Zeitraum überschritten
- Erinnerungen in Form einer Meldung
- Log-In mit Account

3.3 Nice-To-Haves

Falls allenfalls Zeit nach allen erfüllten Must-Haves besteht, werde ich folgende Funktionen einbauen:

- Events nach Tags sortieren
- Events in einem Kalender anzeigen

Da ich mich mit der Mobil-Applikationsentwicklung nur wenig auskenne, kann ich nicht genau einschätzen was möglich ist und was nicht. Deswegen beschränkte ich mich auf einfachere Must-Haves.

3.4 Use-Cases

Da mein Projekt nicht viele Funktionen beinhaltet, besteht mein Use-Case-Diagramm aus wenigen Anwendungsfällen.

Zwei wichtige Fälle thematisieren die Benutzerkontos und die Aufgaben.

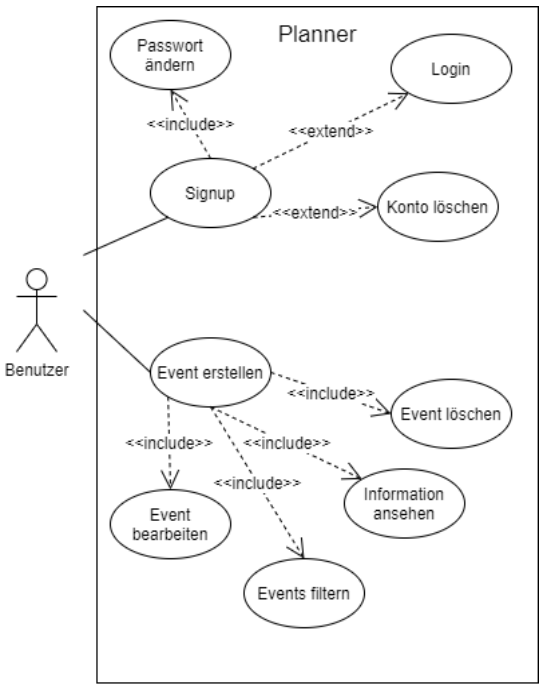


Abbildung 3: Use-Cases

3.5 Meilensteine

Um wichtige Punkte meines Projektverlaufes zu kennzeichnen und klare Ziele zu festlegen, setzte ich mehrere Meilensteine. An den jeweiligen Daten sollte ein Ergebnis vorliegen.

Ursprünglich plante ich fünf Meilensteine ein, jedoch beschränkte ich mich auf drei (ohne Abgabe).

1. 22.11.2019
   Konzept, Planung komplett & dokumentiert (Mockup, Use-Cases, usw.)
2. 29.12.2019
   Grundgerüst der App, Verbindung mit Datenbank, Nachführung der Dokumentation
3. 20.01.2019
   Fertigstellung der App, Testing & Nachführung der Dokumentation
4. 28.02.2020
   Abgabe, Feinschliff

Um mein Vorgehen organisiert zu planen, erstellte ich ein Trello-Board. Somit habe ich eine klare Übersicht über die Ziele welche als nächstes bevorstehen, abgearbeitet wurden und zurzeit bearbeitet werden. Laufend können neue Aufgaben erteilt werden und je nach Status («To Do», «In Progress», «Done») in Spalten gruppiert werden.

Vor jedem Arbeitsabschnitt trage ich die nächsten Ziele im Trello-Board ein.

4 Design

4.1 Usability & Ergonomische Standards

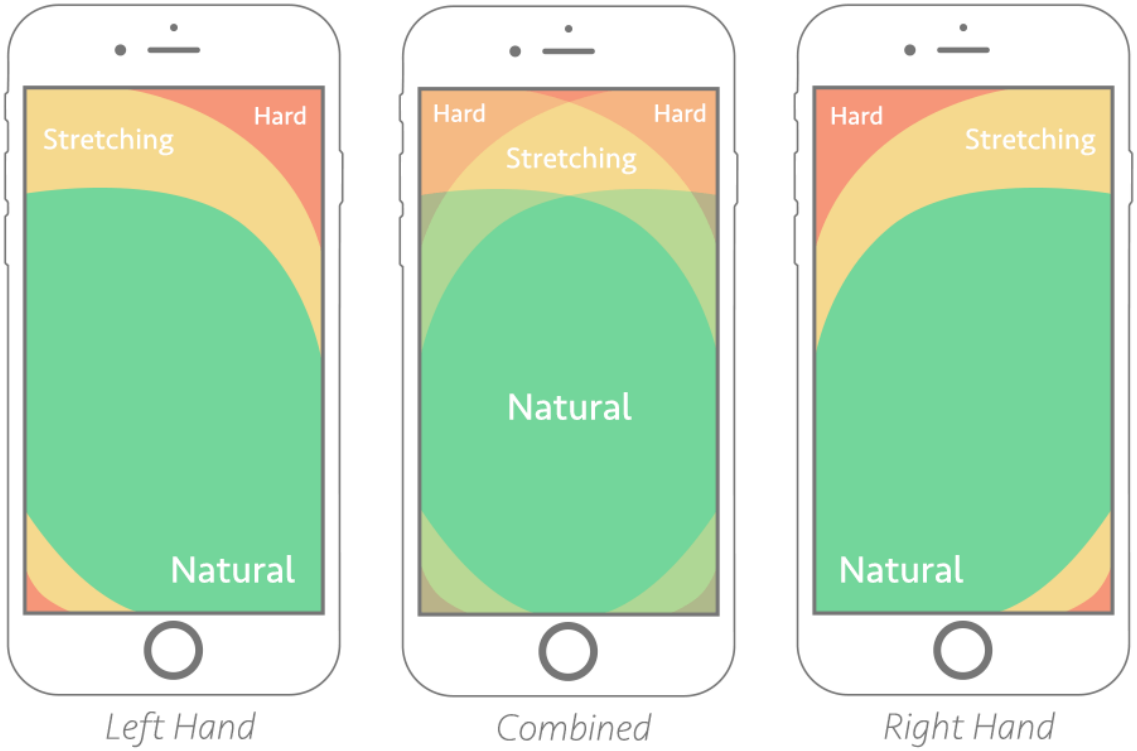


Abbildung 4: Usability Hitmap

In dieser Grafik sieht man eine Hitmap von der Usability eines Handys. Die Buttons sollten in einer Applikation so ausgerichtet werden, sodass die Verwendung einfacher fällt und der Benutzer diese angenehm erreichen kann.

Ausserdem sollte auf bereits Bekanntes und Erlerntes geachtet werden. Elemente- und Bedienprinzipen die bereits in Standard-Apps zu finden ist, sollten auch bei der Entwicklung berücksichtigt werden. So fällt es dem Nutzer leicht die neue App anzuwenden.

Die Applikation sollte natürlich auch nicht überfüllt werden mit Text und Buttons, damit der Benutzer nicht mit dem ersten Anblick überfordert wird. Verschiedene Funktionen und Themen gehören auf verschiedenen Seiten.

4.2 Überlegungen

Ziel war es für mich eine App zu erstellen, welches einfach zu benutzen ist und ein möglichst minimalistisches Design hat. Der Endbenutzer sollte auch immer wissen, wo sie sich in der App genau befinden. Da Nutzer einer To-Do-List-App wahrscheinlich wenig Geduld haben, ist es wichtig, dass sie möglichst schnell dies finden können, wonach sie suchen.

Um Inspiration für das Design zu einzuholen, testete ich diverse ähnliche Apps vom Android Playstore. Dadurch erhielt ich eine Idee von der Platzierung meiner Elemente.

4.3 Mockup

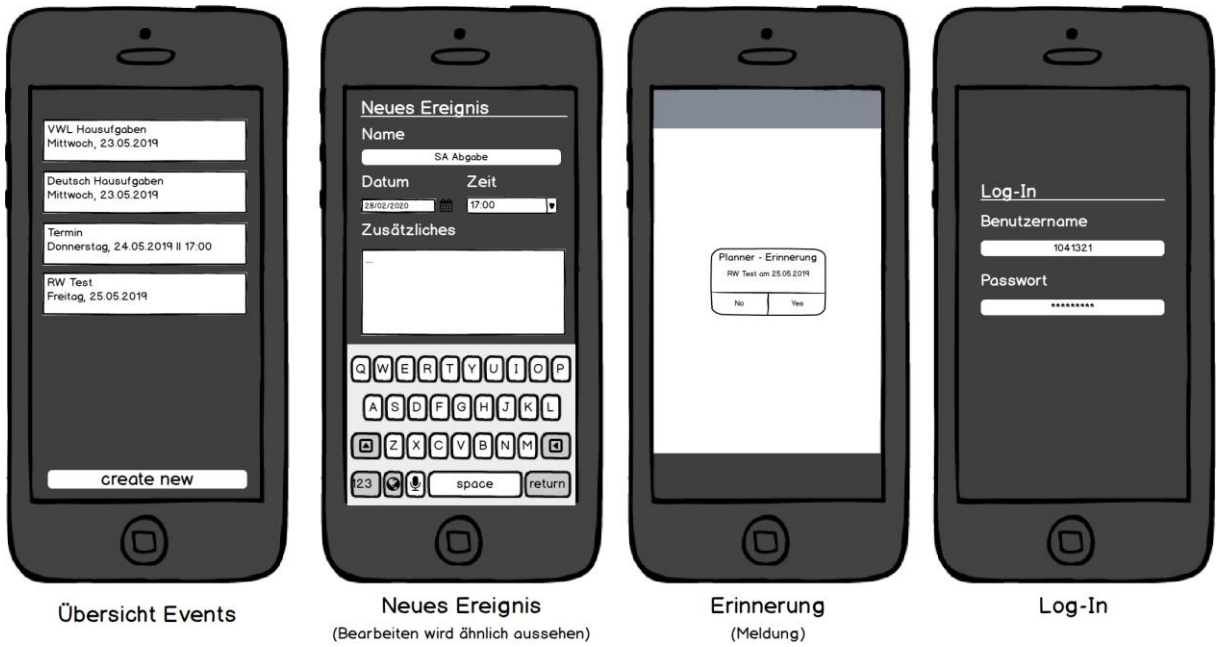


Abbildung 5: Mockup

Für mich war schon von Anfang klar, dass meine App eine Hauptseite haben wird indem die Events aufgelistet werden. Oberhalb der Liste ist eine Toolbar mit Buttons mit kennzeichnenden Icons, welche zu anderen Seiten führt oder Funktionen ausführt. Genauere Informationen sollten nicht auf der Hauptseite direkt gefunden werden, da dies zu einer Überladung von Text führen würde.

4.4 Farbauswahl

Bei der Auswahl von einer Farbpalette sollte darauf geachtet werden, dass die Farben ästhetisch passen und nicht irritierend wirkt. Zusätzlich sollte ein hohes Kontrastverhältnis vorhanden sein, damit die Elemente auch in schlechten Lichtverhältnissen sichtbar sind.

Ursprünglich kolorierte ich meine App in Grün-Tönen jedoch entschied ich mich später um auf die Farben im nachfolgendem Bild.

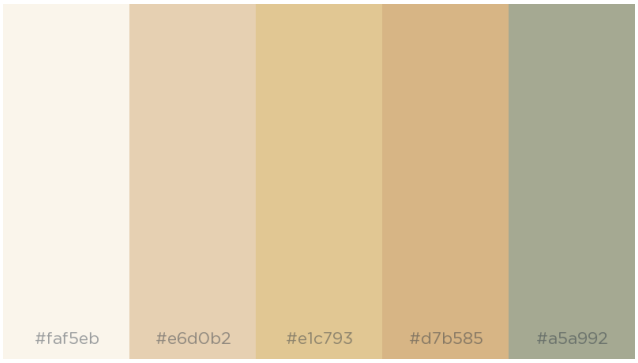


Abbildung 6: Farbschema

4.5 Icons

4.5.1 App

Für meine App habe ich ein Icon gestaltet. Mein Ziel war es, dass es farblich zusammenpasst und den Namen meiner Applikation abbildet.

Abbildung 7: App Icon

4.5.2 Buttons

Anstatt die Buttons zu beschriften, verwendete ich verschiedene selbst-hergestellte Icons, welche ihre Funktion abbilden sollte. So wirkt die Applikation benutzerfreundlicher und interessanter.

Abbildung 15: Kalender Icon

Abbildung 9: Uhr Icon

Abbildung 13: Einstellungen Icon

Abbildung 12: Tag Icon

Abbildung 14: Hacken Icon

Abbildung 11: Stift Icon

Abbildung 10: Suchen Icon

Abbildung 8: Löschen Icon

4.6 Final Design

Abbildung 21: Login

Abbildung 22: Signup

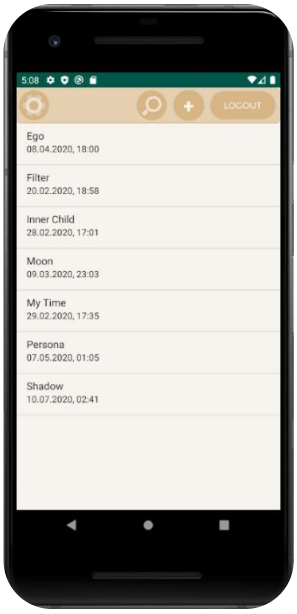


Abbildung 23: Hauptseite

Abbildung 20: New Event

Abbildung 16: Einstellungen

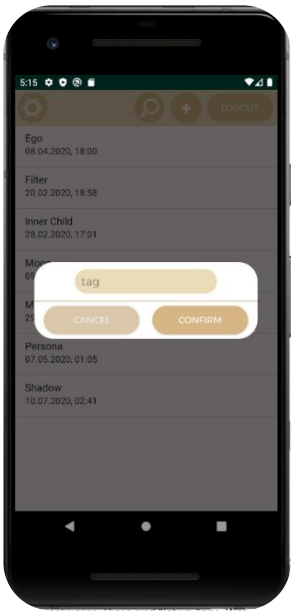


Abbildung 17: Tag Dialog

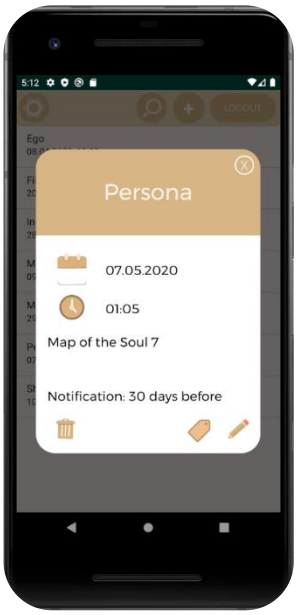


Abbildung 18: Information Dialog

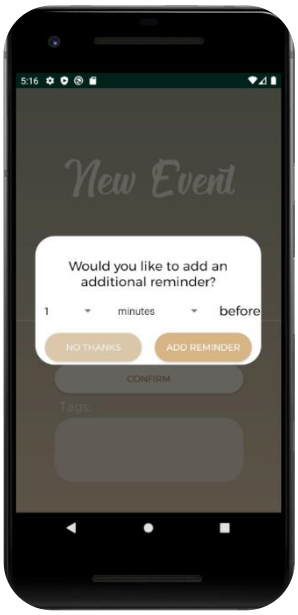


Abbildung 19: Zusätzliche Meldung Dialog

5 Entscheidung

5.1 Bauart

Nach meiner Recherche entschied ich mich eine Native-App zu entwickeln. Der Grund dafür ist, dass ich schon sonst immer Programme im Web-Bereich programmieren musste und nun etwas Neues ausprobieren möchte. Das bietet mir die Native-App-Variante. Zudem auch entschied ich mich für Android, da ich gar nicht

5.2 Verwendete Entwicklungsumgebung

Für die Android-Softwareentwicklung bietet Google die freie Entwicklungsumgebung Android Studio. Auch wenn es diverse Alternativen wie Netbeans oder Eclipse, verwenden die meisten Entwickler Android Studio. Als Grundlage für die Umgebung ist IntelliJ von JetBrains, welches mir bereits bekannt ist.

Abbildung 24: Android Studio

Idea

5.3 Speicherung der Daten

Damit die erstellten Events des Benutzers gespeichert werden können, benötigt mein Projekt eine Datenbank. Die Daten müssen jederzeit aufrufbar sein und da die Struktur nicht allzu komplex aufgebaut ist, suchte ich nach einer einfachen Lösung. Aus meinen vorherigen Projekten kenne ich eine perfekte Variante die Daten zu speichern, nämlich die "realtime Database" von Google. Diese ist NoSQL und unterstützt die Entwicklung von Android Apps was für mich von Vorteil ist.

Abbildung 25: Firebase

Firebase ist eine Plattform, welches von Google gehostet wird. Damit können Android, IOS oder Web Applikationen weiterentwickelt und vermarktet werden. Es bietet verschiedene Tools an, die Entwickler nutzen können um ihre Applikationen zu erstellen. Bei der Verwendung von Firebase wird kein Server Management oder Entwicklung von APIs benötigt.

Für kleine Applikationen, welche wenig Speicherplatz benötigen ist dies Ideal, weil es nicht kostenpflichtig ist. Erst später treten Kosten auf, falls mehr Speicherplatz gewünscht wird.

Die Firebase-Authentifizierung bietet Backend-Dienste, benutzerfreundliche SDKs und vorgefertigte UI-Bibliotheken zur Authentifizierung von Benutzern bei Ihrer App. Es unterstützt die Authentifizierung mithilfe von Passwörtern, Telefonnummern, beliebten Verbundidentitätsanbietern wie Google, Facebook und Twitter und mehr.

6 Realisation

6.1 Datenbank

6.1.1 Struktur

Alle Daten in der realtime Database werden als JSON-Objekte abgelegt.

Im Gegensatz zu einer SQL-Datenbank gibt es keine Tabellen. Wenn Daten dem JSON-Baum hinzufügt werden, wird dieser zu einem Knoten in der vorhandenen JSON-Struktur mit einem zugeordneten Schlüssel.

Meine Idee war es, dass jeder Benutzer ihren eigenen Knoten hat, welche einen weiteren Knoten «events» besitzt in welchen alle Events mit Ihren Daten zu finden sind.

Abbildung 26: Datenbank-Struktur

Die Benutzer-Knoten werden mit ihrer E-Mail gekennzeichnet, da diese einzigartig ist und im Programm jederzeit aufrufbar ist. Falls Daten eines Events verlangt werden, können sie über die E-Mail abgerufen werden.

6.1.2 Verbindung

Um die Firebase Datenbank mit meiner App zu verbinden gibt es zwei Möglichkeiten. Entweder versucht man manuell alles auszuführen oder man nutzt die einfache Firebase Assistant, welche Android Studio zu Verfügung stellt. Da ich Java und allgemein die Entwicklung von Apps nicht kannte, stolperte ich auf viele Fehler auch wenn die Verbindung grundsätzlich sehr simpel ist.

6.1.3 Aufsetzung einer AVD

Um meine App laufend zu testen, musste ich vorerst eine virtuelle Maschine erstellen. So konnte ein Smartphone simuliert werden, welches meine App installiert hatte. Ohne dies, müsste die App immer wieder zusammengepackt, heruntergeladen und installiert werden, welches zeitaufwändiger wäre.

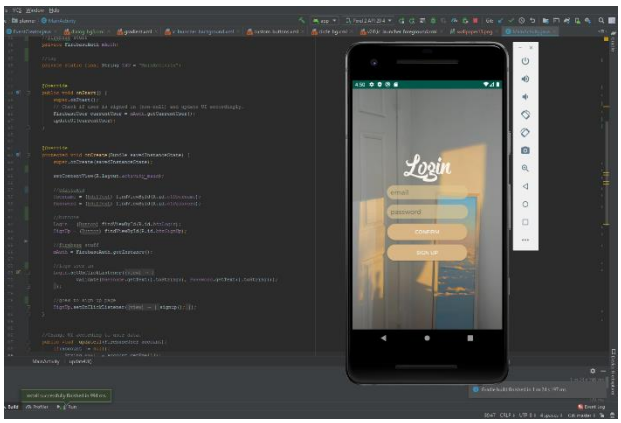


Abbildung 27: AVD

Ein weiterer Vorteil eines solchen Geräts ist, dass die App auf verschiedenen Android Geräten getestet werden kann, auch ohne Besitz der physischen Geräte.

Um eine solche Maschine zu erstellen müssen die meisten Entwickler im BIOS VT-x (Virtualization Technology) aktivieren. Bei mir war das auch der Fall, denn ohne die Einstellung kann die Nachbildung eines Soft- oder Hardware-Objekts nicht ermöglicht werden.

6.2 Klasse Event

In meinem Programm gibt es nur eine, von mir erstellte Klasse. Nämlich die Events.

Alle Datenfelder sind vom Datentyp Strings um die Speicherung zu Vereinfachen.

Es gibt keine besondere Methoden. Alle aufgelisteten Methoden sind ein Konstruktor und viele get & sets.

Die Klasse wurde in der Hälfte der Durchführung des Projektes zu «EventY» umbenannt. Der Grund dafür ist, dass eine Klasse einer Library gleich hiess und um Fehler zu meiden, nannte ich meine eigene Klassen um.

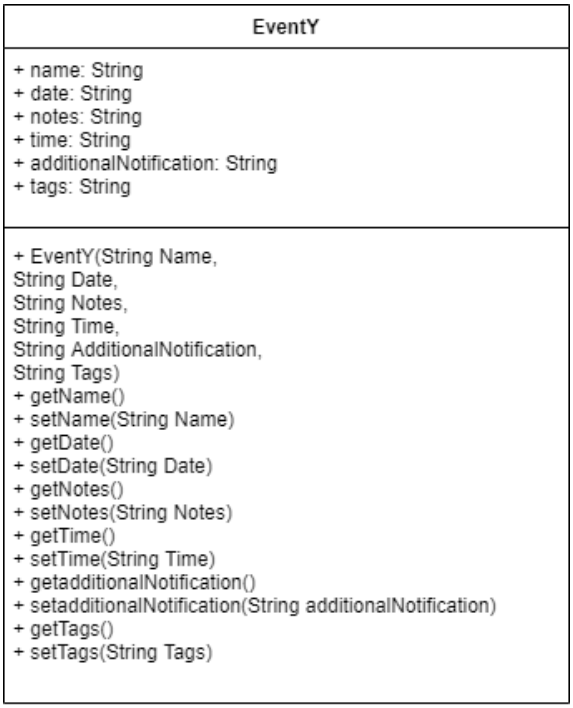


Abbildung 28: Klassendiagramm

6.3 Authentifizierung

Um die Kontoverwaltung zu implementieren, nutze ich das Authentifizierung-System von Firebase. Es verringert nicht nur den Aufwand, sondern es ist auch sicher. Wie bei der realtime Database muss es zuerst mit dem Projekt verbunden werden bevor man die Funktionen nutzen kann.

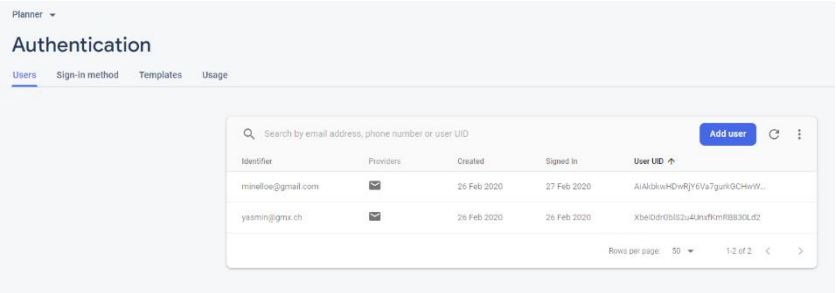


Abbildung 29: Authentifizierung

Unten sind kleine Ausschnitte vom Code, welche die Verwaltung von Kontos ermöglichten.

6.3.1 Wichtige Funktionen

6.3.1.1 Konto Erstellung

Mit diesem Code können neue Kontos erstellt werden.

```
1. mAuth.createUserWithEmailAndPassword(userUsername, userPassword)
```

Falls das Passwort oder die E-Mail nicht die Bedingungen von Firebase erfüllt, erscheint automatisch eine passende Fehlermeldung. Beispiele dafür wären, wenn ein Konto mit der gleichen E-Mail bereits existiert oder wenn das Passwort zu kurz ist.

6.3.1.2 Konto Anmeldung

Ähnlich wie bei der Erstellung gibt es eine Funktion von Firebase, welches den Benutzer anmeldet.

```
1. validate(String userUsername, String userPassword)
```

6.4 Daten speichern

Um einen Event zu speichern, erstellte ich zuerst eine neue Klasse "Event" und konnte so mehrere Event-Objekte erstellen. Diese Objekte konnten danach in die Datenbank gespeichert werden.

```
1.      public EventY(String Name, String Date, String Notes, String Time, String Addi-
    tionalNotification, String Tags){
2.          this.name = Name;
3.          this.date = Date;
4.          this.notes = Notes;
5.          this.time = Time;
6.          this.additionalNotification = AdditionalNotification;
7.          this.tags = Tags;
8.      }
```

Damit ein Event in die Datenbank gespeichert wird, nutzte ich diese Funktion von Firebase:

```
1. mDatabaseReference = mDatabase.getReference().child(EmailToSave + "/events/" + NE-
   Name.getText().toString());
2. mDatabaseReference.setValue(newEvent);
```

6.5 Daten in einer ListView darstellen

Dieser Teil war der aufwendigste Teil meines Projektes und verursachte viele Probleme.

Jedes Element der ListView enthält den Namen auf der ersten Zeile und als «Subitem», also auf der zweiten Zeile, erscheint das Datum und die Zeit.

Um diesen Teil zu ermöglichen musste ich folgend vorgehen:

- Mittels Firebase-Funktionen jeden Event-Namen erhalten
- Name, Datum und Zeit von der Datenbank auslesen
- ArrayAdapter mit zwei Zeilen erstellen und diese mit dem Inhalt der Liste füllen
- Adapter für ListView setzen

Unten ist der Code, den ich dafür kreiert habe.

```
1.        //List with events
2.         final List<String[]> ItemsList = new LinkedList<String[]>();
3.
4.         //custom adapter so that every item in the listview has two rows (+subi-
   tem)
5.         final ArrayAdapter<String[]> adapter = new ArrayAdapter<String[]>(Main-
   Page.this,android.R.layout.simple_list_item_2, android.R.id.text1, ItemsList){
6.             @Override
7.             public View getView(int position, View convertView, ViewGroup parent){
8.                 View view = super.getView(position, convertView, parent);
```

```
9.
10.                 String[] entry = ItemsList.get(position);
11.                 TextView text1 = (TextView) view.findViewById(android.R.id.text1);
12.                 TextView text2 = (TextView) view.findViewById(android.R.id.text2);
13.                 text1.setText(entry[0]);
14.                 text2.setText(entry[1]);
15.                 return view;
16.
17.             }
18.
19.         };
20.
21.
22.         //sets adapter
23.         lvEvents.setAdapter(adapter);
24.
25.         //firebase stuff
26.         FirebaseUser currentUser = mAuth.getCurrentUser();
27.         FirebaseDatabase database = FirebaseDatabase.getInstance();
28.         DatabaseReference myRef = database.getReference();
29.
30.         //gets email, emails are saved with commas instead of dots (fire-
    base doesn't allow dots)
31.         final String email = currentUser.getEmail().replace('.', ',');
32.
33.         //first reference, gets key (name) of every event
34.         myRef.child(email + "/events").addValueEventListener(new ValueEventLis-
    tener() {
35.             @Override
36.             public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
37.
38.                 for(DataSnapshot uniqueKeySnapshot : dataSnapshot.getChildren()){
39.
40.                     //gets key
41.                     String EventName = uniqueKeySnapshot.getKey();
42.                     FirebaseDatabase database = FirebaseDatabase.getInstance();
43.
44.                     //second reference, gets data of event (key)
45.                     DatabaseReference ref2 = database.getRefer-
    ence(email + "/events/" + EventName);
46.                     ref2.addValueEventListener(new ValueEventListener() {
47.                         @Override
48.                         public void onDataChange(DataSnapshot dataSnapshot) {
49.                             String name = dataSnapshot.child("name").get-
    Value().toString();
50.                             String date = dataSnapshot.child("date").get-
    Value().toString();
51.                             String time = dataSnapshot.child("time").get-
    Value().toString();
52.                             String DateTime = date + ", " + time;
53.
54.                             //updates adapter
55.                             adapter.notifyDataSetChanged();
56.
57.                             //every account has an "ig-
    nore" event so that all events can be de-
    leted, this event will not be shown in the listview
58.                             if (name.matches("ignore") != true){
59.                                 //event will be added to the list
60.                                 ItemsList.add(new String[]{name, DateTime});
61.                             }
62.                         }
63.
64.                         @Override
65.                         public void onCancelled(DatabaseError databaseError) {
66.                             Toast.makeText(MainPage.this,"read failed" + databaseEr-
    ror.getCode(),Toast.LENGTH_SHORT).show();
```

```
                        }
                    });
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }


        });
```

6.6 Kalender- & Uhrzeit-Eingabefeld

Um die Eingabe von Daten und Uhrzeiten für den Benutzer zu vereinfachen und sicherzustellen, dass die Werte im richtigen Format sind, werden Dialoge angezeigt. In diesen Dialogen kann die Uhrzeit von einer Uhr und das Datum von einem Kalender ausgewählt werden.

Abbildung 31: Timepicker

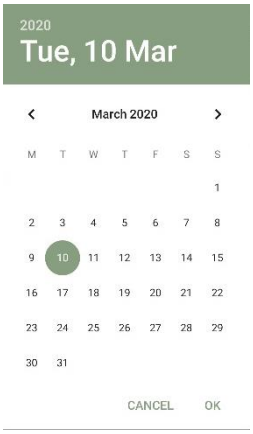


Abbildung 30: Datepicker

6.7 Notifikationen

Ein essentieller Teil meines Projektes sind die Erinnerungen, die auftauchen bei Erreichung eines Datums. Nach langer Recherche und Ausprobieren wendete ich eine Library an, da es alles unkomplizierter machte.

Die Library ist auf GitHub zu finden unter dem Namen «NotifyMe» von jakebonk.

Notifikationen werden mit der Erstellung von Events gesetzt und nachfolgend ist ein Code-Ausschnitt:

```
1.  NotifyMe notifyMeFinal = new NotifyMe.Builder((getApplicationContext()))
2.                          .title(NEName.getText().toString())
3.                          .con-
    tent("Event due! " + NEDate.getText().toString() + ", " + NE-
    Time.getText().toString())
4.                          .color(255,0,0,255)
5.                          .led_color(255,255,255,255)
6.                          .time(dateToSave)
7.                          .addAction(new Intent(), "Snooze",false)
8.                          .key(NEName.getText().toString())
9.                          .addAction(new Intent(),"Dismiss",true,false)
10.                         .addAction(new Intent(),"Done")
11.                         .large_icon(R.mipmap.ic_launcher_round)
12.                         .build(););
```

Mit der «key»-Eigenschaft, welches den gleichen Namen wie das Event besitzt, kann die Notifikation wieder mit cancel() gelöscht werden.

Um eine frühere Notifikation zu erstellen, subtrahierte ich die gewünschte Anzahl Minuten, Stunden oder Tage vom gesetzten Datum und erstellte eine Notifikation mit diesem früherem Datum. Die Anzahl konnte in einem Dialog bei der Erstellung eines neuen Events ausgewählt werden. Der Nutzer der App kann sich auch entscheiden, keine zusätzliche Notifikation zu setzen.

6.8 Kalender

Einer meiner Nice-To-Haves war die Events in einem Kalender abzubilden. Zuerst wollte ich ein Kalender-Widget von Android Studio verwenden, jedoch fand nicht heraus wie die Abbildung von Events funktionierte und ob es überhaupt möglich ist.

Abbildung 32: Calendar-Widget

Die nächste Idee war es eine Library dafür zu verwenden. Ich stiess auf CompactCalendarView von SundeepK auf GitHub (https://github.com/SundeepK/CompactCalendarView).

Es erschien vorerst vielversprechend, da es Video-Anleitungen und eine schriftliche Anleitung dazu gab mit verständlichen Funktionen. Als ich es in mein Projekt einbettete, konnte ich die Events nicht in einer Schleife im Kalender setzen und ich bemerkte, dass nicht mehrere Events an einem Tag gesetzt werden können.

Nach zwei scheiternden Versuchen gab ich schliesslich auf, da mir die Zeit dafür fehlte.

6.9 Tags & Filtern

Das zweite Nice-To-Have meines Projektes war die Zuweisung von Tags, welche das Filtern von Events ermöglichte. Das konnte einfach realisiert werden.

Zu jedem Event können die Tags in einem String zusammengeführt werden und so unter dem Namen «Tags» gespeichert werden (siehe Datenbank-Struktur). Bei der Auslesung der Daten und Füllung der ListView können die Tags jedes Events mit dem gesuchten Tag verglichen werden. Wenn die Tags gleich sind, wird das Event aufgelistet und sonst ignoriert.

Damit die Darstellung von den Kennzeichnungen schöner ist, nutzte ich die Library AndroidTagGroup von 2dxgujun auf GitHub (https://github.com/2dxgujun/AndroidTagGroup). Die Library funktionierte ähnlich wie eine Liste. Es gibt eine «TagGroup» in dem beliebig viele Tags hinzugefügt und gelöscht werden können. Wenn ein Tag angeklickt wird, kann dies mittels einer Funktion erkannt werden.

Abbildung 33: AndroidTagGroup

6.10 .APK erstellen

Um die App auf meinem Smartphone zu installieren, brauchte ich die APK-Datei meines Projektes. Diese APK-Datei kann auf allen Android-Geräten installiert werden, denn Android unterstützt die Installation von externen Applikationen.

Mein Projekt konnte ich auf Android Studio als APK-Datei herunterladen, auf meinem Gerät übertragen und schlussendlich installieren.

7 KONTROLLE

7.1 Rahmen

Nebst die Verwendung einer AVD (siehe Realisierung, AVD) testete ich die Applikation auf zwei verschiedene Android Smartphones. Mehrheitlich nutze ich ein LG G6, jedoch wurden die neuen Versionen meiner App auf ein Honor Lite installiert und getestet.

7.2 Testfälle

Getestet von: Yasmin Häberli
Verwendete Version: Final-Version
Betriebssystem: Android
Test-Datum: 27.02.2020

Tabelle 1: Testfälle

Login				
Test	Erwartetes Ergebnis	Ergebnis	Bestanden	Kommentar
normal				
Geben Sie "test@gmx.ch" im E-Mail-Feld ein und "123456" im Passwort-Feld ein. Nachfolgend drücken Sie auf "confirm"	Die Hauptseite erscheint.	Die Hauptseite erscheint.	☒	-
Drücken Sie auf "Sign Up"	Die Signup-Seite erscheint	Die Signup-Seite erscheint	☒	-
Destruktiv				
Geben Sie "z@gmail.com" im E-Mail-Feld ein und im Passwort-Feld "444" ein.	Es erscheint eine Meldung, dass es dieses Konto nicht gibt.	Es erscheint eine Meldung, welche mitteilt, dass die Authentifizierung fehlgeschlagen ist.	☒	Andere Meldung, jedoch gleiches gewünschtes Resultat
Geben Sie 5-Mal "test@gmail.com" im E-Mail-Feld ein und im Passwort-Feld "1234567" ein.	Der Nutzer wird blockiert und kann sich nicht einloggen.	Der Nutzer wird blockiert und kann sich nicht einloggen.	☒	-
Signup				
normal				

Geben Sie im E-Mail-Feld "neu@gmx.ch" ein und im Passwort-Feld "123456" ein. Nachfolgend drücken Sie auf "confirm"	Ein neues Konto wird erstellt und die Hauptseite erscheint.		☒	-
Destruktiv				
Geben Sie im E-Mail-Feld "neu@gmx.ch" ein und im Passwort-Feld "123" ein. Nachfolgend drücken Sie auf "confirm"	Es erscheint eine Meldung, welche mitteilt, dass das Passwort zu schwach ist.	Es erscheint eine Meldung, welche mitteilt, dass das Passwort zu schwach ist.	☒	-
Geben Sie im E-Mail-Feld "test@gmx.ch" ein und im Passwort-Feld "123456" ein. Nachfolgend drücken Sie auf "confirm"	Es erscheint eine Meldung, welche mitteilt, dass es dieses Konto schon gibt.	Es erscheint eine Meldung, welche mitteilt, dass es dieses Konto schon gibt.	☒	-
Geben Sie im E-Mail-Feld "test@ch" ein und im Passwort-Feld "123456" ein. Nachfolgend drücken Sie auf "confirm"	Es erscheint eine Meldung, welche mitteilt, dass keine gültige E-Mail eingetragen wurde.	Es erscheint eine Meldung, welche mitteilt, dass keine gültige E-Mail eingetragen wurde.	☒	-
Hauptseite				
normal				
Drücken Sie auf das Plus-Symbol.	Eine Seite erscheint, auf welche neue Events erstellt werden kann.	Eine Seite erscheint, auf welche neue Events erstellt werden kann.	☒	-
Drücken Sie auf den Logout-Button.	Die Login-Seite erscheint und das Konto wurde abgemeldet.	Die Login-Seite erscheint und das Konto wurde abgemeldet.	☒	-

Drücken Sie auf den Einstellungen-Button.	Die Einstellungs-Seite erscheint.	Die Einstellungs-Seite erscheint.	☒	-
Drücken Sie auf das erste Event.	Ein Dialog erscheint, in welchem die Informationen des dazugehörigen Events erscheint.	Ein Dialog erscheint, in welchem die Informationen des dazugehörigen Events erscheint.	☒	-
Drücken Sie auf das erste Event und anschliessend auf das Tag-Symbol.	Ein Dialog erscheint, welches die Tags des Events darstellt.	Ein Dialog erscheint, welches die Tags des Events darstellt.	☒	-
Drücken Sie auf das erste Event und anschliessend auf das Stift-Symbol.	Die Daten des jeweiligen Events können nun in einem Dialog bearbeitet werden.	Die Daten des jeweiligen Events können nun in einem Dialog bearbeitet werden.	☒	-
Drücken Sie auf das erste Event und anschliessend auf das Stift-Symbol. Verändern Sie den Namen auf «test» und drücken Sie nachher auf den Hacken-Symbol.	Die Hauptseite erscheint und der Name wurde verändert.	Die Hauptseite erscheint und der Name wurde verändert.	☒	-
Drücken Sie auf das erste Event und anschliessend auf das Abfalleimer-Symbol. (Lösch-Symbol)	Die Hauptseite erscheint, dieses Event wurde gelöscht.	Die Hauptseite erscheint, dieses Event wurde gelöscht.	☒	-
Drücken Sie auf den Such-Button.	Ein Dialog erscheint, in welchem ein Tag eingegeben werden kann.	Ein Dialog erscheint, in welchem ein Tag eingegeben werden kann.	☒	-
Drücken Sie auf den Such-Button und suchen Sie nach «bts»	In der Liste von Events erscheinen nur diese mit «bts» Tags.	In der Liste von Events erscheinen nur diese mit «bts» Tags.	☒	-

Neue Events				
normal				
Geben Sie im Namen-Feld "Test 1", im Datum-Feld auf das aktuelle Datum im Zeit-Feld die aktuelle Zeit + 5 Minuten, im Notiz-Feld "-", bei den Tags «test» & «app» ein und stellen sie eine weitere Notifikation auf «5 minutes before». Nachfolgend drücken Sie auf "confirm"	Ein neuer Termin (Event) wird erstellt, die Hauptseite erscheint mit dem neu eingegebenen Termin. Zwei Notifikationen erscheinen. Eine war die automatisch erstellte Erinnerung und die andere war die zusätzliche.	Ein neuer Termin (Event) wird erstellt, die Hauptseite erscheint mit dem neu eingegebenen Termin. Zwei Notifikationen erscheinen. Eine war die automatisch erstellte Erinnerung und die andere war die zusätzliche.	☒	-
Geben Sie im Namen-Feld "Test 1", im Datum-Feld auf das aktuelle Datum im Zeit-Feld die aktuelle Zeit + 5 Minuten, im Notiz-Feld "-", bei den Tags «test» & «app» ein und stellen Sie keine zusätzliche Notifikation. Nachfolgend drücken Sie auf "confirm"	Ein neuer Termin (Event) wird erstellt, die Hauptseite erscheint mit dem neu eingegebenen Termin. Eine Notifikation erscheint.	Ein neuer Termin (Event) wird erstellt, die Hauptseite erscheint mit dem neu eingegebenen Termin. Eine Notifikation erscheint.	☒	
Drücken Sie auf "date".	Ein kleiner Popup erscheint, auf welchen man das gewünschte Datum in einem Kalender auswählen kann.	Ein kleiner Popup erscheint, auf welchen man das gewünschte Datum in einem Kalender auswählen kann.	☒	
Drücken Sie auf "time"	Ein kleiner Popup erscheint, auf welchen man die ge-	Ein kleiner Popup erscheint, auf welchen man die ge-	☒	

	wünschte Uhrzeit in einer Uhr auswählen kann.	wünschte Uhrzeit in einer Uhr auswählen kann.		
Settings				
normal				
Drücken Sie auf «Reset Password»	Ein Popup erscheint, in dem ein neues Passwort eingegeben werden kann.	Ein Popup erscheint, in dem ein neues Passwort eingegeben werden kann.	☒	-
Drücken Sie auf «Reset Password» und geben Sie «654321» als neues Passwort ein.	Passwort wurde geändert.	Passwort wurde geändert.	☒	-
Drücken Sie auf «Delete Account»	Konto wird gelöscht und die Login-Seite erscheint.	Konto wird gelöscht und die Login-Seite erscheint.	☒	-
Drücken Sie auf «Delete All Events»	Es werden alle Events gelöscht. Die Hautpseite ist leer.	Es werden alle Events gelöscht. Die Hautpseite ist leer.	☒	-

7.3 Known-Bugs

- Wenn man den Zurück-Button drückt bei der Änderung der Daten eines Events, erscheint die ListView auf die Hauptseite leer
- Layout-Unterteilung bei der Einstellungen-Seite wird nicht auf allen Geräten korrekt abgebildet
- Wenn 1 Minute, 1 Stunde oder 1 Tag bei der zusätzlichen Notifikation gewählt wird, bleibt die jeweilige Bezeichnung in plural (z.B. «1 minutes before»
- Wenn eine Event bearbeitet oder gelöscht wird, stürzt die App ab. Um das zu vermeiden, ruft die App die Hauptseite nochmals auf. Die Lösung ist nicht sehr schön
- Daten können ohne Internet-Verbindung nicht abgerufen werden

7.4 Auswertung

Alle meiner Must-Haves und einer meiner Nice-To-Haves wurden erfolgreich implementiert. Durch die Auslassung der Abbildung der Events in einem Kalender konnte ich andere Teile einbauen wie die Einstellungen, Dialoge zur Auswahl von Datum und Zeit und die visuelle Darstellung von Tags.

Natürlich sind kleine Bugs im Programm zu finden aber der Grossteil ist fehlerfrei. Mit ein wenig mehr Zeit könnten die letzten Probleme gelöst werden und das letzte «Nice-To-Have» realisiert werden.

Grundsätzlich bietet die Applikation viele Ausbaumöglichkeiten, aber es fehlt die Zeit dafür.

Wie wahrscheinlich bei vielen anderen fiel es mir schwer mich an den Meilensteinen zu halten. Die ersten zwei konnte ich mehr oder weniger erreichen, nur die vor der Abgabe konnte überhaupt nicht eingehalten werden.

Trotz vielen Problemen und Zeitstress entwickelte ich eine App, worauf ich stolz sein konnte. Meine Applikation könnte ich auch sonst im Alltag verwenden.

8 Quellenverzeichnis

8.1 Abbildungen

8.2 Tabellen

8.3 Libraries

Notifikationen – NotfiyMe: https://github.com/jakebonk/NotifyMe
Tags - AndroidTagGroup: https://github.com/2dxgujun/AndroidTagGroup
Kalender – CompactCalendarView : https://github.com/SundeepK/CompactCalendarView

8.4 Information

Information zu Web-Apps: https://de.yeeply.com/blog/web-app-erstellen-vorteile-nachteile/
Information zu Native-Apps: https://www.mobiloud.com/blog/native-web-or-hybrid-apps/#5

E-Mail Validation: https://www.geeksforgeeks.org/check-email-address-valid-not-java/
Übergänge: https://codinginflow.com/tutorials/android/slide-animation-between-activities

Google Firebase: https://firebase.google.com/docs/
StackOverflow: https://stackoverflow.com/
Draw.io: https://www.draw.io/

8.5 Tutorials

ListView + OnClick Items: https://www.youtube.com/watch?v=kCJv5YWHRXQ
.APK-Export: https://www.youtube.com/watch?v=BfuOn5LuOA4
Notfikationen (Library): https://www.youtube.com/watch?v=oLngUva-Ves

9 Anhang

9.1 Installationsanleitung

Leider ist die App nicht aus dem Android Playstore installierbar, aber die App kann auch auf GitHub gefunden werden. Im «Planner»-Repository ist der Code sowie auch die .APK-Datei, welches die App direkt installiert.

https://github.com/minelloe/Planner/tree/master/app/release

Unter «Planner/app/release/» kann «app-release.apk» gefunden und heruntergeladen werden.

Um die Installation auf dem Smartphone auszuführen muss in den Einstellungen zuerst die Installation von Apps von externen Quellen erlaubt werden. Es werden Warnungen erscheinen, aber da meine App ungefährlich ist, können diese ignoriert werden.

9.2 Benutzeranleitung

Das meiste sollte eigentlich selbsterklärend sein, denn die verschiedenen Funktionen sind ähnlich zu bereits bekannten Apps.

9.2.1 Hauptseite

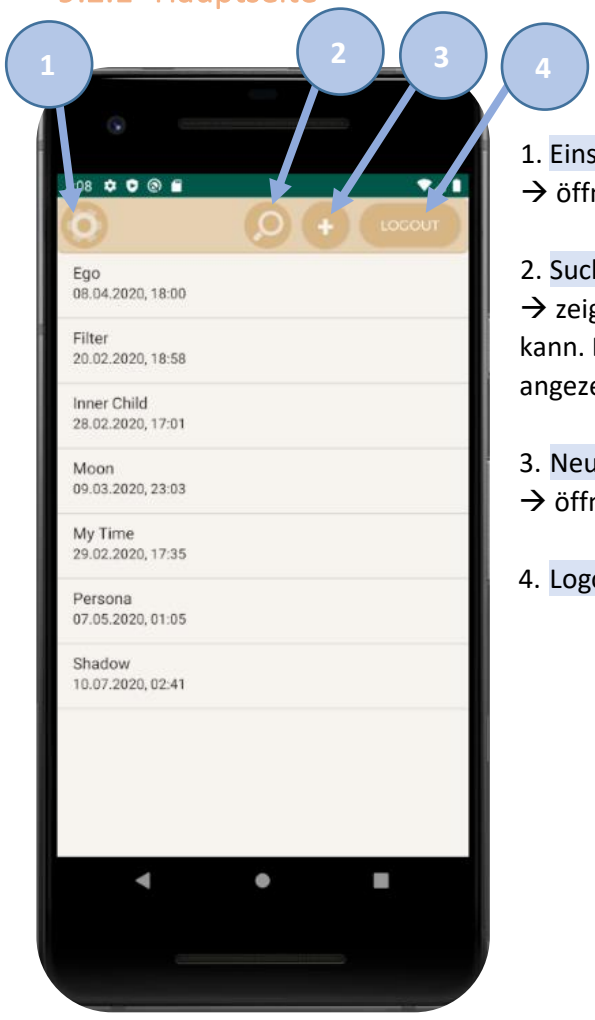


Abbildung 34: Hauptseite

1. Einstellungen
→ öffnet Seite mit Einstellungen

2. Suchen
→ zeigt ein Dialog, in welchem ein Tag eingegeben werden kann. Bei der Bestätigung werden nur Events mit diesem Tag angezeigt werden

3. Neues Event
→ öffnet Seite in dem ein neues Event erstellt werden kann

4. Logout

9.2.2 Popup mit Eventinformationen

1. Cancel
   → schliesst Popup

2. Löschen
   → löscht Event

3. Tag-Übersicht
   → zeigt alle Tags an, die zu dieser Event gehören

4. Event Bearbeiten
   → Daten können mittels Textfelder verändert werden.

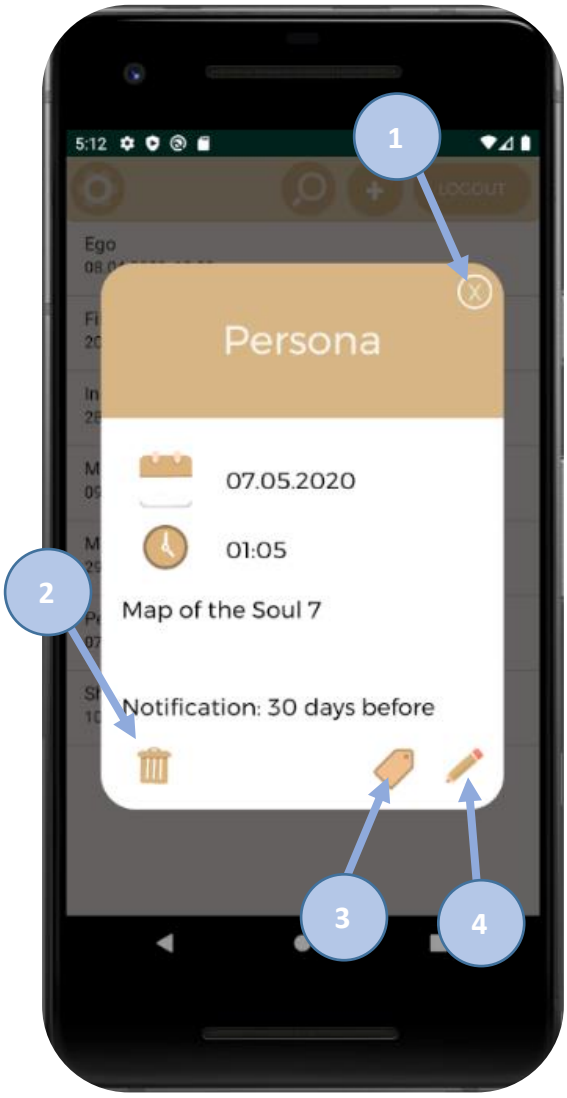


Abbildung 35: Information Dialog

9.3 Arbeitsjournal

	Das Arbeitsjournal				
Datum	Vorhaben	Geleistete Arbeit	Zeit geplant	Zeit effektiv	Bemerkungen, nächste Arbeitsschritte
18.10.2019	• Erstellung der Dokumentation • Anfang Recher-che • Anfang Planung	• Erstellung der Dokumentation • Anfang Recherche • Anfang Planung (Dokumenta-tion) • Vergleich von Bau-arten von Apps (Dokumentation)	2 h	~ 2 h	• Zeitplan muss als nächstes erstellt werden
22.11.2019 (Meilenstein 1)	• Planung weiterführen	• Planung weiterge-führt	1h	1h	• Meilenstein 2 wurde nicht ganz erreicht
28.12.2019	• Use-Cases • Setup And-roid Studio	• Setup And-roid Studio • Aufsetzung einer AVD	2h	1h	• Einfache In-stallation von Android Studio • Probleme bei der Aufsetzung einer AVD
29.12.2019 (Meilenstein 2)	• Anfang Programm	• Layout der App	2h	2h	• Meilenstein 2 mehr oder weniger erreicht
02.01.2020	• Nachführung Dokumentation (Planung & Entscheid)	• Nachführung Dokumentation	2h	3h	-
04.01.2020	• Log-In & Sign Up erstellen • Verbindung al-ler Activities	• Log-In erstellt • Verbindung von Activites	1h	1h	-
05.01.2020	• Verbindung Da-tenbank (Firebase realtime Data-base & Authenti-cation) • LogIn fertigstellen	• Verbindung Da-tenbank (Firebase realtime Data-base & Authenti-cation)	2h	3h	• Imports ver-gessen & dadurch sehr viel Zeit ver-loren
07.01.2020	• LogIn fertigstellen	• LogIn fertigstellen	30min	45min	• SignUp fehlt
09.01.2020	• SignUp erstel-len und mit Lo-gIn verbinden	• SignUp erstellt und mit LogIn verbunden	1.5h	1.5h	
11.01.2020	• Nachführung Dokumentation	• Speicherung der Daten, kurzer Text über Firebase (Dokumentation) • Use-Cases	45min	45min	-
13.01.2020	• Nachführung Dokumentation	• Nachführung Do-kumentation (ein-zelne kleinere Teile)	1h	30min	-

29.12.2019 (Meilenstein 3)	-	-	-	-	Meilenstein 3 nicht erreicht
03.02.2020	• Nachführung Dokumentation	• Nachführung Dokumentation (Realisation)	1.5h	1.5h	-

04.02.2020	• Problem mit Auflistung der Events (List-View) lösen	• Frustration, keine Lösung wurde gefunden	2h	3h	• Keine Lösung für das Problem, möglicherweise muss eine Alternative gesucht werden
05.02.2020	• Nachführung der Dokumentation	• Nachführung der Dokumentation (Testfälle, Realisation)	1.5h	1.5h	-
05.02.2020	• Problem mit Auflistung der Events (List-View) lösen	• Problem gelöst	2h	2h	-
08.02.2020	• Popup erstellen, um Information über Event abzubilden	• Popup erstellt	30min	45min	• Noch nicht fertig
09.02.2020	• Popup fertigstellen & programmieren, sodass die korrekten Daten angezeigt werden	• Popup fertig • Events können bearbeitet werden • Events können gelöscht werden	4h	6h	-
10.02.2020	• Nachführung der Dokumentation, Farbänderung der Icons	• Nachführung der Dokumentation	1.5h	1.5h	• Icons werde ich später machen
11.02.2020	• Farbänderung der Icons, Start mit Programmierung der Notifikationen, Anpassung der Grössen	• Projekt endlich auf GitHub gestellt, Grössen angepasst, Notifikation erstellt	2h	4h	
12.02.2020	• Farbänderung der Icons, Nachführung der Dokumentation	• Farbänderung der Icons, Nachführung der Dokumentation	1.5h	1.5h	-
13.02.2020	• Programmierung, sodass Events im Kalender dargestellt werden	• Anfang Programmierung des Kalenders	1h	1h	• Problem: Klasse einer Library nutzt den gleichen Namen wie meine Klasse

15.02.2020	• Ausbesserung der Notifikationen, Kalender fertig-stellen	• Weiterarbeit am Kalender, Anfang Ausbesserung der Notifikationen	4h	4h	• Problem beim Kalender: Events er-scheinen nicht
17.02.2020	• Ausbesserung der Notifikationen	• Ausbesserung der Notifikationen	1h	1h	• Noch nicht fertig
18.02.2020	• Ausbesserung der Notifikationen	• Ausbesserung der Notifikationen	2h	2h	• Noch nicht fertig
19.02.2020	• Ausbesserung der Notifikationen	• Ausbesserung der Notifikationen	4h	4h	• Noch nicht fer-tig, Problem konnte bis jetzt noch nicht ge-löst werden

20.02.2020	• Nachführung der Dokumenta-tion	• Nachführung der Dokumenta-tion	45min	45min	-
20.02.2020	• Ausbesserung der Notifikationen	• Ausbesserung der Notifikationen	2h	2h	Problem gelöst
22.02.2020	• Ausbesserung der App, Fehlererken-nung, Design, usw.	• Mehrere Fehlerbe-hebungen, List-View neu mit Sub-items (2 Stufen), neues Design, Animationen	6h	6h	-
23.02.2020	• Fehlerbehe-bung, Anfang Tags (Nice-To-Have)	• Animierte Über-gänge, Fehler-behebung, An-fang Einstellungsseite	2h	2.5h	-
24.02.2020	• Weiterbearbeitung der Einstellungen, Dokumentation Nachführung	• Weiterbearbeitung der Einstellungen, Dokumentation Nachführung	1h	1h	-
25.02.2020	• Weiterbearbeitung der Einstellungen, Bearbeitung Nice-To-Have, Doku-mentation Nachführung	• Einstellungen fer-tig, Anfang Tags	5h	6h	-
26.02.2020	• Bearbeitung Tags, Symbole er-setzen, Icon für App fertigstellen, Nachführung der Dokumentation	• Fertigstellung der App	6h	6.5h	-
27.02.2020	• Fertigstellung der Dokumentation, Abgabe	• Fertigstellung der Dokumenta-tion, Abgabe	7h	7h	-
Total			74h	82h	