Audit 3
Entwicklungsprojekt
WS2022/23

Annika Lenneper
Marko Kalaburda
Christian Pankiv

TH Köln Medieninformatik B.A.

Inhalt

- Iterationen
- PoC: WearOS Kommunikation
- PoC: Android Notifications
- PoC: Analyse-Phase (Golden Behaviors)
- Klassenstruktur
- Rapid Prototype User Journey
- Ausblick

Iterationen

- Keine Einbindung zu Datenbank mehr geplant
  - Stattdessen: lokale Speicherung von Daten
  - PoC zur Datenbankverbindung entfällt damit
  - Positiv für den User: persönliche Daten, Informationen zu Gewohnheiten und Handlungen bleiben in eigener Hand

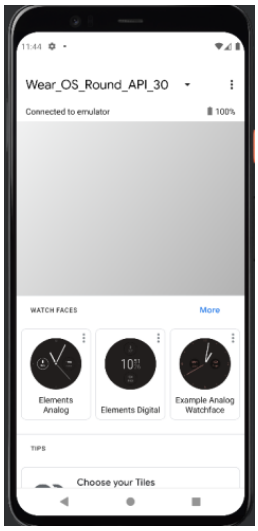
Die vorher in erwägung gezogene Einbindung der Daten in eine Datenbank fällt mit fortschreitendem Projektfortschritt weg, da es keine Notwendigkeit gibt kleinere Datensätze mit NutzerInfortmationen aufwendig zu verwalten.
Ein sich daraus ergebender POC fällt somit weg und das System wird sicherer für Nutzerdaten. Der Nutzer kann bei Entfernen der App alle seine Daten Löschen.

POC: WearOS Kommunikation

Verbindung zwischen App und Smartwatch (in unserem Fall über Browser)
Demonstration der Funktionsweise der Schnittstelle
Exit-Kriterien:
Tile ist bedienbar
Tiles zeigt synchrone Informationen der Smartphone App an
Tile kann Informationen an Smartphone App Senden
Fail-Kriterien:
Verbindung nicht möglich
Datenübertragung nicht möglich
Synchronisation nicht möglich

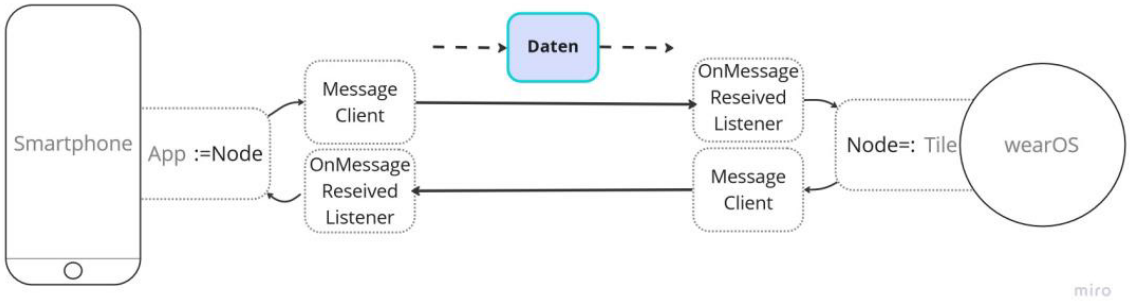
Der POC zur Wearable soll erläutern, ob eine schnelle und synchrone Kommunikation eines Wearable Tiles mit einer App einfach und schnell umsetzbar ist.

POC: WearOS Kommunikation



Für die Darstellung von Informationen auf einem Wearable wurde sich für sogenannte Tiles entschieden: Widget-ähnliche Anwendungen, die schnelle und einfache Informationen anzeigen, ohne eine App im Wearable öffnen zu müssen, da sie ähnlich wie Watchfaces durch einen Swipe auf der Uhr angezeigt werden. Sie sind für den Usecase sehr nützlich.
Die Verbindung von einem Smartphone und einem Wearable wird durch die im Playstore verfügbare WearOS App ermöglicht.
Für den POC genügt es sowohl die Smartwatch als auch das Smartphone in Android Studio durch einen Emulator zu bedienen: ein gleiches Vorgehen ist auch bei echten Geräten möglich.

POC: WearOS Kommunikation



Bei der Kommunikation zwischen WearOS und Smartphone wird auf Nodes zurückgegriffen, die als Schnittstelle der im Netzwerk verfügbaren Systeme dienen.
Über den Message Client können dann Nachrichten an andere Nodes versendet werden. Der Zugriff wird über die jeweiligen Id-Nummern durchgeführt.
Erhalten können diese Nodes dann Nachrichten über die innere Klasse MessageClient.OnMessageReceivedListener, die als Listener auf Nachrichten wartet und anschließend nutzen kann.
Alternativ können Daten auch über den Data-Client versendet und über alle Nodes synchronisiert werden. Es erstellt eine Kopie bevor es die Daten synchronisiert.
Der Programmierte POC Wird im weiteren Verlauf vollständig Entwickelt.

POC: Android Notifications

- Notifications erinnern Nutzer an das System
- Erleichtern die Datenerhebung zu der Einhaltung der Habit-Bildung

POC:

- Exit-Kriterium: Notification erscheint zum geplanten Zeitpunkt und auch bei geschlossener Applikation
- Fail-Kriterium: Notification erscheint nicht, oder nur bei geöffneter Applikation

In dem Android Notifications POC Soll getestet werden, ob und in welchen Umfang Notifications geplant an der Nutzer gesendet werden können, auch wenn das System beispielsweise nicht im Vordergrund geöffnet ist oder das Gerät neu gestartet worden ist.

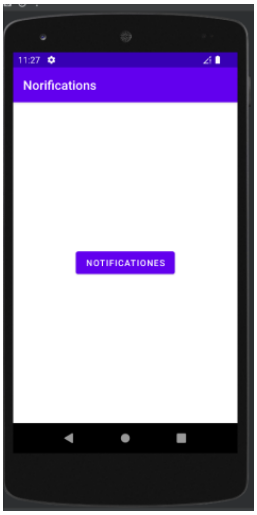
POC: Android Notifications

- Umsetzung des POC´s
- Zeitmanagement durch WorkManager
- Notifications durch NotificationChannel, NotificationManager, NotificationCompat und NotificationManagerCompat
- Änderung der Activities durch Intent und PendingIntent

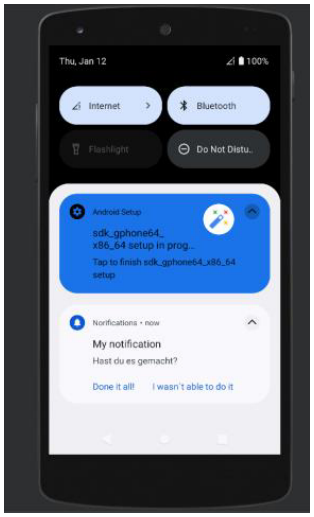
Der WorkManager soll eine persistente Haltung der Notifications ermöglichen: Er ermöglicht das übergeben von Tasks an einen Systeminternen Handler, der Aufgaben zu den jeweiligs festgelegten Zeitpunkten erledeigen kann. So könnenunter anderem Notifications und Erinnerungen an den User gesendet werden, Bsw. Je nach eingestelltem Intervall oder Häufigkeit.
Die Notifications wurden nach der Android Studio Documentation gestaltet.
Um die Activities zu ändern werden Intents benötigt, dies wird auch im oberen Link beschrieben.

POC: Android Notifications

Anfangsbildschirm



Notification



Der WorkManager hat eine Mindestdauer von 15 Minuten, nachdem eine Notifikation versendet wird.
Dies soll in folgenden Implementatinen dazu führen, dass der Nutzer dei Notifikationen selber einstellen kann.
Der Notificationscreen wird durch das Auftauchen des Icons oben links verdeutlicht.
Die Notification gibt einen Text und zwei Optionen wieder, welche jeweils eine andere Activity öffnen.
Im Verlauf des Projekts soll daraus eine Variable gemacht werden, welche persistent gespeichert wird.

POC: Android Notifications

Wahl links

Wahl rechts

Die zwei Optionsmöglichkeiten sind als vorläufige Celebration gedacht bzw. ein Hilfetext bei dem Scheitern der eigenen Ambitionen. Sie sind aus den Buttons der Notification zu erreichen.
Der Programm Code ist im Github Repo auffindbar: https://github.com/Chrspa/EntwicklungsProjektWS2022-23-Chris-Marko-Annika/tree/main/Audit3/EntwicklungsProjektWS2022-23-Chris-Marko-Annika-master

PoC: Analyse-Phase

- Testweise Umsetzung in IntelliJ
  - Lauffähig, jedoch ohne komplexes Fehler-Handling
  - Soll in Zukunft noch mehr Fälle abdecken/ detailliertere Rückmeldungen geben
  - Texte noch vorläufig
- Erster Aufbau der Klassenarchitektur
  - Klassen, Eigenschaften und Methoden

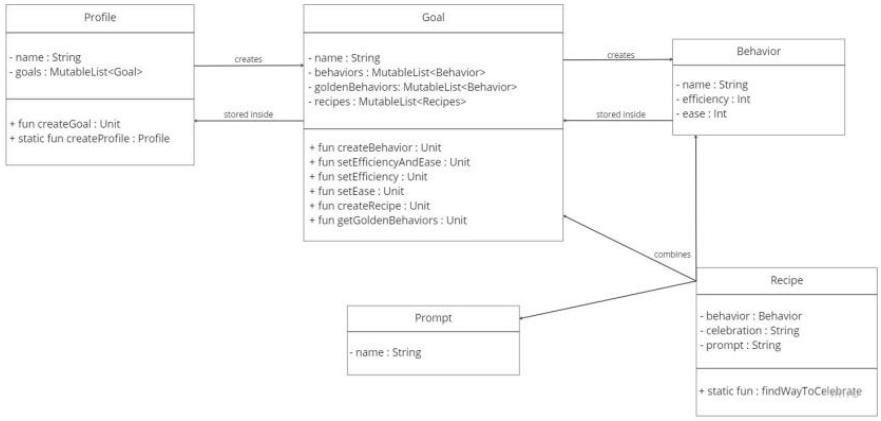
Durch den PoC zur Analyse-Phase sollte der Weg vom Setzen eines Ziels über das Sammeln und Auswerten von Verhaltensweisen hin zur Ausgabe der Golden Behaviors in Code umgesetzt werden. Die Implementierung erfolgte zunächst in IntelliJ, da zuerst einmal Ablauf und Struktur festgelegt werden und Schwierigkeiten mit dem User Interface oder der Parallelität von Abläufen an diesem Punkt noch keine Rolle spielen sollten. Für die Umsetzung wurden erste Klassen, Eigenschaften und Methoden definiert. Die Auflistung folgt auf den nächsten Folien und entspricht dem für den hier besprochenen PoC nötigen Umfang. In den nächsten Arbeitsschritten sollen, wo nötig, weitere Klassen und Methoden definiert werden. Die Ansprache des Users und die Textausgaben sind an dieser Stelle als vorläufig anzusehen. Der PoC soll hier entsprechend des definierten Ziels aus dem zweiten Audit zeigen, dass Golden Behaviors aus einem Set von Behaviors nach den Kriterien "Effizienz" und "Leichtigkeit der Ausführung" ausgewählt werden können

PoC: Analyse-Phase/ Use Case

- Anschließend Versuch der Umsetzung als Use Case in Android Studio, Vorläufer des Prototypes
  - Ein- und Ausgaben über editView/ textView
  - Prototypische Umsetzung auf einem Screen
  - Callbacks zum Aufruf der jeweils nächsten Funktion
  - Bisher keine Unterscheidung von verschiedenen Activities
- Schwierigkeiten
  - Parallelität von Prozessen
  - Abwarten von Nutzereingaben
  - Navigation

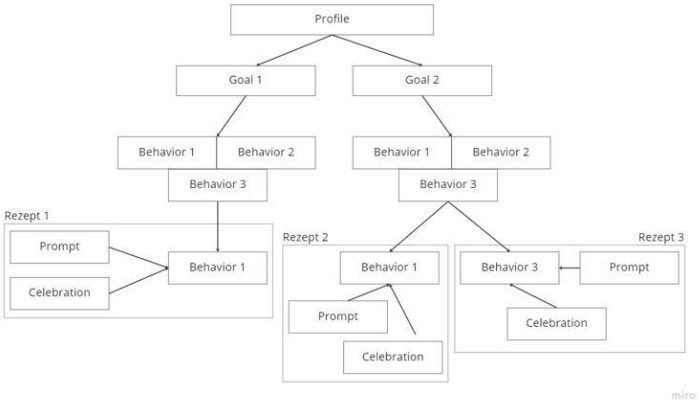
Bei dem an den PoC anschließenden Versuch, die Code-Bestandteile in Android Studio für eine erste Use-Case-Demonstration umzusetzen, mussten einige Änderungen vorgenommen werden. Die Interaktion mit dem User erfolgt hier durch das User Interface von Android, weshalb das Handling der Ein- und Ausgaben entsprechend geändert muss. Die Umsetzung sollte zunächst prototypisch erfolgen. Es wurde die mainActivity mit einem Textfeld, einem Eingabefeld und einem Button genutzt. Um auf dem Android-Betriebssystem lauffähig zu sein, sind noch weitere Anpassungen notwendig. So führen die betriebssystem-inhärenten Besonderheiten dazu, dass Prozesse parallel ausgeführt werden und an wichtigen Stellen nicht auf den User-Input gewartet wird. Eine nähere Auseinandersetzung mit Android, Kotlin-Koroutinen und Event-Handlern ist daher für die weitere Entwicklung unausweichlich.

Klassenstruktur



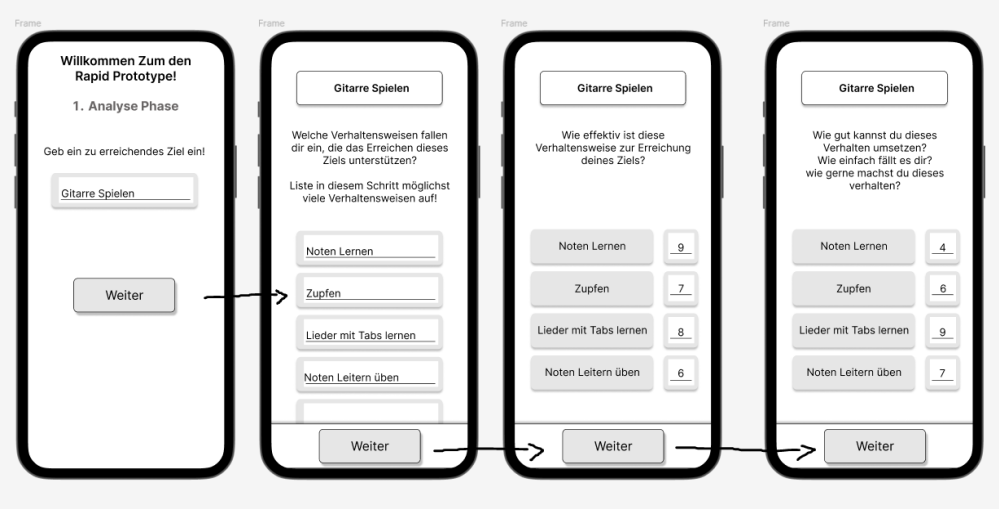
Das Diagramm zeigt den Aufbau der bis zu diesem Zeitpunkt entwickelten Klassenstruktur mit den Klassen "Profile", "Goal", "Behavior", "Prompt" und "Recipe", ihren Eigenschaften und Methoden. Es handelt sich um eine vorläufige Darstellung. Die Entwicklung reicht nur wenig weiter, als für die Umsetzung des Analyse-Parts notwendig. In der späteren Review-Phase werden beispielsweise Werte für die Effektivität von Prompts und/oder Recipes benötigt, die über spezielle Methoden gesetzt und verändert werden können. Diese sind kritisch, um den User passend anzuleiten und das Programm entsprechend der Erfolge oder Misserfolge anzupassen.
Die Diagramme sind auch im Miro bei der Klassenstruktur einsehbar:
https://miro.com/app/board/o9J_lJqd05Y=/?share_link_id=222737666303

Klassenstruktur



Das Baumdiagramm zeigt, wie die einzelnen Instanzen der Klassen miteinander verbunden sind. Zu einem Profil können mehrere Ziele gehören, wobei jedes Ziel eine Sammlung von ihm dienlichen Verhaltensweisen beinhaltet. Diese werden nach dem Prinzip der Fogg'schen Focus Map nach den Kriterien "Effizienz" und "Leichtigkeit der Ausführung" gefiltert, so dass eine geringere Anzahl übrig bleibt (die "Golden Behaviors" nach Fogg). In den Rezept-Instanzen werden schließlich konkrete, zu einem Ziel gehörende Verhaltenweisen mit einem Prompt und einer Celebration verbunden.

- Rapid Prototype User Journey



Es wurde Zusätzlich noch eine User Journey als Prototyp in Figma Erstellt. der Tatsächlichen Prototype wird im nächstem Schritt in Android Studio Code Implementiert und fertiggestellt. Die Code Teile aus den POCs dienen dazu als Bauteile.
Die in dem Prototypen dargestellten Schritte entsprechen den Phasen der UserJourney für die Erstellung eines Habits.(https://miro.com/app/board/o9J_lJqd05Y=/?share_link_id=129687251729)
Der Gesamte Prototyp ist in Figma Interaktiv Einsichtlich: https://www.figma.com/proto/qdb1cWMb2tdoxvQTF65oxg/Untitled?node-id=1%3A2&scaling=scale-down&page-id=0%3A1&starting-point-node-id=1%3A2

Ausblick

- Implementierung eines Use Cases als Prototype
  - Weiterentwickeln der Klassen und Methoden für die Review-Phase
  - Erweiterung der Funktionen bis zum Review inklusive Iterationen
  - Aufteilen der Phasen und Abschnitte auf verschiedene Activities
  - Prototypischen UI-Design
  - Sinnvolles Zusammenführen der Erarbeiteten Use Cases
  - Textgestaltung im Sinne der Fogg'schen Maximen, gute User-Ansprache, Interaktion und Zusatzinformationen/ Hilfestellungen entwickeln

Im Laufe der weiteren Entwicklung des Projektes soll ein Use Case prototypisch in Android Studio implementiert werden. Dazu werden die Klassen und Methoden weiterentwickelt, um auch die Review-Phase abzudecken. Außerdem werden die technischen PoCs zur Verbindung von Smartwatch und App sowie zum Senden von Notifications in der Praxis-Phase eingebunden, um den User bei der Ausführung der Rezepte zu begleiten. Die verschiedenen Phasen und/oder Code-Abschnitte sollen auf eigene Activities aufgeteilt und entsprechende UIs entwickelt werden.

Einen weiteren wichtigen Punkt stellen die Texte dar. Es muss eine geeignete Sprache gefunden werden, die beim User auf positive Resonanz stößt und ihn bei der Ausführung der Rezepte und der Interaktion mit dem System motiviert. Zudem sollen an verschiedenen Stellen Hilfestellungen angeboten werden, beispielsweise dazu, wie man eine Verhaltensweise konkretisiert oder im Bedarfsfall "verkleinern" kann.