Praktikumsbericht

Florian Häusler
InterMedCon GmbH

26. Mai 2019

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Bereits vor meinem Praktikumsbeginn habe ich vom 1. Dezember 2018 bis 31. Januar 2019 als Werksstudent bei der InterMedCon GmbH gearbeitet. Da mir der Zeitraum für ein Praktikum ziemlich kurz erschien, ein neues Projekt im Dezember 2018 startete und ich im letzten Semester nur zwei Kurse besuchte. Hat es sich angeboten als Werksstudent in das Team und das neue Projekt einzusteigen, um eine spätere Einarbeitung zu vermeiden und das Projekt von Beginn an unterstützen zu können.

1.1 Praktikumsbetrieb

Die InterMedCon GmbH hat Standorte in Münster, Kiel und Berlin. Ein Großteil der Entwickler befinden sich zurzeit in der Dorotheenstraße Berlin, wo auch ich meinen Arbeitsplatz bekommen habe. Ein Umzug in größere Büros steht aber schon fest. Da die Bürokapazitäten noch begrenzt sind, ist den Mitarbeitern freigestellt ihre Arbeit auch im Home Office zu erledigen. Auch wenn der Platz im Büro manchmal etwas enger wurde, war ein konstruktives und gutes Arbeiten möglich. Vor Ort ist der Technische Leiter, zwei Webentwickler und ein weiterer Android Entwickler sowie Mitarbeiter von einem Tochterunternehmen. Die Personalabteilung und Geschäftsführung befinden sich in Münster bzw. Kiel.

Die InterMedCon GmbH hat sich auf eHealth Anwendungen spezialisiert, der Kundenstamm umfasst Private und Gesetzliche Krankenkassen sowie Pharma-Unternehmen. Dabei ist die InterMedCon GmbH in zahlreichen öffentlichen Forschungs- und Förderprogrammen vertreten und arbeitet mit internationalen Partnern zusammen u.a in Dänemark, Schweden und der Niederlande. Gerade für die Organisation und Aufbereitung Internationaler Projekte in Medizinischen Studien werden neue Softwarelösungen benötigt.

2 Projekt

2.1 Projektbeschreibung iSpine

Beim iSpine-Projekt handelt es sich um ein von der Europäischen Union gefördertes Projekt, welches im Verbund mit den Dänischen Partnern Sens Motion durchgeführt wurde. Es sollte eine Intelligente Selbstüberwachung für eine effiziente ambulante Behandlung von ambulanter Rückschmerzen entwickelt werden. Dafür sollte von der Firma Sens Motion ein neuer Sensor entwickelt werden, der verschiedene Bewegungsarten erkennen kann. Weiter sollte eine App zur Motivation zur Unterstützung eines aktiveren Lebens entwickelt werden, eine Anpassung für weiter Therapien und Krankheitsformen sollte gegeben sein. Die Sensoren sollen mittels Pflaster an den Oberschenkel angebracht werden, Sie sind für den einmaligen Gebrauch und min. 14 Tage wasserfest. Die Bewegungsarten die ein Sensor ermitteln soll sind: Ruhend, Gehen, Laufen, Fahrradfahren, Schrittzähler.

Die ersten Klinischen Tests bei unserem Partner in Kopenhagen waren für April 2019 geplant. Hierfür sollten 100 Patienten mit unspezifischen und chronischen Rückenschmerzen ausgesucht werden und wurden in 2 Gruppen eingeteilt. Die eine Gruppe sollte den Sensor und die App erhalten und die zweite Gruppe nur den Sensor ohne die App. Die Begleitung der Studie war für einen Zeitraum für 6 Wochen angelegt. Die groben Anforderungen an die App waren dabei: Sie sollte möglichst leicht bedienbar sein, da auch gerade älter Patienten und Einwanderer die App bedienen sollen. Es sollte die Möglichkeit geben Daten wie Alter, Gewicht, Größe zu hinterlegen, um so später Rückschlüsse auf Erkrankungen zu erkennen. Die Daten die von den Sensor von Sens Motion ermittelt wurden, sollten in die App integriert werden und der Patient sollte zu diesen Daten Feedback erhalten. Das Aktivitätsniveau sollte er dabei selbst definieren können.

2.2 Teammitglieder und deren Aufgaben

Beim iSpine Projekt handelt sich um die erst native Android-App die von der InterMedCon GmbH entwickelt wurde. Die Zusammenstellung des Teams erfolgte erst kurz vor dem Start des Projektes, im November. Die Projektführung wurde zum großen Teil von unserem Technischen Leiter aus Berlin und wurde später durch eine Wissenschaftliche Mitarbeiterin aus Kiel unterstützt. Des Weiteren waren zwei Werksstudenten /-in als Android Entwickler sowie eine Designerin für die Mockups eingestellt worden, die ihren Sitz nicht in Berlin hatten und ihre Arbeit im Homeoffice verrichteten. Ab Januar ist ein weiterer Android Entwickler, erst als Teilzeit und später als Vollzeitkraft zum Team in

Berlin hinzugekommen. Wie unter 1 Einleitung bereits erwähnt, habe ich erst als Werksstudent und ab Februar im Praktikum als Android-Entwickler das Team unterstützen dürfen.

2.3 Projektablauf

2.3.1 Beginn/Vorbereitung

Projektstart für die App-Entwicklung war der 3.Dezember.Das Gesamtprojekt startete aber bereits weit vor diesem Datum, mit der Entwicklung des Sensors und der API[1] für den Zugriff auf die Daten durch die Dänischen Partner. Es wurden bereits Studien geplant und erste Auswertungen mit der Hilfe einer Web-App durchgeführt. Zielgruppen wurden analysiert und grobe Anforderungen spezifiziert.

Wie bereits oben erwähnt startete am 3. Dezember mit einer Kickoff Veranstaltung in Form einer Telefonkonferenz mit den Dänischen Partnern und der Projektleitung die Entwicklung der App. Ich durfte auch an diesem Gespräch teilnehmen, hierbei wurde uns die Programmschnittstelle erläutert und Testzugänge eingerichtet. Weiter wurden Technische Anforderungen und eine grobe Zeitplanung besprochen. Dabei war gerade aus unser Position eine konkrete Aussage zu treffen sehr schwer. Da viele Faktoren unklar waren, wie zum Beispiel Personalstärke, Fähigkeiten und Funktionsweise der API und Auswertung der Daten. Hierbei wurde auch entschieden, dass wir nicht die Synchronisation des Sensors vornehmen, sondern dies durch eine bereits vorhandene App realisiert werden soll. Um durch den begrenzten Zeitraum bis zum Studienbeginn zu erleichtern. Ein erster Prototyp sollte bereits Mitte bis Ende Januar fertig sein und die erste Studie sollte dann Ende Februar beginnen.

Kurz darauf folgte die erst Telefonkonferenz zwischen dem in Kapitel 2.2 beschrieben Entwickler-Team statt. Hierbei wurden die Inhalte der Telefonkonferenz mit den Dänischen Partnern ausgewertet.Weiterführend musste sich auf die einzusetzenden Werkzeuge geeinigt werden, da das Team an mehreren Standorten ihre Arbeitete verrichtete.

Nachfolgend wurde die Arbeit mit folgenden Arbeitsmitteln unterstützt:

Slack zur Kommunikation innerhalb des Teams, als auch mit den Dänischen Partnern.

GIT als Versionsverwaltungssoftware, was das Arbeiten am selben Code sehr vereinfacht.

Taiga als Projektmanagementsoftware zur agilen Softwareentwicklung. Später konnte hier auch ein Bug-Report eingerichtet werden, wo die Dänischen Partner Fehler hinterlegen konnten.

GoTo-Meeting für die Wöchentlichen Sprints die als Telefonkonferenzen durchgeführt wurden.

Android-Studio Eine von Google und IntelliJ basierende Entwicklungsumgebung.

[1] Aplication Programming Interface

Es wurde sich auch auf eine Wöchentliche Iteration geeinigt, wo jeder seine Arbeitsfortschritte und Probleme im Arbeitsprozess vortragen konnte.

2.3.2 Durchführung

Wie bereits erwähnt war unsere App abhängig von den Daten die wir von der API bekommen. Daher mussten einige Punkte im Vorfeld geprüft werden:

- Wie erfolgt die Authentifizierung?
- Wie und wer legt neue Patienten an?
- Welche Daten stehen uns in der API zur Verfügung?
- Welche Daten brauchen wir?
- Welche Request müssen gesendet werden, um an die gewünschten Daten zu bekommen?

Hierbei kam es gerade in der Anfangszeit zu Problemen. Da es sich bei der API um eine Beta Version handelte, weitere Änderungen vorgenommen wurden und die Verfügbarkeit eingeschränkt war. Wir haben dafür eine kurze API Dokumentation bekommen und Testanfragen über Postman realisiert, um das Verhalten der API zu analysieren. Dabei war eine funktionierende Kommunikation mit den Dänischen Partnern wichtig! Nachdem die Fragen geklärt waren konnten erste Mockups und User Storys erstellt werden. In der APP-Entwicklung konnten dann die ersten Anmeldeseiten erstellt werden. Außerdem wurden Bibliotheken recherchiert, die das Anfrage und Verarbeiten der Daten von der API erleichtern sollten. Die Betaversion der API und neue Anforderungen machte ein lokales Zwischenspeichern der Daten unverzichtbar. So kam es zu Zeiträumen in der die API nicht erreichbar war, oder es nach der Synchronisation des Sensors 10-20 min dauerte bis Daten von der API geladen werden konnten. Damit war das vorhergehende Speichermodell nicht mehr möglich/nötig und es musste auf eine lokale Datenbank umgestellt werden. Weitere Anforderungen wie eine Hintergrundaktualisierung oder Notifikation machte den Einsatz von LiveData (Observer) Notwendig. Dabei wurden Daten die durch die API in der Datenbank verändert wurden, in der aktuellen Ansicht aktualisiert. Es sollten auch eigene Daten mit Hilfe der API hochgeladen werden. Damit die Nutzer bei dem Wechsel ihres Gerätes nicht ihre Persönlichen Daten erneut eingeben müssen.

2.3.3 Abschluss

Zwei bis drei Wochen vor dem Start der Studie, bekamen die Kliniker die App ausgehändigt um erste Tests durchzuführen. In einer Wöchentlichen Telefonkonferenz wurden kleine Änderungen,Erweiterungen oder Probleme besprochen. Die wie bereits beschrieben im Bug-Report festgehalten und gegebenenfalls in die App intrigiert oder behoben wurden. Hierbei kam es auch zur Anpassungen an der API, so dass der Anmeldeprozess

umgestellt wurde, von Nutzernamen und Password auf Telefon und SMS/Pin. Durch den Einsatz der Bibliotheken war die Umstellung nicht so ein großes Problem wie erst vermutet. Es mussten nur wenige Klassen und UI-Elemente angepasst werden. Weiter musste getestet werden ob alle Notifiakations funktionieren, ob die Hintergrundsynchronisation mit der externen App funktioniert. Da die Studie in Dänemark durchgeführt wurde, mussten alle UI-Elemente übersetzt und deren konkrete Übertragung getestet werden. Da in der Studie Probanden auch ihre eigenen Geräte nutzen durften, war es wichtig die App auch für verschiedene Displaygrößen auszurichten. Ein großer Teil der Studie nutzte allerdings bereitgestellte Smartphones, diese Geräte mussten beschafft konfiguriert und mit einer MDM[2]ausgestattet werden. Damit spätere Updates oder Erweiterungen schneller eingespielt werden können. Hierbei wurden alle erkannten Bugs erfolgreich behoben und die Studie konnte lediglich durch eine kurze Verzögerung durch die Lieferung der Smartphones gestartet werden. Während der Studienphase gab es weitere Technischem Gespräche mit den Dänischen Partnern. Dabei kamen lediglich nur noch kleinere Fehler zum Vorschein, die schnell behoben werden konnten. Die Studie konnte dann erfolgreich starten, die Evaluation allerdings steht noch aus.

2.4 Aufgaben am Projekt

Wie in Kapitel 2.3.1 beschrieben, durfte ich an der ersten Kickoff Veranstaltung mit den Dänischen Partnern teilnehmen, um auch erste technische Fragen bezüglich der einzusetzenden API stellen zu können. Außerdem war es wichtig das Dänische Team kennenzulernen, da die Umsetzung eine enge Zusammenarbeit voraussetzt. Dies war wichtig da ich nachfolgend im Team für die Datenbeschaffung über die API verantwortlich sein durfte.

Dabei war es erst mal notwendig den Request Ablauf der API herauszufinden, um die in Kapitel 2.3.2 gestellten Fragen beantworten zu können. Dies ging einerseits über eine kleine Webdokumentation und da nicht alle Endpunkt richtig beschrieben waren über Postman. Hiermit konnte erstmals außerhalb der Anwendung das Verhalten der API analysiert werden, die gewünschten Request anfragen gefunden und die ersten Daten empfangen werden. Nachfolgend sollte ich eine Lösung finden, die Daten von der API in unsere App zu übermitteln. Hierbei hat sich die Bibliothek Retrofit sehr nützlich herausgestellt. Bei Retrofit werden die Anfragen in einem Interface beschrieben wie Header, Anfrage (POST, GET), Response Objekt , Request Objekte und Endpunkt der API. Retrofit setzt mit der Hilfe der Basisadresse die Anfrage zusammen und liefert die Antwort wieder zurück. Diese Request und Response Objekte werden als POJO[3] Objekte beschrieben und die Bibliothek kümmert sich um die richtige Serialisierung. Ein manuelles parsen der JSON Objekte ist somit nicht mehr notwendig, dies hat sich gerade bei Umstellungen bei API Anfragen oder bei Veränderungen von Objekten vorteilhaft erwiesen.

[2]Mobile-Device-Management
[3]Plain Old Java Object

Da sich die API wie in 2.3.2 beschrieben im Verlauf des Projektes als nicht zuverlässig herausgestellte und wir zum Vorantreiben und testen der UI-Elemente sowie der weiteren Logik auf Daten angewiesen waren, haben wir uns für ein Caching der Daten entschieden. Ich durfte mir darüber Gedanken machen und wir entschieden uns für auf die Room Bibliothek. Es handelt sich dabei um eine Relationale Datenbank und man kann die von der API bereits verwendeten POJO Objekte für das ORM[4] wiederverwenden. Die Bibliothek macht den Zugriff auf die Datenbank einfach über ein DAO[5] möglich und übernimmt ebenfalls das OR[6]-Mapping. Der Einsatz von Room erleichterte die Programmierung, durch den Einsatz von Konvertern. Die es ermöglichten auch nicht Primitive Datentypen zu Speichern . Durch den Umstieg auf die Room Datenbank wurde das vorhandene Speicherkonzept verworfen und die Funktionen in die neue Datenbankstruktur migriert. Vorher haben wir unsere Daten in SharedPreferences für die dauerhafte Speicherung und während der Laufzeit auf das Singleton-Muster gesetzt. Ein ER-Modell der Datenbank befindet als Abbildung:5.1 im Anhang.

Wie in 2.3.3 beschrieben kamen es durch Veränderungen an der API noch zu weiteren Anpassungen innerhalb unserer App. So wie unten erwähnt sollte der Authentifizierungsprozess geändert werden, es sollten eigene Metadaten gesendet und verarbeitet werden. Wir haben weitere Metadaten erhalten, die einen Feedback Zeitraum beinhalteten. So konnte die in 2.1 beschriebene Feedbackperiode hinzugefügt und die persönlichen Daten auf dem Server gespeichert werden.

Weitere Anforderungen und der Einsatz von Hintergrundaktualisierungen machte weitere Veränderungen notwendig. Hierbei durfte ich mich mit dem Einsatz von LiveData beschäftigen. LiveData ist ein Observer-Muster, es ermöglichte LifeCycle Konflikte zu verhindern und das Zusammenspiel zwischen API und Room-Datenbank zu verbessern. Da man die Tabellen einer Datenbank direkt als Observable Objekt deklarieren kann, werden Veränderungen die durch die API hervorgerufen werden direkt an die aktuelle View weitergegeben.

Zum Ende des Projektes durfte ich noch bei Frontend Arbeiten unterstützen. Dabei ging es hauptsächlich um responsive Anpassungen zwischen den verschiedenen Display Größen(Smartphone,Tablet), kleinere Fehlerbehebungen und das Implementieren weiterer Fitnessübungen in Form von Bildern (Siehe Abbildung: 5.3). Final wurden Smartphones beschafft, die installiert, konfiguriert und versendet werden mussten. Dies Stellte den Abschluss des Projektes da.

[4]Object Relatoin Mapping
[5]Data Access Object
[6]Object Relatoin

3 Bezug zum Studium

Dadurch dass wir in der Universität in zahlreichen Projekten bereits schon mit Git, Android Studio und anderen Entwicklungsumgebungen, Slack und Projekt-Management Tools wie Trello gearbeitet haben, brauchte es in diesen Bereichen keine lange Einarbeitungszeit und es konnte sich auf inhaltliche Probleme und dessen Umsetzung konzentriert werden.

Wie durch die vorherigen Kapitel erkennbar, wurde die Entwicklung mithilfe von Scrum vorangetrieben. Mit dem Project Owner in Kiel, den Scrum Master in Berlin und das Entwicklungsteam zu dem ich auch gehörte. Allerdings gab es keine täglichen Sprints, sondern nur Wöchentliche, da unser Entwicklungsteam nicht Vollzeit und in Deutschland verteilt arbeitete. Dabei half es die einzelnen Rollen zu kennen, sich auf die Wöchentlich Sprints Vorzubereiten bzw. den Scrum Master bei Problemen rechtzeitig informieren zu können. Damit Probleme bei nachfolgenden Sprints und im Projekt Plan berücksichtigt werden konnten.

Ich denke bei meiner Arbeit mit der Rest API konnte ich am besten aus dem Wissen von Komponentenbasierte Entwicklung und Verteilte System zurückgreifen. Gerade bei der Serialisierung von Objekten mit POJOs oder das OR-Mapping waren mir aus den Vorlesungen bekannt und halfen mir bei der Umsetzung in diesem Projekt. Bei der Rest API hatte ich bereits ein grundlegendes Verständnis wie die Kommunikation abläuft, wie die Fehlercodes aussehen, was sie bedeuten sowie Anfragen und Antworten ablaufen.

Da ich den Schwerpunkt im Studium auf Mobile-Anwendungen gelegt habe, hatte ich bereits Grunderfahrungen in der Entwicklung von Smartphone Apps in iOS sowie Android. Wir hatten uns bei der Entwicklung auf die Sprache Java und gegen Kotlin geeinigt, so musste auch hier kein großes Umlernen erfolgen, da quasi 80 % des in der Universität produzierten Codes Java war. Hier half auch das der Umgang mir Android Studio und deren Struktur, sowie die Paketverwaltung Gradel, deren Problem mir bekannt waren.

Da wir auch eine Datenbank zum Caching einsetzten, konnte ich vom Wissen aus dem Modul Datenbanken zurückgreifen. Hinsichtlich der Beziehungen zwischen Tabellen und Erstellen von Datenbank abfragen

Weiterhin wirkte sich das Umsetzen von kleineren Projekten innerhalb der Kurse aus, womit erlernte Fähigkeiten verfestigt wurden und wieder schneller ins Gedächtnis gerufen werden.

Neu war der Umgang auch mit fremden Code der interpretiert und verstanden werden musst. Wo man sich vorher zum größten Teil nur mit seinem eigenen Code auseinandersetzen musste, war es nun wichtig zu berücksichtigen das auch andere diesen Code lesen und verstehen müssen. Hierbei war die Vergabe von Variablen, Klassennamen usw. sowie das Dokumentieren wichtig. Dies wurde im abschließenden Refactoring Prozess sichtbar.

4 Ausblick

Nach einem erfolgreichen Abschluss der Studie, wird mit den Dänischen Partnern überlegt die Arbeit in Folgeprojekten (iSpine 2.0) fortzusetzen. Hierbei wird darüber nachgedacht die App zur Synchronisation des Sensors zu integrieren. Weitere Überlegungen sind es auch die Auswertung nicht in der Cloud zu erledigen sondern auf dem Gerät selbst. Dem Nutzer sollte auch noch eine bessere Auswertung seiner Daten geboten werden, wie zum Beispiel vergleichen der Aktivitäten bestimmter Tage.

Es wird hierfür noch nach Partnern bei Krankenkassen oder weiteren Studien gesucht.

5 Anhang

5.1 ER-Modell

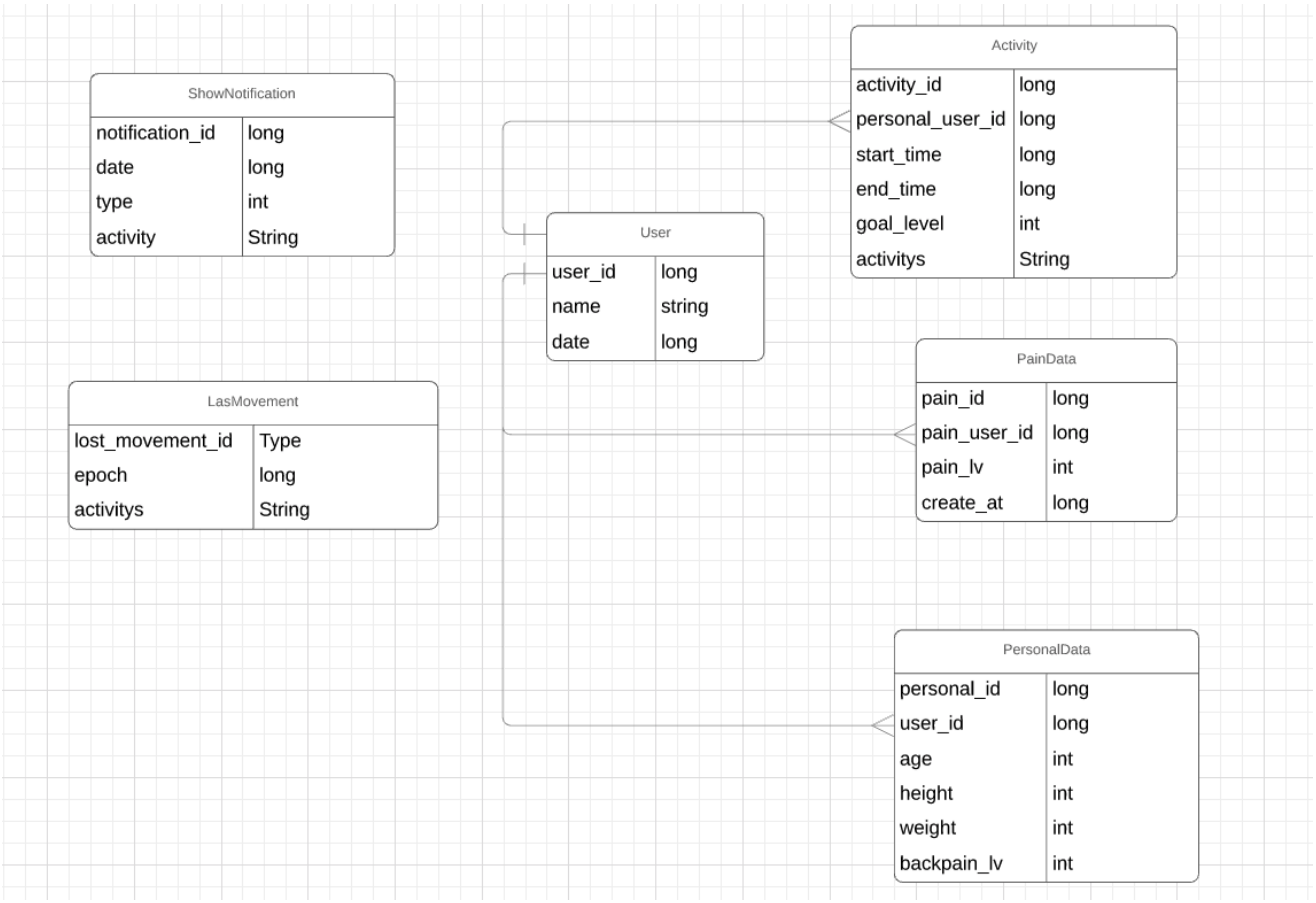

Abbildung 5.1: Grobes Datenbankmodell

5.2 Abbildungen

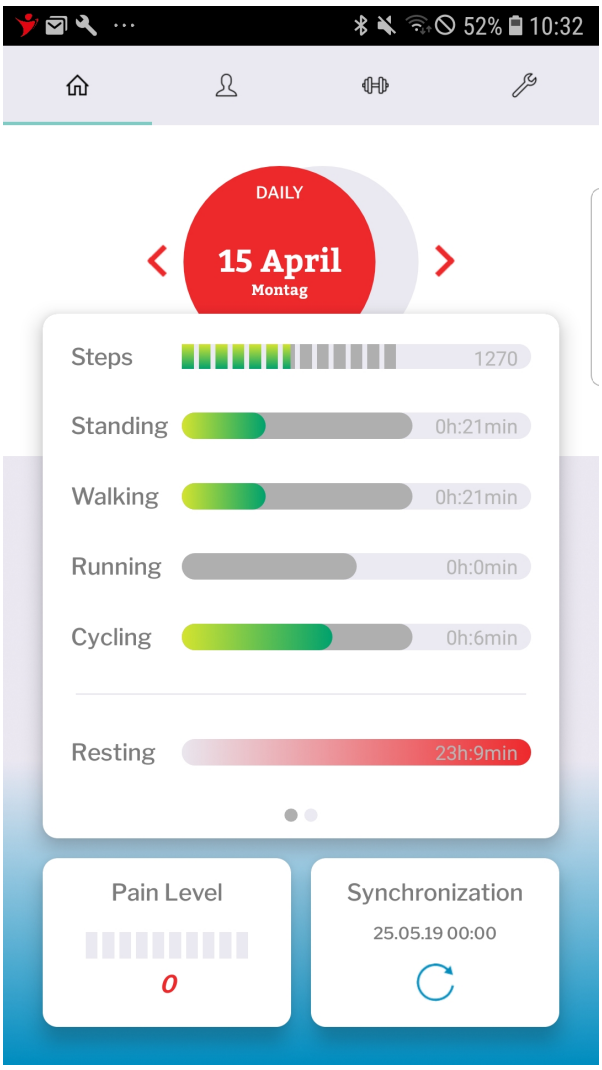

Abbildung 5.2: Aktivitäten mit den Daten von der API

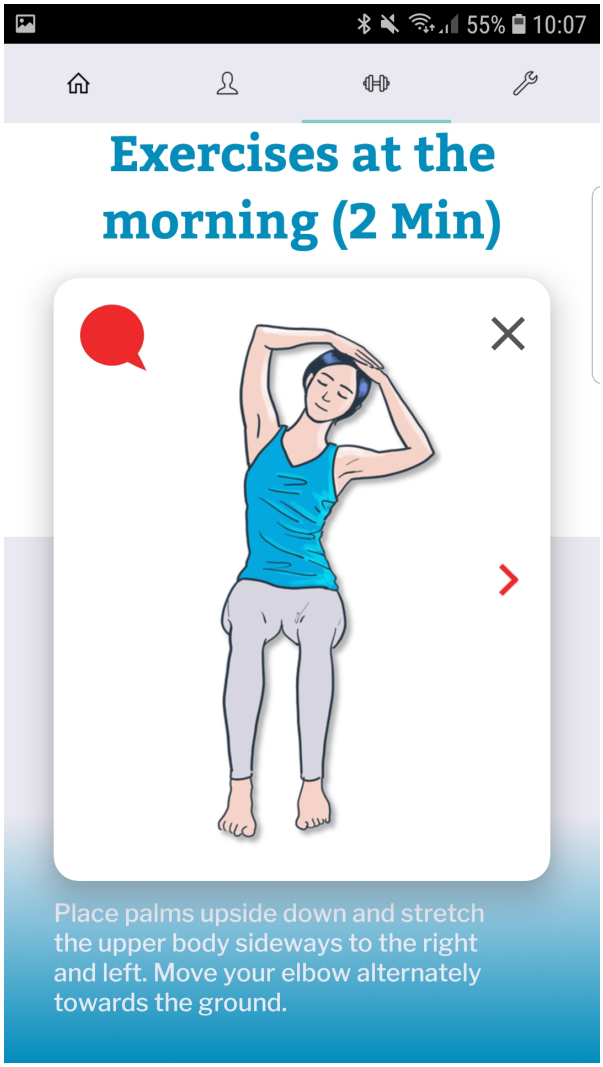

Abbildung 5.3: Fitnessübungen zu einem bestimmten Schmerzbereich

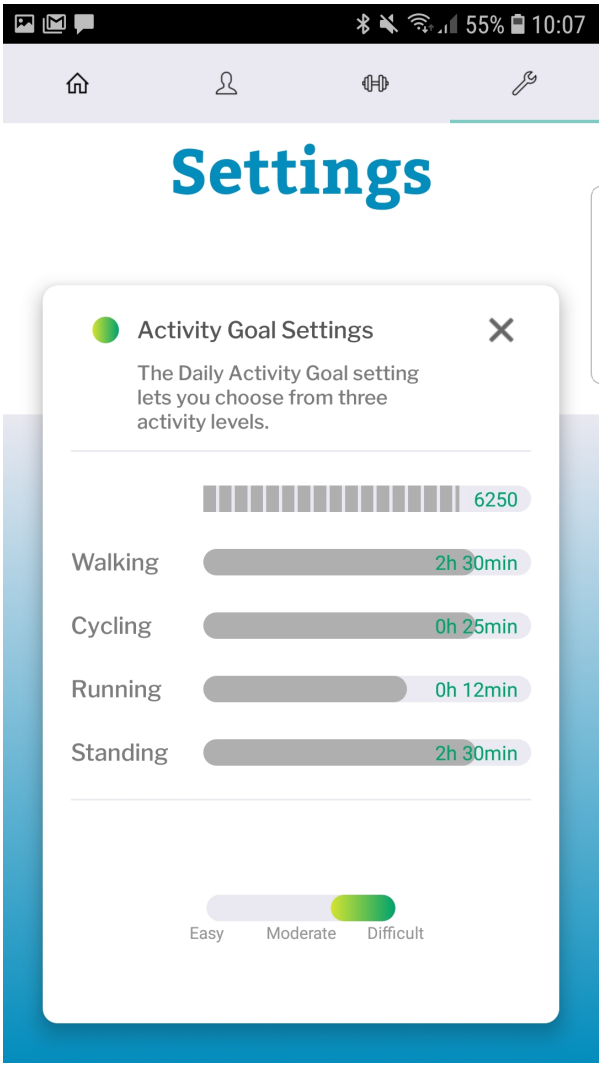

Abbildung 5.4: Einstellung der Schwierigkeitslevels

6 Abkürzungsverzeichnis

POJO Plain Old Java Object

ORM Object Relatoin Mapping

OR Object Relatoin

DAO Data Access Object

API Aplication Programming Interface

MDM Mobile-Device-Management

7 Glossar

Bug Fehler in einer Software. 8

GIT Ist ein freis software zur verteilten Versionsverwaltung von Dateien. 6

LiveData Live Data folgt dem Beobachter-Muster. Ist eine Klasse die Daten beinhaltet und mit einem gegebenen Lebenszyklus diese Beobachtet. Der Lebenszyklus Besitzer wird nur im Status STARTED oder RESUMED bei Veränderungen informiert.. 7

Postman Ist ein Werkzeug zum Testen von REST API. 7, 8

Retrofit Ist eine Bibliothek zur Verarbeitung von API Anfragen. 8

Room Ist eine Bibliothek für eine lokale Relationale Datenbank und OR-Mapper. 9

SharedPreferences Persistente Speicherung von Daten in eine XML Datei als Key, value werte. 9

Ort, Datum	Unterschrift Praktikant	Ort, Datum	Unterschrift Praktikumsbetrieb