



TEKO Luzern

Pilatusstrasse 38

6003 Luzern

Promille-Rechner „Kalkohol“

Eine Software-Applikation zur

Berechnung des Blutalkoholgehaltes

Projektarbeit 2.Semester

in der Weiterbildung zum

Dipl. Techniker/in HF Informatik

vorgelegt von: Patrick Wyser Janik Schilter
Huobmattstrasse 10 Chilenmattli 7
6045 Meggen 6055 Alpnach Dorf
patrickwysi@hotmail.com janik.schilter@gmail.com

Gutachter: Fachlehrer Projektmanagement
Bruno Hammer
teko@dasycon.ch



1 Management Summary

Es stehen bereits diverse Promillerechner zum Download bereit, aber bis dato keine kostenlose und werbefreie Applikation für Mobiltelefone, welche den Blutalkoholgehalt ohne Angabe der vergangenen Zeit, seit der Einnahme des Getränktes errechnet. (Stand: Mai 2020)

Aus dieser Idee entstand das Projekt Kalkohol. Es wurde eine «Live-Tracking»-Funktion, welche eine unkomplizierte Bedienung ermöglicht entwickelt. Der Benutzer wählt ganz einfach sein derzeitiges Getränk aus und die Applikation registriert den Zeitpunkt der Einnahme und berechnet Promillewert, Alkoholabbau und den Zeitpunkt des nüchternen Zustandes.

Kalkohol wurde als reine Android-Applikation in der IDE Android Studio entwickelt. Die Vorteile von Android Studio sind:

- vorgegebene Projektstrukturen für Android-Applikationen
- Kompatibilität mit Android-Geräten
- schneller Emulator

Die fertiggestellte Applikation überzeugt durch ansprechendes Design, intuitive Handhabung und innovativer «Live-Tracking»-Funktion. Die Wünsche und Erwartungen der Projektbeteiligten wurden erfüllt.

Nach Fertigstellung der Software-Entwicklung waren alle Musskriterien vollumfänglich erfüllt. Die Wunschkriterien wurden aus zeitlichen Gründen nicht berücksichtigt. Die Projektbeteiligten behalten sich vor, die Wunschkriterien zu gegebener Zeit, ausserhalb der Projektarbeit zu erfüllen.



Abbildung 1 - GUI Vorstellung



2 Inhalt

1	Management Summary	2
3	Abbildungsverzeichnis	5
4	Einleitung	6
4.1	Vorwort	6
4.2	Erwartungen	6
4.3	Die Projektbeteiligten	6
5	Richtlinien / Aufgabenstellung	7
6	Analyse	8
6.1	Ist-Zustand	8
6.1.1	Beispiel App: Promille von Christoph Riepe	8
6.2	Konzept	9
6.2.1	Funktionsübersicht	9
6.2.2	Entwurf GUI-Design	10
6.2.3	Die Getränke	11
6.3	Vorstudie	14
6.4	Termin- und Zeitmanagement	15
7	Pflichtenheft	16
7.1	Zielbestimmung	16
7.2	Produkteinsatz	16
7.3	Produktfunktionen	16
7.4	Qualitätsanforderungen	17
7.5	Benutzeroberfläche	17
7.6	Technische Produktumgebung	17
8	Softwareentwicklung	18
8.1	GUI	18
8.1.1	GUI Design	18
8.1.2	Entwicklungsumgebung	21
8.1.3	GUI-Elemente	22
8.2	Backend	26
8.2.1	BAK-Formel	26
8.2.2	Alkoholabbau	27
8.2.3	Kalkulation im Backend	28
9	Testbericht / Auswertung	35
9.1	1. Testphase	35
9.2	2. Testphase	36
10	Veröffentlichung	37



10.1	Google Play-Store	37
11	Schlussstein	38
11.1	Sachergebnisse	38
11.1.1	Ursprüngliche Aufgabenstellung und Ziele.....	38
11.1.2	Veränderungen der Aufgabenstellung / Zielsetzung	38
11.1.3	Erarbeitete Sachergebnisse	38
11.2	Projektverlauf	39
11.2.1	Überblick Kosten- und Zeitmanagement.....	39
11.2.2	Planungsqualität.....	40
11.3	Ausblick.....	41
11.3.1	Restaktivität.....	41
11.3.2	Ergänzungen und Erweiterungen.....	41
11.3.3	Folgeprojekte.....	41
11.4	Schlusswort / Reflexion.....	42
11.4.1	Patrick Wyser	42
11.4.2	Janik Schilter	42
12	Anhang	43
12.1	Projektunterlagen	43
12.2	GIT-Repository	43
12.3	Quellenangaben	43
12.4	Sitzungsprotokolle	44



3 Abbildungsverzeichnis

Abbildung 1 - GUI Vorstellung	2
Abbildung 2 - Patrick Wyser	6
Abbildung 3 - Janik Schilter	6
Abbildung 4: Beispiel App - Zeitraum	8
Abbildung 5: Beispiel App - Getränkeauswahl	8
Abbildung 6: Beispiel App - Ergebnis	8
Abbildung 7: Flussdiagramm	9
Abbildung 8: Entwurf GUI-Design	10
Abbildung 9: Stange	11
Abbildung 10: Kübel	11
Abbildung 11: Pitcher	11
Abbildung 12: Rotwein	12
Abbildung 13: Weisswein	12
Abbildung 14: Rosé	12
Abbildung 15: Cuba Libre	12
Abbildung 16: Long Island	12
Abbildung 17: Vodka Lemon	12
Abbildung 18: Vodka Shot	13
Abbildung 19: Gin Shot	13
Abbildung 20: B52	13
Abbildung 21: Android default	14
Abbildung 22: Zeitplan komprimiert	15
Abbildung 23: GUI Hinweis	18
Abbildung 24: GUI Main	18
Abbildung 25: GUI User	19
Abbildung 26: GUI Wertanzeige	19
Abbildung 27: Toast Getränk	20
Abbildung 28: Toast Profil	20
Abbildung 29: Toast Reset	20
Abbildung 30: Entwicklungsumgebung	21
Abbildung 31: Autofill hint color	22
Abbildung 32: Constraint_Layout	23
Abbildung 33: TextView Beispiel	23
Abbildung 34: Button User	24
Abbildung 35: ExpandableList	24
Abbildung 36: NumberPicker	25
Abbildung 37: RadioButton Kalkohol	25
Abbildung 38: RadioButton default	25
Abbildung 39: Widmark-Formel	26
Abbildung 40: Kalkulation	28
Abbildung 41: Widmark-Formel	29
Abbildung 42: Timestamp	31
Abbildung 43: Timer	32
Abbildung 44: Anzeige – wieder nüchtern	33
Abbildung 45: Anzeige BAK-Wert	34
Abbildung 46: Projektverlauf Zeitplan	39



4 Einleitung

Während der Weiterbildung zum Dipl. Techniker/in HF Informatik werden mehrere Projektarbeiten durchgeführt. Wir befinden uns nun im 2. Semester unserer Weiterbildung und dürfen hiermit unsere erste Projektarbeit erarbeiten. Ziel dieser Arbeit ist, ein Projektthema aus der Weiterbildung zu wählen und bei der Durchführung unsere Kenntnisse in verschiedenen Bereichen der Weiterbildung zu festigen und erweitern.

Der inoffizielle Leitspruch dabei lautet: „Fehler machen – und aus Ihnen Lernen!“

4.1 Vorwort

Wir haben uns dazu entschieden eine Software-Applikation zu entwickeln, um unsere Kenntnisse im Bereich der Applikationstechnik zu erweitern und vertiefen. Nach reichlichem Überlegen sind wir zum Schluss gekommen, dass wir eine reine Unterhaltungs-Applikation entwerfen möchten und entschieden uns dazu, einen Promille-Rechner zu entwickeln und diesen im Google Play Store zum Download zur Verfügung zu stellen.

4.2 Erwartungen

Die Applikation Kalkohol ist der ideale Begleiter für jeden, der gerne ab und zu Alkohol trinkt. Bestimmt jeder Trinkbegeisterte hat sich schon gefragt, wie viele Promille er eigentlich intus hat und wie lange es dauert bis man wieder nüchtern ist.

Obwohl wir unsere Applikation kostenlos zum Download zur Verfügung stellen, würden wir uns natürlich über angemessene Downloadzahlen und zufriedene Benutzer freuen.

4.3 Die Projektbeteiligten

Projektleiter

Patrick Wyser
Huobmattstrasse 10
6045 Meggen
patrickwysi@hotmail.com



Abbildung 2 - Patrick Wyser

Auftraggeber

Janik Schilter
Chilenmattli 7
6055 Alpnach Dorf
janik.schilter@gmail.com



Abbildung 3 - Janik Schilter



5 Richtlinien / Aufgabenstellung

Untenstehend finden Sie eine Zusammenfassung der Richtlinien für die Projektarbeit Techniker HF Informatik 2020:

Umfang: Die Richtzeit beträgt ca. 40 Stunden pro Studenten.

Projektthema: Themen aus einem der folgenden Gebiete:

- Software-Engineering
- Datenbank
- Systemtechnik
- Netzwerktechnik / Telekommunikation

Betreuung durch: Bruno Hammer, Fachlehrer Projektmanagement

Start Projektarbeit: Montag, 18. Mai 2020

Besprechungstermine:

- 1. Termin: Mittwoch, 3. Juni 2020, 16:30 Uhr
- 2. Termin: Mittwoch, 26. August 2020, 16:30 Uhr

Abgabe der Arbeit: Samstag, 19. September 2020

Präsentation: Samstag 26. September 2020, 13:00 Uhr

Publikation: Management Summary → wird im Internet publiziert
Gesamte Arbeit → nur nach Absprache mit Betreuer



6 Analyse

6.1 Ist-Zustand

Um den Ist-Zustand zu definieren haben wir die Plattformen App Store von Apple® und den Play Store von Google® durchsucht. Es gibt bis dato, keine kostenlose oder werbefreie Applikation für Mobiltelefone, welche den Blutalkoholgehalt ohne Angabe der vergangenen Zeit, seit der Einnahme des Getränkes errechnet. (Stand: Mai 2020) Es gibt zwar viele Blutalkoholgehalt Rechner, keine Applikation beinhaltet jedoch eine «Live-Tracking» Funktion.

Die meisten Applikationen funktionieren wie im folgenden Beispiel gezeigt:

6.1.1 Beispiel App: Promille von Christoph Riepe

Auf der [Abbildung 5](#) ist zu sehen, dass ein eingestelltes Profil mit Alter, Geschlecht und Gewicht erstellt worden ist.

Nach der Erstellung kann man die Menge diverser alkoholhaltigen Getränke eingeben. Als nächstes kann man die Zeitspanne angeben, welche seit der Einnahme der eingestellten Getränke vergangen ist. ([Abbildung 4](#))

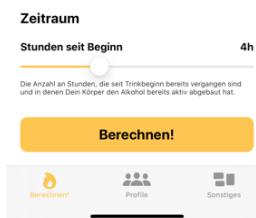


Abbildung 4: Beispiel App - Zeitraum



Abbildung 5: Beispiel App - Getränkeauswahl



Abbildung 6: Beispiel App - Ergebnis



6.2 Konzept

6.2.1 Funktionsübersicht

Dieses Flussdiagramm zeigt die Funktionen der geplanten Applikation und stellt die Abläufe vereinfacht dar.

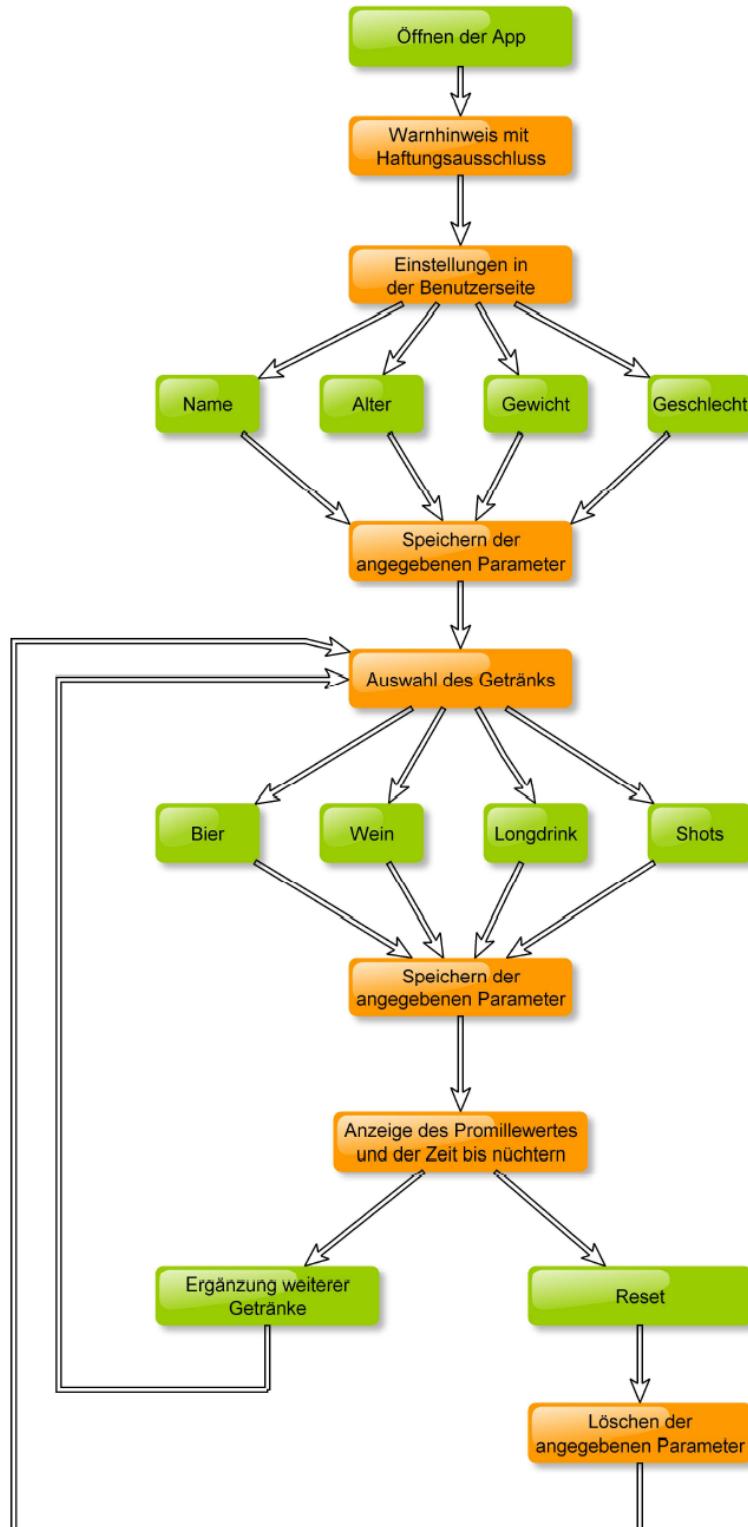


Abbildung 7: Flussdiagramm



6.2.2 Entwurf GUI-Design

Damit wir zielgerichtet das Projekt vorantreiben konnten, war es wichtig, bereits vor Beginn der Arbeiten in der Entwicklungsumgebung (IDE) ein Entwurf vom GUI-Design zu erarbeiten. Diesen Entwurf haben wir mittels Microsoft PowerPoint erarbeitet. Somit konnten wir unsere Ideen und Wünsche betreffend Design miteinander abgleichen.

Das Design der Applikation möchten wir in etwa gemäss untenstehendem Bild realisieren.



Abbildung 8: Entwurf GUI-Design



6.2.3 Die Getränke

Da wir uns für die im Abschnitt [8.2.1 BAK-Formel](#) beschriebene Widmark-Formel entschieden haben, war der nächste Schritt, uns für einige Getränke welche wir in der App anbieten möchten, zu entscheiden.

Zunächst haben wir uns für folgende Kategorien entschieden:

- Bier
- Wein
- Longdrinks
- Shots

Anschliessend haben wir pro Kategorie je drei Getränke ausgewählt, die unserer Meinung nach am häufigsten konsumiert werden.

Wir möchten an dieser Stelle darauf hinweisen, dass wir im Rahmen dieser Projektarbeit bewusst eine minimalistische Produktpalette anbieten, da ein grösseres Angebot mit den uns zur Verfügung stehenden 80 Arbeitsstunden nicht möglich ist.

Nachfolgend ist eine Auflistung unserer getroffenen Auswahl zu finden:

Bier:



Abbildung 9:
Stange

Getränk:	Stange Eichhof Lager
Volumen:	300 ml
Alkoholvolumenanteil:	4.8% VOL.



Abbildung 10:
Kübel

Getränk:	Kübel Eichhof Lager
Volumen:	500 ml
Alkoholvolumenanteil:	4.8% VOL.



Abbildung 11:
Pitcher

Getränk:	Pitcher Eichhof Lager
Volumen:	1800 ml
Alkoholvolumenanteil:	4.8% VOL.



Wein:



Abbildung 12:
Rotwein

Getränk:	Rotwein
Volumen:	100 ml
Alkoholvolumenanteil:	13% VOL.



Abbildung 13:
Weisswein

Getränk:	Weisswein
Volumen:	100 ml
Alkoholvolumenanteil:	12% VOL.



Abbildung 14:
Rosé

Getränk:	Rosé
Volumen:	100 ml
Alkoholvolumenanteil:	10% VOL.

Longdrinks:



Abbildung 15:
Cuba Libre

Getränk:	Cuba Libre
Volumen:	40 ml Rum
Alkoholvolumenanteil:	24% VOL.



Abbildung 16:
Long Island

Getränk:	Long Island
Volumen:	300 ml diverse Spirituosen
Alkoholvolumenanteil:	21% VOL.



Abbildung 17:
Vodka Lemon

Getränk:	Vodka Lemon
Volumen:	40 ml Vodka
Alkoholvolumenanteil:	40% VOL.



Shots:



Getränk: Vodka

Volumen: 20 ml

Alkoholvolumenanteil: 40% VOL.

Abbildung 18:
Vodka Shot



Getränk: Gin

Volumen: 20 ml

Alkoholvolumenanteil: 40% VOL.

Abbildung 19:
Gin Shot



Getränk: B52

Volumen: 20 ml

Alkoholvolumenanteil: 55% VOL.

Abbildung 20:
B52

Damit eine allfällige Ergänzung weiterer Getränke leichter von der Hand geht, haben wir die verwendeten Getränke mittels Kommentare direkt im Code definiert.

Beispiel der Kategorie Bier:

```
/***
 * childPosition = 0:
 * Beverage: Stange Eichhof Lager (URL:
https://www.eichhof.ch/biere/klassiker/lager)
 * Volume (V): 300 ml
 * Volume percentage (e): 4.8 % VOL. = 0.048
 * Density of ethanol (ρ): 0.8 g/ml
 * <p>

 * childPosition = 1:
 * Beverage: Chöbel Eichhof Lager (URL:
https://www.eichhof.ch/biere/klassiker/lager)
 * Volume (V): 500 ml
 * Volume percentage (e): 4.8 % VOL. = 0.048
 * Density of ethanol (ρ): 0.8 g/ml
 * <p>

 * childPosition = 2:
 * Beverage: Pitcher Eichhof Lager (URL:
https://www.eichhof.ch/biere/klassiker/lager)
 * Volume (V): 1800 ml
 * Volume percentage (e): 4.8 % VOL. = 0.048
 * Density of ethanol (ρ): 0.8 g/ml
 *
 * @param childPos
 * @return
 */
```



6.3 Vorstudie

In der Vorstudie haben wir uns das Ziel gesetzt, die für unser Projekt, geeignete IDE zu finden. Da wir beide bis jetzt noch nicht grosse Erfahrungen in objektorientierter Programmierung haben, stellten wir uns ein Zeitlimit für die Analyse. Wegen fehlendem Vorwissen konnten wir auch nicht viele IDE's vergleichen und haben uns auf Ratschläge anderer fokussiert.

Für uns in Frage kam Netbeans, IntelliJ und Android Studio.

Netbeans benutzen wir zwar an der TEKO, jedoch haben uns diverse Software Entwickler IntelliJ empfohlen. In Verbindung mit Android Studio, welche man zur Emulation von Android-Geräten benötigt, sei dies ihre favorisierte IDE.

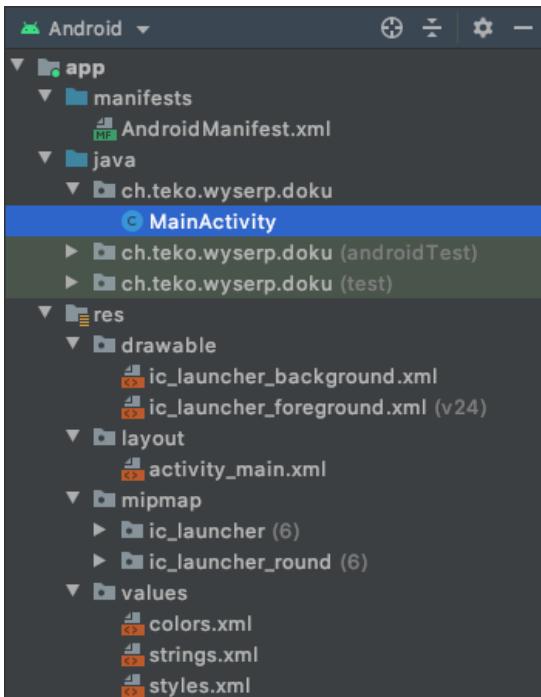
Wir haben definiert, dass wir eine Applikation für Android-Mobilgeräte entwickeln werden. Als Wunschkriterium haben wir die Kompatibilität mit iOS Geräten gesetzt. Dass dieses Wunschkriterium überhaupt möglich ist, mussten wir dementsprechende Abklärungen treffen.

Es gibt diverse «Multi-OS» Plattformen und Tools, welche eine möglichst einfache und gemeinsame Entwicklung für Android und Apple versprechen. Wir haben uns durch verschiedene dieser Tools durchgearbeitet und diese analysiert.

Da wir jedoch bis jetzt noch gar keine Erfahrung in diesem Bereich sammeln konnten, haben wir uns während dem Vorstudienprozess dazu geeinigt, dem Wunschkriterium nicht nachzugehen. Dies, weil für uns rein der Ermittlungsprozess des für uns am besten geeigneten «Multi-OS» Tool, den Rahmen unserer Projektarbeit sprengen würde.

Wir werden mit Android-Studio eine reine Android-Applikation entwickeln, welche wir mit dem Android-Emulator und schliesslich auf dem Mobiltelefon von Janik testen werden.

Ein neu erstelltes Projekt in Android Studio ist in der [Abbildung 21: Android default](#) gezeigt und die wichtigsten Elemente erklärt:



Manifest:

Im Manifest Ordner wird die `AndroidManifest.xml` Datei erstellt. In dieser sind die wichtigsten Informationen für das Android - "Build-Tool", OS und den Google Play Store.

Java:

Im Java Ordner sind alle Java Klassen, wie z.B. die `MainActivity`. In diesen wird der Java Code programmiert.

Res (Ressourcen):

Unter dem Ordner res sind diverse XML Dateien. Im Ordner drawable sind Grafiken und Vektoren, welche im Projekt verwendet werden. Unter dem Ordner layout sind die Anzuzeigenden Seiten der App. Im Ordner Mipmap werden alle Icons für die App erstellt, bzw. abgelegt.

Unter values sind XML Dateien, welche Übergreifende Werte enthalten. Z.B.

Abbildung 21: Android default

werden in der `colors.xml` Datei Farben definiert, welche im ganzen Projekt dann mit einer Verknüpfung auf diesen Wert übernommen werden können.



6.4 Termin- und Zeitmanagement

Eine komprimierte Version des Zeitplanes:

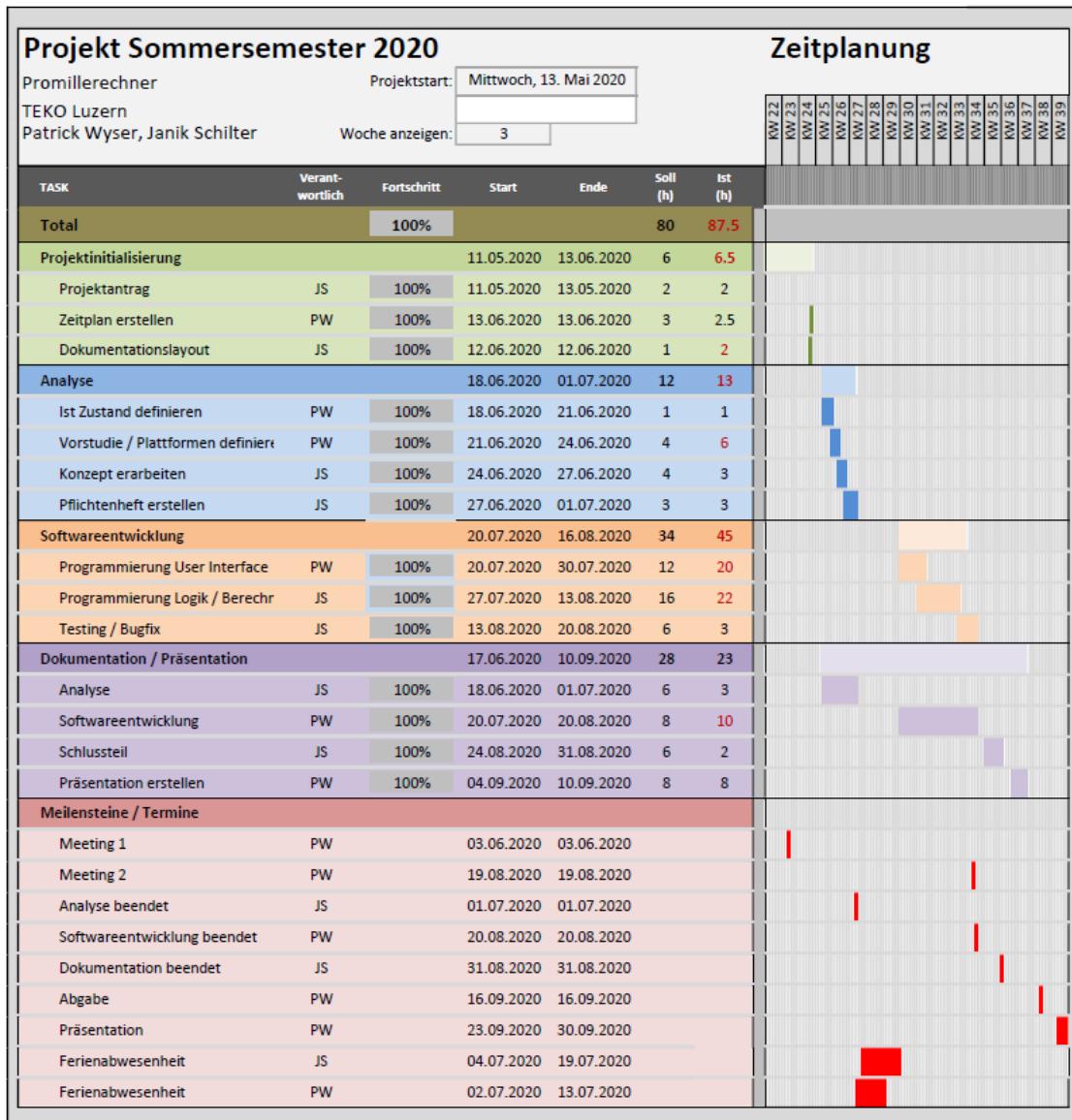


Abbildung 22: Zeitplan komprimiert

Eine Auflistung der wichtigsten Meilensteine und Termine:

Meilensteine / Termine	Verantwortung	Start	Ende
Meeting 1	PW	03.06.2020	03.06.2020
Fertigstellung der Analyse-Phase	JS	01.07.2020	01.07.2020
Ferienabwesenheit Patrick Wyser	PW	02.07.2020	13.07.2020
Ferienabwesenheit Janik Schilter	JS	04.07.2020	19.07.2020
Fertigstellung Softwareentwicklung	PW	20.08.2020	20.08.2020
Meeting 2	PW	26.08.2020	26.08.2020
Fertigstellung der Dokumentation	JS	31.08.2020	31.08.2020
Abgabe der Dokumentation	PW	19.09.2020	19.09.2020
Präsentation	PW	26.09.2020	26.09.2020



7 Pflichtenheft

7.1 Zielbestimmung

Musskriterien

- Anzeige des Blutalkoholgehalts mittels Eingabe konsumierter Getränke
- App-Kompatibilität mit Android-Endgeräten

Wunschkriterien

- App-Kompatibilität mit iOS-Endgeräten
- Festlegung einer Promille-Obergrenze, bei welcher eine Warnung erscheint.
- Möglichkeit zur Erstellung benutzerdefinierter Getränke

Abgrenzungskriterien

- Die Software soll keine wissenschaftlich belegte Werte anzeigen, sondern nur zur Unterhaltung des Benutzers dienen.
-

7.2 Produkteinsatz

Anwendungsbereiche

Android-Smartphones

Zielgruppen

Jede volljährige Person, die diese App downloaden und nutzen möchte.

7.3 Produktfunktionen

Personalisierung

Damit ein genauerer Promillewert errechnet werden kann, gibt der Benutzer bei erstmaliger Nutzung der App folgende Parameter an. Diese können jederzeit wieder angepasst werden.

- Geschlecht
- Gewicht

Auswahl des Getränks

Dem Benutzer soll eine Auswahl an diversen alkoholischen Getränken zur Verfügung stehen:

- Bier
- Wein
- Longdrinks
- Shots



Ausgabewert

- Anzeige des errechneten Promillewertes.
- Anzeige des errechneten Zeitpunktes des ausgenüchterten Zustandes

Ergänzung weiterer Getränke

Nun können weitere Getränke ausgewählt werden. Diese werden dem bisherigen Ergebnis angerechnet.

Reset

Der errechnete Promillewert und die ausgewählten Getränke werden gelöscht. Die personalisierten Parameter sollen jedoch beibehalten werden.

7.4 Qualitätsanforderungen

Zeitrechnung

Der Zeitpunkt der Konsumierung eines Getränkes wird von der Software erfasst. Damit soll der Alkoholabbau im Körper des Benutzer in die Berechnungen miteinbezogen werden können.

Reset-Knopf

Der Reset-Knopf lässt sich nur durch längeres Drücken betätigen. Dadurch wird gewährleistet, dass nicht versehentlich alle Daten gelöscht werden.

7.5 Benutzeroberfläche

Design

Das Design der Benutzeroberfläche wird gemäss den Anforderungen und Wünschen des Kunden erstellt.

Struktur

Die Struktur und der Aufbau der App wird gemäss den Anforderungen und Wünschen des Kunden erstellt.

7.6 Technische Produktumgebung

Entwicklungsumgebung (IDE)

Android-Studio

Vertriebsplattformen

- Zwingend: Google-Playstore
- Optional: Apple Appstore



8 Softwareentwicklung

8.1 GUI

8.1.1 GUI Design

Dialog

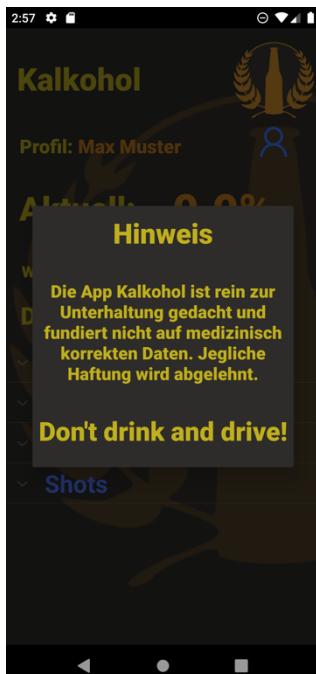


Abbildung 23: GUI Hinweis

Beim Öffnen der Applikation wird als erst ein Dialog mit einem Hinweis angezeigt. Dieser Hinweis dient zum Haftungsschutz der Entwickler.

Die Hauptseite ist folgendermassen gestaltet:

Gelbe Schrift:

Texte in gelber Schrift sind Titel, welche vom Benutzer nicht bearbeitet werden können.

Orange Schrift:

Texte in oranger Schrift sind Texte, welche sich während dem benutzen der App verändern können. Der Profilname (hier Max Muster) kann in der Benutzerseite verändert werden. Die Anzeige vom Alkoholpegel und der restlichen Zeit zum nüchternen Zustand wird anhand der konsumierten Getränke und der vergangenen Zeit errechnet.

Blaue Elemente:

Alle Elemente in blauer Farbe lösen Aktionen der App aus. Das Benutzersymbol ist zur Navigation zur Benutzerseite. Bei Druck auf den Reset-Knopf werden der Alkoholpegel und die konsumierten Getränke zurückgesetzt.

Hauptseite



Abbildung 24: GUI Main



Benutzerseite



Abbildung 25: GUI User

In der Benutzerseite gilt die gleiche Farbgebung wie auf der Hauptseite.

Bei Druck auf den aktuellen Profilnamen öffnet sich die Android Tastatur. Danach kann ein neuer Name eingegeben werden.

Das Alter und das Gewicht können mit einer Wischbewegung nach oben oder unten eingestellt werden.

Das Geschlecht kann beim "Toggle-Button" umgeschaltet werden.

Hauptseite mit Werten

In der [Abbildung 26: GUI Wertanzeige](#) werden nach Eingabe einer Stange die Werte neu berechnet und angezeigt. Die Werte werden alle 60 Sekunden neu errechnet und mit einem Zeitstempel abgespeichert. So werden auch nach einem Neustart der App oder des Mobiltelefons die aktuellen Werte angezeigt.



Abbildung 26: GUI Wertanzeige



Mit Hilfe von "Toasts", werden kleine Anzeigen dargestellt.

Beim Hinzufügen eines Getränks:

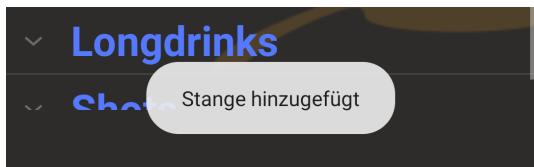


Abbildung 27: Toast Getränk

Beim Abspeichern des Profils:

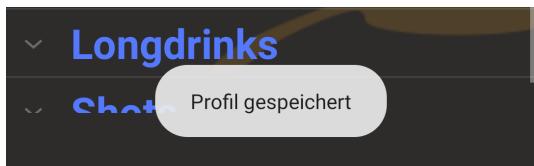


Abbildung 28: Toast Profil

Beim Rücksetzen aller Werte:

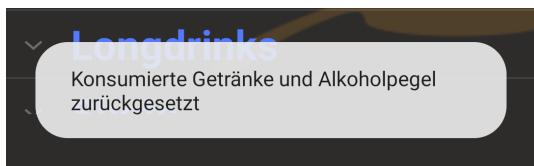


Abbildung 29: Toast Reset



8.1.2 Entwicklungsumgebung

Das GUI haben wir direkt im Android-Studio entwickelt. Ein sehr tolles Feature dieser Entwicklungsumgebung ist, dass die Bedienoberfläche grafisch erstellt und dargestellt werden kann, jedoch auch direkt im XML Code bearbeitet werden kann. So kann ein Element wie z.B. ein Button ins grafisch dargestellte Design gezogen werden und z.B. die Grösse angepasst werden. Attribute wie z.B. die Android-ID oder die Schriftgrösse können aber auch direkt im XML-Code bearbeitet werden. In der sogenannten «Split-View» Ansicht kann dann der XML-Code neben der grafischen Entwicklungsumgebung angezeigt werden. So kann man mit Leichtigkeit auf der grafischen Oberfläche ein Element anwählen, sodass die Codeansicht direkt zu diesem springt, um die Attribute im XML-Code anzupassen oder zu betrachten.

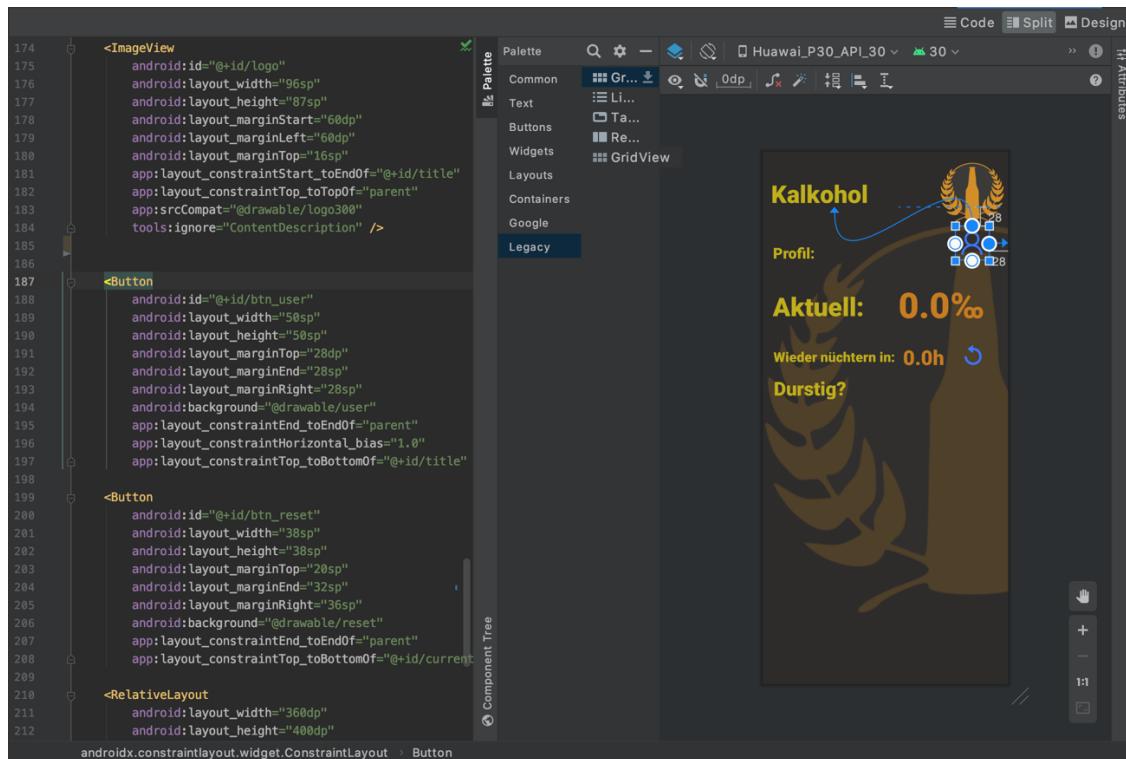


Abbildung 30: Entwicklungsumgebung

In der Abbildung 30 ist links der XML-Code und rechts die grafische Oberfläche zu sehen. In der Mitte ist die Palette zusehen. Aus dieser können diverse Elemente ins Design gezogen werden. Die von uns in der Applikation verwendeten Elemente werden unter 8.1.3 *GUI-Elemente* beschrieben und ihre Eigenschaften erklärt.



8.1.3 GUI-Elemente

8.1.3.1 Allgemeine Attribute

Diverse Elemente haben verschiedene Attribute, welche ein Element beschreiben und definieren können.

Die wichtigsten werden direkt beim Erstellen des Elements erzeugt. Wenn aber zum Beispiel ein TextView Element im Design hinzugefügt wird, ist die Farbe des Textes einfach schwarz. Um dies oder andere Eigenschaften zu verändern können wie in der Abbildung 31: *Autofill hint color* gezeigt, diverse Attribute mit Hilfe von einfachen Stichwörtern hinzugefügt werden.

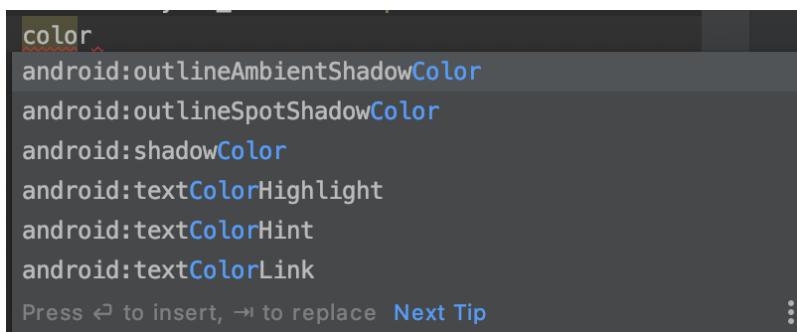


Abbildung 31: Autofill hint color

Ein paar der wichtigsten Attribute sind folgend aufgelistet:

android:id

Die Android-ID wird verwendet, um jedes Element eindeutig identifizierbar zu machen. So kann in einem späteren Schritt im Java-Code auf Attribute eines Elements zugegriffen werden, um z.B. den Text eines TextView Elements zu ändern. Daher ist es zu empfehlen sprechende Namensgebungen für die Android-ID zu verwenden.

android:layout_width/height

Das Attribut layout_width und layout_height wird verwendet, um die Breite, bzw. Höhe eines Elements zu definieren.

Dies kann einen fix definierten oder auch dynamische Werte annehmen. Wenn zum Beispiel "wrap_content" definiert wird, wird die Grösse des Elements, der Grösse des Inhalts wie z.B. dem angezeigten Text angepasst.

android:text

Beim Attribut android:text kann der anzuzeigende Text eines Elements hinterlegt werden. Dieser kann direkt definiert werden, was jedoch nicht zu empfehlen ist. Texte werden am besten in dem File "strings" hinterlegt, um z.B. eine mehrsprachige App zu entwickeln. So können die Texte in "strings" angepasst werden und im Design mit der Eingabe **@string/beispieltext** verknüpft werden.

So können auch mehrere Elemente auf den gleichen String verknüpft sein, sodass man den Text nur an einem Ort anpassen muss.

android:color

Die Farbe eines Elements, Schattens oder eines Textes kann hier angegeben werden. Die Farben können auch in einem eigenen File eingestellt werden und mit einer Verknüpfung übernommen werden. So kann auch im Nachhinein sehr schnell, die ganze Farbwahl über mehrere Elemente hinaus angepasst werden.



app:layout_constraint_XXX

Mit den layout_constraint_XXX Attributen können die Positionierungen von Elementen voneinander abhängig gemacht werden. Somit können Gruppierungen geschaffen werden, sodass diese zusammen verschoben werden können. Das Arbeiten mit dem "Constraint_Layout" erleichtert auch die Entwicklung für mehrere Geräte, da die Elemente nicht statisch auf eine Koordinate gebunden sind und somit bei einem Gerät mit einem grösseren Bildschirm dynamisch den richtigen Platz einnehmen.

Das Vorgehen beim "Constraint Layout" ist mit folgendem Beispiel und der Abbildung 32: Constraint_Layout erklärt:

Der Button und der Text sind an die Position vom Bild in der Mitte gebunden. Die geschwungenen Pfeillinien zeigen die Verknüpfung.

Das Attribut "layout_constraintStart_toStartOf" vom Text ist auf die ID vom Bild verknüpft.

Mit dem Attribut android:layout_marginTop="12dp" wird der Abstand zum mit dem im "constraint_layout" definierten Element angegeben.

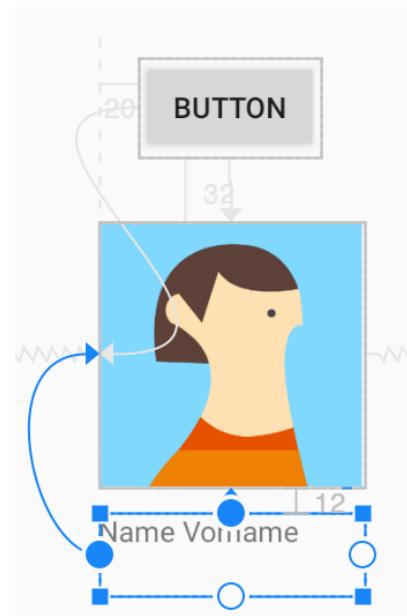


Abbildung 32: Constraint_Layout

8.1.3.2 TextView

Ein TextView-Element ist ein Text, welcher in der Bedienoberfläche angezeigt wird.

Die Abbildung 33: TextView Beispiel zeigt das Element, welches den aktuellen Wert des Blutalkoholgehaltes anzeigen soll. Dieser haben wir neben den Eigenschaften wie Grösse und Positionierung folgende Attribute vergeben:

- android:id="@+id/state_data"
- android:text="@string/current_state_data"
- android:textColor="@color/colorTextLight"



Abbildung 33: TextView Beispiel

Die ID verwenden wir um aus dem Java Code mit folgender Methode den Inhalt dieser Textanzeige ändern können:

```
public void setDisplay(String bac, String time){
    TextView actBca = (TextView) findViewById(R.id.state_data);
    TextView actTime = (TextView) findViewById(R.id.time_data);
    actBca.setText(bca);
    actTime.setText(time);
```

Der Text "time_data" verweist auf die im Bild nicht ersichtliche Anzeige der Restdauer bis zum nüchternen Zustand.



8.1.3.3 ImageView

Mit dem ImageView Element können Bilder ins Design integriert werden. Diese müssen zuerst in dem Verzeichnis unter "drawable" hinzugefügt werden. Unter dem dort angegebenen Namen kann mit dem Verweis "app:srcCompat="@drawable/xxx" im Layout das Bild dargestellt werden.

8.1.3.4 Button

Ein Button ist ein mit Berührung bedienbares Element, um diverse Aktionen auszulösen. Die Aktion wird jedoch im Java Code und nicht im Layout hinterlegt.

In unserem Beispiel wurde dem Button "user" mit folgendem Attribut noch eine selbst erstellte Grafik hinterlegt:

```
android:background="@drawable/user"
```



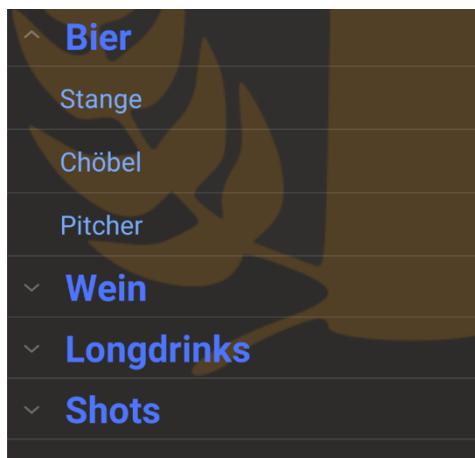
Abbildung 34:
Button User

8.1.3.5 Expandable-List

Die Expandable-List ist eine vertikale Liste welche zwei Ebenen darstellen kann. So können "Drop-Down Listen" erstellt werden. Es können "Parents" und dessen "Childs" erstellt und mit Hilfe des folgenden Codes ermittelt werden, welches Element gedrückt wird:

```
public boolean onChildClick(ExpandableListView parent, View v, int groupPosition, int childPosition, long id) {
    float ethanol = 0.0f;

    if (groupPosition == 0) {
        ethanol = this.calcBeerChild(childPosition);
    } else if (groupPosition == 1) {
        ethanol = this.calcWineChild(childPosition);
    } else if (groupPosition == 2) {
        ethanol = this.calcLongdrinksChild(childPosition);
    } else if (groupPosition == 3) {
        ethanol = this.calcShotsChild(childPosition);
    }
}
```



Mit Hilfe der Expandable-List und dem Relative-Layout, welches unter [Relative-Layout](#) erklärt wird, wurde diese Drop-Down Liste erstellt.

Abbildung 35: ExpandableList



8.1.3.6 Relative-Layout

Mit Relative-Layout können in diesem Baustein definierte Elemente in relativen Positionen dargestellt werden. So können zum Beispiel dynamische Anzeigen wie eine Expandable-List anhand ihrer Grösse korrekt und in dem definierten Bereich dargestellt werden. Denn im Beispiel Expandable-List, ist die Grösse nicht immer gleich. Somit wird mit dem Relative-Layout diese Expandable-List mit einer Scrollbar erweitert und alle "Childs" der Expandable-List können angezeigt werden.

8.1.3.7 EditText

Ein EditText Element funktioniert ähnlich wie ein TextView Element. Dieses ist aber nicht nur zur Anzeige, sondern auch zur Eingabe gedacht. Das EditText Element kann auf Berührung die Tastatur von Android aufrufen.

8.1.3.8 NumberPicker

Ein NumberPicker ist ein Element welches Nummern mit Hilfe von Wischbewegungen eingestellt werden können. Mit den Funktionen ".setMinValue()" und ".setMaxValue()" können solche Elemente im Wertebereich eingeschränkt werden. Mit der Funktion ".getValue" kann der eingestellte Wert ausgelesen werden.

NumberPicker wurden wie in [Abbildung 36: NumberPicker](#) gezeigt für die Eingabe von Alter und Gewicht verwendet.

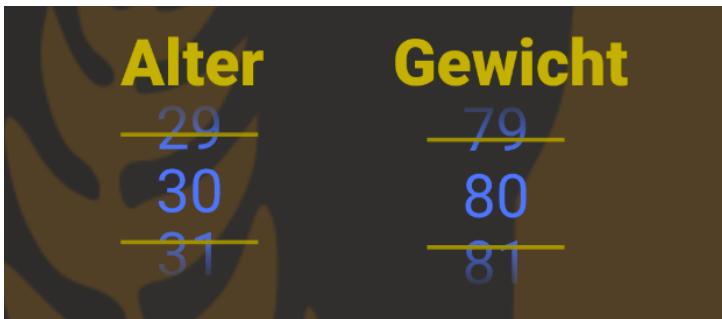


Abbildung 36: NumberPicker

8.1.3.9 RadioGroup / RadioButton

Mit RadioButtons in einer RadioGroup können einfach Auswahlmöglichkeiten dargestellt werden. Meist werden diese mit sogenannten "Checkboxes" realisiert. Mit den Funktionen ".setChecked" und ".isChecked" können die Auswahlmöglichkeiten gesetzt oder ausgelesen werden.

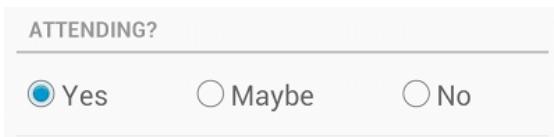


Abbildung 38: RadioButton default

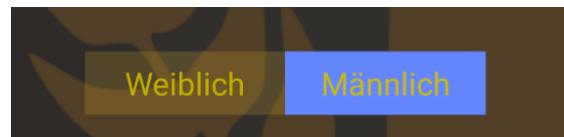


Abbildung 37: RadioButton Kalkohol



8.2 Backend

8.2.1 BAK-Formel

Die Menge an Alkohol im Blut bezeichnet man als Blutalkoholkonzentration (BAK). Diese wird in Gramm Reinalkohol pro Kilogramm Blut als Promille angegeben. Eine Blutalkoholkonzentration von 1‰ bedeutet, dass ein Kilogramm Blut 1 Gramm reinen Alkohol enthält.

Die Höhe der Blutalkoholkonzentration lässt sich mit der sogenannten Widmark-Formel berechnen. Diese Formel berücksichtigt sowohl das Gewicht als auch das Geschlecht. Für eine realistische Berechnung ist das Geschlecht unerlässlich, da die durchschnittliche Menge an Körperflüssigkeit von Mann zu Frau unterschiedlich ist. Der Anteil an Körperflüssigkeit beträgt bei Männern ca. 68%, bei Frauen ca. 55%.

Widmark-Formel zur Berechnung der Blutalkoholkonzentration:

$$\text{BAK } (\%) = \frac{\text{getrunkener Alkohol (g)}}{\text{Körpergewicht (kg)} \times \text{Anteil Körperflüssigkeit}}$$

Abbildung 39: Widmark-Formel

Getrunkener Alkohol:

Für die Formel muss das Gewicht des reinen Alkohols in Gramm bestimmt werden. Zur Berechnung der Masse A des Alkohols, wird das Volumen V (in ml) des Getränkes mit dem Alkoholvolumenanteil (z. B. Bier: e=0,05) und der Dichte von Alkohol ($\rho \approx 0,8\text{g/ml}$) multipliziert werden: $A = V \cdot e \cdot \rho$.

Ein Beispiel:

Hat beispielsweise ein Liter (1000 ml) Bier einen Volumenanteil von 0,05 (5%) Alkohol, so entsprechen die 50 ml Alkohol einem Gewicht von 40 g.

Körpergewicht:

Das Körpergewicht beeinflusst die Blutalkoholkonzentration in erheblichem Maße und wird daher in der Widmark-Formel verwendet. Hierbei wird das Körpergewicht in Kilogramm (kg) in die Formel eingesetzt.

Anteil Körperflüssigkeit:

In der Widmark-Formel wird ein durchschnittlicher Anteil an Körperflüssigkeit für Mann oder Frau eingesetzt:

Frauen: 0,55 (Anteil Körperflüssigkeit 55%)

Männer: 0,68 (Anteil Körperflüssigkeit 68%)



8.2.2 Alkoholabbau

Konsumiert man Alkohol, so benötigt die Leber eine gewisse Zeit, um die Blutalkoholkonzentration zu verringern. Dabei werden in Abhängigkeit von Geschlecht und Gewicht der Person pro Stunde in etwa 0,1 bis 0,2 Promille abgebaut.

Der Alkoholabbau dauert bei Frauen generell etwas länger als bei Männern. Sie verfügen über geringere Mengen des alkoholabbauenden Enzyms ADH und benötigen daher im Vergleich zu Männern länger, um wieder nüchtern zu werden.

Die exakte Berechnung der Blutalkoholkonzentration gestaltet sich schwierig, da noch viele weitere Faktoren wichtig sind. Gene, Alter, Stoffwechsel oder Aufnahme von Wasser und Essen vor, während oder nach dem Konsum von Alkohol haben einen beträchtlichen Einfluss auf die effektiv vorhandenen BAK-Werte.

Da wir aber mit unserer Projektarbeit in erster Linie eine Software-Applikation zur Unterhaltung entwickeln möchten, stellen wir nicht den Anspruch eine wissenschaftlich fundierte Berechnung der BAK in der Applikation zu implementieren. Zudem weichen die erwähnten Erkenntnisse und Werte je nach Informationsquelle voneinander ab.

Somit haben wir uns dazu entschieden, zur Simulation des Alkoholabbaus mit folgenden Werten zu arbeiten:

Männer: Alkoholabbau von 0,16 Promille pro Stunde (0,16‰ / h)

Frauen: Alkoholabbau von 0,14 Promille pro Stunde (0,14‰ / h)



8.2.3 Kalkulation im Backend

Mit dem Entscheid für die Widmark-Formel und der Definition der Getränke galt es nun, die Berechnung für die Software zu entwickeln. Nachfolgend sind die interessantesten Methoden aufgelistet:

8.2.3.1 Methode "calcBeerChild"

Berechnung der Alkoholmenge in Gramm:

```
private float calcBeerChild(int childPos) {
    if (childPos == 0) {
        return this.calcNewDrink(300f * 0.048f * 0.8f);
    } else if (childPos == 1) {
        return this.calcNewDrink(500f * 0.048f * 0.8f);
    } else if (childPos == 2) {
        return this.calcNewDrink(1800f * 0.048f * 0.8f);
    }
    return 0.0f;
}
```

Zur Dokumentation dieser Methode haben wir als Beispiel die Getränkekategorie Bier verwendet. Mittels dieser Methode wird die Alkoholmenge in Gramm berechnet.

Die Formel hierzu lautet:

$$A = V \cdot e \cdot \rho$$

A = Masse in Gramm (g)

V = Volumen im Milliliter (ml)

e = Alkoholvolumenanteil als Dezimalzahl

ρ = Dichte von Alkohol in Gramm pro Milliliter (g/ml)

Die Berechnung einer Stange Bier lautet also wie folgt:

$$V = 300 \text{ ml}$$

$$e = 0.048$$

$$\rho = 0.8 \text{ g/ml}$$

$$A = 300 \cdot 0.048 \cdot 0.8 = \underline{\underline{11.52 \text{ Gramm reiner Alkohol}}}$$



Abbildung 40: Kalkulation



8.2.3.2 Methode "calcNewDrink"

Berechnung der Blutalkoholkonzentration:

```
public float calcNewDrink(float AlcWeight) {
    SharedPreferences sharedpreferences;
    sharedpreferences = getSharedPreferences(user, Context.MODE_PRIVATE);
    String actWeight = sharedpreferences.getString(weight, "");
    String actGender = sharedpreferences.getString(gender, "");

    assert actWeight != null;
    float fWeight = Float.parseFloat(actWeight);
    float result = 0.00f;

    assert actGender != null;
    if (actGender.equals("male")) {
        float fGender = 0.68f;
        result = AlcWeight / ( fWeight * fGender );
    }

    if (actGender.equals("female")) {
        float fGender = 0.55f;
        result = AlcWeight / ( fWeight * fGender );
    }
    return result;
}
```

Mit dieser Methode wird die zusätzliche Blutalkoholkonzentration bei Auswahl eines Getränkes berechnet. Die hier verwendete Berechnungsformel wurde bereits im Abschnitt [8.2.1 BAK-Formel](#) detailliert beschrieben.

$$\text{BAK (\%)} = \frac{\text{getrunkener Alkohol (g)}}{\text{Körpergewicht (kg)} \times \text{Anteil Körperflüssigkeit}}$$

Abbildung 41: Widmark-Formel

BAK	=	Blutalkoholkonzentration in Promille (%)
AlcWeight	=	Alkohol in Gramm z.B. aus Methode "calcBeerChild"
fWeight	=	Körpergewicht in Kilogramm (kg) → gemäss Eingabe im Nutzer-Profil
fGender	=	Anteil Körperflüssigkeit (in Abhängigkeit vom Geschlecht) → Geschlecht gemäss Einstellung im Nutzer-Profil

Trinkt nun eine 70 kg schwere Frau eine Stange Bier, lautet die Berechnung wie folgt:

AlcWeight	=	11.52 g	(siehe 8.2.3.1 Methode "calcBeerChild")
fWeight	=	70 kg	
fGender	=	0.55	(siehe 8.2.1 BAK-Formel)
BAK	=	$\frac{11.52}{70 \cdot 0.55}$	= <u>0.299 %</u>



8.2.3.3 Methode "alcReduction"

Berechnung des Alkoholabbaus:

```
public void alcReduction(float elapsedTime) {

    SharedPreferences sharedpreferences;
    sharedpreferences = getSharedPreferences(user, Context.MODE_PRIVATE);
    String actGender = sharedpreferences.getString(gender, "");

    float reduction = 0.00f;

    assert actGender != null;
    if (actGender.equals("male")) {
        float fGender = 0.16f / 60;
        reduction = fGender * elapsedTime;
    }

    if (actGender.equals("female")) {
        float fGender = 0.14f / 60;
        reduction = fGender * elapsedTime;
    }

    this.setDisplay((-1) * reduction);
}
```

Mit dieser Methode wird der Alkoholabbau in Abhängigkeit von Geschlecht und vergangener Zeit berechnet. Wie bereits im Abschnitt [8.2.2 Alkoholabbau](#) beschrieben, haben wir uns zur Berechnung des Alkoholabbaus dazu entschieden mit folgenden Werten zu arbeiten:

Männer: Alkoholabbau von 0,16 Promille pro Stunde (0,16‰ / h)

Frauen: Alkoholabbau von 0,14 Promille pro Stunde (0,14‰ / h)

reduction = abgebaute Blutalkoholkonzentration in Promille (‰)
fGender = Alkoholabbau pro Stunde (in Abhängigkeit vom Geschlecht)
elapsedTime = vergangene Zeit seit letzter Berechnung

Da wir diese Methode mittels einer Timer-Funktion (siehe [8.2.3.4 Methode "initTimer"](#)) einmal in der Minute ausführen beträgt der Wert der Variable *elapsedTime* immer 1 (1 Minute).

Die Berechnungsformel bei einem Mann wäre demnach wie folgt:

$$\text{reduction} = \frac{f\text{Gender} \cdot \text{elapsedTime}}{60} = \frac{0.16 \cdot 1}{60} = \underline{\underline{0.0026\%}}$$



8.2.3.4 Methode "initTimer"

Timer-Funktionen:

```
public void initTimer() {  
  
    Timer timer = new Timer();  
  
    // alcReduction every 60 sec:  
    timer.scheduleAtFixedRate(new TimerTask() {  
        @Override  
        public void run() {  
            alcReduction(1);  
        }  
    }, 0, 60000);  
  
    // Timestamp every 60 sec for calculating alcReduction while app closed:  
    timer.scheduleAtFixedRate(new TimerTask() {  
        @Override  
        public void run() {  
            actTimestamp = new Timestamp(System.currentTimeMillis());  
  
            // timestamp im lokalen Speicher speichern  
            SharedPreferences sharedpreferences;  
            sharedpreferences = getSharedPreferences(user,  
Context.MODE_PRIVATE);  
            SharedPreferences.Editor editor = sharedpreferences.edit();  
            String str_timestamp = actTimestamp.toString();  
            editor.putString(time, str_timestamp);  
            editor.apply();  
        }  
    }, 2000, 60000);  
}
```

Diese Methode beinhaltet zwei Timer-Funktionen:

Der erste Timer führt die beschriebene *Methode "alcReduction"* einmal pro Minute aus und gibt den Parameter 1 mit.

Der zweite Timer generiert jede Minute einen Timestamp und speichert diesen mittels *SharedPreferences* lokal auf dem Gerät des Benutzers. Dieser Timestamp ermöglicht es, die Berechnung des Alkoholabbaus bei geschlossener Applikation zu gewährleisten. Dies wird im Abschnitt *8.2.3.5 Methode "closedApptime"* genauer erläutert.



Abbildung 42: Timestamp



8.2.3.5 Methode "closedApptime"

Berechnung des Alkoholabbaus bei geschlossener Applikation:

```
public void closedApptime() {
    float passedTime;

    Sharedpreferences sharedpreferences;
    sharedpreferences = getSharedpreferences(user, Context.MODE_PRIVATE);
    String actTime = sharedpreferences.getString(time, "0000-00-00
00:00:00.000");

    assert actTime != null;
    if (!actTime.equals("0000-00-00 00:00:00.000")) {
        this.actTimestamp = Timestamp.valueOf(actTime);
        passedTime = this.timestampOnCreate.getTime() -
this.actTimestamp.getTime();
        alcReduction(passedTime / 1000f / 60f);
    }
}
```

Die *Methode "alcReduction"* und die *Methode "initTimer"* ermöglichen die Berechnung des Alkoholabbaus ausschliesslich während die Applikation geöffnet ist. Daher entwickelten wir zusätzlich die *Methode "closedApptime"*. Diese ermöglicht es uns, die Zeitdifferenz zwischen dem Schliessen und dem Erneuten Öffnen der App festzustellen. Diese Zeitdifferenz übergeben wir ganz einfach der Methode *"alcReduction"* in der Masseinheit Minuten.

Wie bereits im vorherigen Abschnitt erklärt, generiert die *Methode "initTimer"* einmal pro Minute einen Timestamp und speichert diesen lokal auf dem Gerät des Benutzers. Dies ermöglicht ein Abruf des Wertes nach geschlossener Applikation. Wird die Applikation nun geschlossen und nach gewisser Zeit wieder geöffnet, wird der letzte abgespeicherte Wert vom Gerät gelesen und für nachfolgende Formel verwendet.

In der *Methode "onCreate"* haben wir einen zweiten Timestamp platziert, welcher beim Öffnen der App erstellt wird.

<i>passedTime</i>	=	Dauer von geschlossener App in Minuten
<i>actTimestamp</i>	=	letzter gespeicherter Timestamp vor dem Schliessen
<i>timestampOnCreate</i>	=	Timestamp beim Starten der Applikation

Daraus ergab sich folgende Berechnungsformel:

$$\text{passedTime} = \frac{\text{timestampOnCreate} - \text{actTimestamp}}{1000 * 60}$$



Abbildung 43: Timer



8.2.3.6 Methode "setNewTimeTillSober"

Anzeige: "Wieder nüchtern in:"

```

private void setNewTimeTillSober() {
    if (this.currentBAC <= 0) {
        TextView actTime = findViewById(R.id.time_data);
        actTime.setText("0.0f + "h");
        return;
    }

    SharedPreferences sharedpreferences;
    sharedpreferences = getSharedPreferences(user, Context.MODE_PRIVATE);
    String actGender = sharedpreferences.getString(gender, "");

    float fGender = 0.0f;

    assert actGender != null;
    if (actGender.equals("male")) {
        fGender = 0.16f;
    }

    if (actGender.equals("female")) {
        fGender = 0.14f;
    }

    this.tts = this.currentBAC / fGender;
    float roundedTts = Math.round(this.tts * 10.0f) / 10.0f;

    TextView actTime = findViewById(R.id.time_data);
    actTime.setText(roundedTts + "h");
}

```



Abbildung 44: Anzeige – wieder nüchtern

Diese Methode berechnet die benötigte Zeit, bis die Blutalkoholkonzentration im Körper wieder 0.00% Promille erreicht.

Dafür arbeiten wir mit folgenden Werten: (siehe 8.2.2)

Männer: 0,16 Promille pro Stunde (0,16‰ / h)
 Frauen: 0,14 Promille pro Stunde (0,14‰ / h)

Die Formel zur Berechnung:

tts = Zeit bis wieder nüchtern (timeTillSober)
 $currentBAC$ = aktuelle Blutalkoholkonzentration
 $fGender$ = Alkoholabbau pro Stunde
 (in Abhängigkeit vom Geschlecht)

$$tts = \frac{currentBAC}{fGender}$$

Danach wird das Ergebnis mittels *Math.round()* gerundet und mit *set.Text* ausgegeben.



8.2.3.7 Methode "setDisplay"

Anzeige der aktuellen Blutalkoholkonzentration:

```
private void setDisplay(float bac) {
    TextView actBac = findViewById(R.id.state_data);

    this.currentBAC += bac;
    if (this.currentBAC <= 0) {
        this.currentBAC = 0;
    }

    // der currentBAC im lokalen Speicher speichern
    SharedPreferences sharedpreferences;
    sharedpreferences = getSharedPreferences(user, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedpreferences.edit();
    String str_currentBAC = Float.toString(this.currentBAC);
    editor.putString(BAC, str_currentBAC);
    editor.apply();

    this.setNewTimeTillSober();

    float roundedBAC = Math.round(this.currentBAC * 100.0f) / 100.0f;
    actBac.setText(roundedBAC + "%");
}
```

Dieser Methode werden die berechneten Änderungen der Blutalkoholkonzentration übergeben.

Sie besitzt folgende Funktionen:

- Addiert die aktuelle BAK mit dem neuen Wert
- kontrolliert, dass Ausgabewert nicht < 0 beträgt
- Speichert den berechneten BAK-Wert lokal auf Gerät
- rundet berechneten BAK-Wert auf zwei Dezimalstellen
- Ausgabe des berechneten BAK-Wert mit *set.Text*

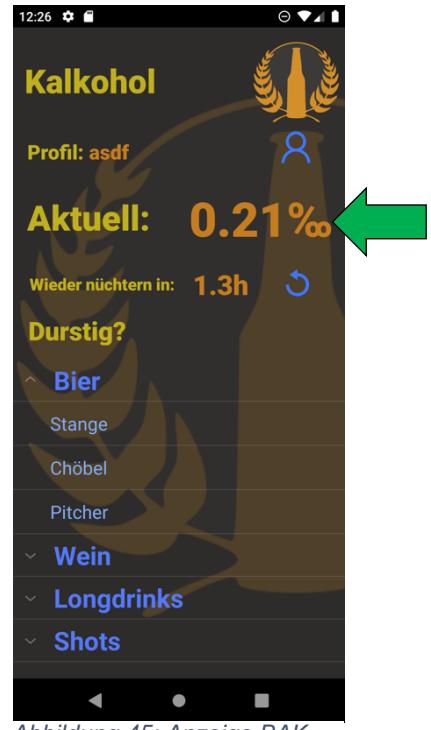


Abbildung 45: Anzeige BAK-Wert



9 Testbericht / Auswertung

9.1 1. Testphase

Die erste Testphase umfasst das Testen während der Softwareentwicklung. Dazu wurde der im Android Studio integrierte Emulator genutzt.

Einstellungen im Emulator:

Specs	Einstellung
Gerät	Huawei P30
Auflösung	6.1" // 1080px * 2340px
Density	xxhdpi
API-Level	30
Android	10.0+ Google Inc.
System Image	x86

Der Emulator wurde bereits während der Entwicklung der Software genutzt. Dadurch erkennt man allfällige Probleme oder Schwierigkeiten frühzeitig und kann Fehler viel schneller eruieren.

Folgender Fehler ist an mehreren Orten zu tragen gekommen:

Die lokal abgespeicherten Werte (sharedPreferences) müssen abgefragt werden:

```
SharedPreferences sharedPreferences;
sharedPreferences = getSharedPreferences(user, Context.MODE_PRIVATE);
String actAge = sharedPreferences.getString(age, "");
String actWeight = sharedPreferences.getString(weight, "");
String actGender = sharedPreferences.getString(gender, "");
```

Wenn mit diesen Werten gerechnet werden muss, ist es natürlich notwendig, dass immer auch ein Wert hinterlegt ist. Im Falle das kein Profil abgespeichert ist können keine mathematischen Funktionen ausgeführt werden, welche zum Beispiel auf den Wert vom Gewicht angewiesen sind. Falls in diesem Fall kein Wert hinterlegt ist, stürzt die Applikation ab.

Fehlermeldung:

2020-09-14 13:01:49.225 4948-4948/ch.teko.wyserp.gui E/AndroidRuntime: FATAL EXCEPTION: main

Process: ch.teko.wyserp.gui, PID: 4948
java.lang.NumberFormatException: empty String

Um dieses Problem zu beheben können die Strings beim Abfragen initialisiert werden. Diese in grün eingegebenen Werten werden nur übernommen, falls keine Werte im lokalen Speicher hinterlegt sind. Dies ist grundsätzlich nur beim erstmaligen Öffnen der Applikation der Fall.

```
SharedPreferences sharedPreferences;
sharedPreferences = getSharedPreferences(user, Context.MODE_PRIVATE);
String actAge = sharedPreferences.getString(age, "25");
String actWeight = sharedPreferences.getString(weight, "80");
String actGender = sharedPreferences.getString(gender, "male");
```



9.2 2. Testphase

Die zweite Testphase umfasst das testen nach beendeter Softwareentwicklung. Dazu verwendeten wir folgendes Smartphone:

Specs	Einstellung
Gerät	Huawei P30
Modell	ELE-L29
Build-Nummer	9.1.0.260 (C432E4R2P2)
EMUI-Version	9.1.0
CPU	Huawei Kirin 980
RAM	6.0 GB
Auflösung	6.1" // 1080px * 2340px
Density	xxhdpi
Android	9.0 Google Inc.
System Image	x86

Durch das Testing auf dem Smartphone wurden wir auf folgendes Problem aufmerksam. Es handelt sich hierbei um ein ähnliches Problem, wie bereits in der ersten Testphase entdeckt wurde.

Die Applikation konnte auf dem Smartphone nicht gestartet werden, da der im sharedpreferences gespeicherte String falsch formatiert war.

Fehlerhafter Code:

```
SharedPreferences sharedpreferences;
sharedpreferences = getSharedPreferences(user, Context.MODE_PRIVATE);
String actTime = sharedpreferences.getString(time, "0000-00-00 00:00:00");

assert actTime != null;
if (!actTime.equals("0000-00-00 00:00:00")) {
    this.actTimestamp = Timestamp.valueOf(actTime);
    passedTime = this.timestampOnCreate.getTime() -
this.actTimestamp.getTime();
    alcReduction(passedTime / 1000f / 60f);
```

Korrigierter Code:

```
SharedPreferences sharedpreferences;
sharedpreferences = getSharedPreferences(user, Context.MODE_PRIVATE);
String actTime = sharedpreferences.getString(time, "0000-00-00
00:00:00.000");

assert actTime != null;
if (!actTime.equals("0000-00-00 00:00:00.000")) {
    this.actTimestamp = Timestamp.valueOf(actTime);
    passedTime = this.timestampOnCreate.getTime() -
this.actTimestamp.getTime();
    alcReduction(passedTime / 1000f / 60f);
```

Zusätzlich wurde nun noch mit einer if-Abfrage kontrolliert, ob auch wirklich ein Wert in den sharedpreferences abgespeichert ist.



10 Veröffentlichung

10.1 Google Play-Store

Um eine Applikation auf den Google Play-Store zu laden muss ein Developer Account erstellt werden. Nach einer einmaligen Registrierungsgebühr von 30 USD werden grundsätzlich keine Ausgaben mehr benötigt.

Als registrierter Entwickler können nun Applikationen für die Veröffentlichung beantragt werden. Bei den Applikationen sind jedoch diverse Angaben zu tätigen, damit eine Veröffentlichung von Google freigegeben wird.

App-Zugriff:

Hier muss angegeben werden ob die App z.B. nur standortabhängig oder erst nach einer Anmeldung / Registrierung verwendet werden kann oder ob diese eingeschränkten Funktionen hat.

In unserem Fall: *Alle Funktionen sind ohne speziellen Zugriff verfügbar.*

Anzeigen:

Hier wird angegeben ob die Applikation Werbung enthält.

In unserem Fall: *Nein*

Einstufung des Inhalts:

Unter diesem Punkt muss ein Fragebogen ausgefüllt werden, sodass die Applikation betreffend Altersgruppen eingestuft werden kann.

Themen des Fragebogens:

- *Gewalt*
- *Angst*
- *Sexualität*
- *Simuliertes Glücksspiel, reales Glücksspiel oder Barauszahlungen*
- *Vulgärsprache*
- *Betäubungsmittel*
- *Derber Humor*
- *Sonstiges*

Im Thema Betäubungsmittel mussten wir natürlich diverse Angaben machen, da unsere Applikation zwar nur zur Unterhaltung dient, jedoch schon einen gewissen Anreiz zu Alkoholkonsum mit sich bringt.

Zielgruppe:

Hier wird angegeben welche Zielgruppen für den Entwickler in Frage kommen.

Organisation:

Hier wird die App einer Kategorie zugewiesen, die Kontaktangaben für den Store-Eintrag gemacht und der Store-Eintrag inkl. Printscreens, Logos und Beschreibung erstellt.

Wenn die Angaben von Google überprüft worden sind, muss die App entweder von ausgewählten Personen oder von beliebigen Nutzern getestet werden, sodass die App dann schlussendlich im Play-Store veröffentlicht wird.



11 Schlussteil

11.1 Sachergebnisse

11.1.1 Ursprüngliche Aufgabenstellung und Ziele

Das Ziel dieser Arbeit war eine Applikation mit einer einfachen und übersichtlichen Eingabe zu erschaffen, welche den aktuellen Blutalkoholgehalt errechnen und anzeigen soll.

Musskriterien:

- Anzeige des Blutalkoholgehalts mittels Eingabe konsumierter Getränke
- App-Kompatibilität mit Android-Endgeräten

11.1.2 Veränderungen der Aufgabenstellung / Zielsetzung

Alle Musskriterien konnten erfüllt werden. Somit mussten die Aufgabenstellung und die Zielsetzung nicht verändert werden.

Die Wunschkriterien wurden jedoch aus Zeitgründen nicht erarbeitet.

Wunschkriterien:

- App-Kompatibilität mit iOS-Endgeräten
- Festlegung einer Promille-Obergrenze, bei welcher eine Warnung erscheint.
- Möglichkeit zur Erstellung benutzerdefinierter Getränke

11.1.3 Erarbeitete Sachergebnisse

Die Anzeige des Blutalkoholgehaltes mittels Eingabe konsumierter Getränke konnte durch unsere entwickelte Applikation Kalkohol erreicht werden. Mittels Android Studio konnten wir eine Android-Applikation entwickeln, mit Emulatoren und einem Android Gerät testen und schlussendlich via Google-Playstore veröffentlichen.



11.2 Projektverlauf

11.2.1 Überblick Kosten- und Zeitmanagement

Zeitmanagement:

Folgend ein Ausdruck aus unserem nachgeführten Zeitplan:

TASK	Verantwortlich	Fortschritt	Start	Ende	Ressourcen Soll (h)	Ressourcen Ist (h)
Total		100%			80	87.5
Projektinitialisierung			11.05.20	13.06.20	6	6.5
Projektantrag	JS	100%	11.05.20	13.05.20	2	2
Zeitplan erstellen	PW	100%	13.06.20	13.06.20	3	2.5
Dokumentationslayout	JS	100%	12.06.20	12.06.20	1	2
Analyse			18.06.20	01.07.20	12	13
Ist Zustand definieren	PW	100%	18.06.20	21.06.20	1	1
Vorstudie / Plattformen definieren	PW	100%	21.06.20	24.06.20	4	6
Konzept erarbeiten	JS	100%	24.06.20	27.06.20	4	3
Pflichtenheft erstellen	JS	100%	27.06.20	01.07.20	3	3
Softwareentwicklung			20.07.20	16.08.20	34	45
Programmierung User Interface	PW	100%	20.07.20	30.07.20	12	20
Programmierung Logik / Berechnung	JS	100%	27.07.20	13.08.20	16	22
Testing / Bugfix	JS	100%	13.08.20	20.08.20	6	3
Dokumentation / Präsentation			17.06.20	10.09.20	28	23
Analyse	JS	100%	18.06.20	01.07.20	6	3
Softwareentwicklung	PW	100%	20.07.20	20.08.20	8	10
Schlussteil	JS	100%	24.08.20	31.08.20	6	2
Präsentation erstellen	PW	100%	04.09.20	10.09.20	8	8

Abbildung 46: Projektverlauf Zeitplan

Wie in der [Abbildung 46: Projektverlauf Zeitplan](#) zu sehen ist haben wir unsere mit 80 Stunden budgetierte Zeit um ca. 10% überschritten.

Die Projektinitialisierung haben wir knapp überschritten, die Erstellung des Dokumentationslayouts hat mehr Zeit als eingeplant gekostet.

In der Analyse konnten wir trotz weniger Aufwendungen in der Erstellung des Konzeptes und des Pflichtenheftes das Zeitbudget knapp nicht erreichen. Die Vorstudie hat wegen Beachtung von Wunschkriterien länger gedauert als geplant, bzw. hätte noch länger gedauert. Deshalb haben wir uns dazu entschieden, das Wunschziel iOS-Kompatibilität nicht zu erarbeiten, sodass wir das Zeitbudget nicht noch mehr überschreiten würden.

Die Zeitintensität der Softwareentwicklung wurde klar unterschätzt. Rückblickend ist diese Arbeit jedoch sehr lehrreich gewesen und vor allem können wir in einem evtl. neuen Projekt in derselben Richtung von unseren Erfahrungen profitieren und die Softwareentwicklung besser einschätzen.



Den Zeitplan für die Dokumentation und die Erstellung der Präsentation konnten wir grundsätzlich einhalten.

Die Einhaltung der geplanten Start- und Enddaten konnte bis zum Zeitpunkt der Softwareentwicklung eingehalten werden. Da wir jedoch genug Reserve bis zur Abgabe der Arbeit eingeplant haben, konnten wir diese Zeit nutzen, um die Softwareentwicklung abzuschliessen.

Kostenzusammenstellung:

Kostenpunkt	Kriterium	Budgetiert	Kosten	Eingehalten
Registrierungskosten für Google Developer Account	Soll	20 USD	20 USD	ja
Kosten Apple Developer Account	Wunsch	100 USD (jährlich)		Ja
Total		120 USD	20 USD	Ja

Da wir das Wunschkriterium iOS-Kompatibilität nicht erarbeitet haben, sind auch die Kosten für den Apple Developer Account entfallen.

11.2.2 Planungsqualität

Die Planung wurde zu Beginn des Projektes grundsätzlich genug detailliert erarbeitet. Jedoch fehlte ein Punkt für die allgemeinen Sachen der Dokumentation. Die Dokumentation hatten wir in die Punkte Analyse, Softwareentwicklung und Schlussteil aufgeteilt. Das Niederschreiben von z.B. der Einleitung, dem Konzept oder anderen Sachen wurde zeitlich nicht eingeplant. Dank genügend eingeplanter Reserve konnte dies jedoch in den anderen Punkten untergebracht werden. Daher haben wir den Schluss gezogen, in Zukunft die Dokumentation im Zeitplan entweder allgemeiner zu halten oder bis ins letzte Detail aufzuteilen.



11.3 Ausblick

11.3.1 Restaktivität

Für die Arbeit sind ausser der Präsentation am Samstag 26. September 2020 keine Restaktivitäten eingeplant.

11.3.2 Ergänzungen und Erweiterungen

Aus unserer Sicht sind folgende Ergänzungen bzw. Erweiterungen des Projektes möglich: (Dem Realisierungsaufwand geordnet, von tief zu hoch)

- Festlegung einer Promille-Obergrenze, bei welcher eine Warnung erscheint.
- Hinzufügen von zusätzlichen Sprachen
- Möglichkeit zur Erstellung benutzerdefinierter Getränke
- Grafische Darstellung der Alkoholpegelkurve
- App-Kompatibilität mit iOS-Endgeräten

11.3.3 Folgeprojekte

Folgeprojekte sind bis jetzt noch keine geplant.



11.4 Schlusswort / Reflexion

11.4.1 Patrick Wyser

Für mich war diese Projektarbeit ein toller Start in die Java Programmierwelt. Da ich in meinem jetzigen Job als Softwareentwickler für AV-Systeme (Crestron) viel mit der Erschaffung von Bedienoberflächen zu tun habe, viel es mir leicht, ein von meiner Sicht aus ansprechendes und einfach zu bedienendes GUI zu gestalten.

Die Arbeit in Android Studio finde ich sehr intuitiv und spannend. Ich habe diese Arbeit jedoch unterschätzt, da beim GUI von einer Android App viel mehr dahintersteckt als nur das Design selbst.

Im Projektteam konnten wir ein sehr angenehmes Arbeitsklima schaffen und uns gegenseitig bei Fragen unterstützen. Das Arbeiten mit einem Git-Repository, was ich bis vor dem Beginn dieser Ausbildung nicht kannte, vereinfacht das gemeinsame Arbeiten an einem Projekt sehr.

11.4.2 Janik Schilter

Meine bisherigen Programmierkenntnisse beliefen sich vor dieser Projektarbeit überspitzt gesagt auf «Hello World». Daher war es mein Ziel, mit dieser Projektarbeit tiefer in die Welt der Softwareentwicklung einzutauchen.

Zu Beginn der Projektarbeit hatte ich grössere Schwierigkeiten, da meinerseits schlichtweg zu wenig Know-How vorhanden war. Aber mithilfe von diversen Websites, lehrreichen Gesprächen im Projektteam und mit ausgebildeten Informatikern, war es mir möglich, das nötige Wissen für das Projekt anzueignen.

Die Zusammenarbeit im Projektteam empfand ich als sehr angenehm und unkompliziert. Durch regelmässige Projektbesprechungen im Microsoft Teams konnten wir trotz Corona-Krise eine einwandfreie Kommunikation und Zusammenarbeit gewährleisten.



12 Anhang

12.1 Projektunterlagen

Alle Projektunterlagen inkl. JavaDoc sind im ELAD unter folgendem Link abgelegt:

<https://elad.ch/olat/auth/RepositoryEntry/77234196/CourseNode/101466525741700/path%3D~/0>

12.2 GIT-Repository

Der Quellcode ist auf dem GIT-Repository unter folgender URL abgelegt:

<https://elad.ch/gitblit/summary/L-TIN-19-T-a!2-PA!Team06.git>

12.3 Quellenangaben

Informationen:

<https://de.wikipedia.org/wiki/Blutalkoholkonzentration>

<https://www.kenn-dein-limit.info/promille-bei-frauen.html>

https://www.suchtschweiz.ch/fileadmin/user_upload/DocUpload/alkohol_koerper.pdf

<https://developer.android.com>

Erstellung Logo:

<https://www.logomaker.com/de/>

Abbildungen:

- Abbildung 28: <https://developer.android.com/guide/topics/ui/controls/radiobutton>



12.4 Sitzungsprotokolle

Sitzungsprotokoll // Meeting 01 // 03. Juni 2020

Anwesende: Adrian Birrer, Albert Domgjoni, Andi Moser, Bruno Hammer, Daniel Betschart, Janik Schiler, Markus Kilian, Patrick Wyser, Pascal Rusca, Robin Egg, Silvio Lienhard, Yannik Blum

Abwesende: -

Sitzungsort: TEKO / Bei jedem Zuhause via Teams

Sitzungsdatum: 03. Juni 2020, Start: 16:45 Uhr

Protokollführer: Patrick Wyser

Moderation: Andi Moser

Traktanden:

1. Begrüssung
 2. Vorstellungsrunde Projektteams (Team 1-6)
 3. Rückblick pro Team
 - Bereits erledigte Arbeiten
 - Erlebte Schwierigkeiten
 - Stand im Zeitplan
 4. Ausblick (pro Team)
 - Nächste Ziele
 - Kommende Schwierigkeiten
 5. Offene Runde
 6. Abschluss
-

Besprechung:

Thema / Traktandum

1. Begrüssung

Andi begrüßt die Anwesenden und stellt nochmals die Traktanden vor.

2. Vorstellungsrunde Projektteams (Team 1-6)

Team 1: Markus Kilian – Daniel Betschart

Liederauswahl vereinfachen für Guggenmusik

Team 2: Adrian Birrer - Robin Egg

Netzwerk Überwachung

Team 3: Yannik Blum - Silvio Lienhard

Pflanzenbewässerungssystem



Team 4: Albert Domgjoni - Andi Moser

Youtube Video abspielen

Team 5: Pascal Rusca

Ferngesteuertes Fahrzeug steuern

Team 6: Janik Schilter – Patrick Wyser

Promille-Rechner

3. Rückblick pro Team

Team 1:

Erledigt:

Produktspezifikation

Dokumentation aufgesetzt

Probleme:

Koordination

Stand im Zeitplan:

Eng, jedoch bis jetzt gut eingehalten

Team2:

Erledigt:

Grober Terminplan besprochen

Probleme:

Noch keine

Stand im Zeitplan:

Noch keiner erstellt

Team3:

Erledigt:

Materialliste erstellt

Probleme:

Evtl. Kosten

Stand im Zeitplan:

Noch nicht gestartet, da Yannik am Zügeln, Start am 10.06



Team4:

Erledigt:
Projektplan angefangen

Probleme:
Wenig Zeit momentan

Stand im Zeitplan:
Erst in Erstellung

Team 5:

Erledigt:
Diverse Abklärungen

Probleme:
Bis jetzt noch keine

Stand im Zeitplan:
Noch nicht erstellt

Team 6:

Erledigt:
Vorbesprechungen betreffend z.B. Programmiersprache

Probleme:
Bis jetzt noch keine

Stand im Zeitplan:
Noch keiner erstellt

4. Ausblick (pro Team)

Team 1:

Nächste Ziele:
Wissen aneignen
Gewisse Ressourcen schon aufsetzen

Kommende Schwierigkeiten:
Evtl. Wissensaneignung



Team 2:

Nächste Ziele:
Zeitplan erstellen
Aufsetzen der Projektstruktur

Kommende Schwierigkeiten:
Einarbeiten
Unterschied Wissensstand

Team 3:

Nächste Ziele:
Zeitplan
Meilensteine definieren
Konzept erstellen

Kommende Schwierigkeiten:
Lieferschwierigkeiten

Team 4:

Nächste Ziele:
Zeitplan definieren
Verantwortungen definieren

Kommende Schwierigkeiten:
Welche Hilfsmittel werden eingesetzt, Balance zu finden für den Zeitaufwand zur Evaluierung von Tools und Sprachen...

Team 5:

Nächste Ziele:
Zeitplan erstellen
Konzept

Kommende Schwierigkeiten:
Kontroller, Remote
Ob Website oder anderes

Team 6:

Nächste Ziele:
Zeitplan
Konzept
Verantwortlichkeiten bestimmen

Kommende Schwierigkeiten:
Evaluation Software, Programmiersprachen

**5. Offene Runde**

Andi eröffnet die Offene Runde:

- Expertenforum erstellen: So können Personen sich als Experten für Themen melden, bei denen sich dann andere melden können. So kann sich gegenseitig unterstützt werden.
- Klasse findet diesen Vorschlag top.
- Bruno wendet noch ein, Idee gut, jedoch als Tipp: nicht zu viel Zeit in die Expertenfindung stecken. Die Klasse kennt sich ja gegenseitig schon ziemlich gut.

Bruno:

- Projektarbeit dient zum Lernen
- Nicht erst zwei Wochen vor der Abgabe starten 😊

7. Abschluss

Es hat niemand mehr zusätzliche Bemerkungen:

- Protokolle in OLAD Ordner laden.

Pendenzenliste:

Pendenz	Verantwortlich	Datum
Protokolle ins OLAD laden	Alle Protokollführer	ASAP

Ende der Sitzung, 03. Juni 2020, 17:15 Uhr

Luzern, 5. September 2020

Protokollführer: Patrick Wyser

Moderation: Andi Moser

Unterschrift:



Sitzungsprotokoll // Meeting 02 // 26. August 2020

Anwesende: Adrian Birrer, Andi Moser, Bruno Hammer, Janik Schiler, Markus Kilian, Patrick Wyser, Pascal Rusca, Robin Egg, Yannik Blum

Abwesende: Silvio Lienhard, Albert Domgjoni, Daniel Betschart

Sitzungsort: TEKO / Bei jedem Zuhause via Teams

Sitzungsdatum: 26. August 2020, Start: 16:30 Uhr

Protokollführer: Patrick Wyser

Moderation: Adrian Birrer

Traktanden:

1. Offene Punkte/Fragen aus letzter Sitzung
2. Rückblick/Ausblick
 - a. Rückblick: Stand bisheriger Arbeit, Einhaltung Terminplan, vorhandene Schwierigkeiten
 - b. Ausblick: Was ist noch offen, kann Abgabetermin gehalten werden
3. Unklarheiten, offene Fragen

Besprechung:

Thema / Traktandum

1. Offene Punkte / Fragen aus letzter Sitzung

Team 1: Markus Kilian – Daniel Betschart

Bewertung von Lieder

Team 2: Adrian Birrer – Robin Egg

Netzwerk Überwachung

Team 3: Yannik Blum – Silvio Lienhard

Pflanzenbewässerungssystem

Team 4: Albert Domgjoni – Andi Moser

Youtube Video abspielen

Team 5: Pascal Rusca

Ferngesteuertes Fahrzeug steuern

Team 6: Janik Schilter – Patrick Wyser

Promille-Rechner



2. Rückblick / Ausblick

Team 1:

A:

Datenbank wurde mit Raspberry aufgesetzt. 3 Rollen wurden definiert.
Lieder können abgespielt und bewertet werden.
Im Zeitplan gut drin. Bei ein paar Sachen ein bisschen hinten nach, jedoch auch gewisse Sachen schon erledigt.

B:

Abgabe kann eingehalten werden

Team 2:

A:

Das Tool funktioniert + Telegramm an Handy gesendet werden.

B:

Dokumentation ist noch vieles offen.

Team 3:

A:

Pflanzenbewässerungsprototyp läuft. Ein Sensor ist aber evtl. defekt.
Termin kann eingehalten werden.

B:

Sensor austauschen.

Team 4:

A:

Andi hatte leider keine Zeit beim abgemachten Datum.
Dokumentation schon recht weit.

B:

Effektive Umsetzung muss noch erledigt werden.

Team 5:

A:

Doku fast fertig.
Prototyp ist fertig gebaut
Termin kann eingehalten werden.

B:

Testbericht + Verzeichnisse in Doku
Sauber löten und verbauen

**Team 6:**

A:

SW in Bearbeitung, fast fertig.
Termin kann eingehalten werden.

B:

App auf Gerät laden.
Testing und Dokumentation.

4. Unklarheiten / offene Fragen

In welchem Rahmen ist die Dokumentation so durchschnittlich?
30-60 Seiten.

Dokumentation von Software:

Main Methode

Sich selbst die Frage stellen: Was brauche ich, wenn ich den Code in einem Jahr anpassen muss.

Wenn Anleitungen erstellt wurden, welche für einen selbst etwas bringen, können diese drin gelassen werden.

Für Analysen: Wie sollen Evaluationen von Technologien gemacht werden?

Killerkriterien setzen

Kriterien setzen.

Beispiel:

Muss Java sein

Abgabe:

Im Git

Git freischalten für Bruno

Doku als PDF ins Git

Physische Abgabe wird noch von Bruno abgeklärt.

Ende der Sitzung, 26. August 2020, 17:10 Uhr

Meggen, 26. August 2020

Protokollführer: Patrick Wyser

Moderation: Adrian Birrer

Unterschrift: