Architekturentwurf

Fitness-Assistent

von Jonas Barth

Version: 1.0 – 14.06.2020

Versionstabelle

Versionsnummer	Autor	Änderungsvermerk	Zu Anforderungsdokument Version
0.1	Jonas Barth	Initiale Fassung	1.0 – 14.06.2020

I. Inhalt

II. Einleitung

Der vorliegende Architekturentwurf beschreibt die Systemarchitektur der Fitness-Assistenz-App mit dem vorläufigen Namen “SoFit”. Es beschreibt die Eigenschaften und die Beziehung der einzelnen Systemkomponenten.

Der Fitness-Assistent hat die drei Hauptfunktionen Zielsetzung, Training und Monitoring. Alle weiteren Funktionen lassen sich diesen Hauptbereichen unterordnen und sind über die drei Hauptseiten Home, Gym und Stats aufrufbar. Dort werden dem Anwender Informationen aufbereitet, die sich aus ausgelieferten Daten und anwender-abhängigen Daten errechnen. Die Berechnung der Daten ist das Alleinstellungsmerkmal der Fitness-Assistent-App.

Dieses Dokument wurde initial für Softwareentwicklung geschrieben. Es besteht weder ein Anspruch auf Vollständigkeit noch werden Funktionen zugesichert.

III. Glossar

Practice	ein Training, bestehend aus mehreren Übungen (workouts)
Workout	eine Übung
Score	erzielte Leistung in Prozent

IV. Abbildungsverzeichnis

V. Tabellenverzeichnis

VI. Literaturverzeichnis

[LEXDB01] datenbanken-verstehen.de. Stammdaten. https://www.datenbanken-verstehen.de/lexikon/stammdaten/#. Zugriff am 10.06.2010

[WIKISWIFT01] Wikipedia. Swift (Programmiersprache). https://de.wikipedia.org/wiki/Swift_(Programmiersprache) Stand und Zugriff am 10.06.2020

1. Architekturstrategie

Zur Erstellung der Architekturstruktur wurde das Anforderungsdokument zunächst in funktionale Einheiten gegliedert. Diese erleichtern das Identifizieren der vorliegenden Architektur, und die Auswahl von alternativen Technologie-Stacks.

2. Statische Sicht – zu persistierende Daten

Zunächst sollen die persistierenden Daten näher beschrieben werden. Sie bilden das Grundgerüst der Softwarearchitektur.

Die Fitness-Assistent-App hat Stammdatentabellen mit der Fitness-Kategorie (fitnessClass) und mit den Übungen (workout), sowie die Bewegungsdaten der Trainingshistorie (practiceHistory). Jede Entität enthält einen Primärschlüssel mit dem Attribut id. Für die Stammdaten wurde für jede ID auch ein sprechende Name (name) vergeben, der für den Anwender verständlich ist.

In der Tabelle fitnessClass ist eine Kennzahl für den Fortschritt (progress) gespeichert, der über einen Algorithmus die Leistungssteigerung des Nutzers berechnet. Die Leistungssteigerung wiederum wird in den Bewegungsdaten der Tabelle Trainingshistorie (practiceHistory) gespeichert. Der Nutzer hat zu jedem Zeitpunkt genau 1 Fitness-Kategorie, die sich auf m Übungen auswirkt, die Beziehung ist daher 1:m.

Die Übungen einer jeden Trainingseinheit werden aus der Tabelle Fitness-Kategorie (fitnessClass) gelesen. Jede Übung hat eine gewissen Schwierigkeit (level), eine empfohlenen Anzahl Wiederholungen (repetition) und Daten für eine Animation (clips). Da sich die Trainings aus den Übungen zusammensetzen, besteht eine n:m Abhängigkeit, n Übungen werden von m Trainings verwendet.

Die Entitäten enthalten noch viele weitere Attribute, auf die nur im Coding zugegriffen wird. So bleiben die Datensätze in practiceHistory redundant.

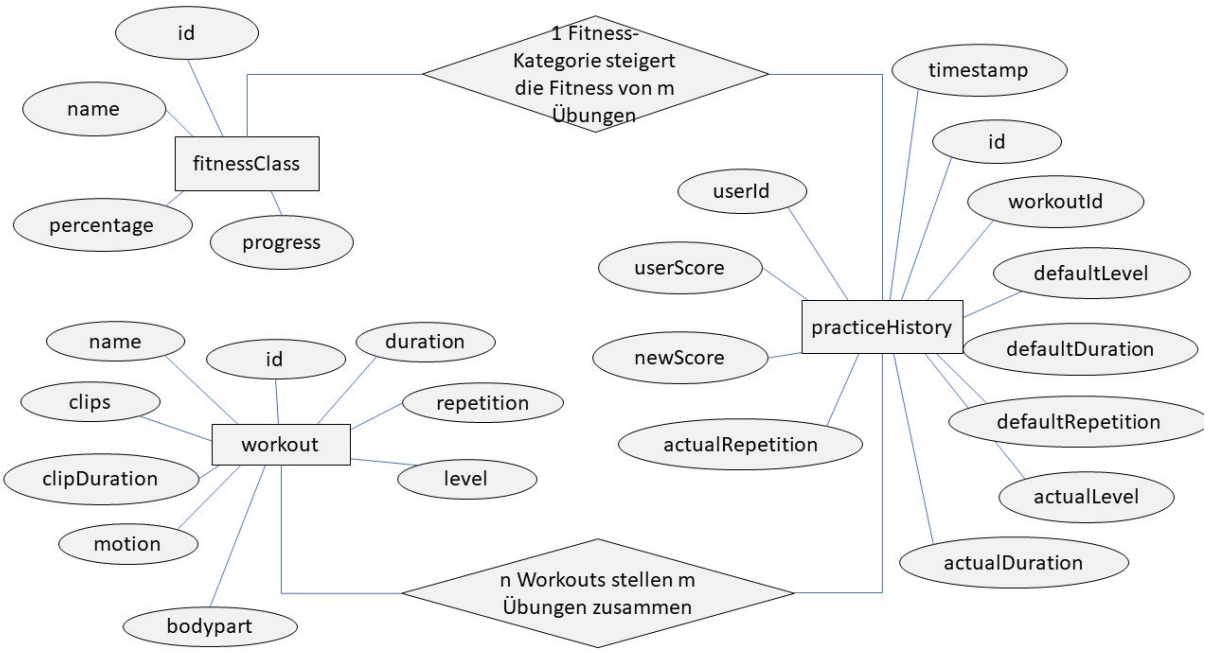

Abbildung 1: ERM-Diagramm

fitnessClass	
id	1
name	Rookie
percentage	1.00
progress	0.25

workout	
id	1
name	Pushups
clips	["pushups_001", "pushups_002"]
clipDuration	0.60
level	1.00
duration	12.00
repetition	10.00
bodypart	chest
motion	strength

practiceHistory	
id	1
timestamp	01.05.2020 12:24
userId	1
userScore	0.75
workoutId	1
defaultLevel	0.80
defaultDuration	4.80
defaultRepetition	8.00
actualLevel	0.80
actualDuration	4.80
actualRepetition	8.00
newScore	0.76

Tabelle 1: Beispielwerte in Tabellen

3. Statische Sicht: Objekttypen zur Laufzeit

In Swift bestehen komplexere Views immer aus kleineren, simpleren Views. Für eine strukturierte Darstellung sollte man sich auf die Haupt-Views beschränken. Die nachfolgende Abbildung zeigt die wesentlichen Views der Fitness-Assistent-App mit ihren Funktionen.

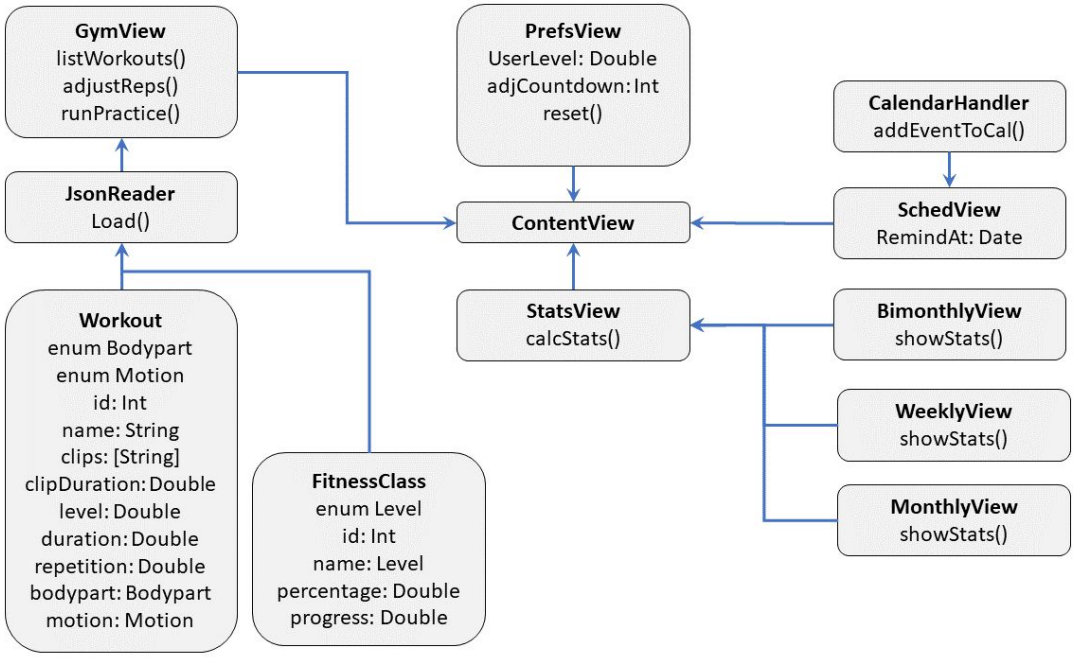

Abbildung 2: UML Klassendiagramm

Im ContentView treffen sich die Pfade aus allen anderen Views. Hier wird nach dem Laden der App die Home-Seite gezeigt, von der alles ausgeht. Im GymView gibt es die Methode listWorkouts(). Diese listet eine Auswahl an Übungen auf und zeigt dazu die empfohlene Anzahl an Wiederholungen an. Die ausgewählten Übungen entsprechen dem Training des Tages. Mithilfe von adjustReps() kann die Anzahl der Wiederholungen pro Übung angepasst werden. Die Methode runPractice() startet das Training und die Übungen beginnen. Die Informationen zu sämtlichen Workout-Objekten, wie auch Fitness-Class-Objekte werden aus den JSON-Files ausgelesen und im ganzen Projekt zugänglich sind. Dies geschieht über load()-Methode der Klasse JsonReader. Im PrefsView können Einstellungen bezüglich des Leistungsfortschritts des Anwenders, seine Pausen zwischen den Workouts und seinen bisher gespeicherten Scores vorgenommen werden. Der StatsView enthält den Algorithmus zum Berechnen der Nutzerstatistiken, die in drei unterschiedlichen Zeitintervallen angezeigt werden können. Schließlich bietet der SchedView die Möglichkeit Erinnerungen an die vom Anwender vorgenommen Trainingseinheiten in seinem Kalender außerhalb der App zu speichern.

4. Dynamische Sicht

Funktionseinheiten

Die Anforderungen des Anforderungsdokumentes lassen sich zu folgenden Funktionseinheiten zusammenfassen:

Nummer	Beschreibung der Anforderung	Referenz
F01	Home-Seite	S.6/7 Nr.1 a,d
F02	Gym-Seite	S.7 Nr.2 a-k
F03	Stats-Seite	S.7 Nr.3 a-h

F04	Präferenzen-Seite	S.6 Nr.1 b
F05	Shedule-Seite	S.6 Nr.1 c

Tabelle 2: Funktionseinheiten

Prozessbeschreibung für Funktionseinheit F01

Die Home-Seite soll den Nutzer ansprechen und mit hilfe eines Motivationsslogans zusätlich zum Trainieren motivieren. Unterhalb des Slogans ist eine Tabbar, mit der man auf die Gym- oder Stats-Seite kommt.

Prozessbeschreibung für Funktionseinheit F02

Bevor ein Training (Practice) beginnt wird dem Nutzer eine Liste der zu absolvierenden Übungen (Workouts) angezeigt. Die Anzahl der Wiederholungen wurde nach einem Algorithmus berechnet, sowohl den den Leistungsstand des Nutzers berücksichtigt, als auch einen Faktor einbezieht, um die Leistung stetig zu steigern. In der Liste kann der Nutzer aber auch die Anzahl der Wiederholungen anpassen, je nach Präferenzen. Somit kann der Nutzer die Leistungssteigerung bremsen oder beschleunigen.

Von der Gym-Seite kommt der Benutzer bei Bedarf direkt zurück auf die Home-Seite. Dieser Absprung entspricht einem Abbruch, die Übungen werden dann nicht gestartet.

Falls Anwender seine Auswahl an Übungen mit der Anzahl der Wiederholungen bestätigt, hat beginnt sein Training. Zuerst startet ein Countdown. Während dessen kann der Anwender noch Änderungen an der Anzahl der Wiederholungen der Übungen vornehmen. Gleichzeitig mit dem Countdown beginnt die Instruktion des Übung. Dabei wird der Name, sowie eine Animation der Übung gezeigt, die die Durchführung des Workouts klar aufzeigt. Sobald der Countdown abgelaufen ist, können keine Änderungen mehr vorgenommen werden und mit etwas Verzögerung wird ein Completed Button aktiviert, mit dem der Nutzer eine Übung vorzeitig beenden kann. Die Anzahl der absolvierten Wiederholungen bis zum Drücken des Completed Buttons wird dann nur gewertet. Die Übung wird dann als absolviert erkannt und es startet als nächstes der Countdown für die folgende Übung. Falls der Anwender den Skip Button drückt, wird die aktuelle Übung nicht gewertet und eine andere Übung wird eingeschoben. Die Übungsnummer bleibt dabei erhalten.

Prozessbeschreibung für Funktionseinheit F03

Von der Stats-Seite kommt der Benutzer ohne weiteres wieder zurück auf die Home-Seite. Der Nutzer kann sich auf dieser Seite ein Diagramm seiner erbrachten Leistung über eine Woche, zwei Wochen oder einen Monat anzeigen lassen. Außerdem kann ein Balken- oder Kuchendiagramm den Anteil der Übungen pro Körperpartie, sowie den Anteil der Übungen pro Motorik (Kraft / Ausdauer) ebenfalls über die drei verschiedenen Zeiträume darstellen. Zusätzlich können die letzten Kennzahlen für einen ausgewählten Zeitraum von einer Woche, zwei Wochen oder einem Monat in einer Liste angezeigt werden. Anfangs werden alle Statistiken im Eine-Woche-Zeitabschnitt angezeigt. Durch einen Druck auf den entsprechenden Button kann der Nutzer die Basis der Stats-Seite zwischen den drei Zeitabschnitten variieren.

Prozessbeschreibung für Funktionseinheit F04

Der Anwender hat auch die Möglichkeit, seine Lieblingsübungen zu favorisieren. Das bewirkt eine erhöhte Chance, diese Übung in den Practices aufzufinden. Ebenfalls kann der Fitness Level angepasst werden. Ein höherer Fitness Level nimmt einen Einfluss auf die vorgeschlagene Anzahl an Wiederholungen eines Workouts. Gleichzeitig beeinflusst der Fitness Level die Geschwindigkeit der Leistungssteigerung. Bei einem niedrigen Fitness Level kommt es zu hohen Leistungssteigerungen, bei höheren Levels werden die Steigerungen nur noch marginal. Die Leistungssteigerung äußert sich über die Anzahl der zu absolvierenden Wiederholungen. So wird der Nutzer motiviert, Übungen maximal zu wiederholen und somit seinen Level zu steigern.

Prozessbeschreibung für Funktionseinheit F05

Die Schedule-Seite ermöglicht es Erinnerungen an das Training zu bestimmten Tagen und Uhrzeiten im eigenen Kalender zu speichern. So kann der Anwender Push-Up Nachrichten der Kalender-App erhalten, die an die Trainingszeit erinnern.

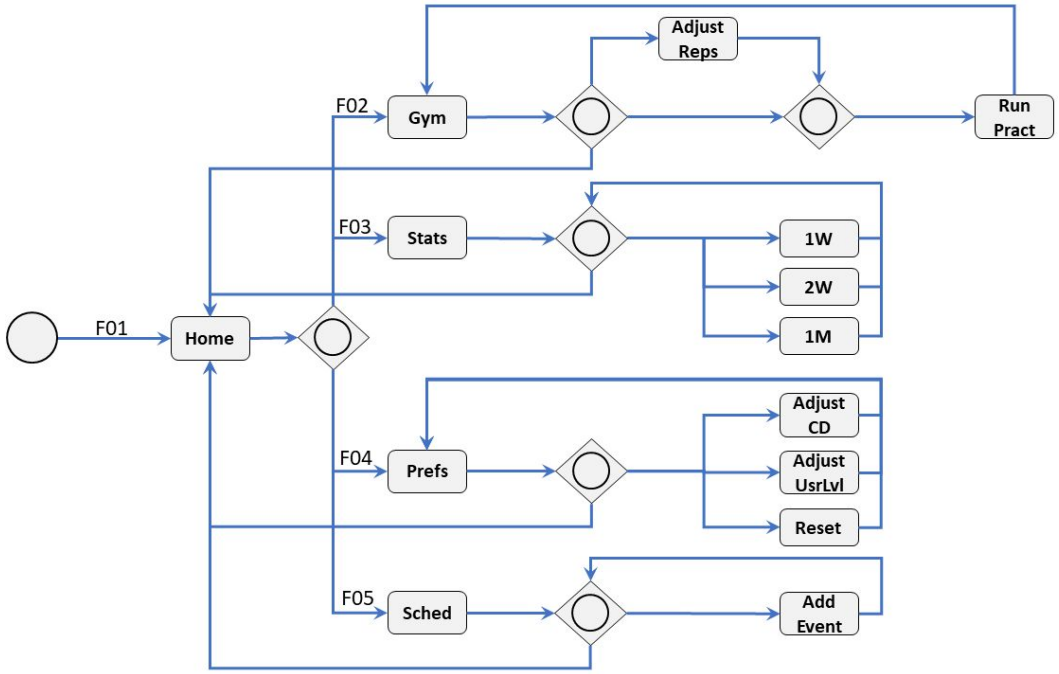

Abbildung 3: Business Process Model & Notation

5. Auswahl des Technologie-Stacks

Um geeignete Architekturen für die Anwendung zu finden, mussten bestimmte Fragestellungen geklärt werden. Hierbei wird knapp die Strategie dargestellt, wesentliche Alternativen vorgestellt und Entscheidungsgründe für und gegen Architekturen genannt. Alternativen und deren Unterschiede untereinander werden durch Entscheidungsmatrizen visualisiert. Im Anschluss werden die Architekturen separat vorgestellt. In einem abschließenden Fazit wird dann die Entscheidung für eine Architektur bekannt gegeben.

Entscheidung der Entwicklungsplattform

Da die Fitness-Assistent-App speziell für iOS Geräte erstellt werden soll, schränkt das die Auswahlmöglichkeit der passenden Architekturen ein. Meine Entscheidung sollte zwischen SwiftUI und Flatter fallen.

	XCode mit SwiftUI	Flutter mit Dart
Persönliches Interesse sich der Architektur anzunähern	Ja	Nein
Auf IOS Apps spezialisiert	Ja	Nein
Auf mehreren Betriebssystemen verfügbar	Nein	Ja

Tabelle 3: Entwicklungsplattform

Flutter:
Flutter ist ein Open-Source UI-Entwicklungs-Kit von Google. Es basiert auf der Programmiersprache Dart, eine moderne Alternative zu JavaScript. Programme, die für Android, iOS, Windows, Linux oder macOS geschrieben wurden, werden speziell für die jeweiligen Plattformen compiliert und in einer Dart-VM ausgeführt.

XCode:
XCode ist eine von Apple geschaffene Entwicklungsumgebung speziell für macOS. Mit XCode lassen sich Programme für macOS, iPadOS, iOS, watchOS und tvOS schreiben. Dabei unterstützt XCode mehrere Programmiersprachen, wie Swift und Objective-C. Am bekanntesten ist es jedoch für Swift, eine “multiparadigmatische Sprache” [WIKISWIFT01], die Ideen von vielen verschiedenen Programmiersprachen aufgreift und vereint.

Fazit:
Beide Entwicklungsplattformen ähneln sich und legen gemeinsamen Fokus auf kurze Entwicklungszeiten und schnelle Ausführungszeiten. Mangels Erfahrung mit beiden Architekturen, fiel Entscheidung für XCode einzig und allein aufgrund persönlicher Interessen, sich in SwiftUI einzuarbeiten.

Entscheidung der zu benutzenden Datenspeicherung

Die zu speichernden Daten lassen sich unterscheiden, in Stammdaten und Bewegungsdaten [LEXDB01]. Stammdaten haben eine lange Gültigkeit, d.h. sie werden nicht sehr häufig aktualisiert. Zum jetzigen Zeitpunkt ist vorgesehen, dass die Stammdaten im Lieferumfang der App enthalten sind. Der Nutzer kann diese Daten nicht ändern. Mögliche Erweiterungen soll es nur durch Updates der Fitness-Assistent-App selbst geben.

Die Bewegungsdaten werden mit jeder Aktion des Nutzers erfasst. Hier gibt es ständig Aktualisierungen. Von Bewegungsdaten werden immer ganze Sätze gespeichert, die einzelnen Datensätze werden später nicht mehr geändert werden.

Aufgrund der unterschiedlichen Anforderungen an die Gültigkeit und Umfang von Stamm- und Bewegungsdaten werden diese in in der Fitness-Assistent-App in unterschiedlicher Form gespeichert:

	JSON Files	Core Data
Auslesen von Daten	Leicht	Leicht
Ändern/Hinzufügen von Daten	Schwieriger	Leicht
Initiales füllen der Tabellen mit Daten	Leicht	Schwer

Tabelle 4: Datenspeicherung

JSON Files:
JSON steht für JavaScript Object Notation und ist ein kompaktes und einfach lesbares Datenformat. Es wird verwendet, um Daten zwischen Anwendungen auszutauschen. JSON ist Skriptsprache unabhängig, was mein Problem ist, da in allen verbreiteten Programmiersprachen Parser existieren.

Core Data:
Core Data ist ein Framework für die persistente Speicherung von Daten, das von Apple in macOS und iOS Geräten zur Verfügung gestellt wird.

Fazit:
Das Einbauen von anderen relationalen Datenbanken als Core Data ist unnötige Arbeit, da Apple dies schon für Programmierer übernommen hat. Somit gibt es keine Alternativen zu Core Data. Das initiale Füllen von Tabellen mit Core Date stellte sich komplizierter heraus als gedacht, da keine hilfreichen Tipps dazu zu finden waren. Der am meist verbreitete Ansatz war sowohl Core Data als auch JSON Files zu verwenden. Dabei werden die Stammdaten in ein JSON File ausgelagert und die Bewegungsdaten werden mit Core Data gespeichert.

Entscheidung der zu benutzenden Bildformate

Zu jeder Übung eines Workouts soll eine Instruktion stattfinden, sodass jeder Nutzer diese Übung ausführen kann. Die möglichen Formate für eine Animation unterscheiden sich nach Größe, Bildqualität und Kompatibilität.

	PNG-Format	GIF-Format	AVI-Format
Größe der Dateien	Klein	Klein	Zu groß
Kompatibel mit SwiftUI	Ja	Nein	Nein

Tabelle 5: Bildformate

AVI-Format:
AVI oder auch Audio Video Interleave ist ein Container-Format. Es kann mehrere Video-, Audio-, und Untertitel-Quellen enthalten, welche auch jeweils anders kodiert sind. Das AVI-Format vereint somit Bild, Ton und Untertitel in einer Datei.

GIF-Format:
GIF ist ein Grafikformat, das im Gegensatz zum PNG-Format bewegte Bilder erlaubt. Dabei werden mehrere Einzelbilder abgespeichert, die von z.B. Webbrowsern als Animationen dargestellt werden können.

PNG-Format:
Das PNG-Format ist das meistverwendete, verlustfreie Grafikformat im Internet. Das PNG-Format wurde entwickelt, um die Grenzen der GIF-Bilder zu heben, in Bezug auf Farbe Unterstützung und Lizenzierung von Patenten.

Fazit:
Da die AVI-Dateien zu groß gewesen wären und SwiftUI leider kein GIF-Format unterstützt, fiel die Entscheidung auf das PNG-Format. Um dennoch eine Animation der Instruktion der Übung zu bekommen, wurde ein Timer programmiert, der nach einer bestimmten Zeit das aktuelle Bild gegen ein anderes Bild austauscht und somit die Übung angemessen einleitet.

6. Fazit

Der Fitness-Assistent differenziert sich von ähnlichen Apps durch die verwendeten Algorithmen. Durch präzises Ausloten der Leistungsgrenzen und motivierende Präsentation der Erfolge, wird der Anwender zur stetigen Leistungssteigerung motiviert. Die Systemarchitektur unterstützt hervorragend die Berechnung von komplexen Algorithmen. Diese werden gespeist über umfangreiche Übungen und Aufzeichnungen der individuellen Leistung der Anwender. So wird die Fitness-Assistant-App zu einem einzigartigen Anwendererlebnis.